Preference Learning for Machine Translation

Dissertation of Patrick Simianer

Heidelberg University, Germany

Dissertation zur Erlangung der Doktorwürde der Neuphilologischen Fakultät der Ruprecht-Karls-Universität Heidelberg vorgelegt von Patrick Simianer, M.A.

Erstgutachter: Stefan Riezler, Universität Heidelberg
Zweitgutacher: Marcello Federico, Fondazione Bruno Kessler

16. Oktober 2018

Abstract

Automatic translation of natural language is still (as of 2017) a long-standing but unmet promise. While advancing at a fast rate, the underlying methods are still far from actually being able to reliably capture syntax or semantics of arbitrary utterances of natural language, way off transporting the encoded meaning into a second language. However, it is possible to build useful translating machines when the target domain is well known and the machine is able to learn and adapt efficiently and promptly from new inputs. This is possible thanks to efficient and effective machine learning methods which can be applied to automatic translation.

In this work we present and evaluate methods for three distinct scenarios: a) We develop algorithms that can learn from very large amounts of data by exploiting pairwise preferences defined over competing translations, which can be used to make a machine translation system robust to arbitrary texts from varied sources, but also enable it to learn effectively to adapt to new domains of data; b) We describe a method that is able to efficiently learn external models which adhere to fine-grained preferences that are extracted from a constricted selection of translated material, e.g. for adapting to users or groups of users in a computer-aided translation scenario; c) We develop methods for two machine translation paradigms, neural- and traditional statistical machine translation, to directly adapt to user-defined preferences in an interactive post-editing scenario, learning precisely adapted machine translation systems. In all of these settings, we show that machine translation can be made significantly more useful by careful optimization via preference learning.

Acknowledgments

> "The difference between the right word and the almost right word is the difference between lightning and a lightning bug."
> [Mark Twain]

First of all I would like to express my gratitude to Stefan Riezler, for introducing me to (statistical) machine translation — which I found to be a worthwhile subject to spend one's professional life with — and for encouraging me to write both Master's and Ph.D. theses on just this topic. I would also like to thank Marcello Federico for hosting an internship whose results turned out to be part of this work, as well as for agreeing to be my second advisor.

During my time at Heidelberg University I was glad to spend time with a number of great people — Carolin, Duy, Felix, Gesa, Julia, Julian, Katharina, Kathrin, Laura, Markus, Mayumi, Samuel, Sandra, Sariya, Sascha, and Shigehiko, amongst others.

Luckily, I was also able to spend some time abroad, where I enjoyed great hospitality and was excellently advised — in Cambridge, UK by Adrià de Gispert, Gonzalo Iglesias and Bill Byrne, and in Trento, Italy by Marcello Federico and Nicola Bertoldi, with support from the *MateCat* project.

Of course, my family and friends also played a large role in the completion of this thesis, always giving me strength and support when I needed it the most.

But, above all, I would like to thank my wife Eugenia for her love and support, without none of this would have been possible.

This work has been supported in part by the German Research Foundation (DFG) under the grants *Cross-language Learning-to-Rank for Patent Retrieval*, and *Grounding Statistical Machine Translation in Perception and Action* (RI-2221/2-1).

Contents

List of Algorithms

List of Figures

List of Tables

Abbreviations

AI	Artificial Intelligence
ALPAC	Automatic Language Processing Advisory Committee
ASR	Automatic Speech Recognition
BLEU	Bilingual Evaluation Understudy
CAD	Computer-Aided Design
CAT	Computer-Aided Translation (or: Computer-Assisted Translation)
DAG	Directed Acyclic Graph
DARPA	Defense Advanced Research Projects Agency
DFG	Deutsche Forschungsgemeinschaft (German Research Foundation)
EPO	European Patent Office
FAMT	Fully Automatic Machine Translation
FAHQMT	Fully Automatic High Quality Machine Translation
FAHQT	Fully Automatic High Quality Translation
FST	Finite-State Transducer
HAMT	Human-Aided Machine Translation
HBLEU	Human-Targeted BLEU
HTER	Human-Targeted Translation Error Rate (Or: ... Edit Rate)
IAMT	Interactive Machine Translation
IPC	International Patent Classification
IR	Information Retrieval
IWSLT	International Workshop on Spoken Language Translation
KSMR	Keystroke and Mouse Action Ratio
KSR	Keystroke Ratio
LMEM	Linear Mixed-Effects Model
MAHT	Machine-Aided Human Translation
MAP	Mean Average Precision
MAR	Mouse Action Ratio
MBR	Minimum Bayes-Risk
MERT	Minimum Error Rate Training
MIRA	Margin-Infused Relaxed Algorithm
ML	Machine Learning
MLE	Maximum Likelihood Estimation
MT	Machine Translation
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
NP	Non-Deterministic Polynomial-Time
NTCIR	NII Testbeds and Community for Information Access Research

OOV	Out-of-vocabulary
PBMT	Phrase-Based Machine Translation
PE	Post-Editing
PRO	Pairwise Ranking Optimization
PWR	Pause to Word Ratio
RNN	Recurrent Neural Network
RR	Repetition Rate
SCFG	Synchronous Context-Free Grammar
SGD	Stochastic Gradient Descent
SMT	Statistical Machine Translation
SVM	Support Vector Machine
TER	Translation Error Rate (or: Translation Edit Rate)
TM	Translation Memory
WFST	Weighted Finite-State Transducer
WIPO	World Intellectual Property Organization
WMT	Workshop on Machine Translation (now: Conference on Machine Translation)
WPA	Next Word Prediction Accuracy
WSCFG	Weighted Synchronous Context-Free Grammar
WSR	Word Stroke Ratio

Notation

N, n	Variables
$\mathbf{M}$	Matrix
$\mathbf{v}$	Vector
$\mathcal{S}$	Set
$\mathbb{R}$	Reals
$\mathbb{N}$	Integers
$\mathbb{N}_+$	Positive natural numbers excluding 0.
$\mathbb{R}^n$	Real vector space with dimensionality n.
$\mathfrak{S}$	Feature space
$\mathbf{X}^{(k,i)}$	Index i relative to offset k in a ranking vector or matrix (real index $j = k + i$).
$\mathcal{S}^{(i)}$	Element at position i of the (ordered) set $\mathcal{S}$.
$\mathbf{v}_i$	Component i of vector $\mathbf{v}$, or vector with index i within a sequence of vectors.
$\mathbf{M}_{ij}$	Component and index i, j in matrix $\mathbf{M}$.
$\mathbf{M}_i$	Column i of matrix $\mathbf{M}$.
$\mathbf{M}^{(j)}$	Row j of matrix $\mathbf{M}$.
$f_{g,h}$	Function f with a dependency on objects g and h.
$\lvert \cdot \rvert_p$	ℓ_p norm
$\lVert \mathbf{W} \rVert_{p,q}$	ℓ_p/ℓ_q matrix norm
iff	If and only if.
$\langle \cdot, \cdot \rangle$	Dot product
$(\cdot)_+$	$\max(0, \cdot)$
$\cdot$	Placeholder

All vectors are assumed to be column vectors unless noted otherwise. Notation can be used differently in some contexts.

1 Introduction

> „Und wenn es auch nur ein kleines und schäbiges Krystall wäre, aber doch eins."
> [Wittgenstein, 1930, Denkbewegungen, a.a.O., S. 21]

Translation is a complex task for humans, but it is a even more daunting task for machines. Much research has been carried out to enable machines to "understand" natural language, arguably without a major breakthrough to date (2017) — possibly because natural language is tremendously different from what machines are typically able to comprehend. Syntax, pragmatics or semantics of natural language seem to be difficult concepts to capture with the precision that computational processing demands.

Translation between a pair of natural languages[1] to be performed automatically by a machine was one of the first applications for computers after the second world war and can be considered a seminal work in the field of natural language processing (NLP). After some first successes, there was a decline in interest, due to the technology seemingly just not being adequate for the problems at hand, which was infamously recorded in the report by the Automatic Language Processing Advisory Committee (ALPAC) [Pierce and Carroll, 1966].

Methods based on statistical modeling breathed new life into the machine translation (MT) research and practice due to their natural capability of dealing with the ambiguities found in natural language. The statistical models of language translation have evolved a lot since then, and today they are a readily available tool in a globalized world for use in a wide range of practical applications.

As powerful as these methods are, the general problem of language and its inherent ambiguity still remains. The statistical models we are concerned with are learned from data, and since language is virtually infinite[2], the data can in any case be only a more or less insufficient sample. To build meaningful models of translation, the data used in (statistical) machine translation (SMT) most commonly originates from texts covering a range of different topics, times, contexts and authors, to maximize coverage and in an attempt to estimate reliable statistics. This poses a new problem: Translations produced with these models tend to represent the lowest common denominator between all the different aspects that constitute the product of translation. In addition, as the sheer combinatorial space that

[1] We will call a pair of natural languages simply a language pair, the languages in the pair are referred to as source and target, respective of the translation direction.

[2] This statement holds true for the combination of existing vocabulary to coherent utterances (sentences), as well as the production of new items in a language's vocabulary.

applies in translation is so large, (some) models in SMT pose strong independence assumptions in their choice of translation units[3], which further aggregates the problems. With machine translation modeled by neural networks, it is possible to build richer statistical models, which rectify this problem to some extent, but still not completely.

While overcoming the immanent need for hand-written rules for translation, the statistical models have their own set of problems. The symptoms of these problems can be acceptable if machine translation is just used as a means for getting a general idea[4] of a text in foreign language, but they can become a severe problem if a translation is meant for direct human utilization, or if it is used as a starting point for a human translator: Contexts may demand certain wording in order to convey the right meaning, or authors may prefer to translate certain passages differently. As variation is endless, these issues are a constant factor in dealing with statistical models for NLP.

In the literature the topic, genre, and style of a text are broadly referred to as its *domain* [van der Wees et al., 2015]. Domain is an important concept in NLP, as well as machine translation where the goal is to produce a semantically equivalent target-language utterance to a given source input, that was in turn generated in a specific context. Capturing the context in translation is difficult as the domain is usually only implicit and its characteristics inherently vague. The alterations in the language going from one domain to another can be manifold: Ambiguities may have to be resolved differently, structural properties such as word order are subject to change, as well as the use of idiomatic expressions. Adapting an existing statistical model to a new domain is called *domain adaptation*, effectively changing a model to better reflect the current domain of interest.

If machine translation is used as a tool for humans[5] e.g. for professional translators, the domain adaptation problem has even higher significance: The event of receiving N different translations when having N people translate the same source text is quite likely, arguably close to certainty, depending on the source text. This imposes another dimension onto the domain adaptation problem as described before, since preferred wordings and expressions of single authors are delicate to capture and model. Furthermore, certain external requirements regarding terminology and consistency are likely to be postulated in a professional environment. So it may be safe to say, that adaptation of MT in this setting is a different compared to general domain adaptation. We argue that both general domain adaptation, and a more specialized adaptation have to be implemented if the goal is to make MT the most useful tool it can possibly be for human translators.

[3] These may be characters, words or phrases depending on the model at hand.
[4] What is sometimes called a “gist” or “gisting”.
[5] Human-machine interaction in the field of translation is most commonly referred to as computer-aided translation (CAT).

In the simplest case the success of domain adaptation efforts in MT are evaluated in terms of translation quality given a set of prefabricated reference translations. Most often, quality is used in the sense of similarity between translation outputs and these reference translations. In the case of CAT, machine translations may also be evaluated in terms of their utility to a translator, and by derived measures such as translation speed.

In the work presented here, we seek possible remedies or solutions for three problems: a) general translation quality improvements[6], by efficiently exploiting large amounts of data from varied sources, learning useful global ranking models e.g. by finding commonalities in the data, b) a more fine-grained adaptation for a CAT scenario, making use of methods that try to automatically learn specific preferences which are given by observed, possibly user- or domain-specific data, and c) learning to adapt to individual translators also in a CAT scenario by directly interacting with translators by exploiting the most direct feedback possible.

The problems a), b) and c) are approached as preference learning problems, and more specifically as ranking problems, where the goal is to learn a ranking function which adheres externally induced preferences.

We now turn to describe our learning framework which we employ for approaching these problems.

1.1 Preference Learning for Machine Translation

Given an sufficient amount of data, the MT systems discussed in this work are able to generate an elusively large set of translation alternatives for any given utterance[7]. It is quite likely that an adequate translation is contained in that set, or at least some acceptable translation candidate. However, as evident by the lack of readily available fully automatic high-quality machine translation (FAHQMT), the best translation hypothesis selected by a system as its output is more often than not the wrong choice. There are a number of different reasons for this behavior: Insufficient statistical evidence in the training data of the MT system, impossible required configuration of translation units, or simply unknown new vocabulary. The most common reasons are however *modeling-* and *search errors*: Due to the huge output space, which renders exact search methods infeasible, traditional machine translation systems heavily rely on independence assumptions and use a simple scoring function for decoding that introduces further approximations through *pruning*.

It is a recurring theme in MT, that while the system is theoretically able to

[6] The developed method can also be used for efficient domain adaptation, improving domain-specific translation quality, which we also empirically substantiate.

[7] The number of possible translations depends of course on the length of the given utterance.

produce very good translations for most utterances, it can just not be scored as such by the model or is lost in search, see e.g. the discussions in [Sokolov et al., 2013] or [Och et al., 2004].

Variants of pruning are implemented in the underlying statistical models, as well as the *scoring function*, that is used to assess the fit of partial translation hypotheses that make up the final translation output (a complete hypothesis). In SMT, this scoring function is generally quite simplistic, i.e. a linear model, and it is prone to select globally non-optimal translations due to its local pruning choices.

Taking a simplified view, the scoring function has to select a number of hypotheses from a list of possible extensions at each step of the translation process. In other words, the scoring function produces a *ranking* of the possible (partial) translations. This insight enables the application of a powerful class of algorithms which can be subsumed under the umbrella term *preference learning*[8]. Fundamentally for our work we may learn preferences from correct rankings (the learning-to-rank approach, specifically pairwise ranking, see e.g. [Herbrich et al., 1999; Joachims, 2002; Watanabe et al., 2007*b*; Hopkins and May, 2011]), as well as from structured representations of translations deemed correct (structured prediction approach, see e.g. [Collins, 2002]).

As already mentioned, correctness is a fuzzy concept in translation, which is why we will resort to similarity-based evaluation metrics for the most part of our work, i.e. what we call translation quality. Both pairwise ranking and structured prediction have in common that they are trying to learn a linear scoring function in such a way that the function assigns higher scores to higher quality translations and lower scores to ones of less quality. This in turn should result in overall better quality in the translation output. To put it slightly more formal: In the pairwise ranking approach we learn a scoring function from rankings of translation hypotheses according to a metric of translation quality, which is evaluated w.r.t. some form of reference translations; In the structured prediction approach the objective is focused on separating a single "best" (or correct) translation and its associated structure from competing translations and their respective structured representations.

We are now in a position to further specify our problem statements with regard to the just outlined proposed learning framework:

a) To improve general translation quality, we seek to learn a robust scoring function from large amounts of data employing a pairwise ranking algorithm for preference learning.

[8]Note that we generally use this term from a ranking perspective, see e.g. [Fürnkranz and Hüllermeier, 2003] or [Fürnkranz and Hüllermeier, 2010].

b) In the second approach, turning to a CAT scenario, we seek to improve translation quality following the same general approach as before, but using a structured prediction approach to directly learn from the translations provided by translators, in an attempt to obtain more specialized models with preference learning.

c) Finally, we explore the most direct possible approach for preference learning by requesting and exploiting direct user feedback to MT outputs, which can then be used for adaptation.

Problems b) and c) need further differentiation: In the CAT scenario it is crucial to effectively learn more fine-grained to effectively adapt the translations outputs, which is why we explore adaptation with two different approaches: Engaging in problem b), we learn to prefer user supplied translations over competing translation hypotheses, by using representations of their exact outputs as target structure in the structured prediction approach. This enables a very direct mode of learning. However, in computer-aided translation, we can exploit even finer-grained feedback since there is a human in the loop. By asking for explicit feedback for MT outputs from translators, we are in a position to effectively learn user-specific preferences directly. In c) we also investigate adaptation of a neural network based MT system.

The problem a) can be considered as a classical tuning task for SMT. In contrast, problems b) and c) are evaluated as post-editing (PE) tasks[9] — revealing a reference translation or user-generated translation to the learning algorithm immediately after evaluating the quality of the originally proposed machine translation. Due to the nature of all of these problems, they can be approached using traditional online learning methods.

1.2 Outline

Our work is structured as follows:

In Chapter 2 we provide the necessary background for the used MT approaches as well as preference learning, which we discuss from the learning-to-rank perspective. For SMT, formal introductions for word-, phrase- and hierarchical phrase-based models are presented, with a focus on the latter since it is the model most prominently used in our work. Also provided is a formalization of neural machine translation (NMT), which we apply for online adaptation in a PE task. We also formulate brief historical overviews of MT in conjunction with CAT, since research and practical application of both fields are closely interleaved. Since our work is

[9] Problem c) allows however for a different set of methods, since it actually considers a human-in-the-loop.

assessed by measuring relative differences in translation quality, manual translation effort and/or translation speed, we provide definitions of various automatic translation metrics, and discuss their relation to human judgments. We further discuss domains in the context of MT, and briefly present the different types of natural language data used in this work, followed by a short discussion of general domain adaptation methods in MT. Hereafter follows an in-depth discussion of optimization for SMT, concentrating on applications of discriminative training techniques. We describe other state-of-the art algorithms related to structured prediction and direct error minimization. The algorithms are presented in the context of ranking. Lastly, preference learning as used in this work is first described from the perspective of the field of information retrieval (IR), concentrating on the problem of learning rankings in the learning-to-rank framework.

After a thorough examination of optimization in SMT, we present our pairwise ranking approach in Chapter 3. The presentation consists of in-depth analyses of all aspects of the algorithm, as well as thorough empirical evaluation. Algorithmic and implementational aspects such as parallelization, asynchronicity, regularization and feature selection are discussed, as well as variants of the underlying translation metric which is used for ranking. We also provide benchmark results on various data sets of the proposed algorithms. This part covers problem statement a).

Our first work in CAT is presented in Chapter 4, where we propose a structured perceptron for reranking k-best output of an SMT system, using a feature representation of the actual reference or post-edit as target without resorting to so-called "oracle" translations[10] defined by a surrogate metric [Och and Ney, 2002]. The approach is described in detail, and simulated PE experiments using either reference translations or post-edited data are presented. We also describe the online learning protocol and simulated PE, which our approach is an instance of, and present related works in general reranking for SMT, adaptation methods in CAT and simulated PE. This work covers problem statement b).

In the last chapter discussing original works (Chapter 5), covering problem statement c), we present our approach for learning preferences from explicit user feedback, which is collected from a human-machine interface for PE. We describe the interface, our adaptation approaches and practical issues. In a second line of work, we show how to adapt a neural statistical machine translation system with post-edited translations. We further describe the two user studies we conducted, and appropriate human-targeted evaluation metrics. Our work is put into perspective by contrasting it to a large body of related works. The outcome of the user

[10] Oracles in this context are translations contained in the search space which represent the best output a system can possibly produce, given a reference translation.

studies are analyzed with a linear mixed effects model (LMEM), which are also defined in this chapter along with an overview of evaluation in CAT.

Finally, we briefly summarize our work in Chapter 6, and give conclusions and ideas for future directions of work.

1.3 Research Contributions

Our contributions to research in preference learning for machine translation can be summarized as follows:

- Review of optimization techniques in SMT with a focus on scaling of both training data and the feature space, as well as a discussion in the context of ranking algorithms [Chapters 2 & 3];
- We present efficient algorithms for (parallelized) online, gradient-based pairwise ranking optimization for SMT [Chapter 3];
- Thorough analysis of linear models for pairwise ranking in application to SMT with extensive experimental evaluation of pairwise ranking varying data and feature sets [Chapter 3];
- An application of multi-task learning via regularization for large-scale SMT tuning, which allows training on the full bitext as well as exploitation of commonalities within the data [Chapter 3];
- Development of a novel k-best reranking method for SMT with a structured prediction objective [Chapter 4];
- Evaluation of the aforementioned reranking technique in a simulated PE setting [Chapter 4];
- Adaptation framework for learning from explicit user feedback including manually generated word-alignments, and development of a suitable user interface [Chapter 5];
- A user study contrasting adaptive to non-adaptive statistical machine translation systems, including other state-of-the art adaptation methods [Chapter 5];
- A user study contrasting adaptive to non-adaptive neural machine translation systems [Chapter 5].

1.3.1 Previous Publications

Parts of the research presented here has been previously published and was peer-reviewed.

Ground-laying work for the pairwise ranking optimization as described in Chapter 3 has already been carried out during a Master's thesis [Simianer, 2012], of which some of the core findings are published in [Simianer et al., 2012]. Our first works on multi-task learning for parameter optimization in SMT are described in [Simianer et al., 2011] as an extension to the classical Minimum Error Rate Training (MERT) algorithm. Further work in this direction [Simianer and Riezler, 2013], as described in Section 3.12.2, uses a similar setup (patent data divided in parts by manual classification) but utilizes a different algorithmic approach and larger-scale data. Simianer and Riezler [2013] also describe vital extensions to the original algorithms in [Simianer et al., 2012]. The usefulness of the pairwise ranking approach, the proposed algorithmic extensions, multi-task learning, and parameter optimization using the full bitext were thoroughly evaluated in [Simianer et al., 2013*a*] (transcribed speech data, two language pairs and three translation directions, cf. Section 3.12.3), [Simianer et al., 2013*b*] (patent data, translating Japanese and Chinese into English, cf. Section 3.12.2.3), and finally in [Jehl et al., 2015] (transcribed speech data, English-to-German, cf. Section 3.12.3).

The reranking approach to learning preferences from post-edits in Chapter 4, was first described in [Wäschle et al., 2013]. A more detailed description of the approach as well as further evaluation can also be found in [Bertoldi et al., 2014].

A short description of our proposed methods for learning directly from user input in statistical machine translation (Chapter 5) is published as [Simianer et al., 2016]. The publication also comprises a description of the conducted user study and the main findings. A further, different analysis of the user study on neural machine translation is published in [Karimova et al., 2017].

2 Background

> "The only things in my life that compatibly exist with this grand universe are the creative works of the human spirit."
>
> [Ansel Adams, An Autobiography, 1985]

Automatic translation has a distinguished place in computer science (or more specific NLP): Translation is a difficult[1] task even for humans which requires long training, and translations of natural language are in any case meant for direct human consumption — it is therefore a daunting task to be carried out by machines.

In board games, classical examples for the application of artificial intelligence (AI), the number of possible board configurations (the *search space*) is typically so high that it is infeasible to explore it completely. There are for example about 10^{45} different legal board configurations in the game of chess[2]. Natural language and translation in particular however have even worse characteristics: Imagine a 20-word utterance in a source language, which is to be translated into another language that we will call the target language. Now, disregarding context, and assuming a non-realistic 1-1 correspondence in words for source and target languages, a conservative estimate for possible translation candidates per word is say 10.[3] We then arrive at 20^{10} possible translations for our original source language sentence, when assuming that the word order between two languages is monotone (identical), and each word occurs exactly once. The former is however infrequently the case, but even more so when languages are more "distant" from each other, e.g. even differing in the fundamental subject-object-verb placements[4]. Assuming completely free word order, there are $(20^{10})!$ possible translations, which are absolutely not manageable in any way. To make matters worse, it is imperative getting word order right — see for example the simplistic sentence depicted in Figure 2.1, where the semantics can be varied in dramatically different ways, inserting the adjective only at any position within the sentence.

An adequate translation system should be able to correctly transfer the meaning of the sentence to a valid target language utterance. Coming up with an algorithm that handles all these cases is bound to be difficult.

[1] Depending however on the subject matter.

[2] Cf. `https://tromp.github.io/chess/chess.html`.

[3] This number will be even higher if translation candidates are automatically determined through co-occurrence in a statistical dictionary or phrase-table.

[4] Subject (S), object (O), verb (V): SOV (e.g. Latin) — SVO (e.g. English) — VSO (e.g. Arabic) — VOS (e.g. Malagasy) — OVS (e.g. Urarina) — OSV (Yoda from the *Star Wars* movie series).

She told him that she loved him.

Figure 2.1: Inserting the word "only" in any of the eight possible positions within the sentence, alters the sentence's semantics, depending on the position and accordingly on the part of speech the word takes.

The wide range of problems in automatic translation has led to the development of manifold approaches to MT, consolidating different lines of research for the sublime goal of adequate and fluent automatically generated translation: Linguistic theories such as syntax and semantics, which are trying to find formalizations of language and translation of languages, have been applied for developing and analysing both rule-based and statistical systems; Developments from graph theory are used to encode and efficiently explore the vast number of possible translations for each given source utterance; Probability theory and statistics are used to mathematically grasp the task of translation; ML and optimization are employed as a means for learning efficient models of translation; And lastly, neural networks provide an alternative way of learning the underlying statistical models.

Aside from the technical formulation of the task of translation, there is a human factor which adds further layers of complexity: Since the fragile rule-based or statistical models underlying machine translation need lots of examples or training data for developing or learning appropriate models, it is almost inevitable that many details and nuances that natural language exhibits are bound to be lost when trying to build a general model for each and every application.

Due to these complex issues, up to date (2017), machine translation still suffers from severe limitations preventing its use for many practical automated applications. There is however significant adoption of machine translation in the field of CAT, interfacing machine translation and human translators.

In this chapter we first give a short history of MT as well as CAT. We then present the different approaches to machine translation that are used in the remainder of this work. We furthermore present methods for evaluating machine translation in the context of reference translations, as well as the data that we use to build the MT systems used in our work. Lastly, we present a range of optimization methods in SMT which are relevant to this work, and describe the fundamentals used in the following chapter.

2.1 A Very Short History of Machine Translation & Computer-Aided Translation

After the Second World War, MT was one of the first applications for the recently invented computers[5]. From that time, one of the most cited sayings by Warren Weaver was conveyed:

> "One naturally wonders if the problem of translation could conceivably be treated as a problem in cryptography. When I look at an article in Russian, I say: 'This is really written in English, but it has been coded in some strange symbols. I will now proceed to decode."
>
> [Weaver, 1949]

Laying the ground work for the information theoretic treatment of the translation problem — considering MT as a cryptographic problem, translating an encoded sequence of strings to a plain text one through decoding.

However, soon after the first conference on machine translation had been held [Hutchins, 1997; Reifler, 1954], the idea of computer-aided translation in the form of PE was formulated by Yoshua Bar-Hillel, proving some doubt in the promise of fully automatic machine translation (FAMT).

Contrariwise, because of the Cold War, the interest in the Russian-English language pair was unbroken in the US, and some successes could be acclaimed, for example adequate translations of specific scientific Russian texts into English [Dostert, 1955].

This lasted until the devastating ALPAC report that was published in the mid 1960s [Pierce and Carroll, 1966], which doubted that FAHQMT could ever be achieved. The report however, explicitly encouraged research in CAT for improving the efficiency of the human translation task.

After this report research in the US slowed down significantly due to reduced funding. Yet, research in other parts of the world continued unaffected, which for example resulted in the interlingua approach to machine translation [Vauquois, 1975][6], completing the famous *Vauquois triangle* [Vauquois, 1968].

During the 1970's and 1980's rule-based MT matured to actual commercial products, for example the *Systran* rule-based system [Ryan, 1989]. Pre- or post-editing was an integral part of most systems [Wagner, 1985].

The concentration on rule- or interlingua-based systems in MT research was eventually broken in the early 1990's, where the idea of automatically exploiting parallel sentence-aligned data with statistical models was first published in a series of papers now referred to as the *IBM models* [Brown et al., 1993]. The following research and development activity lead to the immensely practical statistical phrase-based models [Koehn, 2004*a*; Chiang, 2007], which also were able to produce good

[5] The idea of MT was conceived earlier, see e.g. Hutchins [2007] for a more thorough disquisition of MT history.

[6] This is just an exemplary publication for the field.

translation results with minimal human intervention. Naturally, since machine translations were becoming better and better, this lead to an increased interest in applications of CAT, not later than empirical studies showed potential in real-world tasks.

Due to advancements in hardware technology, neural networks finally became feasible in the late 2000's and, among other things, proved to be the new state-of-the-art in a large number of ML tasks, including various NLP applications. MT was revolutionized shortly after, due to certain advancements in sequence to sequence algorithms for language modeling, exhibiting much enhanced performance and above all fluency first in hybrid [Devlin et al., 2014] then in fully NMT approaches [Bahdanau et al., 2014] (inter-alia). Due to the performance and the simplicity of these systems as well as their natural fit to certain setups, neural (statistical) machine translation has sparked further development in CAT.

To summarize: The idea of fully automatic, and high quality machine translation has had ups and downs throughout the short history of automatic translation, but the idea of CAT has always been on people's minds, as it is a natural application of MT for practical purposes.

2.2 Statistical Machine Translation

Methods for MT have been iterated a number of times throughout its history, from simple dictionary-based approaches for direct translation in between languages, over complex methods involving a specifically designed *interlingua* which is supposed to capture generic semantic concepts, and variations thereof [Vauquois, 1975].

The most successful approach however, providing generally available, useful automatic translation, is the statistical approach. As in other fields in NLP, statistics cope quite naturally with the ambiguity found in natural language. But since it is such a difficult task, a variety of statistical models of translation have been developed.

We first review the general notion of providing a solution to natural language translation in terms of probability theory and statistics, then briefly discuss traditional word-based models which serve as the starting point for the more useful phrase-based models. Finally we introduce neural (statistical) MT.

2.2.1 Statistical Formulation of Translation

In the following we will describe the principle ideas of conducting translation in a statistical framework, starting from the classical noisy channel [Brown et al., 1988, 1990, 1993], converging to the modern formulation as structured prediction using a maximum entropy formulation [Och and Ney, 2002].

In statistical machine translation, ideally, we would have a good estimate for any conceivable pair of sentences[7] e and f[8] of how likely it is that they are translations of each other. Here, e and f are represented by two random variables E and F, and are associated with a joint probability distribution P

$$P(E, F). \tag{2.1}$$

For now we will ignore that this is an obviously overly confident proposition, and explain how one may arrive at a viable decomposition of that probability.

The general goal in generative modeling is to learn the joint probability distribution P, e.g. in classification over objects $X \in \mathcal{X}$ and labels $Y \in \mathcal{Y}$ represented by random variables X and Y respectively:

$$P(X, Y). \tag{2.2}$$

This distribution should satisfy the following constraints:

$$\sum_{(x,y)\in\{X\times Y\}} P(X = x, Y = y) = 1, \tag{2.3}$$

$$\forall(x, y) \in \{X \times Y\}\ P(X = x, Y = y) \in [0, 1], \tag{2.4}$$

where $\{\mathcal{X} \times \mathcal{Y}\}$ is the set of all possible pairs of assignments for X and Y[9]. In our case $X = E$ and $Y = F$.

Without a concrete idea of how to estimate $P(E, F)$, we could apply the chain rule to decompose P into possibly feasible parts:

$$P(E, F) = P(E|F)P(F). \tag{2.5}$$

There is no particular interest in estimating $P(F)$, since we want to be able to translate any utterance in the foreign language. Therefore we focus on the more interesting conditional probability $P(E|F)$.

Going further, using Bayes' rule, we arrive at a feasible decomposition of $P(E|F)$:

$$P(E|F) = \frac{P(F|E)P(E)}{P(F)}. \tag{2.6}$$

[7] Implicitly, we defined the principal but intuitive simplification of statistical machine translation: translational equivalence based on sentences.

[8] Historically e as in *English* is the target sentence, and f as in *French* is the source sentence.

[9] Note that in natural natural language, i.e. $X = E$ and $Y = F$, these variable are most certainly not independent, thus $P(E, F) \neq P(E)P(F)$.

For translation, we are still not interested in $P(F)$, and since there is no dependence on E, the denominator can simply be dropped[10]. This leads us to the *noisy channel*[11] formulation:

$$P(E|F) \propto P(F|E)P(E). \tag{2.7}$$

This reformulated generative model frees us from the burden of having to estimate a single single model $P(E|F)$, but instead we have two models that we can build independently. Brown et al. [1990] refer to $P(F|E)$ as the *translation model*, and to $P(E)$ as the *language model*. Actual implementations of these two models will be presented later in this section.

Given a concrete model of the noisy channel $P(F|E)$, as well as a model of general strings $P(E)$, we are interested in finding, for any f, the most probable translation $\hat{e}$:

$$\begin{aligned} \hat{e} &= \arg\max_e P(E = e|F = f) \\ &= \arg\max_e P(F = f|E = e)P(E = e). \end{aligned} \tag{2.8}$$

This search is called decoding[12]. In the noisy channel model, the arg max operation can be understood as the attempt of recovering the original message. For $P(F = f|E = e)$ we will write $p(f|e)$ from now on, as well as for all other probabilities.

The translation model $p(f|e)$ is however still underspecified, as there is no direct connection between the source f and its translation e. The model described in [Brown et al., 1993] introduces the notion of an implicit, *hidden alignment* between e and f. The translation model becomes:

$$P(F|E) = \sum_{a \in \mathcal{A}} P(F, A = a|E), \tag{2.9}$$

[10] This is due to the fact that in actual translation, performing an arg max operation over $P(E|F)$, $P(F)$ is constant and can thus be dropped.

[11] The noisy channel origins from the work on information theory of Shannon [1948] — a message E is sent through a channel in which it is distorted, resulting in the corrupted message F. In this view, $P(F|E)$ is the noisy channel, and $P(E)$ is a prior on the "original" message. Church and Gale [1991], Mays et al. [1991] and Kernighan et al. [1990] first introduced this probabilistic view into an NLP task, namely for spelling correction. Before that, the formulation was already used in speech recognition [Bahl et al., 1983; Brown et al., 1994].

[12] This term stems from the famous quotation of Warren Weaver, embracing natural language translation as a cryptographic task: "I will now proceed to decode."

and thus:

$$P(E|F) = \sum_{a \in \mathcal{A}} P(F, A = a|E)P(E), \tag{2.10}$$

where $\mathcal{A}$ is the set of possible alignments between E and F.

The consequence of introducing an alignment between E and F is, that it allows to break up the definition of translation probability to smaller items, e.g. single words. With this, it is possible to define more useful models, as we will discuss in the following section.

The noisy channel formulation for translation is still limited, as noted in [Och and Ney, 2002]:

a) It is not possible to incorporate further models (possibly defined on a different data set) in a generalized way, so $p(e)$ and $p(f|e)$ have to be excellent, i.e. close estimates of the true distributions, which is unlikely in practice;

b) $p(e)$ and $p(f|e)$ are independent by definition, which also applies to their application in search — a weighted approach, balancing the models based on their performance would be more appropriate.

A different approach, which can be naturally developed from the noisy channel approach, is to directly model the conditional distribution $P(E|F)$ with a *discriminative* model. Och and Ney [2002] introduce this idea for MT and propose the use of a *log-linear*[13] model for this purpose.

This model applied to the formulation above results in the following definition of the joint distribution $P(F, E)$:

$$P(F = f, E = e) = p(f, e) = \frac{\exp\left(\log p(f|e) + \log p(e)\right)}{\sum_{f'} \sum_{e'} \exp\left(\log p(f|e) + \log p(e)\right)}. \tag{2.11}$$

However, in the log-linear framework, we can directly derive the conditional distribution $P(E|F)$:

$$P(E = e|F = f) = p(e|f) = \frac{\exp\left(\log p(f|e) + \log p(e)\right)}{\sum_{e'} \exp\left(\log p(f|e) + \log p(e)\right)}, \tag{2.12}$$

that is, for any given f, we simply seek a distribution over possible values e, using the previously defined $p(f|e)$ and $p(e)$ as *features*. Due to the normalization,

[13] Or: Maximum entropy model.

we can also add weights w_1, w_2 to the features, possibly balancing their relative quality:

$$P(E = e|F = f) = p(e|f) = \frac{\exp\left(w_1 \log p(f|e) + w_2 \log p(e)\right)}{\sum_{e'} \exp\left(w_1 \log p(f|e) + w_2 \log p(e)\right)}. \tag{2.13}$$

This approach can be generalized to use arbitrary *feature functions* $h_i(f, e)$, with associated weights $w_i = \mathbf{w}$ for $i = 1 \ldots M$, and using $\log p(f|e)$ and $\log p(e)$ as *feature functions*:

$$p(e|f) = \frac{\exp \sum_{i=1}^{M} w_i h_i(e, f)}{\sum_{e'} \exp \sum_{i=1}^{M} w_i h_i(e, f)}. \tag{2.14}$$

Conveniently this equation can be formulated compactly by using a dot product and by introducing a *feature mapping* function $\phi(f, e)$ that takes a source/translation pair (f, e) and maps it to a vector[14] of features $[w_i, \ldots, w_M]^T = \mathbf{w}$:

$$p_{\mathbf{w}}(e|f) = \frac{\exp\langle \mathbf{w}, \phi(e, f)\rangle}{\sum_{e'} \exp\langle \mathbf{w}, \phi(e, f)\rangle}. \tag{2.15}$$

The *decision rule* for finding the optimal $\hat{e}$ for a given f is just the denominator:

$$\begin{aligned} \hat{e} &= \arg\max_{e} p_{\mathbf{w}}(e|f) \\ &= \exp \sum_{i=1}^{M} w_i h_i(e, f) \\ &= \sum_{i=1}^{M} w_i h_i(e, f) \\ &= \mathbf{w}\phi(e, f). \end{aligned} \tag{2.16}$$

Note that, most commonly in MT, the feature map $\phi(\cdot)$ is defined over a triple (f, e, a) (and thus $h(f, e, a)$), including the alignment between e and f. This allows to define features of the *derivation* of a translation. This is due to the *maximum approximation* used in the arg max computation, which is used to ensure feasibility of the search problem.

2.2.2 Word-Based Models and Statistical Word Alignment

The so called *IBM models* introduced by Brown et al. [1993], provide a number of decompositions of the probability distribution $P(E|F)$ (as outlined before) and appropriate procedures for learning the distribution from data.

The central idea of these models is breaking up E and F into sequences of words, and defining an alignment between them:

[14] In some contexts we thus write $\boldsymbol{\phi}$ instead of ϕ to make this clear.

- $\mathbf{e} = [0, e_1, \ldots, e_l]$, a vector of l random variables representing a target sentence, as well as a fixed special *Null* token at 0,
- $\mathbf{f} = [f_1, \ldots, f_m]$, a vector of m random variables representing a source sentence,
- and an alignment $\mathbf{a} = [a_1, \ldots, a_m]$, defined from $\mathbf{f}$ to $\mathbf{e}$.

With this we can write the translation model as:

$$p(f_1, \ldots, f_m, a_1, \ldots, a_m | e_1, \ldots, e_l, m) = p(\mathbf{f}, \mathbf{a} | \mathbf{e}, m), \tag{2.17}$$

and

$$p(\mathbf{f}|\mathbf{e}) = \sum_{\mathbf{a} \in \mathcal{A}} p(\mathbf{f}, \mathbf{a}|\mathbf{e}), \tag{2.18}$$

where $\mathcal{A}$ is the set of all possible $(l+1)^m$ alignment vectors.

The most simple models, *IBM model 1* and *2*, which we describe here, combine a lexical distribution $t(f|e)$ and a distribution over alignments $u(j|i, l, m)$, $(l, m) \in \mathrm{N}_+$, $i \in \{1, \ldots, m\}$, and $j \in \{0, \ldots, l\}$:

$$\begin{aligned} p(\mathbf{f}, \mathbf{a}|\mathbf{e}, m) &= p(\mathbf{a}|\mathbf{e}, m) p(\mathbf{f}|\mathbf{a}, \mathbf{e}, m) \\ &\hat{=} \prod_{i=1}^{m} u(a_i|i, l, m) t(f_i|e_{a_i}). \end{aligned} \tag{2.19}$$

Note that the second equation introduces two important independence assumptions: 1) the alignment only depends on the lengths of the French and English sentences, not on the words, and 2) that the lexical distribution does not depend on the alignment.

In IBM model 1 ($M1$) the distribution over alignments is uniform, thus:

$$p_{\mathrm{M1}}(\mathbf{f}, \mathbf{a}|\mathbf{e}) = \frac{p(m|\mathbf{e})}{(l+1)^m} \prod_{j=1}^{m} t(f_j|e_{a_j}), \tag{2.20}$$

assigning the same probability to all alignments.

Since manually word-aligned data is infeasible to produce, the IBM models cannot be simply estimated by using direct maximum likelihood estimation (MLE), i.e.:

$$t_{\mathrm{ML}}(f|e) = \frac{c(f, e)}{\sum_{f'} c(f', e)}, \tag{2.21}$$

where $c(f, e)$ counts alignments links between f and e in a word-aligned corpus. Accordingly for the alignment:

$$u_{\mathrm{ML}}(j|i, l, m) = \frac{c(j, i, l, m)}{\sum_j c(j, i, l, m)}, \tag{2.22}$$

where $c(j, i, l, m)$ counts all occurrences of parallel sentences of lengths l and m respectively, where the French word i is aligned to an English word j.

Therefore the IBM models are estimated using a variant of the expectation maximization algorithm [Dempster et al., 1977].

Decoding, i.e. $\arg\max_{\mathbf{e}} p(\mathbf{f}|\mathbf{e})p(\mathbf{e})$, with the IBM models is NP-complete [Knight, 1999], hence approximations are needed [Wang and Waibel, 1997; Riedel and Clarke, 2009; Germann et al., 2001].

A by product of a fully trained word-alignment is the so called *Viterbi alignment* $\hat{\mathbf{a}}$, defined as [Brown et al., 1993]:

$$\hat{\mathbf{a}} = \arg\max_{\mathbf{a}\in\mathcal{A}} p(\mathbf{f}, \mathbf{a}|\mathbf{e}), \tag{2.23}$$

which can be generated by selecting:

$$\hat{\mathbf{a}}_i = \arg\max_i u(a_i|i, l, m)t(f_i|e_{a_i}) \tag{2.24}$$

for each position i in the French sentence. The result (an unique alignment for a given sentence pair) can be visualized in an *alignment matrix*, see Figure 2.2 for an example.

The word-based models are not very useful for actual translation due to the harsh independence assumptions and the comparatively complex word-based decoding. They are however, as we will see in the phrase-based models, a good starting point for more advanced approaches. For this purpose, *alignment symmetrization* is an important concept, as first proposed by Och and Ney [2003]: Since the individual word-alignment models only produce many-to-one alignments[15] and not many-to-many alignments, which would be more appropriate for natural language. A simple symmetrization approach is to train two models, $p(f|e)$ and $p(e|f)$, generating Viterbi alignments from both, and joining the alignments using set operations like union, intersection or derivations thereof[16]. The resulting alignment is a proper

[15] In the formulation above, each French word f aligns to exactly one English word e or the Null token.

[16] A popular symmetrization method, referred to as `grow-diag-final-and` [Koehn et al., 2003], starts with the intersection and subsequently adds alignment links from the intersection.

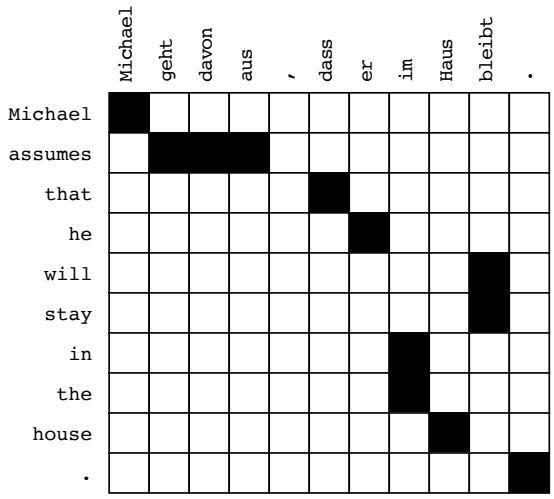

Figure 2.2: Example of a word alignment matrix: The source and target word positions correspond to rows and columns respectively. Alignment links are visualized by cells with a black background at the according positions.

many-to-many alignment.

A popular choice for a not oversimplified word alignment model is IBM model 2, which is why there are modern implementations and extensions available, see e.g. [Dyer et al., 2013] or [Gao and Vogel, 2008].

2.2.3 Phrase-Based Model

Phrase-based machine translation (PBMT) models [Koehn et al., 2003] partially overcome the strong independence assumptions which are imposed to make word-based models efficient. By using phrases instead of words as atomic translation units, local contexts can be efficiently encoded.

Starting point for the phrase-based approach are symmetrized many-to-many word alignments generated by running a word alignment algorithm in both translation directions. From this joint alignment, *phrase-pairs* (aligned sub-sequences of source and target words) can be extracted. A standard requirement for phrase-pairs is consistency with the word alignment, as defined in Och et al. [1999]: A phrase-pair is consistent iff all source words of a phrase are only aligned to words of the target phrase (including Null) and vice-versa.

Estimation of the translation model is trivial for this model, since an explicit alignment is given. It can simply be estimated through MLE:

$$p(\tilde{f}|\tilde{e}) = \frac{c(\tilde{f}, \tilde{e})}{\sum_{\tilde{f}'} c(\tilde{f}, \tilde{e})}, \tag{2.25}$$

slightly abusing notation, $\tilde{\cdot}$ being phrases. After extracting phrases over a large sentence-aligned corpus (bitext), the result of this process is a so-called *phrase-table*, which contains all phrase-pairs that could be extracted and stores them in an efficient manner.

Decoding in this model breaks down into two phases: First, the source sentence to be translated is segmented according to the source-sides of phrases in the phrase table. All segmentations are considered equally likely at first. In a second step, each source phrase is translated into a target phrase according to the entries of a phrase-table, making sure that each source phrase is translated exactly once. Source phrases can be processed in any order, and the target translation hypothesis is built left to right. This implies that *reordering* of phrases is allowed on the target-side. Since most natural languages do not demand this level of flexibility, Koehn et al. [2003] propose to incorporate a *distortion model*:

$$d(a_i - b_{i-1}) = \alpha^{|a_i - b_{i-1} - 1|}, \tag{2.26}$$

where α is a free parameter, a_i is the start position of the source phrase yielding the ith target phrase, and b_i denotes the end position of the $i-1$th target phrase.

This allows penalizing large "jumps" in the source sentence. Furthermore, another parameter ω is added to the model, counting the number of produced words on the target side, the so-called word penalty.

The different models and sub-models in the original phrase-based approach of Koehn et al. [2003] are combined without explicit weighting during decoding (omitting source segmentation):

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e}} p(\mathbf{e}|\mathbf{f}) = p(\mathbf{e})\omega^{|\mathbf{e}|}\left[\prod_i p(\mathbf{f}_i|\mathbf{e}_i)d(a_i - b_{i-1})\right], \tag{2.27}$$

where $|\mathbf{e}|$ is the length of the current (partial) translation hypothesis, and $\mathbf{f}_i$ is the ith source-phrase with its corresponding target side $\mathbf{e}_i$. The log-linear model, as introduced by Och and Ney [2002] for phrase-based statistical machine translation, enables to use a (learned) weighted combination of the different models:

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e}} w_1 \log p(\mathbf{e}) + w_2 \log \omega^{|\mathbf{e}|} + \left[\sum_i w_3 \log p(\mathbf{f}_i|\mathbf{e}_i) + w_4 \log d(a_i - b_{i-1})\right]. \tag{2.28}$$

On the one hand this allows discriminatively learning task-specific weights [Och, 2003] as well as efficient weighted decoding, on the other hand, from a modeling perspective, it enables adding arbitrary sub-models as features to make the model more expressive. The decision function can, analoguosly to Equation 2.16, be written more compactly as a single dot product:

$$\hat{\mathbf{e}} = \arg\max_{\mathbf{e}} \langle \mathbf{w}, \phi(\mathbf{e}, \mathbf{f}) \rangle, \tag{2.29}$$

where ϕ is a function mapping the hypothesis to a joint feature space, just as previously described for the word-based models.

It is important to note that the combinatorial issues when decoding are similar to the ones with word-based models. This is why a variety of approximate search techniques have to be applied in order to efficiently find a good translation hypotheses $\hat{\mathbf{e}}$, such as beam search and hypotheses recombination for stack-based decoding [Och et al., 2001; Koehn, 2004*a*]. Most of the decoding algorithms for statistical machine translation are instances of dynamic programming, and as such, they produce a *search graph* in the form of a probabilistic finite state transducer (FST), which can be exploited using general graph algorithms, such as efficient generation of k-best hypotheses or sampling.

In a search graph, vertices represent a (partial) hypothesis and its internal state[17]. Edges are associated with phrase applications, covering parts of the source. Complete hypotheses form a complete path through the search graph, covering the whole source.

It is important to note, that the approximations for decoding in phrase-based statistical machine translation significantly divert from the originally proposed model: Since the sum in $\arg\max_{\mathbf{e}} \sum_{\mathbf{a}} p(\mathbf{f}, \mathbf{a}|\mathbf{e})$ is intractable, we instead seek $\arg\max_{\mathbf{e}} \max_{\mathbf{a}} p(\mathbf{f}, \mathbf{a}|\mathbf{e})$, maximizing over all possible segmentations and alignments. However, as Och and Ney [2002] note, this can be alleviated by including $\mathbf{a}$ in the joint feature map, defining feature functions using the alignment $\mathbf{a}$, i.e. binary word translation features or identifiers for phrase-pairs.

The most widely used implementation of phrase-based statistical machine translation is the *Moses* toolkit [Koehn et al., 2007], which implements all of the discussed algorithms.

2.2.4 Digression: Language Modeling

For the previous description of statistical machine translation we ignored an important part of the model: the language model $p(\mathbf{e})$. It plays a viable role in most SMT approaches by selecting fluent translations.

A general language model assigns probabilities over sequences of variable length $\mathbf{e} = [e_1, \ldots, e_k]$:

$$p(\mathbf{e}) = p(e_1, \ldots, e_k). \tag{2.30}$$

For short sequences the probability p can be broken up using the chain rule:

$$p(e_1, e_2, e_3, e_4) = p(e_1)p(e_2|e_1)p(e_3|e_1, e_2)p(e_4|e_1, e_2, e_3). \tag{2.31}$$

This is however not feasible for longer sequences. A feasible alternative are N-gram language models, which introduce independence assumptions, e.g. a unigram language model would break up the probability in a series of independent events:

$$p(e_1, e_2, e_3, e_4) = p(e_1)p(e_2)p(e_3)p(e_4). \tag{2.32}$$

In a bigram language model, in which sequences of conditional probabilities are multiplied (all conditioned on a single previous token, a *history*), short of the first item:

$$p(e_1, e_2, e_3, e_4) = p(e_1)p(e_2|e_1)p(e_3|e_2)p(e_4|e_3). \tag{2.33}$$

[17] In PBMT the state can be boiled down to a coverage vector, denoting the parts of the source that are covered by the current hypothesis.

More general, we can write an N-gram language model for sequences of arbitrary length J as:

$$p(\mathbf{e}) = \prod_{i=1}^{J} p(e_i|\mathbf{e}_1^{i-1}) \approx \prod_{i=1}^{J} p(e_i|\mathbf{e}_{i-N}^{i-1}). \tag{2.34}$$

This is an application of the *Markov Chain*, which can importantly also be encoded in a finite state machine.

N-gram language models can be efficiently estimated from monolingual data with maximum likelihood estimation:

$$p_{\mathsf{MLE}}(e_i|\mathbf{e}_{i-N}^{i-1}) = \frac{c(e_{i-N},\ldots,e_i)}{\sum_e c(e_{i-N},\ldots,e_{i-1},e)} = \frac{c(e_{i-N},\ldots,e_i)}{c(e_{i-N},\ldots,e_{i-1})}. \tag{2.35}$$

However, since higher order N-grams are sparse, smoothing and/or interpolation have to be applied [Chen and Goodman, 1996; Kneser and Ney, 1995], *inter-alia*.

A N-gram language model can be directly integrated into decoding for phrase-based statistical machine translation by keeping track of the language model score $w_{\mathrm{LM}} \log p(\mathbf{e})$ in each search state. But one has to adhere that enough history is stored from previous states to calculate the score for the full N also when crossing phrase boundaries. This enlarges the size of the search graph, since it impairs the independence between states.

2.2.5 Hierarchical Phrase-Based Model

The hierarchical phrase-based model as proposed by Chiang [2005, 2007], is a formalism to carry out machine translation with synchronous context free grammars (SCFG). They are a type of syntax-based translation model, since syntactic structures are built up during translation[18]. Instead of purely lexical phrase-pairs as atomic translation units, grammar-based models use synchronous grammars with non-terminal symbols integrated in source- and target-sides of their translation rules. A complete example of a non-weighted synchronous grammar is depicted in Figure 2.3.

This grammar accepts the well-formed German input strings ich sah ein kleines Haus, and ein kleines Haus sah ich (amongst others), while it can synchronously produce trees on the target side. A translation or *rewrite rule* of a synchronous context free grammar consists of a left-hand side, which is a single non-terminal symbol, and a right-hand side in the form of $\langle\gamma|\alpha\rangle$, where γ and α are composed of terminal

[18] It is however a string-to-string model [Williams et al., 2016], since syntactic structure is not necessarily exploited on either source or target side, and the tree structure is only used for structuring the search space.

$$\mathcal{R} = \left\{ \begin{array}{rl} \text{S} & \rightarrow \langle \text{NP VP} \mid \boxed{1}\ \boxed{2} \rangle \\ \text{NP} & \rightarrow \langle \mathsf{ich} \mid \mathsf{I} \rangle \\ \text{NP} & \rightarrow \langle \mathsf{ein}\ \text{NN} \mid \mathsf{a}\ \text{NN} \rangle \\ \text{NN} & \rightarrow \langle \text{JJ}\ \mathsf{Haus} \mid \boxed{1}\ \mathsf{house} \rangle \\ \text{NN} & \rightarrow \langle \text{JJ}\ \mathsf{Haus} \mid \boxed{1}\ \mathsf{shell} \rangle \\ \text{JJ} & \rightarrow \langle \mathsf{kleines} \mid \mathsf{small} \rangle \\ \text{JJ} & \rightarrow \langle \mathsf{kleines} \mid \mathsf{little} \rangle \\ \text{VP} & \rightarrow \langle \text{V NP} \mid \boxed{1}\ \boxed{2} \rangle \\ \text{V} & \rightarrow \langle \mathsf{sah} \mid \mathsf{saw} \rangle \end{array} \right\}$$

Figure 2.3: SCFG Example.

symbols and co-aligned[19] non-terminal symbols. The alignment allows swapping of sequences in source and target. The left part of the right-hand side of a rule (γ) applies to the source, and the right part (α) to the target. Translation is carried out by bottom-up parsing the source sentence, synchronously building a target tree[20]. The leaves of the target tree form the string representation of the translation.

Formally, the translation formalism can be described as an intersection between a weighted synchronous context free grammar (WSCFG) over a semiring K with a non-weighted directed acyclic graph (DAG) that encodes a single path[21] [Dyer, 2010*b*]. The WSCFG is defined as $G = \langle \Sigma, \Delta, \mathcal{V}, \mathcal{S}, \langle \mathcal{R}, \rho \rangle \rangle$, where Σ is a finite set of input terminal symbols, Δ is a finite set of output terminal symbols, $\mathcal{V}$ is a finite set of non-terminal symbols, $\mathcal{S}$ is a set of start symbols, and $\mathcal{R}$ is a set of synchronous rewrite rules as described above, with an associated weight function $\rho : \mathcal{R} \rightarrow K$, $\rho\,(\text{X} \rightarrow \langle \cdot, \cdot \rangle) = \overline{1} \in K$ if $\text{X} \in \mathcal{S}$ ($\overline{1}$ being the unit of semiring K).

For the example grammar given in Figure 2.3, we have $\Sigma = \{\mathsf{ich}, \mathsf{sah}, \mathsf{ein}, \mathsf{kleines}, \mathsf{haus}\}$, $\Delta = \{\mathsf{i}, \mathsf{saw}, \mathsf{a}, \mathsf{small}, \mathsf{little}, \mathsf{house}, \mathsf{shell}\}$, $\mathcal{V} = \{\text{S}, \text{NP}, \text{VP}, \text{V}, \text{JJ}, \text{NN}\}$, $\mathcal{R}$ defined as above, and $\mathcal{S} = \{\text{S}\}$

The bottom-up parser can be described as a weighted logic program [Lopez, 2009; Dyer, 2010*b*] intersecting the WSCFG with an input string, similar to an *Earley* chart parsing algorithm [Earley, 1970]. The input string is, as already mentioned, encoded as a DAG [Dyer et al., 2008], for our example as depicted in Figure 2.4. The length of the input is l, and a *span* over the string is denoted as $\langle i, x, j \rangle$, where $i \leq j$, and $x \in \Sigma^*$. Finally, $\mathcal{Q} \subset \mathbb{N}_+ \cup 0$ is the set of nodes of the input DAG. The binary operation $\otimes$ is the multiplication of the used semiring K. With this, the parsing can be fully described by axioms, goals and inference rules:

[19] The so-called gap constraint, which allows synchronous parsing due to a one-to-one alignment [Dyer, 2010*b*].

[20] In most practical applications, this will be a *forest* due to the ambiguity in the grammar.

[21] Dyer [2010*b*] describes the intersection with an arbitrary weighted finite state transducer, we however focus on simple strings as input.

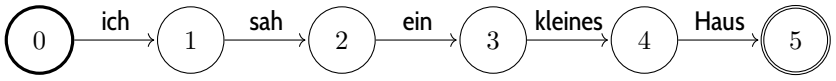

Figure 2.4: Input string as DAG: States are denoted by a number, and edges by a string. The initial state has a bold border, and the accepting state a double border.

Axioms:

$$\frac{}{[\mathrm{X} \to \langle \bullet \beta \mid \zeta \rangle, q, q] : w}, \forall q \in Q \setminus 0, \forall \mathrm{X} \xrightarrow{w} \langle \beta | \zeta \rangle$$

Goals:

$[\mathrm{X} \to \langle \alpha \bullet \mid \zeta \rangle, 0, l]$, $\mathrm{X} \in \mathcal{S} \wedge \langle 0, \alpha, l \rangle$

Inference rules:

$$\text{Shift: } \frac{[\mathrm{X} \to \langle \alpha \bullet x\beta \mid \zeta \rangle, r, s] : w}{[\mathrm{X} \to \langle \alpha x \bullet \beta \mid \zeta \rangle, r, t] : w}, \{r, s, t\} \subset \mathcal{Q}$$

$$\text{Reduce: } \frac{[\mathrm{X} \to \langle \alpha \bullet \mathrm{Y}\beta \mid \zeta \rangle, r, s] : u \quad [\mathrm{Y} \to \langle \gamma \bullet \mid \xi \rangle, s, t] : v}{[\mathrm{X} \to \langle \alpha \mathrm{Y}_{s,t} \bullet \beta \mid \zeta \rangle, r, t] : u \otimes v}.$$

The time complexity of this algorithm is approximately $\mathcal{O}(l^{\mathrm{scope}(R)})$, l being the length of the input, and scope(R) the maximum number of consecutive non-terminals in the set of rewrite rules $\mathcal{R}$ [Hopkins and Langmead, 2010]. In our running example that amounts to $\mathcal{O}(5^2)$.

Alternatively, variants of the CYK chart parsing algorithm [Sakai, 1962] can be employed Chappelier and Rajman [1998]; Sennrich [2014] for parsing with non-binarized grammars.

The resulting chart is equivalent to a directed hypergraph [Klein and Manning, 2004] (or also a *context-free forest*), which renders the approach similar to the finite-state phrase-based models [Koehn et al., 2003]. Furthermore, using different semirings K and shortest path algorithms such as Viterbi- [Viterbi, 1967] or Dijkstra's algorithm [Skiena, 1990], the framework allows to efficiently find the derivation with maximum probability (Viterbi semiring), generalized shortest path (real semiring), or total number of paths (counting semiring) [Goodman, 1999; Huang, 2008]. The algorithms can also be readily extended to return k-best outputs, see e.g. [Huang and Chiang, 2005]. The only condition is that throughout the hypergraph, the parametrization conforms with the optimal substructure property

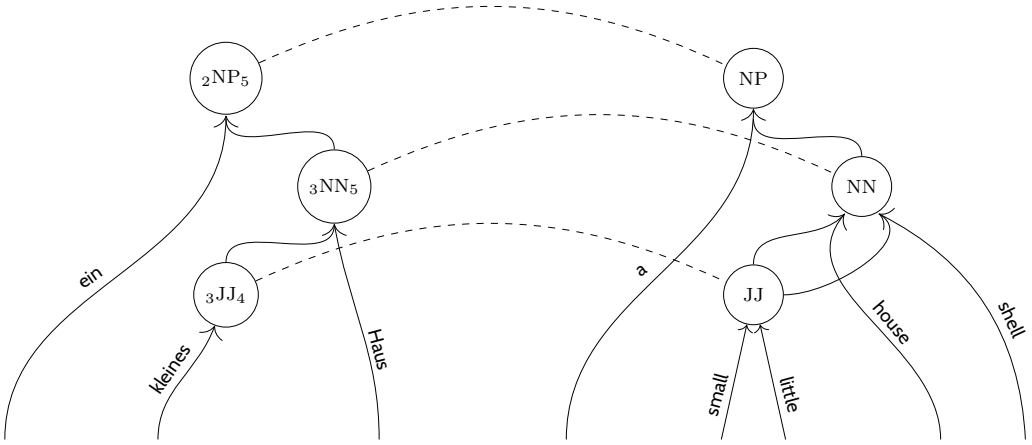

Figure 2.5: Example output for a synchronous parse in a hypergraph representation: The source parse is on the left-hand side, and the target on the right-hand side. Nodes on the source side are annotated with their respective source spans. Source-to-target alignment is shown as dashed gray lines. Annotated edges represent terminal nodes.

of dynamic programming, which also demands linearity in the edge-local features and the scoring function.

The hypergraph is a compact representation of all possible translations of a given source string, as exemplified in Figure 2.5 for the ambiguous part of the running example. However, since SMT utilizes a language model for satisfactory performance[22], another method is needed to integrate language model scores. Chiang [2007] describes an algorithm for a fully weighted intersection[23] and an approximate version including pruning of a language model.

Since the full intersection is often infeasible in practical applications [Dyer et al., 2010], we always resort to the approximate variant. The key idea to these algorithms is to include the relevant target information in the parsing items of the chart, i.e. for a N-gram language model with $N = 3$:

$$[\mathrm{X} \rightarrow \langle \alpha \mid \zeta^{d \star e} \rangle, s, t],$$

[22] See for example [Blunsom and Osborne, 2008].

[23] Dyer [2010*a*] describes the integration of a language model into a synchronous parsing algorithm by representing the language model as a WFST, and defining an intersection between the WSCFG and the resulting WFST.

where d and e are the first and last tokens of the current target-side production. In the approximate algorithm, *cube pruning*, items are reduced selectively, based on their combined score (including language model and the translation rule's features). For maximum flexibility, the integration of the language model or other features can be carried out in a separate step after the initial production of the possible translations under the given grammar. This procedure is referred to as forest rescoring [Huang and Chiang, 2007] and implemented in the toolkit that we use in this thesis [Dyer et al., 2010] (*cdec*).

Although this formalism can be used for all synchronous context-free grammars, in the hierarchical phrase-based approach of Chiang [2005] ("Hiero") the set of non-terminals is limited to a single symbol X, and for efficiency several restrictions on the shape of the grammar rules are imposed [Chiang, 2007]. Additionally, the grammar is extracted in a non-linguistic manner, resembling the method described for the previous phrase-based approach: After extracting all phrase pairs that are consistent with the word alignment, phrases with "gaps" are produced by replacing existing phrases within other phrases with the non-terminal symbol X.

This general approach can be readily used to cope with fully syntactic grammars [Zollmann and Venugopal, 2006], or, exploiting the hierarchical tree structures, used to incorporate (soft) syntactical features [Blunsom and Osborne, 2008; Vilar et al., 2008; Marton and Resnik, 2008], assessing the syntactical well-formedness of the tree derivations of translation hypotheses.

The great promise of parsing-based approaches to machine translation is the greatly improved capability of coping with long-range, non-local reordering phenomena, as for example common in verbal constructions in German. In turn, they eliminate the need for reordering- or distortion models as used in the PBMT approaches. For a fine-grained discussion on reordering see [Bisazza and Federico, 2016]. There are however some inherent limitations to the Hiero approach, either because the restriction to binary grammars, which prevents modeling of some synchronous parse configurations, or, because of the inability to model certain transformations independently, see [Galley and Manning, 2010] for an overview.

2.2.6 Neural Machine Translation

Statistical machine translation with neural networks[24] is a fundamentally different approach compared to the previously presented approaches. With neural networks, it is possible to directly implement the fundamental equation of machine translation

[24] Neural machine translation (NMT).

without further approximations:

$$\hat{e} = \arg\max_{e} p(e|f), \tag{2.36}$$

(note that the vector notation is omitted here). The variant of neural translation model used in this work is closely related to a certain form of neural network language models based on recurrent neural networks (RNN) [Elman, 1990]. The key feature of an RNN is that it is capable of storing an unlimited[25] conditioning history encoded[26] in real vectors of fixed size, which predestines their use in language modeling. A simple RNN can be defined as follows (omitting the case for $t = 0$):

$$\begin{aligned} \mathbf{h}_t &= \sigma(f(\mathbf{x}_t, \mathbf{h}_{t-1})) \\ &= \sigma(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}) \end{aligned} \tag{2.37}$$

where $\mathbf{x}_t$ is the input at time step t, σ is a non-linear function, e.g. $\tanh(\cdot)$, $\mathbf{W}$ and $\mathbf{U}$ are parameter or weight matrices, and $\mathbf{b}$ is a *bias* vector. An RNN iterates over time steps t, updating its *hidden state* vector $\mathbf{h}$ every time. The input at each step is a symbol, i.e. in language modeling a single token. The input $\mathbf{x}_t$ is also a real vector, stored in a column of a so called *embedding matrix* $\mathbf{M} \in \mathrm{R}^{k \times |V|}$, where k is a free parameter and $|V|$ is the size of the used (fixed) vocabulary. The example is extracted by multiplication with a *one-hot* (column) vector $\mathbf{o}$ which is non-zero only at a single index i, $1 \leq i \leq |V|$:

$$\mathbf{x}_t = \mathbf{M}\mathbf{o}. \tag{2.38}$$

The prediction at time step t is carried out by linearly projecting the current hidden state to an intermediate vector with dimensionality $1 \times |V|$, and finally applying a *soft max* operation to generate a probability distribution over vocabulary entries:

$$\mathbf{q}_t = \text{soft}\max(\mathbf{V}\mathbf{h}_t + \mathbf{b}), \tag{2.39}$$

where

$$\text{soft}\max(\mathbf{v}) = \frac{\exp(v_j)}{\sum_j \exp(v_j)}. \tag{2.40}$$

The prediction can then simply be calculated via $\arg\max_j \mathbf{q}_{t,j}$, where $\mathbf{q}_{t,\cdot}$ can be interpreted as probability distribution. The probability of a sequence $[s_1, \ldots, s_i]$

[25] In practice there is a length limit imposed due to bounded memory.
[26] In practice there are problems with overly long sequences, partially caused by the training method.

is thus given by

$$p(s_i|s_1,\ldots,s_{i-1}) = \prod_k \mathbf{q}_{i,j}. \tag{2.41}$$

Neural networks are often trained by stochastic gradient descent (SGD) using the *back-propagation* algorithm [Rumelhart et al., 1988] for efficiently computing the gradients given a loss function. The process in an RNN is however more elaborate, since it involves repeated function applications instead of just a single one. Mozer [1989] presents a variant of the original back-propagation algorithm that handles this structure by unfolding the network (*back-propagation through time*).

The training signal is given by defining a suitable loss function, which in language modeling can simply be the negative log likelihood[27]:

$$l_\theta(\mathcal{X}) = -\frac{1}{|\mathcal{X}|}\sum_{x\in\mathcal{X}} \log p_\theta(x), \tag{2.42}$$

where θ is the set of all parameters of the model, and $\mathcal{X}$ is a set of training data. The loss function is minimized during training until a stopping criterion is met. A language model based on an RNN as defined here can also be found in [Kalchbrenner and Blunsom, 2013; Mikolov et al., 2010].

Similar models can be effectively integrated into PBMT, see e.g. [Devlin et al., 2014], but for using them as a stand alone translation more work has to be done.

Chrisman [1991], Kalchbrenner and Blunsom [2013], Cho et al. [2014] and Sutskever et al. [2014] proposed the *encoder-decoder* approach to model translation. An instance of this type of model uses recurrent neural networks, as outlined above, combing an encoder RNN, providing a fixed vector representation $\mathbf{c}$ of the source sentence, and a decoder RNN, conditioned on both its own history and $\mathbf{c}$ provided by the encoder. Encoder and decoder are both RNN language models as described before, but the decoder RNN receives the last encoder state as additional input in each time step:

$$\mathbf{h}_t = \sigma(f(\mathbf{y}_t, \mathbf{h}_{t-1}, \mathbf{c})), \tag{2.43}$$

the conditional distribution is changed accordingly.[28]

The full network is trained on parallel sentence pairs, the loss is however only defined on the target side, since the encoder RNN has no individual outputs, and only serves to provide context for the decoder RNN. Both encoder and decoder are trained jointly given the target-side loss.

[27] Which is closely related to the cross-entropy loss.

[28] In practice, the context vector is concatenated with the hidden state before applying the soft max operation.

For satisfying performance of the proposed models, several modifications are advisable: The simple RNN suffers from the *vanishing gradients* problem due to the deep structures in back-propagation through time. The long-short term memory RNN [Hochreiter and Schmidhuber, 1997] alleviates the problem by explicitly modeling an adaptive memory. Gated recurrent units [Cho et al., 2014] have similar qualities while being somewhat simpler to implement.

Bahdanau et al. [2014] introduced a learned soft alignment model to NMT by replacing the context vector by a richer, target-context dependent representation: Given a decoder's hidden state at time t, $\mathbf{h}_t$, and a vector of encoder hidden states $[\mathbf{h}'_1, \dots \mathbf{h}'_s]$ (s being the length of the source sequence) a so-called *attention model* calculates a vector $\boldsymbol{\alpha}_t$, with elements $\alpha_{t,1}, \dots, \alpha_{t,j}$ given by

$$\alpha_{t,j} = g(\mathbf{h}'_j, \mathbf{h}_t), \tag{2.44}$$

where $g(\cdot)$ is an arbitrary differentiable function returning a single real number, which can be interpreted as a score[29]. $\boldsymbol{\alpha}_t$ is normalized by applying the soft max function $\boldsymbol{\alpha}_t = \text{soft}\max(\boldsymbol{\alpha}_t)$.

Since each hidden state of the encoder only summarizes the sequence up to that point, Bahdanau et al. [2014] propose to use a bidirectional RNN [Schuster and Paliwal, 1997] for encoding both left and right contexts at each position.

Translation in NMT can be trivially carried out by modifying the training procedure to emit the tokens with maximum conditional probability in Equation 2.39, repeating the process until an end-of-sentence token[30] is emitted. The final translation is then simply the concatenation of emitted tokens. This is referred to as *greedy* decoding, since all tokens are the best possible local choices at each time step of the translation process. Alternatively, a simple variant of the beam search algorithm can be employed, considering b words at each time step of the translation process: The algorithm always maintains b partial hypotheses (prefixes of translations) at each time step, discarding other possible continuations. If any of the b best translations is finished, i.e. the last emission was an end-of-segment token, b is reduced by one, and the process is continued until b is 0. This is first described by Sutskever et al. [2014] for NMT.

Variants[31] of this general architecture are performing favourably compared to traditional statistical machine translation [Luong and Manning, 2015; Bentivogli

[29] Bahdanau et al. [2014] propose a multi-layer perceptron — Neubig [2017] list a number of varieties.

[30] Parallel data for NMT models is augmented to include a distinct end-of-segment token in source and target segments for being able to explicitly learn appropriate sentence lengths.

[31] The attention models defined in Neubig [2017] and Bahdanau et al. [2014] differ significantly in how the context vector $\mathbf{c}_t$ is applied.

et al., 2016*b*; Wu et al., 2016; Bojar et al., 2017] *inter-alia*.

A major caveat in NMT is the fixed vocabulary, which cannot be altered after a model is trained. This poses issues for practical applications, for example in PE tasks where users may input unknown words. This can however be alleviated by using vocabulary based on *sub-words*, as described by Sennrich et al. [2015] or previously Schuster and Nakajima [2012]. The sub-word vocabulary ideally includes the complete alphabet of the target language to be able to reliably back off. But the general problem remains, since there are much more possible atomic symbols than could possibly be fitted in a fixed-size vocabulary, e.g. the ever extended number of *emojis* in the *Unicode* character coding standard [Unicode Staff, 1991].

2.3 Evaluation of (Machine) Translation

Evaluation is a topic of great importance in ML and NLP in particular. Aside from the trivial 0-1 error or other related error measures, evaluation measures in information retrieval (IR) are non-trivial due to the complexity of the task, e.g. due to the more complex task of comparing rankings, or other structures such as trees.

However, in NLP and also MT there is another layer of complexity in evaluation: As language is generative, ambiguous and infinite in some ways, there is hardly a single true reference to compare to available. In the comparatively simple task of part-of-speech (POS) tagging this is easy to demonstrate: In Figure 2.6 there are two equally correct labels for the word "duck" (noun and verb). In evaluation the correct sequence entirely depends on the context, which may or may not be given. Evaluation measures in NLP should be able to account for that, as simple measures based on 0-1 error may not give any meaningful insights.

This is more pronounced in MT where the bilingual aspect makes matters more complicated, as there are a huge number of equally correct translations for every non-trivial sentence, which possibly differ only in nuances. However, most automatic evaluation of MT is done in the context of a single or a couple of *reference translations* per segment or sentence[32], which are considered the ground truth. All evaluation measures arguably try to measure some form of semantic equivalence.

In this section we will first present the issues that arise in human evaluation of translation, and then present the methods we use for automatically evaluating translation outputs, and discuss their correlations with human judgments. Lastly we discuss methods for comparing translation outputs as well as providing statistical

[32] We will use these terms interchangeably, as the language pairs examined in this thesis have identical definitions of sentence boundaries and we are not concerned with translation in wider contexts than single sentences.

	We	saw	her	duck	.
Interpretation #1	Pronoun	Verb	Pronoun	noun	Punctuation
Interpretation #2	Pronoun	Verb	Pronoun	Verb	Punctuation

Figure 2.6: POS-tag sequences for two interpretations for the English sentence "We saw her duck.".

significance for comparison of results.

2.3.1 Human Evaluation

Manual human evaluation by experts is the gold-standard for evaluation in MT. This type of evaluation is most commonly split into two separate ratings: *adequacy* and *fluency*, as proposed in the ALPAC report by Pierce and Carroll [1966][33], and also later in the context of the DARPA machine translation initiative [White et al., 1994].

The question for the adequacy of a translation seeks to assess whether, and to what extent the meaning of a source sentence is properly reflected in a proposed translation. This implies that an assessor is able to to understand both source and target languages, and is also able understand the subject matter. Fluency, on the other hand only answers the question whether a translation is a valid utterance in the target language, in terms of grammatical correctness, proper choice of words, spelling and other more stylistic aspects. Evaluation for fluency can be assessed by monolingual evaluators, possibly without domain knowledge. Both adequacy and fluency evaluations are typically carried out on a per-sentence basis. Ratings are assessed by using a coarse ordinal scale [LDC, 2005], or continuous scale [Graham et al., 2013*a*].

While the adequacy-fluency human evaluation is a preferred type of human evaluation, it is costly to carry out as bilingual experts need to be recruited[34]. Additionally, fluency seems to be a subjective measure, as low inter-annotator agreement is common [Graham et al., 2013*b*] and it is overall difficult to assess in a consistent manner [Koehn and Monz, 2006*a*; Turian et al., 2006; Denkowski and Lavie, 2010]. However, since fluency can be assessed by monolingual non-experts, successful methods for sourcing assessments from non-experts have been developed [Graham et al., 2013*a*; Callison-Burch, 2009].

A method for cost- and time-efficient human evaluation to contrast MT systems can be carried out by collecting pairwise preferences and establish a ranking

[33] In the report, fluency corresponds to intelligibility, and adequacy to fidelity.
[34] This constraint can be relaxed if reliable reference translations are available.

[Sakaguchi et al., 2014; Bojar et al., 2014*a*]. Even simpler methods have been proposed, e.g. retrieving human assessments of acceptability by per-sentence questionnaires with yes/no responses [Callison-Burch et al., 2008].

Other types of human evaluation include error analysis [Stymne and Ahrenberg, 2012; Koponen, 2010; Vilar et al., 2006] and measuring post-editing effort [Snover et al., 2006; Bentivogli et al., 2016*b*].

Overviews of human evaluation for MT can be found in [Graham et al., 2017], [Han and Wong, 2016] and [Uszkoreit, 2007].

2.3.2 Automatic Evaluation

Due to the challenges of human evaluation, as well as the fact that development of MT systems requires repeated evaluation of translation quality, a wide range of automatic measures for automatic evaluation of MT outputs have been developed.

Banerjee and Lavie [2005], propose a number of properties an automatic measure should exhibit: a) Sensitivity to differences in quality to successfully discriminate similar but sufficiently different systems while consistently identifying similar results; b) Reliable results over a wide range of data and system setups, producing similar results on similar setups; c) Correlation with human judgments. Su et al. [1992] additionally suggest low cost and high speed of computation as desirable properties of an automatic evaluation metric.

The most widely used metrics can be assorted in two classes: edit-distance based, measuring lavishness of string transformations, and precision-based. In the following we present the predominant approaches for both directions, which are employed in our work.

2.3.2.1 Edit Distance-Based Evaluation

The most basic evaluation distance-based metric applicable to machine translation evaluation is *word error rate* (WER), proposed by Su et al. [1992], which is a variant of the Levenshtein distance [Levenshtein, 1966] operating at word-level instead of string level. By definition it is the smallest possible number of operations to transform an array of words H (the hypothesis) into another array of words R (the reference). Possible operations are insertions I, deletions D, and substitutions R (replacements):

$$\mathrm{WER}(H, R) = \max_{I+D+R} \frac{I + D + R}{N}, \tag{2.45}$$

where N is the number of words in the reference translation. If there are several reference translations, the average number of words is used in the denominator, and the minimal number of edits to arrive at any reference is used in the enumerator.

Note that all operations have a single shared cost of 1.

A variant of WER, which is more common in MT evaluation is presented in [Snover et al., 2006] and referred to as *translation error rate* (TER)[35]:

$$\mathrm{TER}(H, R) = \min_{I+D+R+S} \frac{I + D + R + S}{N}, \tag{2.46}$$

where I, D, R and N are consistent with their definitions in WER. However, a overly harsh reduction of scores due to simple displacement of groups of words is prevented by introducing shifts S as another class of possible editing operations. It allows to move groups of words at once, incurring only the cost of a single operation, instead of two operations per word (deletion and insertion for shifting a single word).

2.3.2.2 Precision-Based Evaluation

Precision and recall are fundamental metrics in many applications of NLP. However, without modification they can be non-informative in MT, as their naïve application ignores word order, which is inarguably not negligible when trying to measure semantic equivalence. To account for word order one may extract ordered subsets of words from both hypothesis and the reference translation, i.e. N-grams, and calculate precision relative to these subsets. The BLEU[36] score [Papineni et al., 2002] implements just that. For each N we have:

$$q_N = \frac{\sum_{g \in n_N(c)} \delta_r(g)}{|n_N(c)|}, \tag{2.47}$$

where c is a candidate translation, $n_N(c)$ is the set of N-grams of length N in c, and $\delta_r(g)$ is 1 iff the N-gram g also appears in the reference translation r. The count in the enumerator is *clipped* by the maximum number that each N-gram appears in the reference translation, to not reward nonsensical repetitions.

These single precision scores are summed up to some maximum N, which is commonly set to 4.

The original BLEU score is defined on a corpus basis, i.e. summing over all sentences in a given corpus $\mathcal{C}$:

$$Q_N = \sum_{c \in \mathcal{C}} q_N. \tag{2.48}$$

[35] TER was initially proposed as a metric for estimating technical translation effort in post-editing.
[36] Bilingual Evaluation Understudy

Different N are combined in a geometric average[37]:

$$P = \left(\prod_{i=1}^{N} Q_i \right)^{\frac{1}{N}}. \tag{2.49}$$

As only using precisions would in practice possibly skew the score to prefer shorter translations, a *brevity penalty* is introduced to penalize hypotheses that are shorter than the reference[38]:

$$\mathrm{BP} = \begin{cases} 1 \text{ if } |c| > |r| \\ \exp^{(1-|r|/c)} \text{ else,} \end{cases} \tag{2.50}$$

where $|c|$ is the length of the candidate translation, and $|r|$ the length of the reference.

The total score is then simply calculated as the product of the precision term and the brevity penalty:

$$\mathrm{BLEU} = \mathrm{BP} \times P. \tag{2.51}$$

Closely related to the BLEU score is the NIST translation evaluation metric [Doddington, 2002], which additionally includes a term to weigh "informativeness" of N-gram matches, weighing less frequent N-grams more in the precision calculation. Agarwal and Lavie [2008] propose a variant of BLEU allowing fuzzy N-gram matches.

Another precision-based measure is the METEOR score proposed by Lavie et al. [2004], which is presented as an alternative to BLEU with directly taking recall into account. It does so by generating a one-to-one alignment at the unigram level, calculating precision, recall and finally an F-score by counting correct and incorrect alignment links. Multiple references are handled by calculating the scores separately and taking the best scoring alternative. Alignment is carried out in several stages, which includes alignment of word stems and other word forms. The final score includes a factor for preferring longer N-gram matches. Note that the score requires language-specific resources for the alignment.

It is important to note that the BLEU and NIST metrics are designed to evaluate performance on the corpus level, which poses special challenges when sentence-wise

[37] Note that the geometric average is zero if any of the factors is 0, which makes the BLEU score problematic for sentence-level scoring.

[38] The alternative, recall (and in combination with precision F-score) can not be directly applied here, since the BLEU score is designed to support multiple reference translations, and recalling translation candidates for the same source span should not be rewarded. METEOR handles this issue with the alignment.

scores are needed. The BLEU score can be used to score with multiple reference translations per candidate by altering the clipping procedure to use the maximum number of occurrences of any N-gram in any reference, and by using average lengths when calculating the brevity penalty.

2.3.2.3 Correlations with Human Judgments

Correlation with human judgments provides the vital credibility of automatic measures for evaluation of translation quality.

Papineni et al. [2002] show good correlations between BLEU and document-level adequacy (rated by bilinguals), as well as fluency ratings on a scale of one to five, using a range of correlation coefficients (Pearson Correlation [Pearson, 1895], Spearman rank Correlation [Spearman, 1904], Kendall's τ [Kendall, 1938]). Doddington [2002] also show good correlations with document-level adequacy, fluency and informativeness[39] for the closely related NIST score. Furthermore, good correlations (Pearson and Spearman correlation coefficients) with human judgments of adequacy and fluency are shown for the TER score and some variants in [Snover et al., 2009].

Banerjee and Lavie [2005] present a study on correlations of a number of evaluation metrics including the BLEU and NIST scores, correlating segment-level measurements to human judgments of adequacy and fluency (on a scale from one to five) with the Pearson correlation coefficient. Tan et al. [2015] discuss a disparity of the BLEU score compared to human judgments (assessed by pairwise preferences of non-experts), which primarily arose from BLEU being overly sensitive to lexical choices. The study in [Agarwal and Lavie, 2008] suggests the same, showing improved correlations by allowing more flexible N-gram matches. Callison-Burch et al. [2006] also suggest that not being able to employ flexible matching, word-level importance weighting, and omitting recall renders the BLEU score insufficient as an evaluation metric. They also show low correlations with human judgments in a fluency-adequacy study, again with ratings from one to five.

Despite these issues, the BLEU score has proven to reliably discriminate better from worse translation systems in a wide range of evaluations, e.g. [Bojar et al., 2017], [Nakazawa et al., 2014] or [Cettolo et al., 2014*b*], *inter-alia*. Furthermore, the score has been able to precisely reflect advances in machine translation technology, e.g. improvements by NMT applied to spoken language translation [Luong and Manning, 2015; Cettolo et al., 2015] or translation of news text [Sennrich et al., 2016; Bojar et al., 2016]. Bentivogli et al. [2016*b*] also highlight the advancement of MT technology with neural machine translation by reporting BLEU scores, while

[39] Informativeness is measured by the ability to answer a set of questions based on a system's translations alone.

providing an in-depth error analysis showing the superiority of the neural systems, which agrees with the automatic evaluation.

2.3.2.4 System Comparison & Significance Testing

Because human evaluation is usually carried out by a number of people, it can be readily verified with inter- and intra-annotator agreements, see e.g. the methods for pairwise judgments in [Callison-Burch et al., 2011*a*]. But automatic evaluation metrics as described above, only provide a single numeric assessment of the quality of the translations, which raises doubts about how reliably outputs of different machine translation systems can be compared. A telling example for this behavior is the BLEU score, but is also applicable to most other automatic scores: It is possible, though unlikely, that two systems may produce a very similar or even identical scores on the same test set, but perform fundamentally different on different parts of the test set [Berg-Kirkpatrick et al., 2012]. This is expected and due to the factorization of the score, counting N-grams and measuring relative length differences.

To assess the true magnitude of score differences[40] of automatic evaluation metrics, appropriate *significance tests* can be used, which provide p-values and confidence intervals for score differences. Two tests are widely used for system comparison in MT: the two-sided bootstrap re-sampling test [Koehn, 2004*b*], and the approximate randomization test [Riezler and Maxwell, 2005]. The bootstrap test in principle explores whether the score difference between two systems holds true on repeated bootstrap samples[41] [Efron, 1992]. The test assumes that the samples represent the true population of the data. In contrast, the approximate randomization test does not make any assumptions, is trivial to implement, and in practice tends to perform similarly to tests based on bootstrap samples [Graham et al., 2014].

The stratified approximate randomization test[42] [Clark et al., 2011], as depicted in Algorithm 1, works by repeatedly calculating the difference in scores with some translations swapped between the two systems and checking whether the difference agrees with the original statistic. For this type of test, the Null hypothesis is that the two systems are actually not different, i.e. the score difference is only by chance. The Null hypothesis is rejected if the returned p-value of Algorithm 1 is lower or equal than a pre-determined threshold, e.g. 0.05 with a confidence interval of 95%. This is the case if the absolute values of the score differences are often greater than

[40] Some significance tests can also be used to assess the output of a single system, e.g. by re-sampling.

[41] Random permutations, with replacement, of the translation outputs of two systems.

[42] The originally proposed application of the approximate randomization test for MT evaluation of [Riezler and Maxwell, 2005] swaps items on the level of the evaluation metric, e.g. N-gram matches for the BLEU score.

the difference of the actual score when swapping translations, which allows for the interpretation that the observed difference likely occurred by chance. The test is approximate in that sense that it does not consider all possible permutations ($2^{\mathsf{size-of-test-set}}$), but rather a fixed number, typically $\geq 1,000$.

Algorithm 1 Stratified approximate randomization test for machine translation system comparison. *Inputs:* Test set, number of random restarts r, system outputs A and B, and evaluation metric. Algorithm adapted from [Riezler and Maxwell, 2005].

Compute actual score difference $d \leftarrow |\text{Evaluate}(A) - \text{Evaluate}(B)|$
$c \leftarrow 0$
for Random restarts $1 \ldots r$ **do**
  for all Segments of test set s **do**
    Swap translations outputs for segment s of system A and B with probability 0.5
  end for
  Compute pseudo-statistic $d' \leftarrow |\text{Evaluate}(A'_r) - \text{Evaluate}(B'_r)|$
  if $d'_r \geq d$ **then**
    $c \leftarrow c + 1$
  end if
end for
Return $p \leftarrow (c + 1)/(r + 1)$

While the approximate randomization test was found to be robust against the for research arguably more relevant type I errors[43] [Riezler and Maxwell, 2005], for some setups is has been shown that its power can be limited [Graham et al., 2014] compared to human judgments.

However, as Søgaard et al. [2014] and Berg-Kirkpatrick et al. [2012] show, significance tests in NLP are very sensitive to sample size and domain shifts, which may require very low p-values or tests on a variety of data sets to provide reliable results. Regarding length of test sets in MT evaluation, Estrella et al. [2007] show that at least about 500 segments are needed for meaningful results.

2.4 Linguistic Materials for Machine Translation

To provide meaningful results for the hypotheses in this work, we consider a wide range of linguistic materials for training and evaluation, i.e. parallel and

[43] Type I errors being the incorrect rejection of the Null hypothesis, i.e. deeming two systems outputs to be significantly different when they are actually not, and type II errors being the incorrect acceptance of the Null hypothesis, i.e. deeming two system outputs not significantly different when they actually are.

monolingual data from a wide range of domains.

2.4.1 Data Domains & Characteristics

A domain in NLP is a somewhat fuzzy concept. We follow a general classification as disseminated for example by van der Wees et al. [2015]: On the top-level, the domain of a text refers to its topic and genre. The topic of a text signifies the general subject, while the genre can be defined by every feature of a text that is not due to its topic. This can include the text's formal structure and the style its been written in. The most prominent effect of a topic is its induced vocabulary. If a topic is known or can be inferred otherwise, it can be used for resolving ambiguity, i.e. provide a means for domain adaptation [Hasler et al., 2014].

Domain adaptation of MT systems is a well known problem, but in SMT, e.g. phrase-based or hierarchical phrase-based MT, there is no single straight-forward way to perform domain adaptation, since these models consist of a number of sub-models which are trained in different ways with different objectives (i.e. translation and language models). The generative models can be adapted by including domain-specific data in the training, or be made more robust by smoothing techniques. The weights of the discriminative log-linear model can be straight-forwardly adapted by training on in-domain data. This can also be done with NMT, in this context often referred to as *fine-tuning*.

We will now discuss the types of text used in this work.

2.4.1.1 News-Style Text

Due to its abundance on the internet, news text is a widely used resource for MT, which is also employed in MT evaluation tasks such as the Conference on Machine Translation (WMT[44]) [Koehn and Monz, 2006*b*; Bojar et al., 2017]. A news text naturally covers a diverse set of topics, but mostly follows a formal style, which depends on the author and the context. Extraction of such data relies on the availability of multi-lingual news outlets. An example are the test sets released for the WMT evaluations, see e.g. [Bojar et al., 2017], and the news-commentary corpus[45].

News articles are written by a manifold of authors and generally do not follow a formal structure. The virtually unlimited range of topics appearing in this type of data adds some difficulty for translation, due to the possible topic/style mismatch.

[44] Formerly known as the Workshop on Machine Translation.
[45] `http://www.casmacat.eu/corpus/news-commentary.html`

A	Human Necessities
B	Performing Operations, Transporting
C	Chemistry, Metallurgy
D	Textiles, Paper
E	Fixed Constructions
F	Mechanical Engineering, Lighting, Heating, Weapons
G	Physics
H	Electricity

Table 2.1: Top-level sections of the International Patent Classification (IPC).

2.4.1.2 Patents

Patent translation is a very active field in both research and practice, partly due to the inherent internationality of the patent process: Patents for the European Patent Office (EPO) have to be either submitted in one of the three official languages (English, French or German), or a translation into one of these languages has to be provided by the applicant[46]. When a patent is granted, a translation has to be provided in an official language for every one of the designated countries [WIPO, 2014], which further increases the translation expenditure. Since patent translation requires a lot of expertise, the process is expensive, and thus the incentive for research towards automation is great.

First efforts for the application of MT began in Europe the 1980s [Johnson et al., 1985], and have continued ever since [Orsnes et al., 1996; Tinsley et al., 2010; España Bonet et al., 2011]. International research interest in MT for patent translation is also high [Goto et al., 2013; Nakazawa et al., 2016] *inter-alia*, notably in Asia.

Patents are a viable application of automatic translation, since they follow a strict structure: Patents are divided into four sections: title, abstract, background descriptions (descriptions for short), and claims, each of which can be considered as a distinct genre of text. While abstracts and descriptions allow free-form text, the title is naturally constrained and the claims follow a very strict form: As denoted in [WIPO, 2014] and further discussed in [Fuji et al., 2015], a claim has three distinct parts, the preamble, the transitional phrase (mostly a verb) and lastly the body of the claim. Due to these constraints, claims can be considered as non-natural or controlled natural language. Structures like this can easily learned by MT systems, which is advantageous for the application of CAT.

Besides the genre, patents cover a vast amount of different topics, which are manually sorted in a number of content classes, as depicted in Table 2.1. This

[46] `http://www.epo.org/law-practice/legal-texts/html/caselaw/2016/e/clr_iii_f_1.htm`

is only the classification on the topmost level — patents are sorted into a very fine-grained classification hierarchy.

Overall, the specifics of patent translation can lead to very good automatic translation results by exploiting the idiosyncrasies of patent text. For our work we use data from two different corpora, Japanese-English [Utiyama and Isahara, 2007] and German-English data [Wäschle and Riezler, 2012*b*,*a*]. Main difficulties in translation of patents include very specialized vocabulary, cross-references (in claims), overly long sentences, and references to figures similar to captions, but found in running text.

2.4.1.3 Legal Texts

The parallel data from the legal domain which is publicly available mostly consists of multi-lingual actual legislation (motions, laws, amendments, and regulatory texts), as well as political speeches, discussions and explanations around legislation. Laws and regulations, similar to patents, follow a strict form. Speeches and discussions, while being formal, not necessarily follow a such strict form. Naturally, this type of data covers a wide range of topics, such as economic and trade, social issues, science and education, or sports, amongst others.

For our work we use to different corpora extracted from publications of the European parliament: The *Europarl* corpus [Koehn, 2005], consisting of texts from the proceedings of the European Parliament, and verbatim reports of the plenary sessions of the European Parliament, i.e. speeches and discussions around legislation [Hajlaoui et al., 2014]. The corpus is available in all of the official languages of the European Union, but we only use the English-German parts. Secondly, we use the English-Italian parts of the *JRC-Acquis* data [Steinberger et al., 2006], consisting of legislations concerning the common rights and obligations of European member states.

Legislation in general is a difficult subject matter, and the wide range of topics and genres is a further challenge.

2.4.1.4 Manuals

Manuals are a another domain using somewhat constrained language, and is often readily available in multiple languages. In this work, we use software manuals and documentation from the *OPUS* corpus [Tiedemann and Nygaard, 2004] for the English-Italian language pair. The data consists of a number of manuals for software products, including documentation of a word processor, a reference for a user interface library, and documentation of a programming language. The data includes a variety of types of text: titles, content listings, descriptions and definitions.

2.4.1.5 Spoken Language

Translation of spoken language is challenging for a number of reasons: It may contain spontaneous speech (which can render it challenging to identify individual segments), and it requires at least one further complex pre-processing step in the form of either manual transcription or automatic speech recognition, which are both error-prone tasks. Such data may also contain speech disfluencies that may cause issues for non-specialized translation systems. Furthermore, the number of topics in this type of data is virtually unlimited.

We use English-German and English-Russian data distributed for the International Workshop on Spoken Language Translation (IWSLT) [IWSLT, 2004]. The data consists of transcribed and translated public talks. Since the data is produced (transcribed and translated) by a number of volunteers, the quality of the data may vary, since there is no standardized way for handling disfluencies, or the sometimes occurring use of informal language.

2.5 Optimization in Machine Translation

Optimization has many applications in SMT: All sub-models included in the log-linear model are optimized in some special manner — phrase translation probabilities, N-gram language models, as well as reordering models are estimated by optimizing maximum likelihood objectives, commonly by relative frequency estimation, seeking an in some sense optimal distribution over sequences of tokens or trees in context. Finally, the decoding algorithms, mostly instances of dynamic programming, solve optimization problems in the form of arg max operations.

In our work however, we focus on finding in some defined sense good and ultimately robust weights for the linear model, which is used for finding appropriate translations. A variety of methods have been proposed for learning suitable weights, but methods that optimize directly towards evaluation have naturally been the most successful approaches overall. This type of optimization is most commonly referred to as *parameter tuning*. The term originates from being easily able to fine-tune a fully trained model to a new domain by training on a in relation small number of additional in-domain segments.

This section covers related work to our own approach in SMT tuning: First, general methods of discriminative training for SMT are discussed. We then cover the first tuning strategy – direct error minimization, namely Minimum Error Rate Training. What follows is an introduction and presentation of related work of tuning methods originating from different directions, including structured prediction and ranking.

2.5.1 Digression: Discriminative Training in Statistical Machine Translation

As introduced in Section 2.2.1, the distribution $P(E|F)$ can be directly learned in a maximum entropy framework, or log-linear model. Models that directly estimate $P(E|F)$ given some training data can be characterized as *discriminative models*. In this section we will shortly discuss how to train these types of models, and explain different approaches to discriminative training in SMT.

Och and Ney [2002] propose the optimization of the parameters of the log-linear model by finding a parameter vector $\mathbf{w}$ (also referred to as the weight vector) which maximizes the log-likelihood of a given parallel training corpus with N examples, by (omitting vector notation for f and e):

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left\{ \sum_{i=1}^{N} \log p_{\mathbf{w}}(e_i|f_i) \right\}. \tag{2.52}$$

This convex optimization problem can be approached with gradient descent, however, since Equation 2.15 involves an intractable normalization term, the approach requires approximating the space of possible translations[47]. A similar proceeding, which however seeks to do without the maximum approximation, is presented by Blunsom et al. [2008] which explicitly include derivations in their model, while also maximizing a log-likelihood objective. Discriminative models, for example the *perceptron* algorithm [Rosenblatt, 1958], enable efficient reranking approaches [Shen et al., 2004], in turn enabling the use of complex feature sets, which can be used in the first pass of decoding. Wellington et al. [2009] present a boosting procedure for tree-structured translation models.

In another line of work, Tillmann and Zhang [2005] present a discriminatively trained translation model, modeling local phrase reordering decisions as a *block sequence model*, utilizing a log-linear model and maximizing likelihood of the training data. Serrano et al. [2009] and Wang et al. [2007] show how the discriminative kernel regression framework can be used for modeling in machine translation.

However, in traditional SMT[48], methods directly[49] optimizing the evaluation metric have been shown to outperform discriminative approaches maximizing the likelihood of the training data: Tillmann and Zhang [2006] show improved results combining the method of Tillmann and Zhang [2005] with a discriminative model, which is trained to prefer high quality translations. Influentally, Och [2003],

[47] Additionally, Och and Ney [2002] introduce the *maximum approximation*, i.e. $\max P(E, A|F) \coloneqq \max P(E|F)$.

[48] In NMT, maximizing log-likelihood is the state-of-the-art, but minimizing the expected risk [Och, 2003; Smith and Eisner, 2006; He and Deng, 2012] can also be used to directly optimize an evaluation metric such as BLEU [Shen et al., 2015].

[49] Direct in the sense of being related to the objective of the optimization.

presents an algorithm to directly learn a log-linear model's weights to optimize the evaluation metric. Despite these results, [Shen and Joshi, 2005] show that in reranking, a likelihood based objective can be effective, and improvements in training performance positively correlate with the evaluation metric.

2.5.2 Direct Error Minimization

In direct error minimization methods, the goal is to conductively optimize the gold-standard evaluation metric of interest, e.g. TER or BLEU scores, on a given set of training examples. The respective objective for optimizing the log-linear model can be formulated as follows:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left\{ \sum_{i=1}^{n} g\left(e_i^*, \max_{j \in \mathcal{Y}(f_i)} p_{\mathbf{w}}(e_{i,j}|f_i) \right) \right\}, \tag{2.53}$$

where $g(\cdot)$ is the evaluation metric, i iterates over the indexes of the training data with n examples, $\mathcal{Y}(f)$ is the (indexed) set of translation hypotheses for a given source segment f, and e_i^* is the reference translation for example i.[50]

Formulated as a loss function, to minimize the true error:

$$L_{\text{true}} = -\sum_{i=1}^{n} g\left(e_i^*, \arg\max_{(e,h) \in \mathcal{Y}(f_i)} m(f_i, e, h) \right), \tag{2.54}$$

where $m(\cdot)$ represents the model score of a translation hypothesis, i.e. $\langle \mathbf{w}, \boldsymbol{\phi}(f, e, h) \rangle$, for a feature representation derived from source f, target e with derivation h. The derivation h corresponds in word-based models to the word alignment, in phrase-based models to a phrase segmentation, and in hierarchical phrase-based models to the co-aligned source and target parse trees.

While TER can be directly optimized in this framework, the BLEU score, being defined on the full corpus, is not evaluable for isolated hypotheses, which either calls for a sentence-wise approximation, or specialized optimization procedures. The predominant approach for directly optimizing BLEU is described by Och [2003]. Other direct approaches include work by Chung and Galley [2012] and Erdmann and Gwinnup [2015].

2.5.2.1 Minimum Error Rate Training

Minimum error rate training (MERT), as described by Och [2003] aims to directly optimize the true loss (Equation 2.54) on a set of translation hypotheses, e.g. a

[50] From now on we will omit the vector notation for e and f if the context is unambiguous.

set of k-best lists of a training set[51]. Given an initial or previous model, a single iteration of the MERT procedure first produces a list of k-best translations for each input of the training data, annotated with model scores and feature values.

Then, MERT, in its simplest form, optimizes the weight of each feature in a linear model $\mathbf{w}$ separately, as a variant of Powell's method [Powell, 1964]: An naïve modus operandi for this single-weight optimization would utilize a grid search, reranking all lists with the new weight and observing the global evaluation score. This is generally infeasible due to the large number of 1-best allocations. Instead, in MERT, an efficient way of finding the optimal weight for a given feature for a fixed set of k-best lists in utilized: While other parameters are fixed, the model score of all entries of every k-best list can be represented as follows (assuming only unique entries in the k-best lists):

$$\langle \boldsymbol{\lambda}, \boldsymbol{\phi}'(f,e) \rangle + \gamma \phi''(f,e), \tag{2.55}$$

where $\boldsymbol{\phi}'$ is the feature representation using only the fixed features, $\boldsymbol{\lambda}$ are the weights for the fixed features, γ is the weights for the current active feature, and ϕ'' the respective feature value. Each hypothesis can thus be represented as a line, with slope $\phi''(f,e)$ and intercept $\langle \boldsymbol{\lambda}, \boldsymbol{\phi}'(f,e) \rangle$. This formulation allows to efficiently determine an optimal global weight for a given feature by generating the upper envelope (UE) of the linear model for each k-best list [Macherey et al., 2008], providing an exhaustive representation for all $\gamma \in \mathbb{R}$:

$$\mathrm{UE}\left(\mathcal{Y}(f)\right) = \max_{e \in \mathcal{Y}(f)} \left\{ \langle \boldsymbol{\lambda}, \boldsymbol{\phi}'(f,e) \rangle + \gamma \phi''(f,e) : \gamma \in \mathbb{R} \right\}. \tag{2.56}$$

Since the upper envelope is piecewise linear and convex, it enables to efficiently determine a finite number of values for γ, namely those where the global evaluation metric actually changes its value. With this insight, a globally optimal score for each feature can be efficiently determined. The process is iterated for a number of epochs, re-decoding the training data each time with the new parameters.

While the algorithm is capable to optimize the weights of a typical SMT system, i.e. less than 30 features, it does not scale to larger feature sets, e.g. when using sparse, lexicalized features, see e.g. [Hopkins and May, 2011].

MERT is extensively studied: The algorithm can be extended to use a larger portion of the search space, i.e. lattices encoding source segmentation in phrase-based MT [Macherey et al., 2008; Galley and Quirk, 2011], or hypergraphs encoding tree-structured derivations in syntax-based MT [Kumar et al., 2009], including more efficient approaches to compute the upper envelope [Sokolov and Yvon, 2011; Dyer,

[51] Even with a small data set and reduced search space, finding optimal weights that maximize the global BLEU score on this sample is still a daunting task, as there are N^k 1-best allocations to consider, N being the size of the training data.

2013]. Regularization can also be employed [Cer et al., 2008]. Other aspects of the algorithm, considering random restarts [Moore and Quirk, 2008], multi-dimensional optimization [Galley et al., 2013], or stability of the resulting weights [Foster and Kuhn, 2009; Clark et al., 2011] have also been thoroughly explored.

Note that, since BLEU is non-differential and piecewise linear [Och, 2003; Papineni et al., 2002], it cannot be optimized directly as some metrics in IR (cf. Section 2.6.1.2). While MERT can optimize the gold-standard score for a given set of k-best lists, it is no guarantee whatsoever that it is a global optimum, since the k-best lists depend on the weights.

2.5.3 Structured Prediction

In general ML, when there are complex output spaces — which is commonplace in MT — a different type of learning algorithms can be used compared to the ones designed for more simple tasks such as (binary) classification. *Structured prediction* can be characterized by an antagonism [Daumé, 2006]: While each output $y \in \mathcal{Y}$ decomposes into an ordered set of states encoded by variable sized vectors, the losses or error metrics do not decompose in the same way, but are evaluated for the full structure. Besides MT, typical applications include other tasks in NLP, such as tagging, parsing or more general sequence learning tasks[52]. Structured prediction problems are abundant in NLP.

The structured prediction framework enables the use of expressive features defined on structured inputs and outputs, and allows for efficient online learning algorithms.

A fundamental algorithm in structured prediction is the *structured perceptron* [Collins and Duffy, 2002; Collins, 2002], a linear model parametrized by a weight vector $\mathbf{w}$, as depicted in Algorithm 2.

Algorithm 2 Structured perceptron. *Inputs:* Learning rate η, set of training examples with size N. Algorithm adapted from [Collins and Duffy, 2002].

for $i \leftarrow 1 \dots N$ **do**
  $\hat{y} \leftarrow \arg\max_y \langle \mathbf{w}_i, \boldsymbol{\phi}(x_i, y) \rangle$
  if $\hat{y} \neq y^*$ **then**
    $\mathbf{w}_{i+1} \leftarrow \mathbf{w}_i + \eta \left(\boldsymbol{\phi}(x_i, y_n^*) - \boldsymbol{\phi}(x_i, \hat{y}) \right)$
  end if
end for
Return $\mathbf{w}_{N+1}$

The algorithm closely resembles the original perceptron algorithm for classification and has the same convergence properties, but the condition for an update is

[52] There are also notable exceptions of this classification, such as POS-tagging where the natural loss is actually decomposable.

different: An update is performed iff the 1-best output structure with the current weights $\mathbf{w}$ is not correct, i.e. not identical to the gold-standard structure.

The algorithm is not directly applicable to traditional SMT: Since one or more reference translations are used for evaluation, an exact match is either unlikely or impossible, which is further complicated by the large search spaces. Additionally, since SMT relies on a Viterbi (maximum) approximation in decoding, scoring a single derivation of a string instead of all possible derivations, the arg max operation adopted in the structured perceptron is not exactly computable. The Viterbi approximation can be straight-forwardly coped with by having the feature mapping $\boldsymbol{\phi}(\cdot)$ include the *hidden variable* h, encoding the derivation, see [Liang et al., 2006*a*]. Liang et al. [2006*a*] further propose to use an approximate reference for the update condition, using a sentence-wise surrogate metric making it possible to pick an oracle translation for each k-best lists [Och and Ney, 2002].

The loss for the structured perceptron in SMT can be compactly formulated following Gimpel and Smith [2012*b*]:

$$L_{\text{struct}} = \sum_{i=1}^{n} -m\left(f_i, \underset{(e,h)\in\mathcal{K}(f_i)}{\arg\max}\ g(e, e^*)\right) + m(f_i, \hat{e}, \hat{h}), \tag{2.57}$$

where $g(\cdot)$ is again a gold-standard evaluation metric, e.g. per-sentence BLEU (slightly abusing notation by letting the function also return a (e, h) pair), and $m(f, e, h) = \langle \mathbf{w}, \boldsymbol{\phi}(f, e, h)\rangle$, $\mathcal{K}(f)$ returns a k-best list for input f, and:

$$(\hat{e}, \hat{h}) \approx \underset{(e,h)}{\arg\max} \langle \mathbf{w}, \boldsymbol{\phi}(f, e, h)\rangle, \tag{2.58}$$

as returned by a SMT decoding algorithm. This loss is also closely related to a *ramp loss* objective [Chapelle et al., 2009; Collobert et al., 2006], as shown by [Gimpel and Smith, 2012*b*][53] for SMT:

$$L_{\text{ramp}} = \sum_{i=1}^{n} - \max_{(e,h)\in\mathcal{K}(f_i)} [m(f_i, e, h) + g(e, e^*)] + m(f_i, \hat{e}, \hat{h}). \tag{2.59}$$

In contrast to the hinge loss of the structured perceptron, the ramp loss is non-convex, but it is a tighter upper bound of the true loss in Equation 2.54.

2.5.3.1 Margin-Infused Relaxed Algorithm

For SMT variants of the margin-infused relaxed algorithm (MIRA) [Crammer and Singer, 2003], more precisely its application to structured prediction [Crammer et al., 2006], have been proposed [Watanabe et al., 2007*b*; Chiang et al., 2008; Chiang, 2012; Gimpel and Smith, 2012*b*]. The key idea of the algorithm is to

[53] Referred to as $\text{loss}_{\text{ramp 2}}$ in [Gimpel and Smith, 2012*b*].

maintain a large margin (margin-infused) between better and worse hypotheses (in terms of a gold-standard evaluation function), which is at least as large as the difference in their evaluation scores, while at the same time keeping the weight updates as small as possible (conservative). The algorithm is commonly formulated as a quadratic program, see e.g. [Chiang, 2012]:

$$\begin{array}{ll} \text{minimize} & 1/2\eta||\mathbf{w}' - \mathbf{w}||^2 + \xi \\ \text{subject to} & \langle \mathbf{w}, \boldsymbol{\phi}(f, e^+, h^+)\rangle - \langle \mathbf{w}, \boldsymbol{\phi}(f, e, h)\rangle - \xi \\ & \quad \geq g(e^+, e^*) - g(e, e^*) \\ & \forall (e, h) \in \mathcal{Y}(f), \end{array} \tag{2.60}$$

where e^+ is a reachable, high-scoring derivation in terms of the evaluation function (referred to as the *hope* derivation):

$$(e^+, h^+) = \underset{(e,h)\in\mathcal{Y}(f)}{\arg\max} \langle \mathbf{w}, \boldsymbol{\phi}(f, e, h)\rangle + g(e, e^*), \tag{2.61}$$

$\xi \geq 0$ being slack variables. This optimization problem can be approached in numerous ways, for example with the cutting plane algorithm proposed by Tsochantaridis et al. [2004], as Chiang [2012] explicate.

Instead of a range of derivations as suggested by Watanabe et al. [2007*b*] and Chiang et al. [2008], Chiang [2012] consider using only a single *fear* derivation (y^-, h^-), which represents the most-violated constraint:

$$(e^-, h^-) = \underset{(e,h)\in\mathcal{Y}(f)}{\arg\max} \langle \mathbf{w}, \boldsymbol{\phi}(f, e, h)\rangle - g(e, e^*). \tag{2.62}$$

Following Gimpel and Smith [2012*b*][54], the optimization problem can be formulated as a ramp loss function:

$$\begin{aligned} L_{\text{mira}} &= \sum_{i=1}^{n} -\left[\max_{(e,h)\in\mathcal{Y}(f_i)} m(f_i, e, h) + g(e, e_i^*)\right] + \left[\max_{(e,h)\in\mathcal{Y}(f_i)} m(f, e, h) - g(e, e_i^*)\right] \\ &= \sum_{i=1}^{n} -\left[m(f_i, e_i^+, h_i^+) + g(e_i^+, e_i^*)\right] + \left[m(f_i, e_i^-, h_i^-) - g(e_i^-, e_i^*)\right]. \end{aligned} \tag{2.63}$$

Being an online learning algorithm, the problem can also be approached in a simpler variant by stochastic gradient descent (SGD) [Martins et al., 2010; Crammer et al., 2006; Eidelman et al., 2013*b*], when using a single constraint, omitting slack variables, and without imposing limits to the magnitudes of updates that the weight

[54] Referred to as $\text{loss}_{\text{ramp 3}}$ in [Gimpel and Smith, 2012*b*].

vector should receive[55]:

$$\mathbf{w}' \leftarrow \mathbf{w} + \eta\boldsymbol{\phi}(f, e^+, h^+) - \boldsymbol{\phi}(f, e^-, h^-), \tag{2.64}$$

which resembles the update of the structured perceptron, as shown in Algorithm 2, but using the notion of *hope* and *fear* derivations.

Mira has first been applied to a structured NLP problem by McDonald et al. [2005]. In SMT, the algorithm has received most interest due to its appeal as a online learning algorithm [Arun and Koehn, 2007; Watanabe, 2012; Watanabe et al., 2007*b*,*a*], and for enabling the use of sparse features [Chiang et al., 2009, 2008; Hasler et al., 2011; Eidelman, 2012] due to its efficiency. Since Mira can be implemented as an online algorithm, it also allows for parallelization [Eidelman et al., 2013*c*,*b*]. Batch variants of the Mira algorithm have also been explored for SMT [Zhao and Huang, 2013; Cherry and Foster, 2012]. As we have shown in our presentation of Mira for SMT, *hope* and *fear* derivations are a way of defining effective constraints. However, by using k-best lists as stand-in for the full search space, some fidelity is lost, which is why Chiang [2012] proposes a cost-augmented inference approach to search for constraints in a larger space. Wisniewski and Yvon [2013] present another variant for the constraints, and Eidelman et al. [2013*a*] propose a variant for the margin definition in Mira. Tan et al. [2013] propose an algorithm, which, similar to Mert, optimizes the exact corpus-level BLEU score. In general structured prediction for SMT, approaches that include the search procedure for learning have been explored thoroughly: Zhang et al. [2008] present an application of search-based structured prediction [Daumé et al., 2009] for SMT, and in another line of work, violation-fixing approaches are presented, which try to counter-act incorrect updates which are due to search errors [Huang et al., 2012; Yu et al., 2013; Liu and Huang, 2014; Zhang et al., 2013; Zhao et al., 2014].

2.6 Preference Learning & Ranking

> "NEW NAVY DEVICE LEARNS BY DOING
> Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser"
> [Special to the New York Times, July 8, 1958]

Preference learning [Cohen et al., 1998] and ranking methods have been extensively studied for IR [Manning et al., 2008], commonly in the context of document retrieval applications, i.e. web search engines [Li, 2011*b*; Liu, 2009]. Oftentimes, learned models are used to provide personalized user experiences, e.g. by learning per-user ranking preferences.

[55] The full optimization problem, as shown in Equation 2.60 for 1-best Mira, can also be optimized using SGD [Martins et al., 2010].

In NLP, ranking methods have also found application. To provide context of our own work, we first briefly introduce related work in ranking for general NLP. A more comprehensive overview of usage of ranking methods in NLP is available in [Li, 2011*a*].

The prevalent variant of ranking methods in NLP is the pairwise approach: Jiang et al. [2009] apply the linear Ranking SVM (support vector machine) [Herbrich et al., 1999; Joachims, 2002] to keyphrase extraction, contrasting it to a linear SVM trained in the pure classification case, showing consistently superior results in several ranking measures (MAP and Precision@$\{1, 3, 5, 10\}$, cf. Section 2.6.1.2) for the ranking method. Similarly, Metzler and Kanungo [2008] employ a Ranking SVM for sentence selection in a summarization task, using the cross product of relevant and irrelevant sentences as training data. Their evaluation shows that the Ranking SVM has comparable performance to ordinal regression and boosting approaches. Surdeanu et al. [2008] present an approach to question answering by ranking, learning from pairwise differences of manually verified answers and the outputs of an IR system. They employ a pairwise ranking perceptron with uneven margins as proposed for SMT tuning by Shen and Joshi [2005].

Directly related to our work, but considering the task of reranking static lists of offline generated translations, Shen et al. [2004] present the first application of a learning-to-rank algorithm to MT: They employ a pairwise ranking perceptron first proposed for parse reranking in [Shen and Joshi, 2004].

Hopkins and May [2011] first applied the learning-to-rank approach for *ranking* in machine translation, that is learning weights for the log-linear model, which are used for decoding. Their batch algorithm closely follows a scheme inspired by MERT, generating k-best lists, a learning step, and re-decoding. But instead of a line search, their algorithm generates training data for an off-the-shelf binary classifier[56]. Since learning in this approach is based on pairwise ranking, which can be carried out with efficient algorithms, the number of features is practically unlimited. The training data consists only of a subset of the total number of possible pairs, by drawing a uniform sample from the data, which is additionally filtered to include pairs that exhibit a maximal difference in the gold-standard scores (per-sentence BLEU). Results indicate on-par performance with MERT when using dense features, and possible additional gains by using an extended feature set. The algorithm is referred to as *PRO*. Bazrafshan et al. [2012] present a similar method, swapping out the binary maximum entropy classifier for a linear regression model. The model is trained to predict the difference in the gold-standard scores, instead of the relative ordering. Their experiments indicate an advantage of about 1 %BLEU for the proposed variant of the PRO algorithm. Green et al. [2013*b*] present an online version of PRO, in the sense of re-decoding after each update.

[56] In their case a maximum entropy model for binary classification [Daumé III, 2004].

They further apply updates with per-coordinate learning rates (*ADAGRAD*, [Duchi et al., 2010]) instead of SGD. Wuebker et al. [2015*a*] apply this method to online domain adaptation. Haddow [2013] use pairwise ranking following Hopkins and May [2011] for learning linear interpolation weights of translation models.

Ganitkevitch et al. [2012] study the effect of varying the loss function used in the optimization framework proposed by Hopkins and May [2011], showing virtually indistinguishable results between the perceptron algorithm for binary classification and two implementations of maximum entropy binary classifiers.

Dreyer and Dong [2015] report on an algorithmic solution to the combinatorial problem of pairwise ranking, which renders the sampling step in the training data generation of Hopkins and May [2011] unnecessary. This is done by applying the idea of Airola et al. [2011], reducing the time complexity of the learning algorithm (a linear Ranking SVM in the primal) from $\mathcal{O}(n^2)$ to $\mathcal{O}(ns + n\log(n))$ (n being the number of training examples and s the average number of non-zero features per example), without resorting to cutting-plane algorithms [Joachims, 2006].

Haddow et al. [2011] present a compromise between the margin-infused relaxed algorithm for SMT (MIRA) [Chiang, 2012] and pairwise ranking by implementing the *SampleRank* [Wick et al., 2011] algorithm for SMT: Instead of k-best list generation or re-decoding, the idea is to sample a number of translation variants from a single translation hypothesis of the decoder by using a Gibbs sampler [Arun et al., 2009], selecting a single oracle translation following Chiang et al. [2008], and finally performing weight updates if the current model disagrees with the gold-standard (in this case corpus BLEU calculated over a small sample of the full corpus).

Watanabe et al. [2006] apply the structured perceptron algorithm to a full pairwise sample from k-best lists for offline reranking, effectively rendering it a ranking approach.

Some variants of listwise ranking for SMT have also been proposed for structured prediction: Niehues et al. [2015] present a reranking approach based on the *ListNet* [Cao et al., 2007] algorithm, a cross-entropy loss which compares the distributions of the gold-standard ranking with the current model's ranking. Zhang et al. [2016] make use of the listwise ranking approach proposed by Pareek and Ravikumar [2014], also for reranking of k-best hypotheses. Finally, Chen et al. [2017] employ both *ListMLE* [Xia et al., 2008] and ListNet algorithms for tuning of a statistical machine translation system in a framework similar to MERT's.

2.6.1 Learning to Rank

Ranking of sets of objects is a widely used concept in ML and NLP. A plain example is an ordered list of search results returned by search engine given a query expressing an *information need*. The internal function of the search engine which defines the ordering of the objects (in this case hyperlinks and their associated

content/documents) in the returned list is the *ranking function*. The representation of the objects and the definition of the ranking function are fundamental problems in IR.

In the following section we will present the basic fundamentals and results of IR necessary to describe our own work on ranking for SMT.

2.6.1.1 Formalization of Information Retrieval

We formally define the basic task of *document retrieval* as outlined before. Considering a set of documents $\mathcal{D}$ and a number of queries $\mathcal{Q}$, the task is to assign each $d \in \mathcal{D}$ a score with respect to any $q \in \mathcal{Q}$. This score can be defined as mapping from the set of documents to the real numbers with respect to each query in $\mathcal{Q}$:

$$f : \mathcal{Q} \times \mathcal{D} \to \mathbb{R}. \tag{2.65}$$

In the *vector-space model* first described by Salton et al. [1975], each $d \in \mathcal{D}$, a sequence of $|d|$ words or items $d = (w_1, w_2, \ldots, w_{|d|})$, is represented as a vector. In the most trivial case, each word[57] w is mapped to an integer by a function u which selects an unique entry in an ordered set $\mathcal{V}$, where $\mathcal{V}$ is the ordered set of all possible words:

$$u : \mathcal{V} \to \mathbb{N}_+. \tag{2.66}$$

For convenience, we can define an auxiliary function $v_{\mathcal{V}}$ which maps an index returned by u to a $|V|$-dimensional sparse vector which has only a non-zero (1 by default) entry at the respective component:

$$v_{\mathcal{V}} : \mathbb{N}_+ \to \mathbb{N}_+^{|\mathcal{V}|}. \tag{2.67}$$

With this we can define a function ϕ which maps a document d to a $|V|$-dimensional vector representation, which sums over the words in the document, effectively counting the occurrences of w in d:

$$\phi_{u,v}(d) = \sum_{i=1}^{|d|} v(u(d_i)). \tag{2.68}$$

More general, we refer to the function ϕ as a *feature map*, mapping from the set of documents to a high-dimensional vector representation:

$$\phi : \mathcal{D} \to \mathbb{R}^k = \mathfrak{S}. \tag{2.69}$$

Note that, since ϕ could also define a more complex map than in the example defined here, the codomain is in the real numbers and the exponent is generalized

[57] Here, intuitively in the sense of a single entry in a word-based vocabulary, or shorter a *type*.

to an arbitrary integer c, in the example $c = |\mathcal{V}|$. We refer to codomain of ϕ as the *feature space* $\mathfrak{S}$, in which all inputs (here: documents) will be represented.

Each component in the feature space corresponds to a distinct feature. Given a concrete *feature vector* for a document, each component of that vector corresponds to a *feature value* for each feature. In the trivial case discussed here, each component i of a feature vector $\mathbf{d}$ of a document d is the count of the word with index i in $\mathcal{V}$ of d. Since queries use the same vocabulary $\mathcal{V}$, they can also be represented in the same feature space using the same joint feature map.

Given the vector representations of a query $\mathbf{q}$ and documents $\mathbf{d}\ \forall\ d \in \mathcal{D}$, a simple ranking can for example be derived from the angle θ between the query and each document:

$$\cos(\theta) = \frac{\langle \mathbf{q}, \mathbf{d} \rangle}{|\mathbf{q}|_2 |\mathbf{d}|_2} = \frac{\sum_{i=1}^{|V|} \mathbf{q}_i \mathbf{d}_i}{\sqrt{\sum_{i=1}^{|V|} \mathbf{q}_i^2} \sqrt{\sum_{i=1}^{|V|} \mathbf{d}_i^2}}, \tag{2.70}$$

where $\langle \cdot, \cdot \rangle$ is a dot product[58] and $|\cdot|_2$ is the ℓ_2 norm of a vector. Cosine similarity always assigns non-zero scores except when vectors are orthogonal (i.e. when they are not sharing any vocabulary) or one of them is the zero-vector $\mathbf{0}$.

A query $\mathbf{q}$ can then be compared to each representation $\mathbf{d}$ in the document collection. The result is a vector of scalars, which represents the ranking under this simple model. By sorting this vector one can then establish an ordering of the documents.

It is worth to note that in a real world application, instead of scoring all documents in $\mathcal{D}$, only a subset of documents $\mathcal{D}' \in \mathcal{D}$ will be actually scored, e.g. by checking principal compatibility between the query and the set of documents. This has however no consequence for the overall approach.

2.6.1.2 Ranking Measures

Our formal description of a simple document retrieval system is not yet complete. The goodness of ranking functions, can only be assessed in reference to a *gold-standard*, which is typically encoded as *relevance labels* or ranks (e.g. an ordinal labels from $1 \ldots n$) of the documents with respect to each query.

Apart from a simple comparison of the top-ranked object, the comparison between two rankings is not trivial. Therefore a number of measures have been

[58] This requires that the feature space is an inner product space, which R^k is for all $k > 1$.

suggested. In the following we will describe several basic approaches.

The fundamental ratios precision, recall and the derived F_1 score provide means to compare two rankings on the set-level, thus ignoring ordering information:

$$\text{Precision} = \frac{|\text{Relevant Documents} \cap \text{Retrieved Documents}|}{|\text{Retrieved Documents}|} \tag{2.71}$$

$$\text{Recall} = \frac{|\text{Relevant Documents} \cap \text{Retrieved Documents}|}{|\text{Relevant Documents}|} \tag{2.72}$$

$$F_1 = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.73}$$

Measuring of precision and recall is more informative when calculated at defined cutoffs of the retrieved documents, taking for example only the 10-best scoring documents into account. This is referred to as Precision@k and Recall@k.

Average precision further generalizes Precicion@k, averaging it over every cutoff k in the ranking:

$$\text{Average Precision} = \frac{\sum_{k=1}^{K} \text{Precision@}k \times \delta_k}{\min(\text{Relevant Documents}, \text{Retrieved Documents})}, \tag{2.74}$$

where δ_k is 1 if the document at rank k is relevant according to the gold-standard, and 0 otherwise.

Mean average precision (MAP) takes the arithmetic mean of average precisions over a number of queries

$$\text{MAP} = \sum_{r=1}^{|Q|} \text{Average Precision}_r. \tag{2.75}$$

Another class of measures for evaluating rankings are rank correlation coefficients, which try to directly measure the similarity of two rankings of the same objects. A widely used method is Kendall's τ, which reduces the problem of comparing rankings to comparing only pairs of true relevance labels (e.g. $1 \dots n$) to the output of the to be evaluated ranking function. Assume that the true labels, and the matching outputs of the ranking function are collected in vectors for n documents, $\mathbf{y}$ and $\mathbf{x}$ respectively:

$$\mathbf{y} = (y_1, y_2, \dots, y_n) \tag{2.76}$$

$$\mathbf{x} = (x_1, x_2, \dots, x_n), \tag{2.77}$$

where entries with the same value refer to the same document, and the rank is given by the index of the component. For this definition, let the gold-standard be $\mathbf{y} = (1, \ldots, n)$, i.e. the value is identical to the rank for the gold-standard. We can then count the number of concordant and discordant pairs by summing over all $\binom{n}{2} = \frac{1}{2}n(n-1)$ possible $[(x_i, y_i), (x_j, y_j)]$ pairings by:

$$\text{Concordant} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \delta_{x_i < y_j}, \tag{2.78}$$

and

$$\text{Discordant} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \delta_{x_i > y_j}, \tag{2.79}$$

where $\delta_{[\text{condition}]}$ is 1 if the condition is true, and 0 otherwise.

Kendall's τ can now simply be calculated[59]:

$$\tau = \frac{\text{Concordant} - \text{Discordant}}{\text{Concordant} + \text{Discordant}}. \tag{2.80}$$

In contrast to most metrics in MT, some of these metrics can be optimized directly, see e.g. [Le and Smola, 2007], [Yue et al., 2007], or [Xu and Li, 2007].

2.6.1.3 Loss Functions and Learning

The cosine similarity ranking function as formulated above has several major drawbacks:

1. It is a static function, which only relies on geometric concepts;
2. In particular, all features in the input space have the same importance, weighting of features is only statically possible;
3. There is no relation at all to the gold-standard measure.

These drawbacks can be resolved by using a function learned from data.
In this work we focus on linear functions[60]:

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle, \tag{2.81}$$

[59] Assuming there are no ties in the data.
[60] Note that biases are omitted in all formulations. If a bias is needed, the first component of data points $\mathbf{x}$ is 1, and the first component of $\mathbf{w}$ is the associated bias.

where $\mathbf{w} \in \mathfrak{S}$ is a learned *weight vector*, which defines a linear transformation for any $\mathbf{x} \in \mathfrak{S}$ to scalar values in $\mathbb{R}$.

To establish our learning framework we first need to make some additional definitions. Let $\mathbf{X}$ be an $m \times c$ matrix of stacked feature (row) vectors drawn from $\mathfrak{S}$, called the *training set*, and let $Y \in \mathbb{R}^c$ be a column vector of gold standard labels. $\mathbf{X}$ replicates the feature vectors of $\mathcal{D}$ $|\mathcal{Q}|$-times, that is $m = |\mathcal{D}||\mathcal{Q}|$. $\mathbf{Y}$ are the relevance labels for all $\mathbf{x} \in \mathbf{X}$ with respect to each query. The feature vectors of documents for each query are constructed by a joint feature mapping $\phi(d, q)$, e.g. calculating cosine similarity and other features for the query-document pair[61]. $\mathbf{X}^{(q,j)} \in \mathfrak{S}$ selects the jth feature vector of the ranking for the qth query, and accordingly $\mathbf{Y}^{(q,j)}$ the respective relevance score. We further assume that the $|\mathcal{Q}|$ replications of the feature vectors $\mathbf{X}^{(q,\cdot)}$ are sorted in descending order according to the gold-standard scores in $\mathbf{Y}^{(q,\cdot)}$ for all $q = 1, \cdots, |Q|$. $\mathbf{X}^{(q,*)}$ and $Y^{(q,*)}$ selects all vectors and scores for query q.

By contrasting the concrete implementation of the two proposed ranking functions it becomes apparent that cosine similarity:

$$\frac{\langle \mathbf{X}^{(q,i)}, \mathbf{X}^{(q,j)} \rangle}{|\mathbf{X}^{(q,i)}|_2 |\mathbf{X}^{(q,j)}|_2} \; \forall \; q = 1, \ldots, |\mathcal{Q}|, \; i = 1, \ldots, |\mathcal{D}|, \; j = i + 1, \ldots, |\mathcal{D}| \tag{2.82}$$

is static[62] compared to the linear function:

$$\left\langle \mathbf{w}, \mathbf{X}^{(q,i)} \right\rangle + b, \; \forall \; q = 1, \ldots, |\mathcal{Q}|, \; i = 1, \ldots, |\mathcal{D}|. \tag{2.83}$$

The bias b can be dropped in ranking for a single query since it is not relevant for the ordering of the output:

$$f(\mathbf{X}^{(q,*)}) = \left\langle \mathbf{w}, \mathbf{X}^{(q,*)} \right\rangle. \tag{2.84}$$

But how can we learn $\mathbf{w}$? We first define a *loss function* which quantifies the amount of error of the function $f(\cdot)$ with respect to the gold-standard scores/ranks in $\mathbf{Y}$:

$$\begin{aligned} L(f(\mathbf{X}), \mathbf{Y}) &= \sum_{q=1}^{|Q|} L(f(\mathbf{X}^{(q,*)}), \mathbf{Y}^{(q,*)}) \\ &= \sum_{q=1}^{|\mathcal{Q}|} \sum_{i=1}^{|\mathcal{D}|} L(f(\mathbf{X}^{(q,i)}), \mathbf{Y}^{(q,i)}). \end{aligned} \tag{2.85}$$

[61] As our application is information retrieval, the concrete construction of feature vectors in the case of information retrieval is out of the scope of this work.

[62] In the sense of being **only** dependent on the data.

This function can be derived from an arbitrary ranking measure as previously introduced, e.g. $(1 - \tau)$ for Kendall's τ, or $(1 - \text{MAP})$ for mean average precision. As L is directly related to the actual ranking measure that the ranking function will be evaluated with, L is referred to as the true loss.

By summing over the full training set we arrive at the *empirical risk* for the ranking function f:

$$R(f) = \frac{1}{|\mathcal{Q}|} L\left(f(\mathbf{X}^{(q,*)}), \mathbf{Y}^{(q,*)}\right). \tag{2.86}$$

With this global loss formulation we should be able to formulate our learning procedure. However, as $R(\cdot)$ is not continuous[63], it is not differentiable. Thus we would have to resort on direct error minimization, which is generally hard to do and very problem specific. We may however define a differentiable surrogate loss function $l(\cdot)$ to define a risk that can be minimized by gradient descent:

$$R'(f) = \frac{1}{|\mathcal{Q}|} \sum_{q=1}^{|\mathcal{Q}|} l\left(f(\mathbf{X}^{(q,*)}), \mathbf{Y}^{(q,*)}\right). \tag{2.87}$$

Three different types of loss functions have been established for ranking problems:

- Pointwise, defining a loss for any object in isolation: $l_{\text{pointwise}}((\mathbf{X}^{(q,i)},\ \mathbf{Y}^{(q,i)}))$;
- Listwise, defining a loss for full lists: $l_{\text{listwise}}(\mathbf{X}^{(q,*)},\ \mathbf{Y}^{(q,*)})$;
- And pairwise, where the loss is defined by contrasting rankings of elements of the same query, e.g. all pairs:

$$l_{\text{pairwise}}((\mathbf{X}^{(q,i)}, \mathbf{Y}^{(q,i)}), (\mathbf{X}^{(q,j)}, \mathbf{Y}^{(q,j)})),$$
$$\forall\ q = 1, \ldots, |Q|, |\ \ i = 1, \ldots, |D|,\ \ j = i + 1, \ldots, |D|.$$

Pointwise Loss

Pointwise losses are defined on single examples, i.e. triples of feature vector, score of the ranking function, and gold-standard label or rank:

$$(\mathbf{X}^{(q,i)}, f(\mathbf{X}^{(q,i)}), \mathbf{Y}^{(q,i)}). \tag{2.88}$$

A loss for pointwise ranking algorithms can simply be defined in terms of the difference of predicted and true relevance label:

$$l_{\text{pointwise}} = \frac{1}{2} \sum_{q=1}^{|\mathcal{Q}|} \sum_{i=1}^{|\mathcal{D}|} (f(\mathbf{X}^{(q,i)}) - \mathbf{Y}^{(q,i)})^2, \tag{2.89}$$

[63] The ranking function can only be evaluated for the data points that can be processed by the feature mapping.

which is just the mean squared error between the prediction of $f(\cdot)$ and the true label. Plugging in our proposed model we obtain:

$$l_{\text{pointwise}}(\mathbf{w}) = \frac{1}{2}\sum_{q=1}^{|Q|}\sum_{i=1}^{|D|}(\langle \mathbf{w}, \mathbf{X}^{(q,i)}\rangle - \mathbf{Y}^{(q,i)}). \tag{2.90}$$

As this is a simple regression problem, an optimal $\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} l_{\text{pointwise}}(\mathbf{w})$ can be found through gradient descent:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla l_{\text{pointwise}}(\mathbf{w}), \tag{2.91}$$

with

$$\nabla L_{\text{pointwise}} = \left((\mathbf{X}\mathbf{w} - \mathbf{Y})^T\mathbf{X}\right)^T, \tag{2.92}$$

for the batch update, or on a per-example basis as stochastic gradient descent:

$$\nabla l_{\text{pointwise}} = (\langle \mathbf{X}^{(q,i)}, \mathbf{w}\rangle - \mathbf{Y}^{(q,i)})\mathbf{X}^{(q,i)}. \tag{2.93}$$

This loss can be used when $\mathbf{Y}$ are ordinal ranks or relevance labels. In former case the underlying problem is referred to as ordinal regression.

Another possibility to learn a linear ranking function in the pointwise approach is to use a variant of the perceptron algorithm. The perceptron as a classical binary classification algorithm, assumes that there are two classes, $+1$ and -1, and it seeks to learn a function f that is able to discriminate between data points belonging to different classes. The perceptron is a straight-forward implementation of such a function:

$$f(\mathbf{x}) = \text{sign}\left(\langle \mathbf{w}, \mathbf{x}\rangle\right), \tag{2.94}$$

where $\text{sign}(z)$ is -1 if $z \leq 0$, and $+1$ if $z > 0$. Geometrically, $\mathbf{w} \in \mathbb{R}^c$ can now be interpreted as dividing $\mathbb{R}^c$ in two half-spaces by defining an hyperplane through $\langle \mathbf{w}, \mathbf{x}\rangle + b$, or through the origin $[0]^n$ when omitting the bias b as noted before.

We can formulate it in the context of empirical risk minimization as to minimize a *hinge loss*:

$$l_{\text{hinge}} = \left((-\langle \mathbf{w}, \mathbf{x}\rangle)y\right)_+, \tag{2.95}$$

where $(\cdot)_+ = \max(0, \cdot)$. The (sub-)gradient[64] of this loss is very simple:

$$\nabla l_{\text{hinge}} = \begin{cases} -\mathbf{x}y & \text{if } \langle \mathbf{x}, \mathbf{w}\rangle y \leq 0, \\ 0 & \text{else.} \end{cases} \tag{2.96}$$

[64] l_{hinge} is not differentiable at 0.

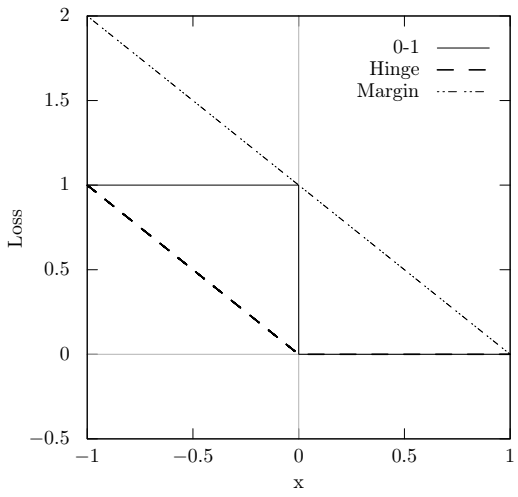

Figure 2.7: Two hinge losses and the zero-one loss.

Again, the loss can be minimized with stochastic gradient descent.

A related loss function, that can be interpreted as ensuring a minimum margin (1.0 in this case) between the nearest classified examples and the hyperplane, can be defined by a small modification to the original hinge loss:

$$l_{\text{margin}} = ((1 - \langle \mathbf{x}, \mathbf{w} \rangle) y)_+ , \tag{2.97}$$

with the subgradient:

$$\nabla l_{\text{margin}} = \begin{cases} -\mathbf{x}y & \text{if } \langle \mathbf{x}, \mathbf{w} \rangle y \leq 1.0, \\ 0 & \text{else.} \end{cases} \tag{2.98}$$

Both variants of the hinge loss, as well as the zero-one loss are contrasted in Figure 2.7.

The *PRank* algorithm is a straightforward application of the perceptron for learning a linear pointwise ranking function: By using a shared weight vector $\mathbf{w}$ and $r - 1$ separate biases b_i ($\mathbf{b}$) it learns to separate r ranks or relevance levels, represented as integers. Effectively, the algorithm learns $r - 1$ parallel hyperplanes, separating the space in r areas. Given a learned model and an ordered set of ranks $1, \ldots, r$, the prediction is as follows:

$$\hat{y} = \min_{i \in 1,\ldots,r} \{ i : \langle \mathbf{w}, \mathbf{x} \rangle - b_i < 0 \} . \tag{2.99}$$

Learning is reduced to the binary classification case: The plane defined by the bias that corresponds to the true label y (with index t), b_t, divides the space into two half-spaces labeled $+1 \; \forall \; y' \leq y$ and $-1 \; \forall \; y' > y$.

Iff the prediction is incorrect ($y \neq \hat{y}$), i.e.

$$(\langle \mathbf{w}, \mathbf{x} \rangle - b_i) \, y' \leq 0 \tag{2.100}$$

a standard perceptron update is performed. Effectively the weight vector $\mathbf{w}$ is updated $r = |\hat{y} - y|$ times by:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta r \, (-\mathbf{x}) \, , \tag{2.101}$$

as well as all biases $b_{i=1,\ldots,r-1}$ corresponding to incorrect classifications.

Listwise Loss

In the listwise approach, in contrast to the previously described pointwise approach, the loss function is defined on complete rankings, i.e. ordered lists of feature vectors, instead of single points. An algorithm that implements this approach is ListNet. The fundamental difference between the two approaches manifests itself in the generic listwise loss function:

$$l_{\text{listwise}} = \sum_{q=1}^{|\mathcal{Q}|} l(\mathbf{Y}^{(q,*)}, [f(X^{(q,1)}, \ldots, f(\mathbf{X}^{(q,|D|)}]^T), \tag{2.102}$$

which is defined on full vectors of labels and scores. ListNet is a probabilistic approach, defining probability distributions over possible rankings (or more general permutations) π for a given set of documents D, and a single query q:

$$P_q = \prod_{i=1}^{|\mathcal{D}|} \frac{\exp(g(\pi_{(q,i)}))}{\sum\limits_{j=i}^{|\mathcal{D}|} \exp(g(\pi_{(q,j)}))}, \tag{2.103}$$

where $g(\cdot)$ is the real-valued gold-standard relevance score[65], and $\pi_{(q,i)}$ selects the feature vector of the document ranked at ith position in the permutation π with respect to a query q. Similarly, another distribution S_q can be defined in terms of the linear ranking function $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$:

$$S_q = \prod_{i=1}^{|\mathcal{D}|} \frac{\exp(f(\mathbf{X}^{(q,i)})}{\sum\limits_{j=i}^{|\mathcal{D}|} \exp\left(f(\mathbf{X}^{(q,j)})\right)}. \tag{2.104}$$

[65] Note that this approach needs to be slightly altered to work with ordinal rankings.

We now can define a loss capturing the difference between the gold-standard ranking and the model's output, e.g. the *cross-entropy* loss:

$$l_{\text{Xentropy}} = \sum_{q=1}^{|\mathcal{Q}|} - \sum_{i=1}^{|\mathcal{D}|} P_q(\pi_{(q,i)}) \log(S_q(\pi_{(q,i)})), \tag{2.105}$$

where the permutation π is given by the gold-standard. P_q and S_q are probability distributions defined over lists (with respect to queries $q \in \mathcal{Q}$), as defined before.

Finally, the gradient of the cross-entropy loss for a single query q and with respect to the parameters of our the linear ranking function is (with j fixed to 1 in the denominators of Equations 2.103 and 2.104):

$$\nabla l_{\text{Xentropy}} = -\sum_{i=1}^{|\mathcal{D}|} P(\pi_{(q,i)}) \mathbf{X}^{(q,i)} + \frac{1}{\sum_{i=1}^{|\mathcal{D}|} \exp\left(f(\mathbf{X}^{(q,i)})\right)} \sum_{i=1}^{|\mathcal{D}|} \exp\left(f(\mathbf{X}^{(q,i)})\right) \mathbf{X}^{(q,i)}. \tag{2.106}$$

For each list in the training data the parameters are learned with SGD:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla l_{\text{Xentropy}}. \tag{2.107}$$

Pairwise Loss

Instead of trying to define a meaningful loss on single items or fully ranked lists, the third approach is to only consider relative rankings between pairs of items, hence the pairwise approach.

Consider a indexed set of documents $\mathcal{D} = \{d\}$ ranked by a gold-standard function g (excluding ties) with respect to a set of queries $\mathcal{Q}$:

$$d^{(q,i)} \succ_g d^{(q,j)} \; \forall i, j \in \mathcal{D}, i \neq j \wedge \forall q \in \mathcal{Q}, \tag{2.108}$$

where $a \succ_g b$ denotes a transitive greater than relation between a and b as given by $g(\cdot)$.

A set of pairs of relative rankings can be extracted directly from list:

$$\left\{ \left[d^{(q,1)}, d^{(q,2)}\right], \left[d^{(q,2)}, d^{(q,3)}\right], \ldots, \left[d^{(q,|\mathcal{D}|-2)}, d^{(q,|\mathcal{D}|-1)}\right], \ldots, \left[d^{(q,|\mathcal{D}|-1)}, d^{(q,|\mathcal{D}|)}\right] \right\}, \tag{2.109}$$

i.e. the set $\mathcal{P}$:

$$\mathcal{P} = \left\{ [d^{(q,i)}, d^{(q,j)}] \,\middle|\, d^{(q,i)} \succ_g d^{(q,j)} \right\}. \tag{2.110}$$

Due to the transitivity of the relation $\succ_g$, the set $\mathcal{P}$ truthfully represents the full original ranking.

For a linear learning problem, the to be learned ranking function f should ideally fulfill the following inequalities:

$$\begin{aligned} d^{(q,i)} \succ_g d^{(q,j)} \Leftrightarrow & \quad f(\phi(d^{(q,i)})) > f(\phi(d^{(q,j)})) \\ \Leftrightarrow & \quad \langle \mathbf{w}, \phi(d^{(q,i)}) \rangle > \langle \mathbf{w}, \phi(d^{(q,j)}) \rangle \\ \Leftrightarrow & \quad \langle \mathbf{w}, \phi(d^{(q,i)}) \rangle - \langle \mathbf{w}, \phi(d^{(q,j)}) \rangle > 0 \\ \Leftrightarrow & \quad \langle \mathbf{w}, \phi(d^{(q,i)}) - \phi(d^{(q,j)}) \rangle > 0 \\ \Leftrightarrow & \quad \langle \mathbf{w}, \mathbf{X}^{(q,i)} - \mathbf{X}^{(q,j)} \rangle > 0. \end{aligned} \tag{2.111}$$

The last line of Equation 2.111 provides us with an apparent solution for finding an optimal linear function f parametrized by a weight vector $\mathbf{w}$: As f merely needs to ensure that the difference vector between two related feature vectors results in a margin > 0, the problem can be cast as a one-class[66] classification problem.

Again, we can define a minimization problem in terms of a hinge loss, in this case without requiring a label[67]:

$$l_{\text{pairwise}} = \left(-\langle \mathbf{x}, \mathbf{w} \rangle\right)_+ , \tag{2.112}$$

with the subgradient:

$$\nabla l_{\text{pairwise}} = \begin{cases} -\mathbf{x} & \text{if } \langle \mathbf{x}, \mathbf{w} \rangle \leq 0, \\ 0 & \text{else.} \end{cases} \tag{2.113}$$

Or with a margin:

$$l_{\text{pairwise/margin}} = \left(1 - \langle \mathbf{x}, \mathbf{w} \rangle\right)_+ , \tag{2.114}$$

with the subgradient:

$$\nabla l_{\text{pairwise/margin}} = \begin{cases} -\mathbf{x} & \text{if } \langle \mathbf{x}, \mathbf{w} \rangle \leq 1.0, \\ 0 & \text{else.} \end{cases} \tag{2.115}$$

There are many approaches to pairwise ranking: Joachims [2002] uses a SVM to solve the same problem as we described above[68] (Ranking SVM). Herbrich et al.

[66] In most literature this is taken to be a binary classification problem, i.e. by including also $-(\mathbf{X}^{(q,j)} - \mathbf{X}^{(q,i)})$ in $\mathcal{P}$. This however effectively just doubles the amount of data without introducing any new information since $-\langle \mathbf{w}, -(\mathbf{X}^{(q,j)} - \mathbf{X}^{(q,i)}) \rangle = -\langle \mathbf{w}, (\mathbf{X}^{(q,i)} - \mathbf{X}^{(q,j)}) \rangle$.

[67] As the label is always $+1$ in the one-class problem.

[68] As the margin perceptron and the SVM are related [Collobert and Bengio, 2004], our formulations are also very similar.

[1999] use an identical formulation, but apply it to ordinal regression. Burges et al. [2005] present a probabilistic approach, defining a cross-entropy loss on pairs. A combination of a pairwise and a pointwise loss is presented by Sculley [2010], where the pointwise loss is simple linear regression, and the pairwise loss is a hinge loss as presented above.

3 Learning Preferences from Static Reference Translations

> "What I cannot create, I do not understand."
> [Richard Feynman]

In this Section we fully formalize and evaluate our online pairwise ranking approach to parameter optimization (tuning) in SMT, based on the classical perceptron algorithm applied for ranking objectives.

We present a motivation for pairwise ranking and link the theoretical background presented in the previous chapter to our application. The approach to pairwise ranking is thoroughly defined and its challenges are elaborated. Next, we will present the baseline algorithm of our approach, which is subsequently extended up to state-of-the-art performance. Each aspect of our approach and its modifications are empirically evaluated. This covers selection of training data, parallelization, the underlying question of producing gold-standard rankings, regularization, amongst others. Finally, we will present a wide range of experiments on real-world translation tasks to prove the feasibility of the presented algorithms.

First we start discussing the individual challenges that occur in a static mixed-domain environment.

3.1 Learning from Static References

Learning preferences from static reference translations in MT poses a number of unique problems: Since parallel data sets are most commonly concatenated from a range of different data sets, the source segments and their respective translations are most likely generated by a multitude of writers and translators, and cover a number of different domains. However, the statistical models underlying SMT assume that the data is produced from a single distribution. Because of this mismatch, domain adaptation and multi-domain adaptation are an active research topic in machine translation [Cuong and Sima'an, 2018]. Since the concept of a domain in natural language is a complex problem, covering style, jargon, vocabulary and other aspects, domain adaptation is an issue that is ideally reconsidered for every application.

A natural example for this problem is patent translation: Patent applications follow a strict structure and tend to heavily make use of idiomatic expressions, resulting in a controlled use of language which is ideal for the application of the SMT machinery. However, general patent translation systems needs to cover a

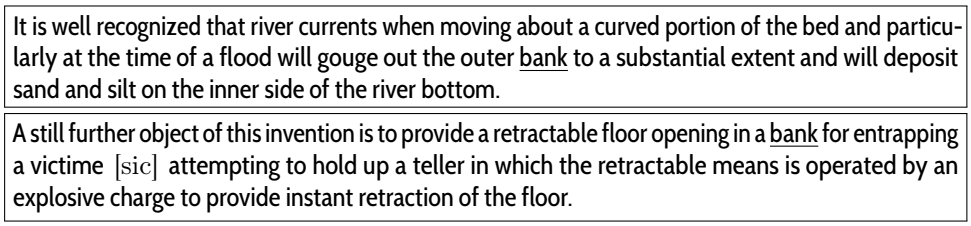

Figure 3.1: Top: Granted US patent US-3386250-A, titled "Water current controlling means", classified under major section E ('Fixed Constructions'); Bottom: Granted US patent US-3313250-A, titled "Trap to prevent robbery of a bank", classified under major section G ('Physics').

wide range of different thematic domains, which can result in very different uses of language, e.g. technical terms or more general jargon. However, to be the best utility for professional translators, it is important to also get these subtle phenomena right. A variation of the classical linguistic example for ambiguity for the word "bank" is shown in Figure 3.1: Without further context, translating from English into German, a generic translation system could possibly not produce produce the preferred translation *Flussufer* (river bank) for the term *bank* in the first sentence.

In traditional SMT, when using a single fixed general domain system, this is a hard problem because of the strong independence assumptions that are imposed during the training of the system and the translation process. From the above example, it becomes quite clear, that when training data is a mixture of several domains, one should prefer not to learn too confident statistics for this particular term, leaving this to specialized systems.

In what follows, we propose a large-scale learning algorithm for SMT which is able to learn robust models that do not try to model such contradicting aspects, but instead reinforce commonalities found within the data which generalize well throughout different sources of source material to be translated. Inherently, the approach is also able to capture fine-grained aspects, which makes it also a suitable algorithm for domain adaptation.

3.2 Learning to Rank for Statistical Machine Translation

The (structured) perceptron algorithm [Rosenblatt, 1958; Collins, 2002] has been well studied in application to discriminative training for SMT: First introduced for reranking by Shen et al. [2004] who propose a splitting algorithm, which effectively performs pairwise ranking between two sets of translations (good and bad) for k-best lists. Watanabe et al. [2006] also perform reranking, but with a structured

perceptron algorithm using BLEU and WER for defining the gold-standard score. Tillmann and Zhang [2006] propose a structured perceptron algorithm for training the parameters of a novel phrase-based translation model. Liang et al. [2006*a*] also employ a structured perceptron variant, and experiment with different strategies for relaxing the condition for the update rule. Finally, Yu et al. [2013] and Zhao et al. [2014] shed further light on this issue by applying the *maximum-violation perceptron* [Huang et al., 2012] to phrase-based and hierarchical SMT. Tian [2015] apply a Plackett-Luce model [Plackett, 1975; Luce, 1959] for learning in SMT. All these approaches have in common that they 1) show competitive test evaluation performance, and 2) they enable to use a much larger feature space than with other algorithms such as MERT, using millions of features instead of a couple of dozens.

The perceptron algorithm enables efficient, large-scale discriminative training for SMT, with arbitrary types and numbers of possibly overlapping features. This gives an incentive to train on very large data sets — which are in turn also needed for estimating meaningful weights for sparse, lexicalized features.

Pairwise ranking is attractive since it enables to use large parts of the search space of SMT system for learning, potentially more than with some of the structured prediction approaches, which consider only a single pair of translations per k-best list. Additionally, since the pairwise preferences are independently processed, training data selection can be done effectively. Also, since the larger part of the search space consists of inadequate and/or disfluent translations, pairwise ranking can make the best use of the available data, by leveraging pairwise differences. Finally, pairwise ranking as proposed here is trivial to implement.

3.2.1 Pairwise Ranking for Statistical Machine Translation

For the applications discussed in this thesis[1], pointwise and listwise ranking methods try to either solve a too simple or a too complex problem. To understand why we came to this conclusion let us first discuss some idiosyncrasies of the MT approach. In principle, the MT system ranks translations according to a simple linear function, i.e. a dot product $\hat{\mathbf{e}} = \arg\max_{\mathbf{e}} \langle \mathbf{w}, \mathbf{e} \rangle$ in a feature space. This allows to directly use the weight vector learned by techniques as described in the previous sections. In translation, one seeks for the best translation candidate using this decision function in a search space. As the space of possible translations given an input string is exponential, the search heavily relies on pruning. For translation, training, and evaluating MT systems, so-called k-best lists are often used, which are (approximated, after pruning) lists of the top k translations according to the linear decision function. As k is typically small[2] in contrast to the full search

[1] Which is ranking lists of similar translation candidates.
[2] Ranging from a few hundred to a few thousand items in most works.

space, a listwise-based optimization would solve an overly complex problem in this setup, which is hardly relevant, since the entries of the k-best lists are subject to significant changes every time $\mathbf{w}$ changes[3], since the SMT search space is defined by $\mathbf{w}$.

Additionally, the task setup is significantly different in SMT compared to IR: Instead of a static list of documents where each item is projected to a feature space according to compatibility to a query, in SMT the search space is built up from partial hypotheses which are compared and pruned if necessary.

Finally, due to the Viterbi approximation used during search, the feature representation of otherwise identical strings may be different, as the arg max operation only ranges over structures, not strings. As a consequence, a pointwise ranking approach will have difficulties learning from these examples, as they will have identical gold-standard scores and thus identical ranks[4].

We thus concentrate our efforts on the most well suited pairwise approach.

For the sake of completeness we now define the pairwise ranking task for SMT tuning.

The principal goal of pairwise ranking for SMT is to render the following conditions true for all possible pairs of translations e with feature representations $\mathbf{x} = \phi(f, e, h)$, h being the derivation, that can be generated given a source segment f. We denote this set of translations as $\mathcal{Y}(f)$, which in our work is a list of k-best translations[5]. We have for any pair of translations $i, j \in \mathcal{Y}(f),\ i \neq j$, where translation $i = (1)$ is preferred over translation $j = (2)$:

$$\begin{aligned} g(\mathbf{x}^{(1)}) > g(\mathbf{x}^{(2)}) &\Leftrightarrow f(\mathbf{x}^{(1)}) > f(\mathbf{x}^{(2)}) \\ &\Leftrightarrow f(\mathbf{x}^{(1)}) - f(\mathbf{x}^{(2)}) &&> 0 \\ &\Leftrightarrow \langle \mathbf{w}, \mathbf{x}^{(1)} \rangle - \langle \mathbf{w}, \mathbf{x}^{(2)} \rangle &&> 0 \\ &\Leftrightarrow \langle \mathbf{w}, \underbrace{(\mathbf{x}^{(1)} - \mathbf{x}^{(2)})}_{=\bar{\mathbf{x}}} \rangle &&> 0 \\ &\Leftrightarrow \langle \mathbf{w}, \bar{\mathbf{x}} \rangle &&> 0, \end{aligned} \tag{3.1}$$

where $f(\cdot)$ is the linear model parametrized by a weight vector $\mathbf{w}$, and $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x} \cdot \mathbf{y} = \sum_i \mathbf{x}_i \mathbf{y}_i$ is a dot product between feature representations $\mathbf{x}$ generated by the feature map $\phi(\cdot)$. The function $g(\cdot)$ maps arbitrary translations (in isolation) to a real valued score, e.g. commonly per-sentence-bleu($\mathbf{x}^{(1)}$) > per-sentence-bleu($\mathbf{x}^{(2)}$) given an implicit reference translation. Another obvious choice would be $-$TER.

[3] However, Chen et al. [2017] still describe a successful application of a tuning method based on listwise ranking methods to SMT.

[4] This is also true for other forms of ties.

[5] Note that this is in contrast to the previous presentation in the context of IR, as $\mathcal{Y}(f)$ generates possibly disjoint sets of translations for any two sufficiently different sentences f, which is why there is no global list of items and also no dependency on some form of query.

With the shorthand $\overline{\mathbf{x}} = \mathbf{x}^{(1)} - \mathbf{x}^{(2)}$ it becomes evident that the problem is equivalent to a binary classification perceptron with only positive examples[6]. This way, the problem becomes a one-class classification problem.

If $\langle \mathbf{w}, \overline{\mathbf{x}} \rangle$ is ≤ 0 the model score predicts the wrong order of the translations, and the model needs to be updated. The resulting weight vector $\mathbf{w}$ can be readily used as the (log-)linear weights of the SMT system, since $\hat{e} = \arg\max_e \langle \mathbf{w}, \phi(f, e, h) \rangle$.

This is an equivalent formulation to the pairwise ranking problem as discussed before in the context of IR. It is also equivalent to a pairwise hinge loss (shown for a single training example):

$$l_j(\mathbf{w}) = (-\langle \mathbf{w}, \overline{\mathbf{x}}_j \rangle)_+ = \max(0, -\mathbf{w} \cdot \overline{\mathbf{x}}_j). \tag{3.2}$$

We will be minimizing the empirical risk of a training or development set (comprised of n sentences) to find suitable weights $\mathbf{w}$:

$$R_{\text{empirical}}(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^{n} l_i(\mathbf{w}; \overline{\mathbf{x}}_i). \tag{3.3}$$

For SMT however, we generate a set of pairwise preference pairs $\mathcal{P}_i$, from the possible set of translations $\mathcal{Y}(f_i)$ using a function Q, which depends on the gold-standard score:

$$\mathcal{P}_i = Q_g(\mathcal{Y}(f_i)). \tag{3.4}$$

Including this, the empirical risk formulation becomes:

$$R_{\text{dtrain}}(\mathbf{w}) = \frac{1}{\sum_i |\mathcal{P}_i|} \sum_{i=1}^{n} \sum_{j=1}^{|\mathcal{P}_i|} l_{i,j}(\mathbf{w}; \overline{\mathbf{x}}_{i,j}). \tag{3.5}$$

The overall loss can accordingly be written as follows:

$$L_{\text{dtrain}} = \sum_{i=1}^{n} \sum_{j=1}^{|\mathcal{P}_i|} l_{i,j}(\mathbf{w}; \overline{\mathbf{x}}_{i,j}). \tag{3.6}$$

The objective function is thus:

$$H(\mathbf{w}) = 1 / L_{\text{dtrain}}(\mathbf{w}). \tag{3.7}$$

There are several other loss functions that could be used here, e.g. we may introduce the large-margin principle to take correctly ranked examples with insufficient

[6] In the actual binary case we would have $\langle \mathbf{w}, \mathbf{x} \rangle y > 0$, y being a class label $\in \{-1, 1\}$.

margins into account.

The subgradient of the hinge loss is[7]:

$$\nabla l_j(\mathbf{w}) = \begin{cases} -\bar{\mathbf{x}}_j & \text{if } \langle \mathbf{w}, \bar{\mathbf{x}}_j \rangle \leq 0, \\ 0 & \text{otherwise,} \end{cases} \tag{3.8}$$

which is exactly the one-class perceptron algorithm.

This is in principle a convex[8] optimization problem which may be approached with batch subgradient descent:

$$\mathbf{w}' \leftarrow \mathbf{w} - \eta \frac{1}{\sum_i |\mathcal{P}_i|} \sum_{i=1}^{n} \sum_{j=1}^{|\mathcal{P}_i|} \nabla l_{i,j}(\mathbf{w}), \tag{3.9}$$

or iteratively by stochastic subgradient descent:

$$\mathbf{w}' \leftarrow \mathbf{w} - \eta \nabla l_j(\mathbf{w}), \tag{3.10}$$

for all pairs j.

3.3 Baseline Algorithm

A straightforward implementation of our proposed approach is depicted in Algorithm 3: Given parallel data, a pair generation algorithm, a gold-standard ranking function as discussed previously in Section 3.8, a learning rate, and the number of epochs for SGD, the algorithm processes pairs generated from the decoder's output in a strictly iterative way. After the last step the final weight vector is returned, which can then be directly used for decoding.

Although the algorithm itself is simple, there are some implications that need clarification in the context of SMT:

Convexity Similar to learning in speech recognition, although the loss function is convex, the overall learning problem is not, since the data used for learning ($\mathcal{Y}(f)$) actually directly depends[9] on the learned function. Thus, every solution on an approximated space, only represents a local minimum with respect to the full search space. Specifically, due to the use of k-best lists, the data not free of search errors and naturally represents only a small part of the overall space of possible translations. However, considering only a fixed data set, the problem is convex for online as well as for batch learning. But note that, in our algorithms,

[7] The hinge loss only has a subgradient as the gradient is undefined at 0.
[8] We will show in the next section why it is actually not a convex problem when applied to SMT.
[9] Also discussed in [Green, 2014, p. 83].

Algorithm 3 Baseline online pairwise ranking algorithm (adapted from [Simianer et al., 2012]). *Inputs:* Number of epochs T, parallel training data $\mathcal{I}$, learning rate η, gold-standard function $g(\cdot)$, pair generation algorithm Q.

procedure DTRAIN
  Initialize $\mathbf{w}_{0,0,0} \leftarrow \mathbf{0}$.
  for epochs $t \leftarrow 0 \dots T-1$: **do**
    for all $i \in \{0 \dots |\mathcal{I}|-1\}$: **do**
      Decode i^{th} input with $\mathbf{w}_{t,i,0}$.
      Generate training examples $\mathcal{P}$ using algorithm Q_g.
      for all pairs $x_j, j \in \{0 \dots |\mathcal{P}|-1\}$: **do**
        $\mathbf{w}_{t,i,j+1} \leftarrow \mathbf{w}_{t,i,j} - \eta \nabla l_j(\mathbf{w}_{t,i,j})$
      end for
      $\mathbf{w}_{t,i+1,0} \leftarrow \mathbf{w}_{t,i,|\mathcal{P}|}$
    end for
    $\mathbf{w}_{t+1,0,0} \leftarrow \mathbf{w}_{t,|\mathcal{I}|,0}$
  end for
  return $\mathbf{w}_{T,|\mathcal{I}|,|\mathcal{P}|}$
end procedure

neither rescoring of the current k-best list nor re-decoding of the current source segment f are performed after weight updates, thus changes in the weights are only recognized after the processing of a full sentence.

Gold-standard The gold-standard function used for learning is significantly different from the actual target metric due to the sentence-level approximation.

Separability There is no guarantee whatsoever that the data used for training the perceptron is linearly separable, i.e. it can not be guaranteed that the proposed algorithms will be able to find a solution in a finite number of steps. Thus only weaker generalization bounds apply [Freund and Schapire, 1999] than the ones available for the perceptron in the separable case.

Despite these issues, the potential advantages of the algorithm are manifold: Perceptron learning enables the use of arbitrary local[10] feature functions in large numbers, as well as the use of large data samples due to the algorithm's inherent efficiency. The linear model $\mathbf{w}$ learned with the perceptron algorithm can also be directly incorporated into the machine translation decoder.

[10] Local in the sense of computable with respect to a single source–translation pair.

Data Set	# Segments
Training	130K
Tuning	1K
Dev. Test	1K
Test	2K

Table 3.1: Statistics for News-commentary (NC) German-to-English data.

Before extending the algorithm and presenting empirical evaluations of its performance, we first present our experimental setup.

3.4 Experimental Setup

For the experiments in this chapter we use a range of data sets for the German-English and Russian-English language pairs, which we describe here for clarity. The used MT system is also described, as well as the basic setup used for all the presented experiments.

3.4.1 Data

Brieftaubenreisevereinigung
[Compound word in German]

To provide reliable empirical results, we use a total of four data sets to evaluate the aspects of the proposed algorithms, considering two diverse translation directions: German-to-English and Russian-to-English. For German-to-English we are using three data sets, for small, medium and large scale experimentation, namely NC, EP, WMT13, and a single data set for Russian-to-English, WMT15. All numbers depicted in the tables are rounded to the nearest thousand, ten thousand, hundred thousand, or million, and abbreviated by K (thousand) and M (million).

The small scale data NC is useful for fast experimentation and verification of hypotheses with a fast turn-around time. The corpus is well studied [Koehn and Monz, 2006*a*] and has been used to show domain adaptation behaviour of various MT systems [Koehn and Schroeder, 2007], *inter-alia*. The data is furthermore attractive for experimentation, since a minimal system trained only using the about 130K parallel segments of training data already produces sensible outputs on the in-domain test sets. The basic statistics of this data are depicted in Table 3.1. The training set is the parallel data distributed for the WMT'11 (Workshop on Machine Translation, 2011) [Callison-Burch et al., 2011*b*] translation task. Tuning, development test and test data are *nc-dev2007*, *nc-devtest2007* and *nc-test2007*

Data Set	# Segments
Training	1.7M
Tuning	2K
Dev. Test	2K
Test_1	2K
Test_2	2K

Table 3.2: Statistics for *Europarl* (EP) German-to-English data.

respectively, from the data provided for the WMT'07 translation task [Callison-Burch et al., 2007]. Three different versions of grammar extractor were used for the experiments with NC data set, which is why there are different baselines. Result tables with different versions are annotated with *, ** or $^{@}$.[11] Note that the results reported on different versions are not comparable.

The medium sized data set for German-to-English is denoted as EP, a collection of speeches given in the European parliament as described by Koehn [2005]. The corpus is also well studied and widely used, and it's training data is particularly interesting since it is a large body of very homogeneous textual data. The training data is a magnitude larger than that available for the small data set NC, as depicted in Table 3.2. We also have two test sets available. The training set is the parallel data distributed for the WMT'11 [Callison-Burch et al., 2011*b*] translation task. Tuning, development test, test_1 and test_2 data are *dev2006*, *devtest2006*, *test2006* and *test2007* respectively, from the data provided for the WMT'07 translation task [Callison-Burch et al., 2007]. Two different versions of grammar extractor were used for the experiments with EP data set, which is why the baselines differ. Result tables with different versions are annotated with * or $^{@}$. Also note that the results reported on different versions are not comparable.

In addition to the in-domain data, we also use another set of development and test sets with EP, which is extracted from the *Common Crawl* data [Smith et al., 2013]. We use a development set for tuning with about 2K segments, and also two test sets, one containing about 2.5K and the other about 3K segments. This data is referred to as CRAWL.

The largest data set we use is denoted as WMT13: The training data is a concatenation of NC, EP and the Common Crawl training data sets. Additional monolingual English data is added from the English Gigaword corpus [Parker et al., 2011] for language modeling. The data was originally distributed for the news translation task described in [Bojar et al., 2013]. For tuning, development test, and test we have each two distinct data sets available. Parallel and monolingual training

[11] The results denoted with $^{@}$ were published in [Simianer et al., 2012].

Data Set	# Segments
Training	4.5M
Train. Mono.	120M
Tuning_S	1K
Tuning_L	10K
Dev. Test_1	.5K
Dev. Test_2	3K
Test_1	3K
Test_2	3K

Table 3.3: Statistics for WMT'13 (WMT13) German-to-English data.

Data Set	# Segments
Training	2M
Tuning	3K
Dev. Test	3K
Test	3K

Table 3.4: Statistics for WMT'15 (WMT15) Russian-to-English data.

data are a concatenation of all data distributed for the WMT'13 translation task [Bojar et al., 2013], as noted above. The monolingual data used for training a language model includes the English side of the parallel data. Tuning_S data is the first half of the *newstest2008* data. Tuning_L is the concatenation of *newstest2008*, *newstest2009*, *newstest2010* and *newstest2011* data. Development test_1 and development test_2 are the *newssyscomb2009* and *newstest2013* sets respectively. The test sets are *newstest2012* and *newstest2014*. All tuning, development test and test sets are distributed for the *WMT'14* translation task [Bojar et al., 2014*b*].

For Russian-to-English, we use the data distributed for the news translation task for [Bojar et al., 2015]. Parallel and monolingual training data is all data provided this task. For tuning we use *newstest2012*, for development test *newstest2013* and for test *newstest2014*.

All data sets presented in this chapter only have a single reference translation per source segment. The data is mostly used as is, without any filtering[12], with the exception of the Common Crawl data for the WMT13 German-to-English data,

[12]Only parallel segments where one of the segments is empty were removed by default.

where the data was filtered by length (maximum of 200 for both source and target) when it is used as tuning data. For training, data is lowercased and tokenized using the scripts distributed with the *Moses* SMT toolkit [Koehn et al., 2007]. When German is the source language, we apply compound splitting either by the methods recommended by [Koehn and Knight, 2003] or Dyer [2009]. For all experiments, tri- or 4-gram language models are used, depending on the size of the data ($N = 3$ for the small data set, $N = 4$ for all others). Language models are estimated either with *lmplz* [Heafield et al., 2013] or *SRILM* [Stolcke, 2002] toolkits using modified Kneser-Ney smoothing [Kneser and Ney, 1995; Chen and Goodman, 1996], and pruning of singleton N-grams as well as backoff interpolation. All language models are binarized using the *kenlm* library [Heafield, 2011].

Tuning is performed on the respective tuning set of the data unless noted otherwise.

3.4.2 Machine Translation Systems

Throughout all experiments we use the implementation of the Hiero approach for SMT [Chiang, 2007] provided within the *cdec* framework [Dyer et al., 2010]. To prevent excess computation while translating, we use a fixed *cube pruning* setting of 200 for rescoring with the language models. Viterbi word alignments in both directions as required for the extraction of grammars [Lopez, 2007] are estimated with either the *GIZA++* toolkit [Och and Ney, 2003] using the wrapper provided with the Moses toolkit. Alignment symmetrization [Liang et al., 2006*b*] is also performed with the Moses toolkit using the *grow-diag-final-and* heuristic. Per-segment grammars[13] and associated features are extracted according to the algorithm proposed by Lopez [2007] with either the original implementation, the implementation described in Baltescu and Blunsom [2014], or the one distributed with cdec. When extracting grammars for the training data, we applied a leave-one-out technique [Zollmann and Sima'an, 2005; Wuebker et al., 2012], excluding the current segment for rule extraction and feature estimation. For each source token, a *pass-through* rule is added to the per-sentence-grammar to enable the translation of unknown source tokens. Finally, a number of *glue rules* are added to the grammars by default, providing re-combination facilities for all possible rule configurations. Two non-terminals X_1 and X_2 are allowed, following Chiang [2007]. The maximal span size for each non-terminal is set to 15 in grammar extraction and for decoding, while the minimum span size is one. Further grammar extraction parameters are: Adjacent non-terminals are disallowed; there may be only five terminal symbols on left- and right-hand side of each rule, and five symbols in total; 300 samples were taken into account for each source phrase.

[13] Compiling all rules applicable to a sentence in a single file, which is in contrast to a global grammar including all possible rules.

All settings as described here apply to all following experiments unless noted otherwise.

3.5 Model Features

We use a number of common features in our model, which are listed below. The features are further grouped into two broad classes:

DENSE: Dense features occur at every hypernode in a translation hypergraph. For each (partial) path their value is the sum of all values in antecedent hypernodes.

SPARSE: Sparse features only apply to a subset of nodes, which is for most features likely $\emptyset$.

3.5.1 Dense Feature Set

Dense features, as defined above, are usually features derived from the generative translation- and language models. The features in the translation model as used in this work are mostly variations of the following quantities [Lopez, 2007]:

- count(f): Absolute count of the occurrence of source phrase f in the whole corpus, or the maximum number of samples.
- count(e): Count of the target phrase e in the whole corpus.
- count(fe): Count of the complete rule, composed of the source phrase f and the target phrase e in the whole corpus. Since the collocations of f are sampled, this count is effectively limited to the number of samples.

The features derived from these quantities are calculated as follows[14] (v_d is the feature's initial weight):

CountEF ($v_d \leftarrow 0.1$):

$$\log_{10}(1 + \text{count(fe)}) \tag{3.11}$$

EgivenFCoherent ($v_d \leftarrow -0.1$):

$$-\log_{10} \frac{\text{count(fe)}}{\text{count(f)}} \tag{3.12}$$

[14] Taken from the implementation described in [Baltescu and Blunsom, 2014], listing adapted from [Karimova et al., 2014].

IsSingletonF ($v_d \leftarrow -0.01$):

$$\begin{cases} 1.0 & \text{if count(f)} = 1 \\ 0.0 & \text{else} \end{cases} \tag{3.13}$$

IsSingletonFE ($v_d \leftarrow -0.01$):

$$\begin{cases} 1.0 & \text{if count(fe)} = 1 \\ 0.0 & \text{else} \end{cases} \tag{3.14}$$

MaxLexFgivenE ($v_d \leftarrow -0.1$):

$$-\sum_{i=1}^{|\mathcal{T}([\text{fe:f}])|} \log_{10} p_{\max}(f_i|e), \tag{3.15}$$

where $\mathcal{T}([\text{fe} : \cdot])$ is the set of terminal symbols in the source or target side of a particular rule, and $p_{\max}$ is the highest translation probability in a lexical distribution given by relative frequency estimation from the symmetrized word alignments.

MaxLexEgivenF ($v_d \leftarrow -0.1$):

$$-\sum_{i=1}^{|T([\text{fe:e}])|} \log_{10} p_{\max}(e_i|f), \tag{3.16}$$

SampleCountF ($v_d \leftarrow -0.1$):

$$\log_{10}(1 + \text{count(f)}) \tag{3.17}$$

Glue ($v_d \leftarrow 0.01$):
Absolute number of usages of glue rules.

PassThrough ($v_d \leftarrow -0.1$):
Absolute number of usages of pass-through rules.

WordPenalty ($v_d \leftarrow -0.1$):
Absolute number of target terminal symbols.

Arity$_{0/1/2}$ ($v_d \leftarrow -0.1$):
Absolute number of rules used with arity[15] 0, 1 or 2.

LanguageModel ($v_d \leftarrow 0.1$):

[15] Corresponds to the number of non-terminals in a rule, i.e. arity-0 rules have no non-terminal symbols.

(1) X → X_1 hat X_2 versprochen | X_1 promised X_2
(2) X → X_1 hat mir X_2 versprochen |
X_1 promised me X_2
(3) X → X_1 versprach X_2 | X_1 promised X_2

Figure 3.2: SCFG rules for translation.

Negative log-likelihood score of the language model.

$LanguageModel_{\text{OOV}}$: ($v_d \leftarrow -1$):
Absolute number of tokens unknown[16] to the language model.

3.5.2 Sparse Feature Set

Sparse features only apply to subset of translations and in search only to a subset of arcs. Thus they should be able to discriminate between different translations, an ideal match for discriminative training. In contrast to e.g. Chiang et al. [2009], who try to find sparse features that cope with very specific phenomena or fix individual problems of the translation system, we seek to find a feature set that can be effectively trained to improve translation quality, without the need for further manual engineering.

To illustrate our proposed features, a sample of three SCFG rules is shown in Figure 3.2.

Rule identifiers (Rule-Id): These features identify each rule by a unique identifier [Blunsom and Osborne, 2008]. Each application of a rule is counted, and the final value of the feature is the sum of its applications in the derivation. Such features roughly correspond to the relative frequencies of rewrites rules used in the dense features described before. Each rule depicted in Figure 3.2 would correspond to a single unique identifier, which is obtained by mapping the rule to a string representation. Combined, these features correspond to the use of a discriminative translation model (grammar).

Rule bigrams (Rule-Bigram): These features identify Bigrams of consecutive items in a rule. We use bigrams on source- and target-sides of rules. Such features identify possible source- and target-side phrases and thus can give preference to rules in- or excluding them.[17] In Figure 3.2, the first rule would fire the following additional features: X_1 hat, hat X_2, and X_2 versprochen on the source-side, and X_1

[16] "Out-of-vocabulary" (OOV) items.
[17] Similar "monolingual parse features" have been used by Dyer et al. [2011].

promised, promised X_2 on the target-side.

Rule shape (Rule-Shape): These features are indicators that abstract away from lexical items by extracting templates that identify the location of sequences of terminal symbols in relation to non-terminal symbols, on both the source- and target-sides of a rule. For example, both rule (1) and (2) in Figure 3.2 map to the same indicator, namely to that of a rule that consists of a (non-terminal, terminal *, non-terminal, terminal *) pattern on its source side, and an (non-terminal, terminal *, non-terminal) pattern on its target side (* denotes zero ore more occurrences). Rule (3) maps to a different template (non-terminal, terminal *, non-terminal), on both source- and target-side.

For some experiments we additionally explore a more syntax-oriented approach for sparse features, or further sparse feature templates. The sparse feature set as described here is denoted by SPARSE, and the dense feature set as described before by DENSE.

3.5.3 Experiments with Features

In a first experiment we explore the feasibility of our SPARSE feature set. The result is depicted in Table 3.5. The largest number of features originates from using the discriminative grammar. Rule bigram features add about 30K features, and rule shapes only account for 51 features. In total, this results in about 180K for the experiment with the full sparse feature set. The best result is achieved combining all features. On the development test data, the DENSE feature set performs best using the small data set for tuning.

Tuning on the full bitext results in a similar picture on the test set, which this time is also resembled by the results on the development test data.

To evaluate our proposed feature sets we compare the DENSE and SPARSE feature sets on all data sets. Results depicted in Tables 3.6 for NC*, 3.7 for EP*, 3.8 for WMT13, and 3.9 WMT15. All results are definitely in favor of the SPARSE feature set, improvements ranging from 0.3 (NC*, Table 3.6) to 1.8 %BLEU (WMT15, Table 3.9). Overall, we observe that tuning with SPARSE features seems to perform better when the underlying training data, used for estimating the generative models, becomes larger.

3.6 Setups for Tuning Methods

In addition to our own work, we compare our efforts to the MERT and MIRA algorithms in implementations we are going to describe below. We refer to our online pairwise ranking tuning method as DTRAIN.

	NC@		
System	Dev. Test	Test	# Features
DENSE	**25.9**	28.0	12
Rule-Id	25.5	†27.6	140K
Rule-Bigram	25.8	†27.4	30K
Rule-Shape	25.9	28.1	51
SPARSE	25.7	**28.2**	180K
DENSE, Bitext	26.1	†27.9	12
Rule-Id, Bitext	26.1	†28.0	3.4M
Rule-Bigram, Bitext	26.3	28.3	330K
Rule-Shape, Bitext	26.4	28.3	51
SPARSE, Bitext	**26.4**	**28.6**	4.7M

Table 3.5: Comparing SPARSE and DENSE feature sets on small-scale NC@ data tuning on a small development (baseline algorithm) set as well as the full bitext (ITERMIXSGD algorithm, cf. Section 3.10): Significance is assessed with an approximate randomization test between experiments in the same group, and significant differences, with $p < 0.05$, to the best result (in bold) are denoted by †. Table adapted from [Simianer et al., 2012].

	NC*	
System	Dev.	Test
DENSE		25.2
SPARSE		**25.5**

Table 3.6: Comparing SPARSE and DENSE feature sets on small-scale NC* data, tuning on a single, small development set.

EP*			
System	Dev. Test	Test_1	Test_2
DENSE	27.8	27.9	28.1
SPARSE	**29.4**	**29.1**	**29.5**

Table 3.7: Comparing SPARSE and DENSE feature sets on medium-scale EP* data, tuning on a single, small development set.

WMT13				
System	Dev. Test_1	Test_1	Dev. Test_2	Test_2
DENSE	18.3	16.7	19.1	19.2
SPARSE	**19.6**	**18.1**	**20.3**	**20.6**

Table 3.8: Comparing SPARSE and DENSE feature sets on large-scale WMT13 data, tuning on a single, small development set (Tuning_S).

WMT15		
System	Dev. Test	Test
DENSE	17.0	20.5
SPARSE	**19.1**	**22.3**
DENSE, Bitext	20.7	25.0
SPARSE, Bitext	**21.9**	**26.1**

Table 3.9: Comparing SPARSE and DENSE feature sets on WMT15 data, tuning on a single, small development set as well as the full bitext.

All methods described in this work rely on k-best lists for approximating the true search space. Throughout this work we use $k = 100$ unique entries. Uniqueness is applied on the string level, always including only the derivation with maximum score.

For all data sets a development test set is used to adjust the various hyperparameters of different methods, reporting scores on test with settings that maximized the score on the respective development test set.

If applicable, all methods use the same amount of epochs[18].

3.6.1 Minimum Error Rate Training

We use an implementation of *hypergraph* MERT, as described by Kumar et al. [2009] which is an adaptation of the lattice-based MERT algorithm [Macherey et al., 2008], but using hypergraphs instead of k-best lists.

An implementation of hypergraph-MERT is provided within the cdec framework. This version requires non-zero initial weights for each feature to be optimized. For our experiments use the weights as depicted as described in Section 3.5.1.

Since MERT uses random initializations, we can account for optimizer instability by repeating the tuning process at least three times, following Clark et al. [2011]. We report the mean scores along with standard deviations if applicable.

3.6.2 Margin-infused Relaxed Algorithm

For the experiments with the MIRA algorithm we use the implementation distributed with cdec which supports a wide range of hyperparameters. This implementation of MIRA uses k-best lists as surrogate for the true search space to select *hope* and *fear* derivations. Hope and fear can be selected by a number of criteria, which we exhaustively explore in our experiments. In one variant we calculate hope and fear as proposed by Chiang [2012] by:

$$\begin{aligned} \text{hope} &= \arg\max_{\cdot} m(\cdot) + g(\cdot) \\ \text{fear} &= \arg\max_{\cdot} m(\cdot) - g(\cdot), \end{aligned} \tag{3.18}$$

where $m(\cdot)$ is the model score and $g(\cdot)$ is the gold-standard score (the higher the better).

In the other variant, hope and fear derivations are simply calculated as $g(\cdot)$ and $-g(\cdot)$ respectively, similar to the local update of Liang et al. [2006*a*]. The gold-standard function is a smoothed per-sentence BLEU [Chiang, 2012].

For MIRA we use the default of $k = 500$ unique translation hypotheses for all experiments. Sentence-level BLEU scores are calculated using a pseudo corpus, as proposed by Chiang [2012], using a decay rate of 0.95.

[18] Epoch referring to a single complete iteration over all training data.

The implementation supports parallelization similar to the *downpour* scheme [Dean et al., 2012], but we use only a single process to obtain deterministic results, since the parallelization resulted in large variance in the results [Simianer et al., 2012]. The algorithm is run for the default of 20 epochs, and final weights are generated by averaging the final weights of each epoch.

We try five different optimizers: SGD, passive aggressive MIRA with selection from cutting plane, cutting plane MIRA, passive-aggressive MIRA, and full MIRA with k-best constraints of hope, fear, and model-best constraints. Optimization is always started from the same initial weights as used for MERT. The weights for the SPARSE feature set are initialized to $\mathbf{0}$.

We always perform a full sweep over step sizes/learning rates over the range $10^{-10} \ldots 1.0$ with a granularity of 10^{-1}.

Optimization is carried out online, with k-best lists of translations re-generated once per each epoch.

3.6.3 Online Discriminative Training with Pairwise Ranking

For all experiments with DTRAIN, the optimization starts from the $\mathbf{0}$ vector. When using a margin, it is fixed at 1.0, and a coarse grid search for an optimal learning rate is performed over the range $10^{-10} \ldots 1.0$ with a step size of 10^{-1}. Without a margin the learning rate is fixed to 1.0. We always use a k-best size of 100 for experiments with DTRAIN, and optimize a smoothed per-sentence BLEU according to Nakov et al. [2012] unless noted otherwise. Examples from all k-best lists are generated by first sorting according to the gold-standard score, then applying a pair extraction algorithm as described in Section 3.8. By default we are using the algorithm depicted in Algorithm 5.

The implementation of the algorithm is online, each segment (excluding the very first segment) is translated with an updated weight vector, unless the data for the previous segment were all correctly classified. The individual updates consist however of the sum of all gradients for a single k-best list. Thus the updates can be considered as mini-batches, since the learning rate applies to a (non-normalized) sum of gradients.

The settings for parallelization are described n the description of the specific experiments.

For evaluation we report %BLEU-4 scores on development test and test data sets. We do not report score on training or tuning sets scores, since scores are calculated on a per-sentence basis. All scores are calculated on lowercased and tokenized data.

	Full		Multipartite	
Training	Test Acc.	Train Err.	Test Acc.	Train Err.
all	97%	0%	100%	0%
100K	90%	0%	96%	0%
10K	74%	0%	85%	0%
1K	64%	0%	71%	0%

Table 3.10: Synthetical experiments using the SPARSE feature set for both full and multipartite settings, reporting accuracies on test data and error rates on training data.

3.7 Experiments with Synthetic Data

To validate our approach and to obliterate the underlying issues of (non-)convexity and linear separability we conduct a series of synthetical experiments: We simulate a classical binary classification problem by considering a fixed set of training examples (pairs), without re-decoding, effectively iterating the innermost loop of Algorithm 3 in a reranking task.

For this experiment we generated training and test data from the Tuning_S development set of the WMT13 data, either with using all pairs or our multipartite scheme (cf. Section 3.8). This resulted in 1.7M pairs in total for the multipartite data, and 4.6M for the full data. For both data settings we conducted 75/25 random splits for generating training and test sets. From the training data we furthermore randomly sampled additional subsets with 100K, 10K, and 1K examples to obtain training sets of diverse sizes. The total number of features is about 110K for both settings.

We trained a perceptron with a hinge loss objective on each data set until convergence, which we define as no observed change in training error for ten consecutive epochs. Training is done for a maximum of one million epochs. The perceptron is effectively a one-class perceptron since all training examples belong to the class +1. Both feature settings are considered: DENSE and SPARSE.

Results of our experiments are depicted in Tables 3.10 and 3.11. Using the SPARSE feature set, all training data splits are perfectly separable, and scaling the training data improves up to 33% accuracy (using the 1K training data as a baseline). The multipartite settings exhibits a test-time advantage of 3%–11% accuracy. With DENSE features, the training data does not seem to be separable under any condition, the error remaining at about 40% when hitting the maximum epoch limit. Test performance is the same for all settings within a 5% range. Multipartite

	Full		Multipartite	
Training	Test Acc.	Train Err.	Test Acc.	Train Err.
all	60%	43%	60%	39%
100K	52%	43%	59%	39%
10K	58%	43%	60%	39%
1K	55%	47%	59%	40%

Table 3.11: Synthetical experiments using the DENSE feature set for both Full and multipartite settings, reporting accuracies on test data and error rates on training data.

data seems to have slight advantages in training error and test accuracy, but only for smaller training data settings.

The main findings are that data appears to be separable when using the SPARSE feature set, but not using only DENSE features, which indicates that our approach may work better using the SPARSE feature set. Furthermore, the multipartite settings seem to show somewhat better performance, selecting only pairs with some margin, without trying to tear apart very close translations.

3.8 Gold-Standard

The translation quality metric which induces the gold-standard ranking is a central aspect of the proposed algorithms. The BLEU metric as introduced by Papineni et al. [2002] is the default choice, since it shows positive correlation with human judgments [Papineni et al., 2002; Coughlin, 2003] and is straight-forward to implement while being easily interpretable due to its simple formalization in terms of averaged of N-gram precisions. It is also mostly language-independent[19], which is important for fast execution and rapid development. Furthermore, Cer et al. [2010] show that the family[20] of BLEU scores is the preferred metric for tuning with MERT.

For a metric to be usable as gold-standard metric in the proposed ranking setup it should have two characteristics: 1) it is evaluable on a per-sentence basis, and 2) fast to evaluate, since it is evaluated n (size of the data set) $\times$ k (number of hypotheses per source segment in the data set) times in a single epoch of training. Metrics based on counting string-level operations converting a hypothesis into the reference translation such as TER satisfy constraint 1), but due to the search

[19] A notable, but for our work dismissed, exception is the question of atomic units which are considered for computing N-grams and their precision.

[20] BLEU-1, BLEU-2, BLEU-3 and BLEU-4 scores.

for the optimal action sequence they are not fast[21] to evaluate. BLEU on the other hand is fast to evaluate, requiring only the generation of N-gram counts and segment lengths. But there is a major caveat — BLEU is designed in such a way that it scores translations at the corpus level [Papineni et al., 2002]. The reason for this is that the N-gram statistics are only likely to be non-zero for all N if more than a single sentence is considered. The problem is evident considering the calculation of the geometric average for the N-gram precision P:

$$P = \left(\prod_{i=1}^{N} Q_i \right)^{\frac{1}{N}}, \tag{3.19}$$

if any Q_i are 0, the whole score will be zero and thus non-informative, even if there are a number of trigram matches for $N = 4$. For training, especially the ranking objectives discussed here, it is important that every hypothesis in the search space has an non-zero, informative score. Intuitively, it would also make sense to be able to calculate a score even though there is no e.g. no matching 4-gram in a candidate with only four words.

To overcome this problem and to evaluate BLEU on the segment level, many smoothing and approximation techniques have been developed. In a discussion on how to automatically assess the quality of machine translation metrics, Lin and Och [2004] address the discussed problem of non-informative zero scores, since they try to score all k-best hypotheses in the output. Their procedure is to simply add 1.0 to all N-gram precision calculations for all $N > 1$:

$$q'_N = \frac{\sum\limits_{g \in n_N(c)} \delta'_r(g)}{|n_N(c)| + N - \max(N-1, 1)}, \tag{3.20}$$

where $\delta'_r(g)$ (clipped N-gram count w.r.t. reference r, see Section 2.3.2.2) is $1 + \delta_r(g)\ \forall N > 1$ iff $\delta_r(g) > 0$ for $N = 1$.[22] This effectively solves the problem, but as Nakov et al. [2012] show, this can have unwanted implications when the metric is used for optimization: Since the method effectively adds $N - 1$ N-gram hits without correcting the brevity penalty, the modification shown in Equation 3.20 leads to a slight bias towards shorter translations. This is because the sum of N-gram precisions gains in importance compared to the brevity penalty term. Thus shorter translations may be preferred when a model optimized with this variant of the per-sentence BLEU score. The remedy as proposed by Nakov et al.

[21] The time complexity of the Levenshtein distance is $O(|\hat{\mathbf{e}}||\mathbf{e}|)$ using a single reference $\mathbf{e}$, $\hat{\mathbf{e}}$ being a candidate translation.

[22] Nakov et al. [2012] refer to limiting the add-one smoothing to precisions $N > 1$ as "grounding", and implement it by subtracting $\left(\prod_1^N \frac{1}{|n_N(c)|+1} \right)^{\frac{1}{n}}$.

[2012] is to simply add 1.0 to the effective reference length:

$$\mathrm{BP}' = \begin{cases} 1 \text{ if } |c| > |r| + 1 \\ \exp^{(1-(|r|+1)/c)} \text{ else.} \end{cases} \tag{3.21}$$

Liang et al. [2006*a*] present another remedy for zero per-sentence BLEU scores in the context of applying the structured perceptron to SMT optimization: Recall, that the original BLEU score is a (weighted) geometric average of N-gram precisions, and that, if there is not a single match for any N, the score will become zero. Liang et al. [2006*a*] propose to use a weighted sum of individual BLEU scores, instead of a single one:

$$\sum_{i=1}^{N} \frac{\mathrm{BLEU}_i(c, r)}{2^{N-i+1}}, \tag{3.22}$$

where BLEU_i is the original geometric average (including brevity penalty) up to $N = i$. Implemented like this, half of the score is accounted for by the full score, e.g. BLEU-4.

A more sophisticated idea for approximating BLEU on the sentence-level is presented in [Watanabe et al., 2007*b*] and [Chiang et al., 2008]: Instead of simple local smoothing, essentially only solving a numerical problem, one could also try to solve the underlying problem of not having any context[23] for calculation of the original BLEU score. The key idea is to use the previous best hypotheses as context for calculation of the BLEU score, thus effectively circumventing zero scores. Therefore, we calculate the precision term Q'_N as a sum of two terms:

$$Q'_N = \left[\sum_{c \in \mathcal{C}'} \sum_{i=1}^{N} q_i^{(c)} \right] + \sum_{i=1}^{N} q'_i, \tag{3.23}$$

where $\mathcal{C}'$ is the set of one-best candidate translations in the *pseudo corpus* (all previous segments already processed in the current epoch), the N-gram precisions $q_i^{(c)}$ are calculated for a given candidate translation c, and q'_i is calculated for the currently to be scored hypothesis. The brevity penalty is calculated similarly:

$$\mathrm{BP}'' = \begin{cases} 1 \text{ if } |c'| + |c| > |r'| + |r| \\ \exp^{(1-(|r'|+|r|/|c'|+|c|))} \text{ else,} \end{cases} \tag{3.24}$$

r' and c' referring to the pseudo corpus.

Watanabe et al. [2007*b*] and Chiang et al. [2008] differ in the definition of the pseudo corpus: While Watanabe et al. [2007*b*] simply define it as the remainder

[23] Context is important in the BLEU score, since statistics calculated on single sentences could be misleading.

of the data set used for optimization[24], switching out only the statistics for the currently considered sentence e.g. when scoring a k-best list. Instead, Chiang et al. [2008] recommend to use a decaying moving average of the previously translated sentences in an online learning setup:

$$Q'_N = \gamma \left[\sum_{c \in \mathcal{C}'} \sum_{i=1}^{N} q_i^{(c)} \right] + \sum_{i=1}^{N} q'_i, \tag{3.25}$$

and

$$\mathrm{BP}''' = \begin{cases} 1 \text{ if } \gamma|c'| + |c| > \gamma|r'| + |r| \\ \exp^{(1-(\gamma|r'|+|r|/\gamma|c'|+|c|))} \text{ else,} \end{cases} \tag{3.26}$$

where γ is typically around 0.9. The pseudo corpus $\mathcal{C}'$ is reset after each epoch of training.

There are more advanced methods to approximate the BLEU score for single sentences: Tromble et al. [2008] propound to use a first-order vector Taylor expansion about an initial guess of the statistics for training a minimum Bayes-risk decoder on lattices — however ignoring the brevity penalty and the clipping of N-gram counts[25].

We experimented with five variants of BLEU approximations as defined above, and on two datasets, two sets of features, and either one or three datasets, respective of the use corpus. The results are depicted in Table 3.3.

1. Papineni: No smoothing, just the original BLEU score calculated on a per-sentence basis;

2. Lin: Simple add-one smoothing as proposed by Lin and Och [2004], Equation 3.20;

3. Nakov: Add-one smoothing and corrected reference length as proposed by Nakov et al. [2012], Equations 3.20 and 3.21;

4. Liang: Combination of N BLEU scores, as described in Liang et al. [2006*a*], Equation 3.22;

[24] That implies that the whole document has to be translated in advance.

[25] Sokolov et al. [2012*a*] extend this approach to include clipping and the brevity penalty, but as an approximation of the NP-hard problem of finding BLEU-oracle hypotheses in translation lattices of the phrase-based machine translation paradigm [Leusch et al., 2008], which is not directly applicable to scoring arbitrary translation hypotheses.

5. CHIANG: Calculation of per-sentence scores in the context of a pseudo corpus of previous 1-best translations, as proposed by Chiang et al. [2008], Equations 3.25 and 3.26. We set γ to 0.9.

The results reported in Table 3.3 report (corpus) BLEU scores. For the NC* data, on both feature sets, the results seem to be very similar, which are however observed at different learning rates[26] on a single development set. The CHIANG approximation gives the best result, by a small margin. Note that none of these results show any problem with brevity penalty, which is in contrast to the results for the PRO algorithm [Hopkins and May, 2011] reported in [Nakov et al., 2012], suggesting there is no bias in the selection of training examples for our proposed approach.

On the EP* data set, using a single development set for parameter tuning, the advantage of the approximation variant using a pseudo corpus has vanished, with about equal performance of CHIANG and NAKOV. The non-smoothed BLEU score shows however deteriorated results in the overall evaluation score.

Due to these results, and the simplicity of the approach, throughout this work we are using the per-sentence BLEU variant proposed by Nakov et al. [2012] for our algorithms, denoted by NAKOV, unless noted otherwise.[27].

3.9 Generating Training Data

Even for the simplest translation problem, the overall search space is too large to be fully explored[28] and the problem of finding the optimal path is NP-complete [Knight, 1999]. A natural choice for approximating the search space, when the true space is too large to fully explore, are k-best lists consisting of the top-k translations by model score. This set can be efficiently found, although with search errors due to pruning. But still, even at modest numbers for k, e.g. 100, considering the full cross-product of pairs from a ordered list $\mathcal{K}$ is still not feasible, since the full set:

$$\mathcal{P} = \left\{ (x^{(i)}, x^{(j)}) | x^{(i)}, x^{(j)} \in \mathcal{K} \wedge i, j \in \mathcal{K}_J \wedge i < j \right\}, \tag{3.27}$$

(where $\mathcal{K}_J$ is the index set of K, with the index of the first item being 1), would require computation of

$$|\mathcal{P}| = \sum_{i=N-1}^{1} i = \frac{N(N-1)}{2} \tag{3.28}$$

[26] The margin perceptron is used for this experiment.

[27] Gimpel and Smith [2012*a*] also report superior results for NAKOV in a single reference evaluation, which we are also considering here.

[28] For word-based machine translation, the most basic case, allowing any reordering of the source $\mathbf{f}$ as input, and having on average l translation options per source word (including ϵ translations), the number of possible paths is $|\mathbf{f}|! \times l$.

NC*						
System	Dev.	Test	BP			
DENSE, NAKOV		26.3	1.0			
DENSE, CHIANG		**26.4**	1.0			
DENSE, LIANG		26.3	1.0			
DENSE, LIN		26.2	1.0			
DENSE, PAPINENI		26.3	1.0			
SPARSE, NAKOV		26.7	1.0			
SPARSE, CHIANG		**27.0**	1.0			
SPARSE, LIANG		26.8	1.0			
SPARSE, LIN		26.7	1.0			
SPARSE, PAPINENI		26.8	1.0			

EP*							
System	Dev.	Test	BP	Test_1	BP	Test_2	BP
DENSE, NAKOV		**30.2**	1.0	**29.9**	1.0	**30.3**	1.0
DENSE, CHIANG		30.1	1.0	29.9	1.0	30.2	1.0
SPARSE, NAKOV		30.1	1.0	30.0	1.0	30.5	1.0
SPARSE, CHIANG		30.1	1.0	**30.1**	1.0	**30.6**	1.0
SPARSE, PAPINENI		29.7	1.0	29.9	1.0	30.2	1.0

WMT15			
System	Dev.	Test	Test
DENSE, NAKOV		21.9	**27.1**
DENSE, CHIANG		21.9	26.8
SPARSE, NAKOV		22.7	27.5
SPARSE, CHIANG		**22.9**	**27.6**

Figure 3.3: Comparing different variants of per-sentence BLEU approximations for the gold-standard function.

pairs per source segment. Taken for itself that is not is not a prohibitive number, but considering that this number of pairs has to be evaluated for every segment in the training data also in addition to decoding costs. In practice this brute force approach is thus prohibitively slow in most cases.

In [Hopkins and May, 2011] this problem is also acknowledged, and an alternative heuristic sampling procedure is proposed: Instead of trying to learn from the full space of pairs with $|\mathcal{P}|$ members, they propose to sample a fixed number of l pairs from the complete set of possible pairs, while enforcing a minimum difference in the gold-standard score for each pair. The sampling is iterated until the set of pairs has l elements. Their algorithm[29] is depicted in Algorithm 4: The k-best list of translations is first ordered according to the output of the gold-standard scoring function $g(\cdot)$. From this list, a set of pairs $\mathcal{P}$ is iteratively constructed by sampling two indexes i, j from $\mathcal{K}_J$ uniformly at random and checking whether their difference in the gold-standard score $g(\mathcal{K}^{(i)}) - g(\mathcal{K}^{(j)})$ is greater or equal than a threshold δ.[30]

This algorithm has been show to have some problematic effects: Since the selection of training data focuses on pairs with a high quality differential[31], the algorithm may (depending on the baseline quality of the translation system) select bad examples for comparison, e.g. translations that have very bad characteristics such as problems with length, as established by Nakov et al. [2013]. Additionally, the algorithm has a number of hyperparameters (gold-standard score difference δ and number of pairs to accept l) which is also problematic, since, ideally, they should be optimized for each data set, as the optimal values may change during optimization, e.g. when overall translation quality is getting better, a smaller δ may be appropriate. Accordingly, Cherry and Foster [2012] report issues finding parameters of the algorithm that work well in general. Overall the algorithm requires the setting of of four hyperparameters: the k-best list of translations $\mathcal{K}$ for a given segment, a gold-standard function $g(\cdot)$, i.e. per-sentence BLEU, a minimal difference in gold-standard score δ, and a maximum number of pairs l. $U\{\cdot\}$ in the algorithm is a uniform distribution.

Lastly, Nakov et al. [2012] suggest that this training data generation may bias

[29] Note that the algorithm is intentionally written in a suboptimal way for the sake of comparability with our own work — instead of the nested iteration over the index set $\mathcal{K}_J$ one could simply draw two different indexes from the index set at random without replacement and skip the first sorting operation. Also, the shown algorithm is missing a stopping criterion due to the chosen formulation. See [Hopkins and May, 2011] for an alternative formulation of the algorithm.

[30] Watanabe et al. [2006] use a similar algorithm for generating the training data for their pairwise ranking rescoring approach: They add pairs $(\mathcal{K}^{(i)}, \mathcal{K}^{(j)})$ of a k-best list $\mathcal{K}$ (which is sorted according to the original model score) to their training data iff a per-sentence BLEU score for $\mathcal{K}^{(j)}$ is larger than $\mathcal{K}^{(i)}$ and the WER of $\mathcal{K}^{(i)}$ is also worse than the one of $K^{(j)}$.

[31] The δ parameter of the algorithm was chosen to be 0.05 in [Hopkins and May, 2011] (5 % points in per-sentence BLEU).

Algorithm 4 The heuristic training data generation of Hopkins and May [2011]. *Inputs:* k-best list $\mathcal{K}$, gold-standard function $g(\cdot)$, threshold δ, and maximum number of pairs l.

procedure SAMPLE PAIRS PRO
  for $i \leftarrow 1 \dots k$ **do**
    $\mathbf{g}_i \leftarrow g(\mathcal{K}^{(i)})$
  end for
  Sort $\mathbf{g}$ in decreasing order.
  Sort $\mathcal{K}$ according to the values of $\mathbf{g}$.
  $\mathcal{P} \leftarrow \emptyset$
  while $|\mathcal{P}| < l$ **do**
    for $i \leftarrow 1 \dots |\mathcal{K}| - 1$ **do**
      for $j \leftarrow i + 1 \dots |\mathcal{K}|$ **do**
        if $X \sim U\{0, 1\} \geq 1 - \frac{l}{|\mathcal{P}|}$ **then**
          if $\mathbf{g}_i - \mathbf{g}_j > \delta \wedge (\mathcal{K}^{(i)}, \mathcal{K}^{(j)}) \notin \mathcal{P}$ **then**
            $\mathcal{P} \cup \left\{(\mathcal{K}^{(i)}, \mathcal{K}^{(j)})\right\}$
          end if
        end if
      end for
    end for
  end while
  Return $\mathcal{P}$
end procedure

the learned model to produce translations that are too short, possibly coming from the pair acceptance criterion of the algorithm.

We propose a simpler algorithm for the generation of training data, with only a single hyperparameter, focusing on the capability of the translation model instead of looking for examples with a large margin in terms of the gold-standard score. This algorithm is depicted in Algorithm 5.

Algorithm 5 Generating training data using multiple quality levels. The algorithm requires a single hyperparameter κ to determine the quality levels. *Inputs:* k-best list $\mathcal{K}$, gold-standard function $g(\cdot)$, parameter κ.

procedure SAMPLE PAIRS MULTIPARTITE
  for $i \leftarrow 1 \dots k$ **do**
    $\mathbf{g}_i \leftarrow g(\mathcal{K}^{(i)})$
  end for
  Sort $\mathbf{g}$ in decreasing order.
  Sort $\mathcal{K}$ according to the values of $\mathbf{g}$.
  $\mathcal{P} \leftarrow \emptyset$
  for $i \leftarrow 1 \dots \lfloor \kappa|\mathcal{K}| \rfloor$ **do**
    for $j \leftarrow \lfloor \kappa|\mathcal{K}| \rfloor + 1 \dots \ |\mathcal{K}|$ **do**
      if $\mathbf{g}_i \neq \mathbf{g}_j$ **then**
        $\mathcal{P} \cup \{(\mathcal{K}^{(i)}, \mathcal{K}^{(j)})\}$
      end if
    end for
  end for
  for $i \leftarrow \lfloor \kappa|\mathcal{K}| \rfloor + 1 \dots |\mathcal{K}| - \lfloor \kappa|\mathcal{K}| \rfloor$ **do**
    for $j \leftarrow |\mathcal{K}| - \lfloor \kappa|\mathcal{K}| \rfloor + 1 \ \dots \ |\mathcal{K}|$ **do**
      if $\mathbf{g}_i \neq \mathbf{g}_j$ **then**
        $\mathcal{P} \cup \{(\mathcal{K}^{(i)}, \mathcal{K}^{(j)})\}$
      end if
    end for
  end for
  Return $\mathcal{P}$
end procedure

In the algorithm the translations of each k-best list are divided into three distinct groups or levels — high, medium and low — as defined by a hyper parameter $0 < \kappa < 1.0$. Since the k-best list is sorted according to the gold-standard score prior to generation of the pairs, these groups may reflect actual levels of translation quality[32]. We refer to this method as *multipartite ranking*. A visualization is

[32]The exact sizes of the three levels may be adjusted if gold-standard scores at the boundaries

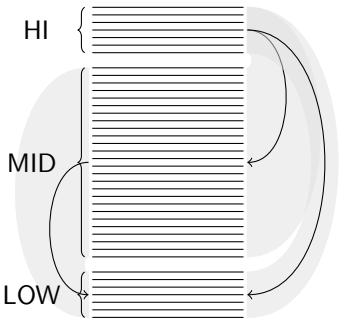

Figure 3.4: Visualization of multipartite pairwise ranking. Figure adapted from [Simianer et al., 2012].

depicted in Figure 3.4: We define three distinct levels of translation quality (the indicated list is sorted according to the gold-standard) — HI, MID and LOW, which are mutually exclusive. In our variant of multipartite ranking, pairs are generated comparing HI with both MID and LOW, and MID just to LOW.

The general intuition behind the algorithm is to ensure that good translations are preferred over bad translations without teasing apart small differences, without having to choose a fixed difference in the gold-standard scores.

This way, we try to focus on getting the most important examples right, i.e. ranking the translations with highest translation quality at the top, since at test time, in most cases, only the 1-best translation is considered. In this case, correct ordering at the bottom of approximated search space is irrelevant. The number of pairs is significantly reduced by about 65%[33], see Equation 3.29.

$$\begin{aligned} &\kappa|K| * (|K| - \kappa|K|) + (|K| - \kappa|K|) * \kappa|K| \\ =&2(\kappa|K| * (|K| - \kappa|K|)). \end{aligned} \tag{3.29}$$

In the most simple case, we could also just extract all pairs. The respective algorithm is depicted in Figure 6. An approach that employs all pairs for optimization is presented in [Dreyer and Dong, 2015], using the algorithm of Lee and Lin [2014] which has a $\mathcal{O}(k \log k)$ time complexity, calculating a hinge loss objective over all pairs.

Another alternative for pair selection is to borrow the idea of hope- and fear translations from Chiang [2012]: In each k-best list we can find a hope translation

are overlapping.

[33] For the proposed multipartite 10-80-10 ($\kappa = 0.1$) scheme: $10 * 90 + 90 * 10 = 1800 \approx 36.36\%$.

Algorithm 6 Full pairwise sample for training data generation. *Inputs:* k-best list $\mathcal{K}$, gold-standard function $g(\cdot)$.

procedure SAMPLE PAIRS FULL
  for $i \leftarrow 1 \dots k$ **do**
    $\mathbf{g}_i \leftarrow g(K^{(i)})$
  end for
  Sort $\mathbf{g}$ in decreasing order.
  Sort $\mathcal{K}$ according to the values of $\mathbf{g}$.
  $\mathcal{P} \leftarrow \emptyset$
  for $i \leftarrow 1 \dots |\mathcal{K}| - 1$ **do**
    for $j \leftarrow i + 1 \dots \ |\mathcal{K}|$ **do**
      if $\mathbf{g}_i \neq \mathbf{g}_j$ **then**
        $\mathcal{P} \cup \{(\mathcal{K}^{(i)}, \mathcal{K}^{(j)})\}$
      end if
    end for
  end for
  Return $\mathcal{P}$
end procedure

e_{hope} with derivation h_{hope} by:

$$(e_{\text{hope}}, h_{\text{hope}}) = \arg\max_{(e,h)\in\mathcal{K}} \langle \mathbf{w}, \phi(f, e, h) \rangle + g(e, e^*), \tag{3.30}$$

and also a fear translation e_{fear} (h_{fear}) by:

$$(e_{\text{fear}}, h_{\text{fear}}) = \arg\max_{(e,h)\in\mathcal{K}} \langle \mathbf{w}, \phi(f, e, h) \rangle - g(e, e^*). \tag{3.31}$$

An update is performed iff $\langle \mathbf{w}, \phi(f, e_{\text{hope}}, h_{\text{hope}}) \rangle > \langle \mathbf{w}\phi(f, e_{\text{fear}}, h_{\text{fear}}) \rangle$, and $e_{\text{hope}} \neq e_{\text{fear}}$ (by string comparison). The algorithm is depicted in Algorithm 7. This is an extreme choice, since in this proposed variant only a single pair is considered for usage in an update. As it resembles ideas from the MIRA inspired structured prediction approaches, as described in Section 2.5.3.1, we refer to this method as *structured* (abbreviated as struct.).

All previously discussed methods for generating training data have differences both in their general justification and their implementation: In the strategy of Hopkins and May [2011], the goal is to end up with a small number of pairs, which all represent a contrast between translations with a large quality difference according to the gold-standard. They employ sampling, which renders the strategy non-deterministic[34]. Experiments using the algorithm as a baseline reported by

[34]Ideally, to control for this variability, experiments with this method should be repeated to

Algorithm 7 Training data generation by selecting hope and fear translations.
Inputs: k-best list $\mathcal{K}$, gold-standard function $g(\cdot)$, model score $m(\cdot)$.

procedure SAMPLE PAIRS STRUCTURED
  for $i \leftarrow 1 \dots k$ **do**
    $\mathbf{g}_i \leftarrow g(\mathcal{K}^{(i)})$
  end for
  hope $\leftarrow \cdot$, $\text{score}_{\text{hope}} \leftarrow -\infty$
  fear $\leftarrow \cdot$, $\text{score}_{\text{fear}} \leftarrow -\infty$
  $\mathcal{P} \leftarrow \emptyset$
  for $i \leftarrow 1 \dots |\mathcal{K}|$ **do**
    if $\text{score}_{\text{hope}} < m(\mathcal{K}^{(i)}) + \mathbf{g}_i$ **then**
      hope $\leftarrow \mathcal{K}^{(i)}$
    end if
    if $\text{score}_{\text{fear}} < m(\mathcal{K}^{(i)}) - \mathbf{g}_i$ **then**
      fear $\leftarrow \mathcal{K}^{(i)}$
    end if
  end for
  if hope $\neq$ fear **then**
    $\mathcal{P} = (\text{hope}, \text{fear})$
  end if
  Return $\mathcal{P}$
end procedure

NC**	
System ↓	Dev. Test
DENSE, Full	23.3
DENSE, Multipartite	**23.5**
DENSE, PRO	23.3
DENSE, Structured	**23.5**
SPARSE, Full	23.7
SPARSE, Multipartite	**23.9**
SPARSE, PRO	23.5
SPARSE, Structured	**23.9**

Table 3.12: Experiments with different pair selection strategies on a small scale experiment based on the small data set (NC**).

Dreyer and Dong [2015] illustrate the problem. Using k-best lists with $k = 1500$, the 50 pairs with the highest difference in gold-standard are selected from a sample of 5,000 out of the total number of 1,124,250 possible pairings[35].

Watanabe et al. [2006] use $k = 1000$, but do not report how many pairs are effectively used according to their proposed selection criterion.

3.9.1 Evaluation

To evaluate pair selection strategies we performed two sets of experiments: 1) comparing the previously described pair selection strategies on a single toy task, and 2) comparing multipartite ranking and the hope-fear strategy on the small and medium data sets.

The results of the first experiment are depicted in Table 3.12. For data we used the NC** data set, tuning is carried out as described previously for all settings. The reported scores are generated using the averaged weights for translation of the development test set. We test both DENSE and SPARSE feature sets.

We observe that the full pairwise sample, as well as the PRO algorithm are behind of both the multipartite and the structured hope and fear approach. In the structured approach we effectively only check a single pair (constraint) in the

account for optimizer instability, see [Clark et al., 2011] or [Cettolo et al., 2011].

[35] The effective number of pairs in the experiments described in [Hopkins and May, 2011] is 100, since the training data is doubled by considering both $(K^{(i)}, K^{(j)})$ and $(K^{(j)}, K^{(i)})$ with opposing labels. This however adds no additional viable information to the training process since $\langle \mathbf{w}, \phi(\mathbf{x}^{(1)}) - \phi(\mathbf{x}^{(2)}) \rangle = -\langle \mathbf{w}, \phi(\mathbf{x}^{(2)}) - \phi(\mathbf{x}^{(1)}) \rangle$.

NC*				
System	Dev.	Test		
DENSE, Multipartite	**26.3**			
DENSE, Structured	26.2			
SPARSE, Multipartite	26.7			
SPARSE, Multipartite	26.7			
EP*				
System	Dev.	Test	Test_1	Test_2
DENSE, Multipartite	**30.2**	**29.9**	**30.3**	
DENSE, Structured	27.1	27.0	27.3	
SPARSE, Multipartite	**30.1**	**30.0**	**30.5**	
SPARSE, Structured	27.5	27.7	28.1	

Table 3.13: Experiments with a structured objective on small and medium scale data using the NC* and EP* data sets with DENSE and SPARSE features.

update rule, which could be more efficient.

Results focusing on the multipartite ranking vs. the hope-fear strategy on the full NC* data set and medium EP* data are shown in Table 3.13. While results are again almost identical on the NC* development test data, the structured strategy does not perform as well on the larger EP* data.

Following these results, throughout this work, unless otherwise noted, training data for discriminative learning are prepared by comparing a 100-best list of translations against a single reference translation using a smoothed per-sentence BLEU variant, i.e. Liang et al. [2006*a*] or Nakov et al. [2012]. We use $\kappa = 0.1$ for the multipartite pair selection method (Algorithm 5) all of our experiments and refer to this setup as 10-80-10[36].

3.10 Parallelization

The complexity of the baseline algorithm as presented before is, as an instance of SGD, maximally linear in sample size [Bottou, 2004; Bousquet and Bottou, 2008]. For algorithms with a fixed size training set or in a true online setting this is either

[36] The top 10% of translations are compared to the full residual of the list, the medium 80% only to the bottom 10%.

acceptable or inevitable. However, for online MT optimization which we seek to improve, this is very inefficient, since it is not possible to perform decoding in parallel which dominates the runtime of the overall algorithm[37].

Various approaches to parallelization for SGD have been proposed: Zinkevich et al. [2010] and McDonald et al. [2010] proposed simple mixing and iterative mixing strategies for training on disjoint parts of large data sets, both providing convergence bounds for convex problems.

Algorithm 8 Parameter mixing for online pairwise ranking optimization algorithm (adapted from [Simianer et al., 2012]). *Inputs:* Number of epochs T, number of shards Z, parallel training data $\mathcal{I}$, learning rate η, gold-standard function $g(\cdot)$, pair generation algorithm Q.

Partition data into Z shards, each of size $S \leftarrow |\mathcal{I}|/Z$ and distribute to machines.
procedure MIXSGD
  for all shards $z \in \{1 \dots Z\}$: **parallel do**
    Initialize $\mathbf{w}_{z,0,0,0} \leftarrow \mathbf{0}$.
    for epochs $t \leftarrow 0 \dots T-1$: **do**
      for all $i \in \{0 \dots S-1\}$: **do**
        Decode i^{th} input with $\mathbf{w}_{z,t,i,0}$.
        Generate training examples $\mathcal{P}$ using algorithm Q_g.
        for all examples $x_j, j \in \{0 \dots |\mathcal{P}|-1\}$: **do**
          $\mathbf{w}_{z,t,i,j+1} \leftarrow \mathbf{w}_{z,t,i,j} - \eta \nabla l_j(\mathbf{w}_{z,t,i,j})$
        end for
        $\mathbf{w}_{z,t,i+1,0} \leftarrow \mathbf{w}_{z,t,i,|\mathcal{P}|}$
      end for
      $\mathbf{w}_{z,t+1,0,0} \leftarrow \mathbf{w}_{z,t,S,0}$
    end for
  end for
  Collect final weights from each machine,

  return $\frac{1}{Z} \sum_{z=1}^{Z} \left(\frac{1}{T} \sum_{t=1}^{T} \mathbf{w}_{z,t,S,0} \right)$.
end procedure

Our implementations of parameter mixing and iterative parameter mixing are depicted in Algorithms 8 and 9 respectively.

The sole difference between the two algorithms is that Algorithm 8 requires absolutely no communication between shards while optimization, while Algorithm

[37] Apart from decoding, which has at least cubic time complexity, the baseline algorithm requires linearithmic sorting, calculation of per-sentence metrics, and dot products, which are all linear operations.

Algorithm 9 Iterative parameter mixing for online pairwise ranking optimization (adapted from [Simianer et al., 2012]). *Inputs:* Number of epochs T, number of shards Z, parallel training data $\mathcal{I}$, learning rate η, gold-standard function $g(\cdot)$, pair generation algorithm Q.

Partition data into Z shards, each of size $S \leftarrow |\mathcal{I}|/Z$ and distribute to machines.
procedure IterMixSGD
    Initialize $\mathbf{v} \leftarrow \mathbf{0}$.
    for epochs $t \leftarrow 0 \dots T-1$: **do**
        for all shards $z \in \{1 \dots Z\}$: **parallel do**
            $\mathbf{w}_{z,t,0,0} \leftarrow \mathbf{v}$
            for all $i \in \{0 \dots S-1\}$: **do**
                Decode i^{th} input with $\mathbf{w}_{z,t,i,0}$.
                Generate training examples $\mathcal{P}$ using algorithm Q_g.
                for all examples $x_j, j \in \{0 \dots |\mathcal{P}|-1\}$: **do**
                    $\mathbf{w}_{z,t,i,j+1} \leftarrow \mathbf{w}_{z,t,i,j} - \eta \nabla l_j(\mathbf{w}_{z,t,i,j})$
                end for
                $\mathbf{w}_{z,t,i+1,0} \leftarrow \mathbf{w}_{z,t,i,|\mathcal{P}|}$
            end for
        end for
        Collect weights $\mathbf{v} \leftarrow \frac{1}{Z} \sum_{z=1}^{Z} \mathbf{w}_{z,t,S,0}$.
    end for
    return v
end procedure

9 requires an average operation after all workers have processed their current data shard. This renders MixSGD trivial to implement, while IterMixSGD requires more effort, collecting, averaging and re-distributing the weights between machines or processes.

3.10.1 Feature Selection, Regularization & Multi-Task Learning

The proposed parallelization schemes give an incentive for an effective feature selection approach. Since the number of sparse features in SMT approaches infinity in the worst case[38], feature selection [Mladenić, 2006] and/or a sparsity inducing regularization such as applying an ℓ_1 norm penalty to the loss [Tsuruoka et al., 2009] are inevitable. Furthermore, as Duh et al. [2010] show, sparse, lexicalized features follow a Zipfian distribution — there are few common features, and a long tail of rarely observed features. It is thus important to have means that counter-act overestimation of these rare features. While parameter mixing helps, effectively averaging weights and thereby attenuating weights of rare features, we show that we can improve upon this simple approach.

In the linear model proposed in this work the issue is actually non-critical, since a single dot-product between two large vectors is efficient, even without specialized hardware. However the MT system's decoder, available disk space and the bandwidth of the network used for storing and distributing the weights are the limiting factors.

In MT feature selection has been predominantly confined to frequency cutoffs in terms of word- or extraction frequencies [Eidelman et al., 2013*a*; Cherry and Foster, 2012; Hopkins and May, 2011; Chen et al., 2017; Chiang, 2012], feature binning [Chiang et al., 2008], or narrowly defined feature templates [Chiang et al., 2009]. Less frequently, frequency cutoffs have been used in conjunction with ℓ_1 [Gimpel and Smith, 2012*b*] or ℓ_2 regularization [Green et al., 2014*a*]. Frequency cutoffs, as instances of *forward selection* [Draper and Smith, 1966], cannot take the actual performance into account, and cannot be integrated in the loss function. Regularization however, is an integral part of a loss function and can thus be more effective. A loss with ℓ_1 regularization:

$$\min_{\mathbf{w}} \sum_i l_i(\mathbf{w}, \mathbf{x}_i) + \lambda |\mathbf{w}|_1 \tag{3.32}$$

(where $|\mathbf{v}|_1 = \sum_i |\mathbf{v}_i|$ is the ℓ_1 norm, and $\lambda \geq 0$ determines the regularization strength) rewards sparser models as well as more conservative updates.

[38] Assuming an online system which picks up new full form vocabulary on the fly while using rule identity features.

Since adding ℓ_1 regularization renders the loss function non-differentiable for zero weights [Schmidt et al., 2009], a wide variety of solutions have been proposed, especially for regression problems: The *Lasso* approach proposed by Tibshirani [1994] interprets minimizing Equation 3.32 as a constrained quadratic[39] optimization problem, with only a single constraint $|\mathbf{w}|_1 \leq \lambda$, where λ is a hyperparameter[40]. [Approximate] (sub-) gradient descent methods have been proposed by Schmidt et al. [2007] and Cai et al. [2010] *inter-alia*. A straight-forward application of ℓ_1 regularization can be achieved by using a sub-gradient (shown here by coordinate):

$$\mathbf{w}_j \leftarrow \mathbf{w}_j + \eta \nabla_j \left(l(\mathbf{w}, \mathbf{x}) - \lambda \operatorname{sign}(\mathbf{w}_j) \right). \tag{3.33}$$

However, as Tsuruoka et al. [2009] show, this approach does not produce sparse results. A very simple approach of ℓ_1 regularization which employs clipping to actually enforce sparsity is proposed by Carpenter [2008], setting weights to zero iff the ℓ_1 penalty would cause a weight to switch its sign, "crossing" zero, see Algorithm 10.

Algorithm 10 ℓ_1 regularization with clipping. Algorithm adapted from [Tsuruoka et al., 2009].

$\mathbf{w}_j \leftarrow \mathbf{w}_j + \eta \nabla_j l(\mathbf{w}^{(t)}, \mathbf{x})$
if $\mathbf{w}_j > 0$ **then**
  $\mathbf{w}_j^{(t+1)} \leftarrow (0, \mathbf{w}_j - \eta\lambda)_+$
end if
if $\mathbf{w}_j < 0$ **then**
  $\mathbf{w}_j^{(t+1)} \leftarrow (0, \mathbf{w}_j + \eta\lambda)_+$
end if

As Tsuruoka et al. [2009] point out, this algorithm is deficient in the stochastic setting, since features observed uniquely at the end of training are overvalued. They propose to use a cumulative penalty to overcome this problem, as demonstrated in Algorithm 11. In the algorithm, $\mathbf{u}_j$ is a scalar corresponding to the amount of penalty a weight could have received in the whole training process — accounting for all examples where the feature did not actually fire:

$$\mathbf{u}_j = \frac{\lambda}{n} \sum_i \eta, \tag{3.34}$$

[39] Using the least squares loss for regression.
[40] Note that, when using a fixed number of k features, one can find a always suitable setting of λ which would have the same effect; An alternative to having to set a fixed regularization strength are regularization path following methods, cf. [Mairal and Yu, 2012].

and $\mathbf{q}_j$ is the sum of penalties the weight has actually received:

$$\mathbf{q}_j = \sum_{s=1}^{t-1} \mathbf{w}_j^{(s-1)} - \mathbf{w}_j^{(s)}. \tag{3.35}$$

Algorithm 11 ℓ_1 regularization with cumulative penalty. Algorithm adapted from [Tsuruoka et al., 2009].

$\mathbf{w}_j \leftarrow \mathbf{w}_j + \eta \nabla_j l(\mathbf{w}^{(t)}, \mathbf{x})$
if $\mathbf{w}_j > 0$ **then**
  $\mathbf{w}_j^{(t+1)} \leftarrow (0, \mathbf{w}_j - (\mathbf{u}_j + \mathbf{q}_j))_+$
end if
if $\mathbf{w}_j < 0$ **then**
  $\mathbf{w}_j^{(t+1)} \leftarrow (0, \mathbf{w}_j + (\mathbf{u}_j - \mathbf{q}_j))_+$
end if

This algorithm effectively and appropriately enforces sparsity in $\mathbf{w}$, which is why we used it for our experiments.

ℓ_2 regularization [Hoerl and Kennard, 1970] on the other hand can be directly used in gradient-based methods (loss shown for SGD with a single example $\mathbf{x}$, omitting bias and label):

$$l(\mathbf{w}) = (-\langle \mathbf{x}, \mathbf{w} \rangle) + \lambda |\mathbf{w}|_2^2, \tag{3.36}$$

(where $|\mathbf{w}|_2 = \sum_i \mathbf{w}_i^2$) with the gradient:

$$\nabla l(\mathbf{w}) = -\mathbf{x} + 2\lambda \mathbf{w}. \tag{3.37}$$

Since ℓ_2 regularization does not induce sparsity, but instead keeps the weights close to the initial weights $\mathbf{w}_0$, we do not include it in our experiments.

3.10.1.1 Multi-Task Learning

Multi-task learning in ML is a way to improve generalization performance of an algorithm by exploiting similarities shared between a number of different tasks [Caruana, 1997]. In a basic application, tasks simply correspond to a number of data sets or domains with a shared output space. The goal of multi-task learning is then to exploit commonalities in the data for improving performance across domains to learn instances of task-specific algorithms. Multi-task learning is a vast field, with a variety of applications.

In contrast to the setup we have just described, multi-task learning can be cast in a very general sense of trying to learn structures of (possibly unrelated)

tasks. Thrun and O'Sullivan [1996] present an approach using k-nearest neighbors clustering to learn similarities between tasks which are encoded in a transfer matrix that can then be used for the benefit of a number of binary classification tasks. Ando and Zhang [2005] also propose learning a common structure of tasks, enabling a semi-supervised learning approach. Multi-task learning of this kind is also used in neural network learning and also for NLP: Collobert and Weston [2008] demonstrate an architecture for using shared network components for predicting outcomes for a number of output spaces, e.g. part-of-speech tags, chunks, named entity tags or semantic roles.

The latter approaches exemplify parameter sharing, between different, but possibly related tasks. Weights for the parameters are learned jointly. This approach goes back to [Caruana, 1993], which suggests a joint neural network with n output nodes for n tasks. Similar to [Collobert and Weston, 2008], this approach can also be used to learn a multi-lingual machine translation system [Johnson et al., 2016].

A different approach is presented by Evgeniou and Pontil [2004], who instead of learning a common set of parameters, propose to use a joint regularization for multi-task learning. The formulation generally is as follows:

$$\mathbf{w} = \mathbf{w}_0 + \mathbf{v}_t, \tag{3.38}$$

where $\mathbf{w}$ are the eventually used weights, $\mathbf{w}_0$ is a shared representation between all tasks, and $\mathbf{v}_t$ are weights for each specific task t. This is multi-task learning by employing a joint regularization over tasks, possibly inducing sparsity. Since we seek an efficient solution, we further investigate in this direction. Evgeniou and Pontil [2004] try to learn a number of SVM systems, considering a joint weight vector and learning to keep all weights close to each other. Cavallanti et al. [2010] present an online multi-task perceptron algorithm, keeping weights close together by joint (half-) updates. Both approaches do not induce sparsity.

In general, following Cavallanti et al. [2010], we can reformulate the multi-task learning problem as follows:

$$\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_Z]^T \tag{3.39}$$

represents a stacked matrix of weight vectors for Z different tasks. The optimization problem is changed accordingly:

$$\mathbf{W}^* = \arg\min_{\mathbf{W}} \left(\sum_{z=1}^{Z} \sum_{n=1}^{N_Z} l(\mathbf{w}_z, \mathbf{x}_n) \right) + \Omega(\mathbf{W}), \tag{3.40}$$

where $\Omega(\cdot)$ is a regularizer for a matrix, and N_Z is the data for a shard Z. A variety of possible matrix norms can be used for Ω: Yuan and Lin [2006] propose a

group Lasso using the ℓ_1/ℓ_2 mixed-norm[41]:

$$\lambda\|\mathbf{W}\|_{1,2} = \lambda \sum_{d=1}^{D} |\mathbf{W}_d|_2.$$

Their approach is effective but groups of features have to be defined, since they consider only a single task. Argyriou et al. [2008], Argyriou et al. [2007] and Obozinski et al. [2010] apply ℓ_1/ℓ_2 regularization to multi-task learning, penalizing the ℓ_1 sum of ℓ_2 norms of the feature columns, effectively reducing the number of non-zero columns in $\mathbf{W}$. They also show that $\ell_1\ell_p$ regularization for multi-task learning is a viable extension of ℓ_1 regularization in single-task learning.

Other norms and grouping methods were also proposed, e.g. Zhao et al. [2009] use the $\ell_1\ell_\infty$ norm[42] and hierarchical groups for linear regression. Quattoni et al. [2009] propose a projected gradient method for $\ell_1\ell_\infty$ regularization.

Multi-task learning is closely related to domain adaptation, e.g. [Daumé, 2009] can be considered as an offline implementation of [Evgeniou and Pontil, 2004], as well as [Finkel and Manning, 2009], using priors for regularization.

ℓ_1 and ℓ_1/ℓ_2 regularization for multi-task learning also have parallels to domain adaptation. Martins et al. [2011] present a group-Lasso approach for feature selection in structured prediction, in which ℓ_1/ℓ_2 regularization is defined over manually selected, non-overlapping groups of features. Lal et al. [2006] present a feature selection framework which covers various regularization approaches. Perkins et al. [2003] also propose regularization for feature selection, in particular a combination of ℓ_2, ℓ_1 and ℓ_0 norms[43] as a sum of regularization terms. Their incremental, iterative variant of ℓ_1 regularization is similar to the iterative algorithm presented by Obozinski et al. [2010]. ℓ_1-based regularization, in the context of feature selection, can be considered as a method to automatically learn a feature set with a task at hand [Perkins et al., 2003].

Multi-task learning has been proven to be an effective technique to make use of related tasks, effectively boosting performance on all tasks, see e.g. the overview presented by Argyriou et al. [2008]. But in NLP and IR, multi-task learning has not been studied extensively for linear models[44]. However, Chapelle et al. [2011] present a boosting approach to ℓ_1 regularization for multi-task learning for IR, and Dredze et al. [2010] propose a perceptron-style algorithm for multi-task learning of a number of NLP tasks. In SMT, multi-task learning is scarcely used.

[41] ℓ_1 sum of ℓ_2 norms of columns.

[42] This penalizes the ℓ_1 sum over the maximum values of each column of each column, instead of the ℓ_2 norm.

[43] The ℓ_0 norm simply sums up all non-zero entries in a vector, i.e. for a vector $\mathbf{v}$ $\sum_i \mathbf{v}_i^0$, iff $\mathbf{v}_i \neq 0$.

[44] But, as noted previously, multi-task learning in the form of parameter sharing is popular in neural network learning.

Domain adaptation is much more in focus, and multi-task learning is only used for multi-domain adaptation, e.g. Cui et al. [2013] propose to use in-domain and general domain language and translation models which are then tuned jointly in a multi-task learning setup. But multi-domain adaptation is otherwise not thoroughly explored[45] for SMT [Sankaran et al., 2012]. A notable exception is presented by Duh et al. [2010], who employ multi-task learning via ℓ_1/ℓ_2 regularization for k-best reranking: They employ ℓ_1/ℓ_2 regularization for discovering a lower dimensional feature space that works well for most parts of the data. Each k-best list is treated as a separate task, and features are selected with ℓ_1/ℓ_2 regularization as noted above.

Implementation

For our implementation we seek a feature selection/multi-task learning method that 1) effectively reduces the number of features in the model, and 2) is compatible with our distributed stochastic gradient-based optimization scheme. Argyriou et al. [2008]'s proposed convex optimization problem cannot be approached using gradient-based optimization. Ando and Zhang [2005]'s gradient-based optimization algorithm requires several iterations with differing hyperparameters, which is infeasible in online tuning for SMT. Yuan and Lin [2006]'s optimization is also not gradient-based. Obozinski et al. [2010]'s ℓ_1/ℓ_2 regularization multi-task learning approach is based on gradients, but implements a form of forward[46] feature selection, which is inefficient for SMT tuning with sparse features, since an empty model (all zeroes) is not effective, and re-evaluating all translation pairs (and possibly re-decoding) is not feasible. However, instead, we may adapt Obozinski et al. [2010]'s gradient-based forward feature selection for use as a backward feature selection method, following the recursive feature elimination algorithm of Lal et al. [2006], using the final weights (with applied gradients) instead of the true gradient.

In our previously proposed algorithms, it is straight-forward to consider each processed shard as a distinct task to apply multi-task learning for feature selection. Our proposed algorithm is depicted in Algorithm 12.

In each epoch of the algorithm, and for each shard, a single iteration of SGD is performed, starting from shared identical weights. After all shards have finished, the shard- or task specific weight vectors $\mathbf{w}_Z$ are collected and stacked as rows in a matrix $\mathbf{W} \in \mathrm{R}^{Z \times d}$, where d is the current number of features (columns) and Z

[45] Mathur et al. [2014] present an application of online multi-task learning [Cavallanti et al., 2010] for multi-user adaptation for SMT.

[46] Forward and backward feature selection [Kohavi and John, 1997] are distinguished by whether they start with a full feature set from which features are removed (backward), or, if the initial feature set is empty and features are subsequently added (forward feature selection).

		w_1	w_2	w_3	w_4	w_5			w_1	w_2	w_3	w_4	w_5	
$\mathbf{W}_1^T$	[	6	4	0	0	0	]	[	6	4	0	0	0	]
$\mathbf{W}_2^T$	[	0	0	3	0	0	]	[	3	0	0	0	0	]
$\mathbf{W}_3^T$	[	0	0	0	2	3	]	[	2	3	0	0	0	]
Column ℓ_2 norm:		6	4	3	2	3			7	5	0	0	0	
ℓ_1 sum:						$\Rightarrow$	18						$\Rightarrow$	12

Figure 3.5: ℓ_1/ℓ_2 regularization enforcing feature selection. Example adapted from [Simianer et al., 2012].

is the number of shards:

$$\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_Z]^T . \tag{3.41}$$

We then may calculate the ℓ_1/ℓ_2 norm of the feature columns of $\mathbf{W}$, as exemplified in Figure 3.5. The number of features kept in the model can either be determined by setting the λ parameter as in:

$$\sum_{z=1}^{Z} l_z(\mathbf{W}^{(z)}) + \lambda\Omega(\mathbf{W}), \tag{3.42}$$

or by selecting a fixed pre-defined[47] number of features K after each epoch. We opt for using the latter since we want the model to have a pre-determined size. The reduced model is then used as a starting point for each shard in the following epoch.

3.10.1.2 Asynchronous Parallelization

In the current implementation of parallelization, a full averaged model computed from the weight matrix $\mathbf{W}$ is only observed after a single epoch of parallel training. Depending on the size of the shards or tasks, this could be a too long delay before obtaining a joint model.

As an alternative to this parallelization scheme, Dean et al. [2012] propose an asynchronous variant of SGD with strong empirical results compared to vanilla synchronous SGD (without parallelization) with largely reduced running time. An asynchronous variant of their proposed scheme adapted for pairwise ranking optimization is depicted in Algorithm 13. In the algorithm, parallel sentences (annotated with the respective per-sentence grammar) are distributed in a round-robin strategy to worker processes running in parallel. The current model parameters $\mathbf{w}$, which are maintained within the main loop, are also distributed to the worker

[47] Each threshold k corresponds to a setting of λ.

Algorithm 12 Iterative mixing algorithm with feature selection (adapted from [Simianer et al., 2012]). *Inputs:* Number of epochs T, number of shards Z, parallel training data $\mathcal{I}$, learning rate η, gold-standard function $g(\cdot)$, pair generation algorithm Q, regularization parameter k.

Partition data into Z shards, each of size $S = \mathcal{I}/Z$ and distribute to machines.
procedure IterMixSelSGD
    Initialize $\mathbf{v} \leftarrow \mathbf{0}$.
    for epochs $t \leftarrow 0 \dots T-1$: **do**
        for all shards $z \in \{1 \dots Z\}$: **parallel do**
            $\mathbf{w}_{z,t,0,0} \leftarrow \mathbf{v}$
            for all $i \in \{0 \dots S-1\}$: **do**
                Decode i^{th} input with $\mathbf{w}_{z,t,i,0}$.
                Generate training examples $\mathcal{P}$ using algorithm Q_g.
                for all pairs $x_j, j \in \{0 \dots |\mathcal{P}|-1\}$: **do**
                    $\mathbf{w}_{z,t,i,j+1} \leftarrow \mathbf{w}_{z,t,i,j} - \eta \nabla l_j(\mathbf{w}_{z,t,i,j})$
                end for
                $\mathbf{w}_{z,t,i+1,0} \leftarrow \mathbf{w}_{z,t,i,|\mathcal{P}|}$
            end for
        end for
        Collect/stack weights $\mathbf{W} \leftarrow [\mathbf{w}_{1,t,S,0}| \dots |\mathbf{w}_{Z,t,S,0}]^T$
        Select top K feature columns of $\mathbf{W}$ by ℓ_2 norm and set
        for $k \leftarrow 1 \dots K$ **do**
            $\mathbf{v}[k] = \frac{1}{Z} \sum_{z=1}^{Z} \mathbf{W}[z][k]$.
        end for
    end for
    return $\mathbf{v}$
end procedure

Algorithm 13 Asynchronous optimization with iterative feature selection, AsyncSGD. *Inputs:* Number of epochs T, number of workers Z, parallel data $\mathcal{I}$, feature selection frequency F, number of features K, learning rate η, gold-standard function $g(\cdot)$, pair generation algorithm Q.

procedure Main Loop($\mathcal{I}, T, Z$)
    $\mathbf{w} \leftarrow \mathbf{0}$
    Prepare Z worker processes
    Setup queue U for incoming weight updates
    for epochs $t \leftarrow 0 \dots T-1$: **do**
        for all $i \in \{0 \dots |\mathcal{I}|-1\}$: **do**
            Send i^{th} input, current weights $\mathbf{w}$, and pointer to queue U to next available worker by round robin allocation.
            for all $u \in U$ **do**
                $\mathbf{w} \leftarrow \mathbf{w} + u$
                if $i+1 \bmod F$ is 0 **then**
                    Select top K feature columns of $\mathbf{w}$ by ℓ_2 norm
                    for $k \leftarrow 1 \dots K$ **do**
                        $\mathbf{w}' = \mathbf{w}[k]$
                    end for
                    $\mathbf{w} \leftarrow \mathbf{w}'$
                end if
            end for
        end for
    end for
    return w
end procedure
procedure Worker(i, $\mathbf{w}$, η, g, Q, U)
    Decode input i with $\mathbf{w}$.
    Generate training examples $\mathcal{P}$ using algorithm Q_g.
    for all pairs $x_j, j \in \{0 \dots |\mathcal{P}|-1\}$: **do**
        $\mathbf{w}_{j+1} \leftarrow \mathbf{w}_j - \eta \nabla l_j(\mathbf{w}_j)$
    end for
    $U \leftarrow U \cup \mathbf{w}_{|\mathcal{P}|}$
end procedure

processes. Model updates from the workers are put into a queue, which is regularly checked within the main process, and its contents are incorporated into the main copy of the model once available. Our proposed implementation is designed in such a way, that a single update consists of a mini-batch, which corresponds to all pairs extracted from a single k-best list.

We also incorporate a heuristic for feature selection, selecting K features after having processed F mini-batches. Since there is no weight matrix available, but only single weight vectors, we propose to select weights simply by their ℓ_2 norm, selecting a fixed number of K features with maximal norm value.

3.10.2 Evaluation

We first explore the effectiveness of our general parallelization scheme on the NC$^{@}$ data set, training on the full bitext. Results are depicted in Table 3.14. Statistical significance between result differences for the test set are assessed with a approximate randomization test for the BLEU score [Riezler and Maxwell, 2005], as described in Section 2.3.2.4. Significant results are annotated by referencing the respective experiment in brackets, where $p \leq 0.05$.

Using only the DENSE feature set, the algorithms[48] show about the same performance, without significant differences. However, using SPARSE features, the iterative mixing approaches perform better than mixing once. Feature selection by ℓ_1/ℓ_2 regularization (selecting 100,000 features[49] after each epoch) results in some minor, but significant gains over all other algorithms. All algorithms are run for 15 epochs and use fixed random shards with a size of 1,000 segments, if applicable.

Another parameter we explore is sharding: Since our general domain data has no obvious partitioning, we use random sharding. There are however two variants to implement this — generate shards once before optimization, or randomly re-sharding after each epoch. Results for these experiments on NC* and WMT13 data sets presented in Tables 3.15 and 3.16 respectively. Since we tune on a smaller tuning set for WMT13 we only select 10,000 features after each epoch.

According to these results there is no difference between randomly sharding once or repeatedly. It is also worth to note that, despite having introduced a random factor in training, there is no large variance observable in the results for repeated experiments, as constituted by standard deviations. All experiments were repeated three times.

Results for the proposed asynchronous optimization algorithm contrasted to the synchronous counterparts are depicted in Table 3.17. Both algorithms periodically selected 10,000 features. Both synchronous and asynchronous variants were trained for 15 epochs, using randomized data (either randomized shards

[48] Note that we omitted the ITERMIXSELSGD algorithm.

[49] In preliminary experiments we determined that a fixed model size of 100,000 represents a practical tradeoff between decoding speed, model size and communication overhead.

NC@		
System	Dev. Test	Test
DENSE, MIXSGD(1)	25.7	27.9
DENSE, ITERMIXSGD(2)	**26.1**	27.9
SPARSE, MIXSGD(1)	26.1	27.9
SPARSE, ITERMIXSGD(2)	26.4	(1)28.6
SPARSE, ITERMIXSELSGD(3)	**26.8**	(1,2)**28.8**

Table 3.14: Comparing different, synchronous parallel optimization schemes on the small NC@ data set, training on the full bitext with DENSE and SPARSE feature sets. Significance is assessed with approximate randomization tests between all experiments in a group, significant improvements are denoted by the number of the respective algorithms. Table adapted from [Simianer et al., 2012].

NC*	
System	Dev. Test
Once	**26.3** ±0.0
Re-shard	26.2 ±0.0

Table 3.15: Random re-sharding per epoch versus sharding once on the NC* data tuning on the bitext with SPARSE features.

WMT13				
System	Dev. $Test_1$	$Test_1$	Dev. $Test_2$	$Test_2$
Once	24.9 ±0.1	**23.1** ±0.2	26.1 ±0.1	25.4 ±0.1
Re-shard	24.9 ±0.0	23.0 ±0.1	26.1 ±0.1	**25.5** ±0.1

Table 3.16: Random re-sharding per epoch versus sharding once on the WMT13 data set using the $Tuning_L$ data for training.

	WMT13			
System	Dev. Test_1	Test_1	Dev. Test_2	Test_2
ITERMIXSELSGD, Once	24.9 ±0.1	23.1 ±0.2	26.1 ±0.1	25.4 ±0.1
ITERMIXSELSGD, Re-shard	24.9 ±0.0	23.0 ±0.1	26.1 ±0.1	25.5 ±0.1
ASYNCSGD, 2 Workers	**25.0** ±0.1	23.1 ±0.1	26.2 ±0.2	25.6 ±0.1
ASYNCSGD, 4 Workers	24.8 ±0.1	23.3 ±0.1	26.4 ±0.1	25.3 ±0.2
ASYNCSGD, 10 Workers	24.8 ±0.1	23.3 ±0.1	26.4 ±0.0	25.6 ±0.1
ASYNCSGD, 20 Workers	23.6 ±0.6	21.8 ±0.9	24.8 ±0.6	24.6 ±0.6

Table 3.17: Synchronous and asynchronous parallelized SGD with ℓ_1/ℓ_2 regularization-based feature selection using the SPARSE feature set on the Tuning_L data.

or random permutations of the training data). We used two, four, ten and 20 parallel workers for the asynchronous algorithm, and ten shards for the synchronous algorithm. The segments of the training data are distributed in a round-robin fashion, skipping workers that did not return yet. Each worker sends its weight vector immediately after each mini-batch, and receives a new segment along with a newly computed global weight vector. Features are selected by the main loop after 100 total segments. All experiments were repeated three times to account for optimizer instability. We use the Tuning_L data set for tuning, and we employ the margin perceptron (cf. Section 3.10.3) with the SPARSE feature set.

Results for both algorithmic variants are very similar, the asynchronous version however breaks down when using more than ten workers, which also results in a slightly increased standard deviation. The variation for the other settings is negligible.

3.10.3 Perceptron Variants

The perceptron algorithm [Rosenblatt, 1958], despite its simplicity, has had tremendous success in various applications of ML and is well studied. Naturally, variants of the original algorithm have been developed.

The voting perceptron of Freund and Schapire [1999] is a maximum margin approach [Vapnik, 1982] of the original perceptron algorithm, which addresses a practical issue: When using a fixed learning rate throughout training, the perceptron has a bias towards the last seen training examples that caused updates. In the voted perceptron algorithm all versions of the weights and biases are stored, along with their survival rate (number of consecutive correct classifications on the training data). The prediction is then carried out by voting between the individual weight

vectors:

$$\operatorname{sign}\left(\sum_{i=1}^{n} c_i \operatorname{sign}(\langle \mathbf{w}_i, \mathbf{x}\rangle)\right), \tag{3.43}$$

where c_i is the survival rate for weight vector $\mathbf{w}_i$ during training. The method works favorably compared to just using the final weight vector. The method is neither memory nor compute efficient since every version of the weights have to be stored and applied for prediction. Alternatively, the predictions may be averaged (or just added), resulting in comparable performance but depicting the same drawbacks:

$$\operatorname{sign}\left(\sum_{i} c_i \langle \mathbf{w}_i, \mathbf{x}\rangle\right). \tag{3.44}$$

Collins [2002] propose the related averaged perceptron algorithm, where the final prediction is done by using averaged weights:

$$\frac{1}{\sum_i c_i}\left\langle \sum_{i} c_i \mathbf{w}_i, \mathbf{x}\right\rangle. \tag{3.45}$$

Collins [2002] also show superior performance over using the final weights.

The perceptron algorithm is deeply related to the support vector machine [Cortes and Vapnik, 1995] when using a hinge loss type objective [Collobert and Bengio, 2004]:

$$l_{\text{hinge}} = ((1 - \langle \mathbf{x}, \mathbf{w}\rangle)y)_+ . \tag{3.46}$$

Geometrically, this loss formulation can be interpreted as requiring a margin of at least 1.0 to the decision boundary. The sub-gradient also reflects that:

$$\nabla l_{\text{hinge}} = \begin{cases} -\mathbf{x}y & \text{if } \langle \mathbf{x}, \mathbf{w}\rangle y \leq 1, \\ 0 & \text{else.} \end{cases}. \tag{3.47}$$

Using this update rule has a caveat compared to the standard perceptron update: The vanilla perceptron algorithm has no relevant hyperparameters when using $\mathbf{0}$ as initial weights, since the scale of the weight vector is irrelevant for the prediction with $\operatorname{sign}(\cdot)$ and the learning rate η can thus be omitted (effectively set to 1.0)[50]. When using the margin perceptron, concrete values of the prediction function determine whether the margin is violated or not, and thus the learning rate η is

[50] This is also applicable to non-zero initial weights, since they "fade away" during the training in some sense.

NC* DENSE	
System	Dev. Test
MERT, DENSE	25.8 ±0.0
DTRAIN, DENSE, Regular	25.2
DTRAIN, DENSE, Margin	**26.3**
DTRAIN, SPARSE, Regular	25.5
DTRAIN, SPARSE, Margin	**26.7**

Figure 3.6: Perceptron variants on NC* data.

NC** Dense	
System ↓	Dev. Test
DENSE, Best	22.4
DENSE, Average	**23.0**
SPARSE, Best	22.2
SPARSE, Average	**22.7**

Table 3.18: Comparing averaged and single best performing weights for DTRAIN on development test.

used to scale the updates and effectively the weight vector. In this work we use a fixed margin of 1.0 and search for adequate learning rates via a coarse grid search. In practice, the learning rate determines the total number of considered pairs, the smaller the rate, the more possible margin violations, the more possible updates.

3.10.3.1 Evaluation

We first evaluate the margin perceptron on different data sets in Tables 3.6, 3.7 and 3.8. Overall, large improvements can be observed for the margin perceptron compared to the standard update rule for all three data sets, but more emphasized on the larger overall data sets, WMT13 and WMT15. The margin perceptron also outperforms the MERT algorithm with DENSE features most of the time. Note that, also when using the margin perceptron, the training procedure is still deterministic and thus has no variation. In Table 3.7 we observe less or no gains with the margin perceptron when using more training data (Tuning_L or the bitext) — here, the advantage of performing more updates seems to be less pronounced.

WMT13				
System	Dev. $Test_1$	Dev. $Test_2$	$Test_1$	$Test_2$
Tuning Data: $Tuning_S$				
MERT, DENSE	23.4 ±0.4	**25.0** ±0.3	21.8 ±0.2	**24.3** ±0.3
DTRAIN, DENSE, Regular	18.3	19.1	16.7	19.2
DTRAIN, DENSE, Margin	**23.5**	24.8	**22.2**	24.1
DTRAIN, SPARSE, Regular	19.6	20.3	18.1	20.6
DTRAIN, SPARSE, Margin	**23.6**	**25.1**	**22.3**	**24.3**
Tuning Data: $Tuning_L$				
DTRAIN, SPARSE, Regular	24.6	25.9	22.9	25.0
DTRAIN, SPARSE, Margin	**24.9** ±0.0	**26.1** ±0.1	**23.0** ±0.1	**25.5** ±0.1
Tuning Data: Bitext				
DTRAIN, SPARSE, Regular	23.4	24.8	22.0	23.9
DTRAIN, SPARSE, Margin	23.4	**25.1**	**22.4**	**24.3**

Figure 3.7: Perceptron variants on WMT13 data.

WMT15		
System	Dev. Test	Test
MERT, DENSE	21.8 ±0.0	26.5 ±0.0
DTRAIN, Regular	17.0	20.5
DTRAIN, Margin	**21.9**	**27.1**
DTRAIN, SPARSE, Regular	19.1	22.3
DTRAIN, SPARSE, Margin	**22.7**	**27.5**

Figure 3.8: Perceptron variants on WMT15 data.

Lastly, we evaluate the performance of the averaged perceptron in different training settings using the NC** data and the regular perceptron. The final weight vectors for all epochs are averaged by summing them up and dividing by the number of epochs. Results are depicted in Table 3.18. For both feature settings, averaging results in an improvement of about 0.5 %BLEU over the best single weights. This is is expected, since that all weights are just a snapshot of the optimization, and the seen training data can change significantly throughout the process. We therefore applied averaging to all other experiments.

3.11 Training on the Bitext

We previously showed that training on the full bitext of the machine translation system can be advantageous under some circumstances, e.g. when having not enough in-domain data for tuning. There have been few attempts to use the parallel training data (the bitext) for more than estimation of feature values for dense translation model features. This has a number of reasons:

1. The raw amount of processing needed for generating meaningful training data for a discriminative training algorithm is time consuming, e.g. generating per-sentence-grammars or k-best lists for the full bitext.

2. Since the bitext is also used for extracting and estimating the generative parts of the translation model, training sparse features on the same bitext that was used for extracting the phrase-table or grammar is prone to overfit [DeNero et al., 2006].

Despite these issues, training sparse features reliably without large data samples appears infeasible, since most features that we previously described occur very infrequently (if at all).

Some attempts have been made to carry out large-scale discriminative training for SMT: Xiao et al. [2011] scale up Blunsom and Osborne [2008]'s discriminative log-linear model using a fast translation forest generation approach, to train a model with 14M features on 500K sentence pairs.

Eidelman et al. [2013c,b] describe a variant of the MIRA algorithm using the *MapReduce* framework [Dean and Ghemawat, 2008] for parallelization, resulting in an approach similar to our proposed ITERMIXSELSGD tuning algorithm. They use portions of the bitext ranging from 5K to 50K segments. To counter act overfitting on the training data, a variant of *jackknifing* [Quenouille, 1956] is employed: Similar to cross-validation, estimating the grammars used for decoding a single fold using the remaining $n-1$ folds. They show however no improvement over simple training on development data. Hasler and Haddow [2012] apply the same technique to train

up to 600K topic features for multi-domain adaptation on about 150K segments, resulting in a gain of 0.1 to 0.5 %BLEU points compared to just using development data.

Maximum expected BLEU training [Smith and Eisner, 2006; Rosti et al., 2010; He and Deng, 2012] has been applied to training sparse features from the bitext. Setiawan and Zhou [2013] train 150M lexical features on a subset of a 550K bitext, resulting in +0.5 %BLEU compared to tuning on a single development set. Auli et al. [2014] present an approach for training parameters of a phrase-based reordering model with 3M features on 300K training sentences with expected BLEU training, resulting in 0.8 %BLEU improvement in a reranking experiment. In Wuebker et al. [2015*b*]'s largest experiment, a model consisting of 45M features, including a discriminative phrase-table[51], is trained on 4M sentence pairs resulting in a 0.6 %BLEU improvement compared to a baseline trained on a single development set. All presented large-scale expected BLEU results were achieved with a single generation step over the training data, i.e. decoding once. On the one hand, this is due to the used loss function which depends on the weights, and due to efficiency on the other hand. In contrast to Setiawan and Zhou [2013], Wuebker et al. [2015*b*] employ a leave-one-out method [Zollmann and Sima'an, 2005; Wuebker et al., 2012] on a per-segment basis when generating their k-best lists for training, showing a slight improvement when using this method.

Green et al. [2013*b*] use pairwise ranking with ℓ_1 regularization to learn from a small portion of the bitext, and show that it is inadequate for peak performance on a in-domain test set. Liu et al. [2013] also employ pairwise ranking, both with and without a feature grouping algorithm, showing deterioration compared to a MERT baseline both with an unregularized and a ℓ_1 regularized pairwise ranking model with sparse and dense feature sets. Only when using the feature grouping method an improvement is achieved (1.2 %BLEU). Flanigan et al. [2013] propose a combination of Gimpel and Smith [2012*b*]'s structured prediction loss and MERT for large-scale learning. In their experiments, using a jackknifing technique, they improve maximal by 1.1 %BLEU using a 300K bitext over a MERT baseline on development set.

Lastly, in a different line of work, Braune et al. [2016] train a large-scale discriminative rule selection classifier on 1.5M sentences, resulting in a gain of 0.4 %BLEU over a baseline without rule selection.

Overall, using the bitext for discriminative training in SMT has resulted in modest gains at best.

For our implementation of training on the bitext, we can straight-forwardly apply the proposed distributed algorithms, using a larger number of shards. We employ

[51] This is the same as the rule-id features used in our work — one feature per phrase-pair.

NC*	
System	Dev. Test
MERT, DENSE	25.8 ±0.0
DTRAIN, DENSE, Regular	25.2
DTRAIN, DENSE, Margin	**26.3**
DTRAIN, SPARSE, Regular	25.5
DTRAIN, SPARSE, Margin	**26.7**
DTRAIN, SPARSE, Regular, Bitext	**26.3**
DTRAIN, SPARSE, Margin, Bitext	26.2

Table 3.19: Highlighting the performance of training on the bitext on NC* data in comparison to MERT and using the margin perceptron loss.

model averaging as well as feature selection. During preliminary experiments we observed strong overfitting effects when using the SPARSE feature set without applying leave-one-out estimation for computing the statistics for the features of the translation model. Applying the improved estimation somewhat diminished the overfitting effect. In contrast to Flanigan et al. [2013], we do not use shard-specific language models, which could lead to inflated language model scores for rare sequences and thus to an overly large weight for associated features. We found that pruning singleton N-grams [Sundermeyer et al., 2011] for all $1, \ldots, N$ was a remedy to the overfitting that we observed.

3.11.1 Evaluation

We evaluate the ITERMIXSELSGD algorithm on NC (130K segments), WMT13 (4.5M segments), and WMT15 (2M segments) data sets. For each setup we provide averaged MERT baselines using the DENSE feature set. Shard size was set to 1K, the number of selected features per epoch to 100K for all experiments.

Results for NC* data are depicted in Table 3.19: Our pairwise ranking algorithm without using the margin perceptron is clearly deficient, but a scaling behavior from tuning DENSE features on the development set to training SPARSE features on the training data is apparent, an improvement of overall 1.1 %BLEU. Using the margin perceptron we can observe almost equal performance for training on the bitext and training DENSE features on the small development set. However, there is no difference training on the bitext using the margin perceptron or the vanilla perceptron loss function. The best result is achieved using the margin perceptron and SPARSE features on the small development set.

	EP@			
System	Dev. Test [EP]	Test$_2$ [EP]	Test$_1$ [CRAWL]	Test$_2$ [CRAWL]
Tuning Set: Tuning [EP]				
DENSE, SGD	25.6	26.4	–	–
SPARSE, SGD	27.8	28.4	–	–
Tuning Set: Bitext [EP]				
SPARSE, ITERMIXSELSGD	**28.0**	**28.6**	**19.1**	**17.3**
Tuning Set: Tuning [CRAWL]				
DENSE, SGD	–	–	15.4	14.4
SPARSE, SGD	–	–	17.8	16.8

Table 3.20: Highlighting the performance of training on the bitext on EP@, especially how the learned weights carry over to two out-of-domain test sets. Table adapted from [Simianer et al., 2012].

We performed a set of experiments with the medium sized EP@ data, see Table 3.20. Besides the largely improved results through the addition of the SPARSE feature set, we observe that training on the bitext with ITERMIXSELSGD gives the best results on in-domain test sets. In further experiments we compare the performance of a system where training and tuning data differ considerably (legal versus news data): For this we used the existing models trained on EP@ data, as well as a model tuned on the in-domain tuning set from the CRAWL data. The in-domain tuning results on two test sets are depicted in the last two rows of Table 3.20. Again the SPARSE features help considerably. The results from the experiment applying the weights learned on the full EP@ bitext are marked in bold in the same columns, and show that the weights learned and selected on EP@ data also perform very well on out-of-domain data, a great improvement over the in-domain tuning results[52].

For WMT15 we omitted the experiment with the margin perceptron on the bitext, since there was no difference on the smaller NC* data. On this data set, results shown in Table 3.21, the standard pairwise ranking is again clearly deficient compared to MERT. But improved results are obtained by scaling the feature set as well as the data, up to an improvement of about 5.6 %BLEU. The margin perceptron is far superior in both feature settings, clearly improving over MERT by about 1 %BLEU when using SPARSE features.

[52]But note that the training data for the generative models is also the one from EP@ experiments.

Wmt15		
Setup	Dev. Test	Test
Mert, Dense	21.8 ±0.0	26.5 ±0.0
Dtrain, Dense, Regular	17.0	20.5
Dtrain, Dense, Regular, Bitext	20.7	25.0
Dtrain, Dense, Margin	**21.9**	**27.1**
Dtrain, Sparse, Regular	19.1	22.3
Dtrain, Sparse, Regular, Bitext	21.9	26.1
Dtrain, Sparse, Margin	**22.7**	**27.5**

Table 3.21: Highlighting the performance of training on the bitext on Wmt15 data in comparison to Mert and using the margin perceptron loss.

Wmt13				
System	Dev. Test_1	Dev. Test_2	Test_1	Test_2
Tuning Data: Tuning_S				
Mert, Dense	23.4 ±0.4	**25.0** ±0.3	21.8 ±0.2	**24.3** ±0.3
Dtrain, Dense, Regular	18.3	19.1	16.7	19.2
Dtrain, Dense, Margin	**23.5**	24.8	**22.2**	24.1
Dtrain, Sparse, Regular	19.7	20.4	18.0	20.3
Dtrain, Sparse, Margin	**23.6**	**25.1**	**22.3**	**24.3**
Tuning Data: Tuning_L				
Dtrain, Sparse, Regular	24.6	25.9	22.9	25.0
Dtrain, Sparse, Margin	**24.9**	**26.1**	**23.0**	**25.5**
Tuning Data: Bitext				
Dtrain, Sparse, Regular	23.4	24.8	22.0	23.9
Dtrain, Sparse, Margin	23.4	**25.1**	**22.4**	**24.3**

Table 3.22: Highlighting the performance of training on the bitext on the large Wmt13 data in comparison to Mert and using the margin perceptron loss.

Results for the largest data set are depicted in Table 3.22. The standard algorithm is once more deficient. Results recover when the margin perceptron loss is used, which also scales well on the development data. Training on the bitext with either the standard- or the margin loss gives similar results, with a slight advantage for the margin perceptron. Since this variant has a hyperparameter this approach is very inefficient, without resorting to auto-adaptive learning rates[53]. However, the performance is merely on par with the SPARSE feature setting on the small development set. The best performance on all test sets is achieved on the large development set with the margin perceptron and SPARSE features, improving over MERT by 1.2 %BLEU on both test sets.

For this set of experiments we can conclude that training on high-quality development sets that are close to the test data is superior to training on very large data sets, even using a large number of SPARSE features, with the exception when there is a emphasized mismatch between the training data of the generative models and the test data.

3.11.2 Efficient Implementation

Our largest experiment required a large amount of computational resources: Since a joint grammar is infeasible to produce, we build per-sentence grammars for every segment in the training data. These grammars have an average size of 1 megabyte when compressed with *gzip*[54]. Training time is a number of weeks.

The proposed sharding scheme fits well in the MapReduce programming model [Dean and Ghemawat, 2008]: Input, are (conjoined) source-target pairs with the respective per-sentence grammar. In the *mapping phase*, shuffled data is automatically distributed to a predefined number of *mappers*, which are instances of per-shard optimizers according to Algorithm 12. The output of each shard is a weight vector, represented as key-value pairs, where the key is a unique string representation, and the value is the learned weight of the feature. In MapReduce, these key-value pairs are automatically and efficiently transferred to a number of pre-defined *reducers*, which are guaranteed to receive every key/value pair with a given key. In our implementation, we employ another phase preceding the reducer, which has the same interface as the reduce phase. In this so-called *combiner*-phase we implement per-column averaging and calculation of the ℓ_2 norm. The following single reducer process then implements the fixed cutoff by ℓ_1 norm and redistributes the reduced weight vector for a new mapping round for optimization.

We use the *hadoop*[55] framework which implements the above described interface

[53] Using auto-adaptive learning rates, the problem of synchronizing learning rates is introduced in the parallel setting.

[54] `https://tools.ietf.org/html/rfc1952`

[55] `http://hadoop.apache.org`

NC*	
System	Dev. Test
MERT	25.8 ±0.00
MIRA	26.0
DTRAIN	**26.3**
DTRAIN, Struct.	26.2

Table 3.23: Results using the MIRA algorithm in comparison to other methods on NC* data and the DENSE feature set.

efficiently, with automatic re-distribution of data processing in case of hardware failures. The proposed scheme can also be implemented using off-the-shelf batch processing systems, such as *Grid Engine* [Gentzsch, 2001] or *Slurm* [Yoo et al., 2003].

3.12 Further Experiments

In this section we present further experiments on a wider range of data sets using the algorithms proposed in the previous sections. We also provide more empirical comparison to two other tuning algorithms, MERT and MIRA.

3.12.1 Comparison to Mira

In Table 3.23 we compare the MIRA algorithm, as implemented in the cdec framework (see Section 3.6.2), to MERT (see Section 3.6.1) and our non-parallelized baseline algorithm using the margin perceptron loss. MIRA's objective is optimized via SGD, and a full sweep over learning rates was performed.

The results in Table 3.23 show that MERT and MIRA perform almost identical, and that there is only a slight advantage for DTRAIN with multipartite training data or the structured prediction approach with hope- and fear translations.

To further investigate the performance of the MIRA and DTRAIN implementations, we carried out further experiments on the large WMT13 data set, using both DENSE and SPARSE feature sets. For this experiments we performed a full sweep over all hyperparameters of cdec's MIRA implementation, as described in Section 3.6.2. When using DENSE features, the results of all three algorithms are similar, only the large variance in the MERT results stands out. In the SPARSE setting, MIRA and DTRAIN perform similarly, with a minor advantage for DTRAIN.

	WMT13			
System	Dev. Test_1	Test_1	Dev. Test_2	Test_2
MERT, DENSE	23.4 ±0.4	21.8 ±0.2	25.0 ±0.3	**24.3** ±0.3
DTRAIN, DENSE	**23.5**	22.2	24.8	24.1
MIRA, DENSE	23.4	**22.3**	25.0	24.1
DTRAIN, SPARSE	**23.6**	**22.3**	**25.1**	**24.3**
MIRA, SPARSE	23.5	22.2	24.6	24.0

Table 3.24: Results using the MIRA algorithm in comparison to other methods on WMT13 data and DENSE and SPARSE feature sets.

These results suggest on the one hand that discriminative training techniques can outperform the traditional, and well-studied MERT algorithm on small and large data settings. On the other hand these results show, that there is not much difference, if at all, using discriminative algorithms based on pairwise differences if hyperparameters are adjusted for peak performance for all compared algorithms.

3.12.2 Multi-Task Learning by Regularization

Until now, we we presented ℓ_1/ℓ_2 regularization as a means for feature selection in a parallelized training algorithm, seeking a robust sparse solution. When translating patent data, we have the possibility to test whether real-world, manual classifications of language material can be used to train overall better performing MT systems by exploiting similarities discovered within the data.

3.12.2.1 Experimental Setup

We used training, development and test data from the *PatTR* corpus [Wäschle and Riezler, 2012*b*] for the English-German language pair. On the top level, patents are classified into eight distinct sections, denoted A-H (see Table 2.1 for a listing). For our experiments, the training data consists of a uniform sample from all sections, which in total results in about 1M parallel sentence pairs. For development and testing we extract 2K parallel sentences each for each individual class A-H from a separate set of patents (split by year and family identifier). Additionally, using the same procedure as for the pooled training data, a pooled test set of 2,000 segments is constructed, with all sections evenly represented.

This setup allows for a number of different data configurations for training:

- INDEPENDENT: For each IPC class, A-H, train individual models using only the respective development set;

	INDEPENDENT^0	POOLED^1	POOLED-CAT^2
Pooled Test	–	51.2	51.2
A	54.9	$^{(1,3)}$55.3	$^{(1)}$55.2
B	51.5	51.5	$^{(1,2)}$51.7
C	$^{(2,3)}$56.3	$^{(3)}$55.9	55.7
D	49.9	$^{(1)}$50.3	$^{(1)}$50.3
E	$^{(2)}$49.2	49.0	$^{(2)}$49.1
F	$^{(2,3)}$51.3	51.0	51.1
G	$^{(2)}$49.6	49.4	49.6
H	49.4	49.5	$^{(1,2)}$49.7
Average	**51.5**	**51.5**	**51.5**

Table 3.25: MERT tuning on the INDEPENDENT, POOLED, and POOLED-CAT configurations. Significance testing is performed by approximate randomization (comparing results within the same row). Significantly superior results are denoted by prefixed indexes referring to the respective tuning set if $p < 0.05$. Table adapted from [Simianer and Riezler, 2013].

- POOLED: Use a joint development set of 2K segments where all IPC sections are evenly represented (each class accounts for 1/8 of the data);
- Concatenated (POOLED-CAT): Train on the full concatenation of the independent development sets, i.e. on 16K segments in total;
- SHARDING: Perform a single randomized sharding of the POOLED-CAT data, then train using the ITERMIXSELSGD algorithm, with the original split;
- RE-SHARDING: Same method as SHARDING, but randomly re-sharding after each training epoch;
- By IPC (IPC): Treat independent development sets as shards, also train with ITERMIXSELSGD.

In addition to individual test sets, we also extract a pooled test set with the same procedure as described for POOLED.

3.12.2.2 Evaluation

We first present MERT baselines using the DENSE feature set in Table 3.25 on INDEPENDENT and both pooled configurations. While there are minor fluctuations between the different configurations, i.e. tuning on INDEPENDENT on sections C

	INDEPENDENT[1]	POOLED[2]	POOLED-CAT[3]
Pooled Test	–	51.3	51.8
A	54.8	54.8	$^{(1,2)}$55.3
B	$^{(2,3)}$52.5	51.3	$^{(2)}$52.2
C	$^{(3)}$56.6	56.7	$^{(2)}$56.1
D	$^{(2)}$50.8	49.9	$^{(2)}$50.6
E	$^{(2)}$49.7	49.2	$^{(1,2)}$49.9
F	$^{(2)}$51.6	51.1	$^{(2)}$51.7
G	$^{(2)}$49.5	49.1	$^{(1,2)}$50.0
H	$^{(2)}$49.8	49.5	$^{(1,2)}$50.6
Average	**51.9**	**51.4**	**52.1**
Model Size	370K	450K	1.5M

Table 3.26: DTRAIN tuning with all data sets using the baseline algorithm (Algorithm 3) and SPARSE features. Table adapted from [Simianer and Riezler, 2013].

and F outperforms both pooled configurations by a small but significant margin, the averaged scores over all sections are almost identical. This also applies to the pooled test data.

For DTRAIN, we first evaluate the baseline algorithm using the SPARSE feature set on the same data configurations as before with MERT tuning. The results are shown in Table 3.26. On average, comparing the MERT baselines with the respective DTRAIN systems, we find that INDEPENDENT and POOLED-CAT perform slightly better with DTRAIN, whereas the POOLED setup is at about the same level. The biggest differences are observed on data from section C[56]. Model sizes are also depicted in Table 3.26. Both pooled configurations lead to significantly larger models than the average size shown for the experiments on INDEPENDENT. Since POOLED and INDEPENDENT are approximately the same size in terms of translation material, this result suggests that there are features that are only used within particular sections, i.e. because of domain-specific vocabulary.

We further experimented with DTRAIN with added ℓ_1 regularization as described in Section 3.10.1. Results are depicted in Table 3.27. Training on the small joint POOLED data set leads to deficient performance, compared to INDEPENDENT and POOLED-CAT data, as well as the MERT baseline and unregularized training

[56] As for example Wäschle and Riezler [2012*a*] show, section C ('Chemistry, Metallurgy') of the patent classification is significantly different from other classes, as it contains large amounts of non-textual data such as chemical formulae.

	Independent[1]	Pooled[2]	Pooled-Cat[3]
Pooled Test	–	50.8	52.1
A	[(2)]55.1	54.3	[(1,2)]55.9
B	[(2)]52.6	50.8	[(2)]52.6
C	56.2	56.1	[(1,2)]56.8
D	[(2)]50.7	49.5	[(1,2)]51.2
E	[(2)]50.3	49.7	[(2)]50.0
F	[(2)]51.7	50.7	[(2)]52.0
G	[(2)]49.9	49.1	[(1,2)]50.5
H	[(2)]50.5	49.2	[(2)]50.5
Average	**52.1**	**51.1**	**52.4**
Model Size	430K	460K	1.6M

Table 3.27: Dtrain tuning with all configurations using the baseline algorithm (Algorithm 3), Sparse features and ℓ_1 regularization as described in Section 3.10.1. Table adapted from [Simianer and Riezler, 2013].

with Dtrain. With regularization, training on Pooled-Cat again works best overall with a slight edge, suggesting that there are commonalities between the different sections that can be exploited better with regularization. We also note that the average result on Pooled-Cat is an improvement of 0.9 %BLEU over the respective Mert experiment. Further, the size of the model grows about 3 fold when training data is 8 times larger. Notably, the model sizes are larger compared to unregularized systems. We suspect that by using regularization, we actually observe more of the search space, which has the effect of obtaining a larger model. However, this contradicts the intention of inducing sparsity with ℓ_1 regularization.

Table 3.28 depicts the results for Dtrain training using the margin perceptron loss function. Overall results are improved compared to the standard perceptron, ℓ_1 regularized perceptron as well as Mert, where observe an improvement of 1.3 %BLEU on Pooled-Cat on average. Model sizes are larger than with both the standard- and the ℓ_1 regularized perceptron: This is expected, since the margin constraint leads to a larger number of updates, introducing more non-zero features into the model.

Lastly, we apply the IterMixSelSGD algorithm using the standard- (Table 3.29) and the margin perceptron (Table 3.30) on the variations of the Pooled-Cat data. Three different sharding methods are compared: Domain specific shards (IPC), random sharding (Sharding) and random re-sharding (Re-sharding) after each epoch. We also employ ℓ_1/ℓ_2 feature selection with a limit of 100,000

	Independent[1]	Pooled[2]	Pooled-Cat[3]
Pooled Test	–	51.3	52.6
A	(2)56.1	55.3	(2)55.9
B	(2)52.5	51.6	(2)52.4
C	(2)57.2	56.9	(1,2)57.5
D	(2)50.5	50.2	(1,2)51.4
E	(2)50.3	49.4	(1,2)50.7
F	(2)52.1	51.2	(1,2)52.6
G	(2)50.0	49.6	(1,2)50.9
H	(2)50.6	49.8	(1,2)51.3
Average	**52.4**	**51.7**	**52.9**
Model Size	420K	480K	1.7M

Table 3.28: Dtrain tuning with all configurations using the baseline algorithm (Algorithm 3), Sparse features and the margin perceptron. Table adapted from [Simianer and Riezler, 2013].

	Independent[1]	Pooled-Cat[2]	IPC[3]	Sharding[4]	Re-sharding[5]
Pooled Test	–	51.8	52.6	52.5	52.6
A	54.8	(1)55.3	(1,2)56.4	(1,2)56.2	(1,2)56.2
B	(2)52.5	52.2	(1,2)52.8	(1,2,3)53.0	(1,2)53.0
C	(2)56.6	56.1	(1,2,4,5)57.8	(1,2)57.3	(1,2)57.4
D	50.8	50.6	(1,2,4,5)51.5	(1,2)51.3	(1,2)51.2
E	49.7	(1)49.9	(1,2)50.5	(1,2)50.5	(1,2)50.4
F	51.6	51.7	(1,2)52.3	(1,2)52.4	(1,2)52.3
G	49.5	(1)50.0	(1,2)50.8	(1,2)50.9	(1,2)50.7
H	49.8	(1)50.6	(1,2)51.2	(1,2)51.1	(1,2)51.1
Average	**51.9**	**52.1**	**52.9**	**52.8**	**52.8**
Model Size	370K	1.5M	100K	100K	100K

Table 3.29: Dtrain tuning with all configurations using the IterMixSelSGD algorithm, Sparse features and the standard perceptron (with two columns replicated from Table 3.26 for comparison). Table adapted from [Simianer and Riezler, 2013].

	INDEPENDENT[1]	POOLED-CAT[2]	IPC[3]	SHARDING[4]	RE-SHARDING[5]
Pooled Test	–	52.6	53.0	53.0	53.0
A	56.1	55.9	$^{(1,2,4,5)}$56.8	$^{(1,2)}$56.6	$^{(1,2)}$56.5
B	52.5	52.4	$^{(1,2)}$53.3	$^{(1,2)}$53.4	$^{(1,2)}$53.2
C	57.2	$^{(1)}$57.5	$^{(1)}$57.5	57.4	57.4
D	50.5	$^{(1)}$51.4	$^{(1,2,4,5)}$52.1	$^{(1,2,5)}$51.8	$^{(1,2)}$51.7
E	50.3	$^{(1)}$50.7	$^{(1,2,4)}$51.1	$^{(1,2)}$50.9	$^{(1,2)}$51.0
F	52.1	$^{(1)}$52.6	$^{(1,2,4,5)}$53.1	$^{(1,2)}$52.8	$^{(1,2)}$52.9
G	50.0	$^{(1)}$50.9	$^{(1,2,4,5)}$51.4	$^{(1,2)}$51.2	$^{(1,2)}$51.1
H	50.6	$^{(1)}$51.3	$^{(1,2)}$51.6	$^{(1,2)}$51.6	$^{(1)}$51.5
Average	**52.4**	**52.9**	**53.4**	**53.2**	**53.2**
Model Size	420K	1.7M	100K	100K	100K

Table 3.30: DTRAIN tuning with all configurations using the ITERMIXSELSGD algorithm, SPARSE features and the margin perceptron. Table adapted from [Simianer and Riezler, 2013].

features for all setups. Comparing the results to the simple concatenation of section-specific data (POOLED-CAT), as well as independent data, we observe significant improvements throughout, while learning a sparser model. Regarding the question whether natural tasks (IPC sections) could provide an advantage over randomized settings, the results indicate slightly better performance for a subset of the sections. This effect is more pronounced with the margin perceptron. Also worth noting is that results for the margin-perceptron outperform the results for the standard perceptron in any setting on average, see Table 3.30.

From these experiments we can conclude that it is possible to exploit commonalities within data sets by ℓ_1/ℓ_2 regularization for overall improved performance.

3.12.2.3 Japanese-to-English Patent Translation

We applied the DTRAIN tuning algorithms to the Japanese-to-English language pair in the patent domain. Training data consists of 3M parallel segments of patent data distributed for the *NTCIR* workshop [Goto et al., 2013]. The translation model and a 5-gram language model were built only from using parallel data. Since the Japanese data is not segmented into tokens, we applied the segmenter of the *MeCab* tool [Kudo, 2005] prior to training.

Three data sets, each containing 2K parallel patent segments, are used for tuning and development. These data sets contain data from all IPC sections, with a

IPC	Dev. 1	Dev. 2	Dev. 3	Dev. Test
A	1.5	3.5	2.7	1.0
B	8.1	11.3	7.2	12.0
C	1.5	1.0	1.2	1.0
D	0.1	0.2	0.3	1.2
E	0.7	0.1	1.0	0.1
F	5.9	7.7	5.5	12.0
G	47.7	43.0	46.3	30.3
H	34.5	33.2	35.8	42.4

Table 3.31: Distribution of IPC classes in % in development data sets for a Japanese-to-English patent translation task. Table adapted from [Simianer et al., 2013*b*].

	Tuning Set			
System	Dev. 1	Dev. 2	Dev. 3	Dev. 1+2+3
MERT	27.9	27.6	27.6	27.8
DTRAIN, DENSE	27.8	–	–	–
DTRAIN, SPARSE	**28.8**	28.1	**28.7**	**29.0**
DTRAIN, SPARSE, ITERMIXSELSGD	–	–	–	28.9

Table 3.32: Tuning results for Japanese-to-English patent translation task. Results in bold are significant improvements over the MERT baseline with $p < 0.01$. Table adapted from [Simianer et al., 2013*b*].

significant bias towards sections G and H. The exact distribution of IPC sections is depicted in Table 3.31. A fourth data set of 2K segments, which exhibits the same bias, was used for testing during development.

In a first experiment we explore DTRAIN with DENSE and SPARSE feature sets on a variety of data configurations and compare it to a baseline established with MERT. Each of the available data sets is used for optimization separately, and in combination by concatenating all data sets. MERT was run several times to account for optimizer instability, the reported scores in Table 3.32 are averages. MERT results are basically invariant with regard to the development set, also when combining all development sets.

For sanity checking, DTRAIN was evaluated by training on the first development set using DENSE features, yielding a similar result as MERT with a score of 27.8 on

the test set. Focusing on the SPARSE feature set, we observe an improvement of about 1 %BLEU, with the exception of development set 2, where the improvement is less pronounced. This may be attributed to the different distribution of IPC sections within this data set, about 10% less of section H compared to the test set. DTRAIN with SPARSE features gains little compared to best single set results when using the concatenated data.

In an additional experiment, we apply the ITERMIXSELSGD algorithm, treating each development set as a single shard. After every one of 10 epochs, 100,000 features are selected. The result on the development test set is similar to using the full concatenated data, but since all data sets could be used in parallel for training, and the model size is reduced by feature selection, training is more efficient.

3.12.3 Spoken Language Translation

Automatic translation of spoken language[57] is a challenging problem, since spoken language is a distinct genre which is significantly different to the genres of written language and includes the error-prone step of transcription, as described in Section 2.4.1.5. Here we present experiments for two language pairs using transcribed data distributed for the translation tasks for the *International Workshop on Spoken Language Translation* (IWSLT) [IWSLT, 2004].

3.12.3.1 German-to-English

For German-to-English, we carried out two distinct experiments: 1) Exploring training on the bitext, as well as training sparse syntactic features, and 2) performing an ablation test for sparse features, adding lexical word-translation features as well.

In the first experiment, parallel data from the IWSLT 2013 is used [IWSLT, 2013][58], which amounts to about 140K segments of transcribed spoken language data. The German side of the parallel data was processed using the compound splitting algorithm suggested by Koehn and Knight [2003] to reduce the number of unknown tokens in the data. The experiments are carried out using three different language models, one 4-gram model built solely from the target-portion of the parallel data, and two 5-gram models, the first one built from 190M segments of the *English Gigaword* corpus [Parker et al., 2011], and another model combining all previous data sets with data from the monolingual data distributed with the WMT benchmarks, as described in Section 3.4.1, resulting in about 300M segments total. The translation model is trained as already described for the other German-to-English experiments.

[57] Here: Transcriptions of spoken language without automatic speech recognition (ASR) component.

[58] See also [Cettolo et al., 2012].

In addition to the SPARSE feature set, with rule identifier, rule bigram and rule shape features, we experiment with soft syntactic constraints to incorporate syntactic knowledge in the hierarchical phrase-based translation system. The idea of incorporating knowledge about source-side syntactic constituency, as introduced by Chiang [2005], is extended to a sparse feature approach by Marton and Resnik [2008]: For each rule application in the first pass of the translation process, a different feature is fired which indicates whether the currently used set of rules adheres or crosses any constituent of a pre-generated parse of the source. This allows to learn (soft) preferences which should identify useful source constituents. For our experiments we employ the *Stanford Parser* for German [Rafferty and Manning, 2008], which uses 20 non-terminal symbols, which results in a total of 40 additional features with this template.

Discriminative training on the bitext was performed on the full set of 140M parallel segments, employing sharding with about 2,200 segments per shard, resulting in 64 total shards. As usual, 100,000 features are selected after each epoch, and the final weights are merged by averaging. In addition to the standard sharding process, we exploited structural data that is available for the training data: The training data is made up of a number of transcribed talks, which exhibit keyword information. We clustered the training data in such a way that evenly sized clusters (similar in size to the random sharding) of related data emerged. This way we could use about 70% of the original data, since some clusters had to be discarded due to being insufficiently small. In an attempt to further prevent overfitting on the training data, instead of the previously described leave-one-out technique for grammar estimation, we employ a technique similar to the folding method described by Flanigan et al. [2013], training shard-specific language and translation models.

The tuning and development sets we use are *dev2010*, and *tst2010* respectively, as distributed by the organizers of the translation task.

The results for the first experiment are depicted in Table 3.33: By using the SPARSE feature set one can outperform the MERT baseline, and using more data for training the language models also clearly leads to improvements, although most of the additional data is not from the spoken language domain. We furthermore observe that training on the full bitext improves over just using a single tuning set by about 0.4 to 0.8 %BLEU. The soft syntactic constraints (*SoftSyntax*) however do not lead to any improvement over just using the SPARSE features in isolation. The clustering approach for sharding seems to provide some advantage over random sharding, but the effect could also stem from a random artifact from using different data.

In the second German-to-English experiment with spoken language data [IWSLT, 2015], we explore feature ablation for the SPARSE feature set and also add word-translation features (referred to as *Lexical* in the results Table) as for example proposed by [Hieber and Riezler, 2015]. These features fire for one-to-one word translations, insertions, as well as deletions. The alignments needed are read off

System	Dev. Test	Test
MERT, DENSE	26.7	–
DTRAIN, SPARSE	27.6	–
DTRAIN, SPARSE, Bitext	28.1	–
DTRAIN, SPARSE, Bitext (Clustered)	28.0	–
DTRAIN, SPARSE, SoftSyntax, Bitext	28.1	–
Large LM		
MERT, DENSE	28.1	–
DTRAIN, SPARSE	28.8	–
DTRAIN, SPARSE, Bitext	29.2	–
DTRAIN, SPARSE, Bitext (Clustered)	**29.4**	–
DTRAIN, SPARSE, SoftSyntax, Bitext	28.9	23.4
Larger LM		
MERT	28.4	–
DTRAIN, SPARSE	28.8	–
DTRAIN, SPARSE, Bitext	29.6	23.9
DTRAIN, SPARSE, Bitext (Clustered)	29.6	**24.1**

Table 3.33: Results for German-to-English spoken language translation for IWSLT'13. Table adapted from [Simianer et al., 2013*a*].

System	Dev. Test	# Features
Baseline (DENSE)	23.1 ±0.1	27
+ Rule-Bigram	+0.3	150K
+ Lexical	+0.4	70K
+ Rule-Id	−0.2	220K
+ Rule-Shape	+0.2	78
+ Rule-Id, Lexical	+0.0	230K
+ Rule-Bigram, Rule-Shape	+0.2	200K
+ Rule-Id, Rule-Shape, Lexical	+0.4	270K
+ Rule-Bigram, Rule-Id, Rule-Shape	+0.6	270K
+ Rule-Bigram, Rule-Shape, Lexical	+0.6	230K
+ Rule-Bigram, Rule-Id, Lexical	**+0.7**	280K
+ SPARSE, Lexical	**+0.7**	260K

Table 3.34: Ablation test for sparse features on German-to-English spoken language data (IWSLT'15). Table adapted from [Jehl et al., 2015].

from translation rules, which are annotated by Viterbi alignments. The features are lexicalized, i.e. each pair of aligned words used within a translation are identified by their lexical items when firing the corresponding feature.

The baseline system used is described in [Jehl et al., 2015]. For tuning we use a concatenation of three small-scale, in-domain data sets[59], with a total of about 4K segments. The data was randomly split into four shards, utilizing ℓ_1/ℓ_2 regularization selecting $100,000$ features after each of 15 epochs for which the algorithm is run. The final model used for testing is again generated by averaging the final weights for each epoch. We observe a standard deviation of 0.1 on the baseline, randomly sharding three times.

The development test results reported in Table 3.34 refer to the *tst2013* data set. All sparse features, with the exception of rule identifiers, lead to gains in translation quality in terms BLEU compared to the baseline. The lexical translation features have the strongest positive effect. Combining two sparse feature templates apparently does not have an additive effect. The combination of three features however, e.g. rule bigrams, rule identifiers, and shape features, lead to improved results. The best improvement is finally achieved either by combining all features, or by combining rule bigrams, lexical translation features, and rule identifiers.

[59] Namely *dev2010*, *tst2011*, and *tst2012*.

System	Dev.	Test	Test_1	Test_2	Test_3
MERT, DENSE		17.5	—	—	—
DTRAIN, SPARSE		17.8	—	—	—
DTRAIN, SPARSE, Bitext		**18.4**	21.2	19.4	21.5
Large TM					
DTRAIN, SPARSE		20.1	—	—	—
DTRAIN, SPARSE, Bitext		**20.7**	24.5	21.5	25.0

Table 3.35: Russian-to-English spoken language translation experiments. Table adapted from [Simianer et al., 2013*a*].

3.12.3.2 Russian-to-English

The Russian-to-English experiment is also carried out within the IWSLT'13 [IWSLT, 2013] evaluation campaign and follows the same general structure as the previous German-to-English experiments.

For training there are 130K in-domain parallel segments available. For this language pair we also try to extend the parallel data, incorporating Russian-English parallel data distributed with the WMT evaluation, resulting in a total of about 2.6M parallel segments. Note that apart from the 130K in-domain parallel data, all other data is most likely non-speech data. For Russian-to-English we re-use the large English 5-gram language model as used for the previous German-to-English experiments. For training on the bitext we sub-sample the parallel data to obtain about 1/3 of the data. We furthermore employ the same folding technique as for German-to-English.

The results for this set of experiments are depicted in Table 3.35. Again, DTRAIN with SPARSE features can outperform a MERT baseline, and training on the bitext results in a further, more pronounced improvement. Using more parallel data for training the translation model clearly improves the experiment with the bitext by 2.3 %BLEU, which indicates that training with the SPARSE features (which are all derived from translation rules) can benefit largely from an improved translation model, even when adding data that is not directly related to the actual domain of interest.

3.12.3.3 English-to-Russian

For the English-to-Russian experiment we use the same data and setup as for the Russian-to-English experiment, but can only use a 5-gram language model built

System	Dev. Test	Test_1	Test_2	Test_3
MERT, DENSE	13.1	—	—	—
DTRAIN, SPARSE	**13.9**	—	—	—
DTRAIN, SPARSE, Bitext	13.2	14.2	12.9	14.6
Large TM				
MERT, DENSE	13.5	—	—	—
DTRAIN, SPARSE	**14.8**	15.5	13.8	16.0

Table 3.36: English-to-Russian spoken language translation experiments for IWSLT'13. Table adapted from [Simianer et al., 2013*a*].

from 22M Russian segments[60], which are mostly out-of-domain.

Translation results are depicted in Table 3.36. Pairwise ranking with the SPARSE feature set again improves considerably over the MERT baseline. Tuning on the bitext however just barely improves over the baseline. This is unexpected since we used the same parallel data as for Russian-to-English for which the results were positive using the bitext as tuning data. We suspect that the trained translation model was deficient. Using more parallel data for the translation model again improves performance.

3.13 Experimental Summary

Due to the large amount of experiments we carried out, we provide a brief recapitulation the main empirical findings below, and give an overview of the experiments for the data sets used for development of our algorithms.

Experiments on Small-Scale Data

The summary for the experiments on the small-scale NC* data is depicted in Table 3.37. The results show clearly the deficiency when not using the margin perceptron loss. With the margin, MERT and DTRAIN are almost on par, with a slight advantage for the pairwise ranking approach. Training on the bitext does not work well for this data set compared to tuning on the development set, which suggests that the development set is close to the test set. We also note that the structured prediction approach works well on this data sets.

[60] For a detailed listing of data see [Simianer et al., 2013*a*].

NC*		
System	Dev.	Test
DTRAIN, DENSE, Regular	25.2	1.0
DTRAIN, SPARSE, Regular	25.5	1.0
MERT, DENSE	25.8 ±0.0	1.0 ±0.0
MIRA, DENSE	26.0	1.0
DTRAIN, SPARSE, Margin, Bitext	26.2	1.0
DTRAIN, DENSE, Margin, Struct.	26.2	1.0
DTRAIN, DENSE, Margin	26.3	1.0
DTRAIN, SPARSE, Regular, Bitext	26.3	1.0
DTRAIN, SPARSE, Margin, Struct.	26.7	1.0
DTRAIN, SPARSE, Margin	26.7	1.0

Table 3.37: Experimental summary for the German-to-English NC* data set.

Experiments on Medium-Scale Data

The compiled list of experiments for the EP* data is depicted in Table 3.38.

For this data the structured prediction approach does not hold up to the other methods. Using MERT for training only the dense features is a competitive baseline for this data set. Again the margin perceptron clearly improves over the previous approach, but also diminishes the gains observed by using the sparse features.

Experiments on Large-Scale Data

The compiled results for our largest data set are shown in Table 3.39. The main outcome of this set of experiments is that, given the same training data and feature sets, all tested algorithms perform similarly, within a range of 0.2 %BLEU. The margin perceptron and the sparse features are clearly an improvement on this data set when combined with large in-domain data. Tuning on the very large bitext does not improve over tuning on a single small development set. We also see that asynchronous and synchronous parallelization schemes perform similarly.

Lastly, the results for the Russian-to-English data set (WMT15) are listed in Table 3.40, the results confirming the previous findings.

Gold-Standard

In Section 3.8 we established that the choice of the evaluation metric used to establish a gold-standard ranking is not of greater importance in our approach, as

	EP*					
System	Dev. Test	BP	Test_1	BP	Test_2	BP
DTRAIN, DENSE, Margin, Struct.	27.1	1.0	27.0	1.0	27.3	1.0
DTRAIN, SPARSE, Margin, Struct.	27.5	1.0	27.7	1.0	28.1	1.0
DTRAIN, DENSE, Regular	27.8	1.0	27.9	1.0	28.1	1.0
DTRAIN, SPARSE, Regular	29.4	1.0	29.1	1.0	29.5	1.0
MERT, DENSE	29.9 ±0.1	1.0 ±0.0	29.7±0.2	1.0 ±0.0	30.0±0.1	1.0 ±0.0
DTRAIN, SPARSE, Margin	30.1	1.0	30.0	1.0	30.5	1.0
DTRAIN, DENSE, Margin	30.2	1.0	29.9	1.0	30.3	1.0

Table 3.38: Experimental summary for the German-to-English EP* data set.

	WMT13			
System	Dev. Test_1	Test_1	Dev. Test_2	Test_2
Tuning_S, DTRAIN, DENSE, Regular	18.3	16.7	19.1	19.2
Tuning_S, DTRAIN, SPARSE, Regular	19.6	18.1	20.3	20.6
Tuning_S, MERT, DENSE	23.4 ±0.4	21.8 ±0.2	25.0 ±0.3	24.3 ±0.3
Tuning_S, MIRA, DENSE	23.4	22.3	25.0	24.1
Bitext, DTRAIN, SPARSE, Regular	23.4	22.0	24.8	23.9
Bitext, DTRAIN, SPARSE, Margin	23.4	22.4	25.1	24.3
Tuning_S, MIRA, SPARSE	23.5	22.2	24.6	24.0
Tuning_S, DTRAIN, DENSE, Margin	23.5	22.2	24.8	24.1
Tuning_S, DTRAIN, SPARSE, Margin	23.6	22.3	25.1	24.3
Tuning_L, DTRAIN, SPARSE, Regular, ITERMIXSELSGD, Re-shard	24.6	22.9	25.9	25.0
Tuning_L, DTRAIN, SPARSE, Margin, ITERMIXSELSGD, Once	24.9 ±0.1	23.1 ±0.2	26.1 ±0.1	25.4 ±0.1
Tuning_L, DTRAIN, SPARSE, Margin, ITERMIXSELSGD, Re-shard	24.9 ±0.0	23.0 ±0.1	26.1 ±0.1	25.5 ±0.1
Tuning_L, DTRAIN, SPARSE, Margin, ASYNCSGD	25.0 ±0.1	23.1 ±0.1	26.2 ±0.2	25.6 ±0.1

Table 3.39: Experimental summary for the German-to-English WMT13 data set.

WMT15		
System	Dev. Test	Test
DTRAIN, DENSE, Regular	17.0	20.5
DTRAIN, SPARSE, Regular	19.1	22.3
DTRAIN, DENSE, Regular,Bitext	20.7	25.0
MERT, DENSE	21.8	26.5
DTRAIN, SPARSE, Regular, Bitext	21.9	26.1
DTRAIN, DENSE, Margin	21.9	27.1
DTRAIN, SPARSE, Margin	**22.7**	**27.5**

Table 3.40: Experimental summary for the Russian-to-English WMT15 data set.

long as some provisions are done to prevent non-informative zero scores. Also the brevity penalty was not an issue in the experiment we presented.

Pair Selection

We have found an effective method to reduce the number of pairs used for training in our pairwise ranking approach: As described in Section 3.9, multipartite ranking based on the gold-standard score is an adequate method to reduce the overall training data, while ensuring that important examples are observed by the algorithm.

We also found that selecting a single pair of hope and fear translations can be an efficient alternative in our framework when using small data sets.

Parallelization

For our work on training on the full bitext for a number of data sets, including one that has about 4.5M segments, we developed parallelization schemes which enable tuning on such large-scale data in the first place.

In Section 3.10 we also found that it is not important in terms of resulting translation quality whether synchronous or asynchronous algorithms are used for optimization. We also did found that there is no significant variance caused by optimizer instability for re-sharding or asynchronous parallelization.

Feature Selection and Multi-Task Learning

We introduced ℓ_1/ℓ_2 regularization for SMT tuning, and highlighted its utility for two applications: 1) feature selection, and 2) multi-task learning.

Throughout our experiments we have shown its effectiveness as a method for feature selection. We have further established the multi-task aspect on patent translation as described in Section 3.12.2, where we successfully improved over in-domain tuning sets as well as concatenated data, by exploiting similarities within the different data sets through ℓ_1/ℓ_2 regularization.

Training on the Bitext

While training on larger amounts of in-domain data is advantageous in the majority of cases, training on the same bitext which was used for training the underlying generative models gives mixed results. This is however in accordance with related work. We also show that, when doing feature selection via ℓ_1/ℓ_2 regularization, the resulting model can be useful for translating out-of-domain data, as shown in Section SEC:DTRAIN-BITEXT. This supports our intuition that the proposed feature selection technique reduces the model's components to the most important ones, which are also likely to be observed on other data than the original training set.

Margin Perceptron

In the experiments in Section 3.10.3, we established that including a large margin objective into the loss function underlying the pairwise ranking approach is clearly superior to the regular approach.

Features

Throughout this work we have shown that using sparse feature in addition to the regular dense feature set is more often than not advantageous in terms of final translation quality. Most noteworthy are the rule identifier features which effectively include a discriminative translation model in the MT system's log-linear model.

Comparison to Other Tuning Methods

Compared to other tuning methods, i.e. MERT or MIRA, we have established that all perform quite similarly. However, MERT is an exception, since it cannot be used to train sparse features. It is however a competitive baseline.

We therefore conclude, since all methods use vastly different algorithms, that training the parameters of the linear model of traditional SMT is inherently limited.

4 Learning Preferences from Post-Edits

> „Das ist eine von den alten Sünden, Sie meinen Rechnen, das sei Erfinden."
> [Johann Wolfgang von Goethe]

Since the earliest beginnings of MT, it has been obvious to many researchers and practitioners in this field of research, that automatic translation is an outstandingly hard problem [Tsujii, 1986] and may always need some form of human participation for sufficient quality [Bar-Hillel, 1960]. This entails that end-to-end[1] MT is possibly infeasible for many use cases, it however could still be used as a tool or an aid for human translators. This view becomes evident reviewing survey literature on machine translation of different decades.

As early as the 1950s, not long after the idea of machine translation was apprehended by a larger audience [Weaver, 1949], Yoshua Bar-Hillel noted:

> "[...] high-accuracy, fully automatic MT is not achievable in the foreseeable future [...]"
> [Bar-Hillel, 1951].

In 1966, the Automatic Language Processing Advisory Committee (ALPAC), asserted that the then current state of MT was amiss. One of their key findings is that general purpose automatic translation was still out of the question:

> "[...] while we have machine-aided translation of general scientific text, we do not have useful machine translation. Further, there is no immediate or predictable prospect of useful machine translation."
> [Pierce and Carroll, 1966]

Instead, the report strongly argued in favor of the view of MT as an aid for human translators.

After almost four decades of research, which explored a manifold of approaches to MT, in 1992 John Hutchins states in an introductory book on MT:

> "We can have either fully automatic translation or high-quality (computer-based) translation, but we cannot have both."
> [Hutchins and Somers, 1992]

[1] Translations meant for direct human consumption without human intervention.

And Martin Kay further states in 1997 in the fittingly titled article *"The Proper Place of Men and Machines in Language Translation"*:

> "The proper thing to do is therefore to adopt the kinds of solution that have proved successful in other domains, namely to develop cooperative man–machine systems."
> [Kay, 1997]

And, most recently, in 2016, Maarit Koponen establishes in an overview article on human-machine interaction in translation:

> "Despite improved MT quality in many language pairs, machine-translated texts are generally still far from publishable quality, [...]"
> [Koponen, 2016].

About 70 years after Bar-Hillel's initial findings, automatic translation systems are still not [yet] able to produce perfect, or, depending on the nature of the domain, even comprehensible translations without human intervention, despite the tremendous efforts invested in research and development during that time.

Nonetheless, as for example shown by Guerberof [2009], the current quality of MT can be perfectly sufficient to be used as an aid for human translators, e.g. by using the MT outputs as a starting point for manual translation in order to to save time, instead of translating from scratch. This mode of machine assisted human translation, referred to as *post-editing* (PE), has had significant impact on the practical application of MT due to its apparent positive effect on cost-efficiency compared to human translation from scratch.

In this chapter, we propose a method for improving the PE experience by providing an efficient method for learning user- or domain-specific models that improve the translation outputs that are used as starting points for PE.

Below we will first provide an extended introduction to CAT, and then discuss the general notion of adaptive MT for human machine translation. We then review related works, and finally present our own approach to an adaptive translation system based on an external k-best reranking approach. Our methods are evaluated in a set of PE experiments with simulated user inputs.

4.1 Computer-Aided Translation

Today (2017), MT outputs can despite large amounts of research, mostly only be used for non-critical tasks such as gisting[2], as we have already elaborated. For other uses of the unaltered outputs, the quality of MT is simply not sufficient for

[2]Gisting means achieving an approximate understanding of a text.

direct consumption[3]. One could argue, that for FAHQMT a much more complex problem has to be solved than just string transduction: If a machine is able to adequately and fluently transform one natural language into another one, it must have means to comprehend the actual underlying meaning, which in turn implies true artificial intelligence (AI).

Since true AI is a far fetched goal, MT may not be solved in the near future. However, as described in 2.1, MT and machine-aided human translation (MAHT) have gone hand in hand throughout the history of MT research and practice.

While human assisted machine translation (HAMT) has an appeal in rule-based MT, where for examples interfaces between man and machine can take away the burden of resolving ambiguity from the machine, current statistical systems have means to handle this autonomously. In contrast MAHT, or the more modern interpretation as CAT[4] has gained more interest in research and in the translation industry with the statistical systems.

In modern CAT, two distinct directions can be identified:

- Traditional PE — correcting automatic translation outputs for improved efficiency;
- Interactive machine translation (IAMT) — providing interactive tools which interface the user and the MT system for generating translations mutually.

While PE has been standardized, and is overall a well studied process, IAMT largely depends on the underlying MT system and is more complicated to implement[5].

In this section we focus on PE, since IAMT requires special user interfaces of some sort, has strict latency requirements due to prefix decoding, and is generally harder to implement, as well as to simulate for evaluation. PE on the other hand is straight-forward to implement and can be readily simulated.

Another type of software for CAT are *translation memories* (TM), which are specialized databases for storing previously generated translations. Prior to the translation of a source segment this database can be queried using the source as input, and all stored target-language translations (if any) are returned. Such systems can handle exact matches as well as *fuzzy*[6] matches, which allow for some variability in the query for generating more varied matches. Translation memories

[3] Most recently, Wu et al. [2016] present a human evaluation showing that there is still a significant gap between human and automatic translation, even for extremely simple inputs.

[4] Analogous to computer-aided design in engineering.

[5] Most approaches in IAMT use the phrase-based approach to MT, since hierarchical phrase-based approaches are less well suited as their parsing-based decoding process can only be used with difficulties for prefix decoding.

[6] Allowing for a certain degree of disagreement between the match and the source.

can help to enforce consistency by storing previously generated translations in context. Related to this use case are *termbases*, which are essentially context-specific dictionaries.

4.2 Post-Editing

Post-editing[7] of MT output [Somers, 2003; Balling and Carl, 2014] is an idea as old as MT itself, going back until the initial steps in MT [Koponen, 2016], and it still is a necessary step if machine translations are to be used for more than just gisting. It has been shown in a number of studies that PE has an positive effect on the productivity of translators.

Early studies on the effects of post-editing on translators found at best mixed results in terms of efficiency and quality comparing PE to translation from scratch — [Orr and Small, 1967] report significantly lower quality of post-edited machine translations, [Pierce and Carroll, 1966] conclude post-editing takes longer and is thus more expensive, and more recently Krings [2001, 1997] also could not find generally applicable gains in efficiency through PE. These results can however possibly attributed to insufficient baseline translation quality, at least to some extent, since it is directly linked to effectiveness[8] of PE [Koehn and Germann, 2014; De Sousa et al., 2011]. Krings [2001, 1997] also employ an arguably non-realistic evaluation method — think-aloud protocols [Jääskeläinen, 2010; Bernardini, 2001] — which, while giving insights to cognitive processes, do not represent a realistic scenario. For low-resource languages, PE has been shown to increase productivity, which however comes with a loss in overall translation quality [Skadiņš et al., 2011]. Garcia [2011, 2010] show the inverse for the Chinese-English language pair, with no or only marginal improvements in productivity, but a loss in the resulting quality of the translations.

More recent studies, mostly employing SMT, have reported encouraging results for PE compared to translation from scratch: Guerberof [2009], Arenas [2008], Flournoy and Duran [2009], Federico et al. [2012] Zhechev [2012], Plitt and Masselot [2010], Koglin [2015] and De Sousa et al. [2011] report large gains in efficiency for constrained well-known domains. Casanellas and Marg [2014] report on a study with positive results for PE, with some constraints with regard to the general applicability of the results. An early result by Baker et al. [1994] also shows possible gains by PE when used with a controlled language in a corporate environment. Koehn [2009] and Koehn and Haddow [2009] show that in general, translators can improve using aids in CAT, as well as that PE showed best results in terms of efficiency and also resulting quality (in terms of coarse human judgments) of translations. Plitt and Masselot [2010] also report improved quality when they use

[7] So to say the inverse view of *pre-translation*.

[8] Tatsumi [2009] find however, that MT metrics are not good predictors for post-editing speed.

PE. Aziz et al. [2012] and De Sousa et al. [2011] provide further evidence for the effectiveness of PE, but do report no effects on quality, whereas Läubli et al. [2013] report gains in quality and translation speed. Prominently, Beaton and Contreras [2010] report an average total cost reduction of 30% by utilizing post-editing. Green et al. [2013*a*] provide a statistical analysis showing improvements in both quality and speed when translators post-edit. Gaspari et al. [2014] also report that faster translation speeds are possible by PE, but also point out discrepancies in the perceived productivity and overall mixed results when comparing PE and fully human translation.

Concerning the comparison between more advanced interactive approaches to CAT, [Sturgeon and Lee, 2015] cannot provide a conclusive result, since it appears to depend on the individual translator. Conversely, Green et al. [2014*b*] report superior speed of PE compared to an interactive approach, but in contrast overall lower quality assessments for translations produced with PE.

Due to these positive results, interest in PE is strong in both research and the translation industry Tatsumi [2010]; Koponen [2016].

The use of MT in the translation industry, especially in the form of PE, is not without controversy: Due to the specious, impressive gains in translation productivity reported in some user studies, translators may be confronted by exaggerated expectations of their throughput, and in turn have to accept dramatic pay cuts. In addition, it is unclear whether these findings can even be generalized to other types of translation jobs, i.e. on different domains with possibly worse baseline translation quality which affects productivity. But translators may nevertheless be forced to accept significant cuts in pay per word regardless, which is one reason why a negative perception of PE prevails e.g. discussed by Guerberof [2013]: The authors report that some translators just do not like the task of PE as such, it being seen as an inherently different task to translation from scratch — in addition to implications on their pay. O'Brien and Moorkens [2014] conclude the same. However, e.g. Lagoudaki [2009] observe that the translation environments are diverse and experiences are thus very subjective and general conclusions again may be difficult[9]. Moorkens and O'Brien [2015] report on a study in which expert translators (in contrast to novices) showed a negative attitude towards PE. Kim and Kankanhalli [2009] present a meta study regarding general attitude towards change in this context.

Nevertheless, CAT and especially PE have increased rates in their adoption in the translation industry, and the process of PE is also an acknowledged and

[9] These difficulties also apply to translation memories and their different modes of application [Wallis, 2006].

officially defined standard[10] in addition to regular[11] translation.

For our work, we conclude that it is imperative to improve baseline translation performance, since it does directly impact the PE productivity [Koehn and Germann, 2014; De Sousa et al., 2011; Sanchez-Torron and Koehn, 2016; Lacruz et al., 2014], and may also be able to enhance translators' perception of the task of PE if they experience it as an interactive experience [Wallis, 2006], where the MT system evidently reacts to user input by continuous learning and adaptation.

4.3 Learning from Post-Edits

Most current MT systems translate, apart from offline domain adaptation efforts, segments in isolation, which means that any context apart from local context given within the segment is not taken into account in translation. This clashes with some basic notions in NLP: *"One sense per discourse"* [Gale et al., 1992] is a well known phenomenon, which is also reflected in concrete frequencies of word occurrences [Hearst, 1997] — without context, a MT system will not not be able adapt to the diction of a given discourse. If there is a mismatch between the system's training data and the translation material at hand, this leads to repeated errors. Since SMT systems tend to be consistent [Carpuat and Simard, 2012], this can pose significant problems in a CAT setup, as errors have to be corrected repeatedly. This can also cause irritations and frustrations with the general PE setup for translators [Macklovitch, 2006; Foster, 1998].

There are methods that implement very fine-grained domain adaptation methods, e.g. by building models on the fly for single segments [Biçici and Yuret, 2011], but in the PE setup a more valuable resource is available — the translations corrected and confirmed by the translator. The translation of related documents by a single translator[12] is a good example where very fine-grained information could be exploited: Translation of a pair related patent documents may benefit greatly from sentence-wise adaptation, as exemplified in Figure 4.1. Given that a particular translation for **glow plug** was deemed to be correct when first translating any of the segments, it would be preferable if the same translation was used in any of the following segments in the text (same patent or related patents). If the translation system however is not able, for any reason[13], to use the correct

[10] ISO 18587:2017: Translation services — Post-editing of machine translation output — Requirements

[11] ISO 17100:2015: Translation services — Requirements for translation services

[12] Or alternatively a group of translators adhering the same guidelines or terminology restrictions, e.g. working for the same company or project.

[13] These include but are not limited to: bad language model scores; translation option not available in the translation model, i.e. not observed in the training data; or the log-linear weights do not allow to use preferred translation rule.

Patent WO-2007000372-A1

> A sheathed element glow plug (1) is to be placed inside a chamber (3) of an internal combustion engine.
>
> The sheathed element glow plug (1) comprises a heating body (2) that has a glow tube (6) connected to a housing (4) [. . .]
>
> [. . .]

Patent WO-2007031371-A1

> A sheathed element glow plug (1) serves for arrangement in a internal combustion engine.
>
> The sheathed element glow plug comprises a heating body [. . .]
>
> [. . .]

Figure 4.1: Excerpts from abstracts of two distinct, but related patent documents about glow plugs for diesel engines.

translation in context, it will inevitably reduce the productivity of the translator (in terms of words per hour) as the translation has to be entered over and over again, possibly on every occurrence. This can also be perceived as an annoyance by the user. These faults could be mitigated if the system was able to learn from the post-edits in real-time. For efficiency and to avoid frustrations it is thus important that the MT systems used in this application are able to adapt and to learn from previous errors that were corrected.

In the following section we will first present a definition of automatic online adaptation, then describe our evaluation framework for our experiments, and present the related work on online adaptation for CAT. Finally we present our work on discriminative reranking for online adaptation, which was partially published in [Wäschle et al., 2013] and significantly extended in [Bertoldi et al., 2014].

4.4 Online Adaptation

All previously defined classification and ranking algorithms were already described as instances of online learning [Cesa-Bianchi and Lugosi, 2006], learning from one instance at a time after receiving the true label, e.g. taking a single step in SGD. The PE scenario considered in the following also fits very well in this setup: Translators typically translate single documents at a stretch, while translating single segments of each document at a time. This results in a stream of new bilingual sentence pairs which can be exploited for adapting the separate models

of the MT system. Note that, as exemplified in the previous section, these new examples can be of major importance for the quality of the translation of the other segments.

4.4.1 Online Learning Protocol

Cesa-Bianchi et al. [2008] defined an online learning protocol for adaptation in an online learning scenario, which we recreated in Algorithm 14.

Algorithm 14 Online learning protocol with online adaptation according to Cesa-Bianchi et al. [2008]. Algorithm adapted from [Wäschle et al., 2013].

Train global model M_g
for all documents $d \in \mathcal{D}$ of $|d|$ segments **do**
  Reset local model $M_d \leftarrow \emptyset$
  for all segments $t = 1, \ldots, |d|$ **do**
    1) Combine M_g and M_d into $M_{g \cup d}$
    2) Receive input segment x_t
    3) Output translation $\hat{y}$ using $M_{g \cup d}$
    4) Receive corrected translation y_t^*
    5) Refine M_d on triple $(x_t, \hat{y}_t, y_t^*)$
  end for
end for

The process starts by learning a global model M_g, which in our case refers to the linear model, and the translation- and language models learned from the static training data. In the presented scheme, we hypothesize a single user working on a single document d at a time, translating each segment t after the next for each document. For producing the output $\hat{y}_t$, as used by the post-editor as starting point, given the source segment x_t, the global model M_g is combined with a local model M_d, which is initially empty. After post-editing, the local model receives a single true label y_t^* (the finalized post-edit). The triple $(x_t, \hat{y}_t, y_t^*)$ can be used for updating M_d. Note, that we included the original translation in the inputs for the update. This original translation can for example be used to learn what exactly the translator changed in the output.

4.4.2 Related Works

[Online] Adaptation for CAT is a well explored topic, with a wide variety of approaches: Adaptation of the underlying generative models was introduced by Nepveu et al. [2004] for language and translation models, implementing age-based caches for both generative models. Levenberg and Osborne [2009] and Levenberg et al. [2010] exploit approximate bloom filters for language modeling, and online

updating of the word-alignment model with an incremental EM algorithm [Liang and Klein, 2009], as well as efficient algorithms for suffix-array based translation models [Lopez, 2007], for providing adaptation capabilities for an MT system. They achieve better results with online learning than with traditional batch retraining. A similar approach for PBMT is presented by Mirkin and Cancedda [2013]. Bertoldi et al. [2013] also suggest a cache-based approach for language modeling. Further work on incremental learning of word-alignment models is presented by Gao et al. [2011]. Ortiz-Martínez et al. [2010, 2011] recommend adaptation by re-estimation of the full feature set derived for the generative models in an PBMT system, including a distortion model.

A notably different approach to online adaptation is taken by Blain et al. [2012], using the word-alignments of source and the machine translation output as a means for learning edits done by the post-editor. Hardt and Elming [2010] use word-alignments between source and the post-edit or a reference translation in order to build a local phrase-table for adaptation.

Conceptually similar to our work is the *bold* structured perceptron-based ranking approach of Liang et al. [2006*a*], which uses "true" feature representations of the reference translations. In contrast to our work, they however discard all non-reachable examples, or use a *local* updating[14] strategy for them. Another tuning is described by Denkowski et al. [2014*a*] and more detailed in [Denkowski, 2015], where a MIRA-based algorithm is used for online learning weights in a PE setup, but without sparse features, and only adapting the weights of the generative models. Wuebker et al. [2015*a*] use sparse features for learning a pairwise ranking model by tuning [Green et al., 2013*b*] in an IAMT setup. A gradient-free variant for online learning in CAT is presented by Mathur and Cettolo [2014]. In a similar setup, Mathur et al. [2014] explore the multi-task perceptron [Cavallanti et al., 2010] for multi-user adaptation. Mathur et al. [2013] employ a variant of MIRA tuning for online adaptation, which is also tested with sparse features [Mathur and Cettolo, 2014]. In general, all tuning methods that involve online learning algorithms such as stochastic gradient descent can be used for online adaptation, if they also allow to use online re-decoding.

An application of a reranker for adaptation is the approach presented by [Cesa-Bianchi et al., 2008], where k-best reranking is employed for online adaptation in a PE environment. They make use of a perceptron-based reranker, learning from oracle BLEU translations as stand in for feature representations of reference translations. They also utilize a sparse feature set similar to the one we will describe by using target words and phrases, as well as full phrase-pairs as features. Martínez-Gómez et al. [2011] also use perceptron-based k-best reranking for online adaptation to post-edits, but use BLEU oracles for learning, and use the original dense feature set of the MT decoder. López-Salcedo et al. [2012] present a ridge

[14] Using an oracle translation as stand-in for the true reference translation.

regression approach in the same setting. Martínez-Gómez et al. [2012] provide empirical evidence that local updating following Liang et al. [2006*a*] is effective.

Online adaptation is also inherently related to the task of domain adaptation [Cuong and Sima'an, 2018; Carpuat et al., 2012], but as we have argued, a much more direct approach is possible in CAT.

Automatic PE, using another MT system [Simard et al., 2007] for correcting the initial outputs of a baseline system, has also been explored in an online adaptive setting by Simard and Foster [2013].

From a theoretical perspective, works on meta-learning that are concerned with incremental learning are related to online adaptation, see e.g. [Giraud-Carrier, 2000] for a discussion of the peculiarities of incremental learning.

Neural statistical MT, due to its inherently simpler approach to learning, can be straight-forwardly adapted to make use of online learning methods [Peris et al., 2017*a*; Turchi et al., 2017].

4.4.3 Simulated Post-Editing

Human evaluation in CAT is inherently difficult due to the many factors involved:

- Human translators have a variety of backgrounds which calls for conducting studies with more than a single translator;
- Evaluation with human translators is costly since employing professional translators has to be paid for;
- Repeated evaluation on the same data is impossible due to exposure bias;
- More general, it is difficult, if not impossible to repeat an experiment with the same variables (data as well as translators), using different MT systems on all levels;
- Translators may translate segments in arbitrary order, which leads to differing training sets for the learning algorithms;
- Humans have a learning curve which has to be taken into account;
- A usable interface has to be designed to provide a realistic scenario, which may include additional tools such as translation memories or termbases.

Hardt and Elming [2010] avoid these issues by proposing a *simulated post-editing* scenario exploiting a given set of reference translations: Following the online learning protocol defined in Algorithm 14, the evaluation is carried out using the set of 1-best outputs $\hat{y}$ of the (adapted) MT system and calculating corpus level metrics such as BLEU or TER on the final corpus using the original reference translations.

4.5 Online Adaptation by Reranking

In the following we describe our approach to online adaptation by learning to rerank in a simulated PE scenario, using a feature representation of the post-edits as learning signal. First k-best reranking is defined and related works are presented, then our approach is described and finally we present a constrained search method used to get a representation of the post-edits instead of relying on pairwise preferences defined by gold-standard scores.

4.5.1 Reranking

Reranking, as first described by Och et al. [2003*a*] for SMT, is the task of learning a ranking function from a set of outputs of a SMT system, commonly a set of fixed k-best translations. The reranking approach opens up the possibility to learn powerful discriminative models using external inputs or globally defined features, which could not be used in the ranking approach, where features can only be used if they can be incorporated into the search. Using a fixed feature set, which is independent of the underlying SMT system, it is possible to learn portable models that generalize over the underlying MT system, i.e. by defining a joint feature representation over input and output strings, disregarding the derivation provided by the MT system. The features of the baseline MT system can however also be used for reranking, e.g. for training another discriminative model on top of an already trained system which produced the k-best lists.

K-best reranking is a well studied method in machine translation. Shen et al. [2004] and Och et al. [2003*a*] describe algorithms inspired by the perceptron that use pairwise preferences based on the BLEU score for ordinal regression. In further work splitting algorithms and uneven margins are applied to reranking [Shen and Joshi, 2005, 2004]. Carter and Monz [2010] apply the structured perceptron of Collins and Duffy [2002] to SMT by using per-sentence BLEU as a ranking measure for learning a rich syntactical reranking model. Mizumoto and Matsumoto [2016] also apply the structured perceptron using BLEU to select oracle translations to avoid grammatical errors produced by the underlying SMT system. Watanabe et al. [2006] present a variant of the structured perceptron using a combination BLEU and TER for learning from full pairwise samples of the k-best lists produced by a SMT system, training a new reranking model with the same features as used in the underlying baseline model on a domain specific development set. Since the original MERT algorithm also makes use of k-best lists, it can be used for learning in a reranking step: Och et al. [2004] present a wide variety of features for reranking using MERT, and Hasan et al. [2007] explore using high values for k when using MERT for reranking. Listwise approaches have also been applied to reranking in SMT [Zhang et al., 2016].

Since reranking is a very flexible method, it can also be used to learn a discrimi-

native ranking model from the outputs of a rule-based MT system [Velldal and Oepen, 2005]; to learn with non-linear features from pairwise preferences in terms of BLEU [Sokolov et al., 2012*b*]; to train with interleaved multi-task learning steps for learning sparser models [Duh et al., 2010]; or to train using gradient-free methods, e.g. using a gradient approximation by perturbation [Lambert and Banchs, 2006].

Reranking was also applied to learn from users in a CAT setup: Cesa-Bianchi et al. [2008] present an application of the structured perceptron to learn from user-provided post-edits, using BLEU for selecting oracle translations.

4.5.1.1 Reranking with the Structured Perceptron

In contrast to parse reranking [Collins and Koo, 2005], the correct translation or structure is more often than not not included in the k-best list of translations in SMT. While there are acceptable or even good translations, matching a given string exactly is problematic. This has a number of reasons: K-best lists are very coarse approximations of the vast true search space of the system, and since the linear model used for search is error prone and hard to learn, it is no wonder that the correct one[15] is not to be found in the list. Furthermore, a single string (the surface form of a translation) may be derived in a number of ways, leading to ambiguous feature representations. No single true derivation of the reference translation can be used for learning because of this. Fundamentally, if the reference translation contains vocabulary that was not observed during training of the translation system — and is not made up from tokens that can be simply copied to the output by passing-through — the reference translation is definitely not reachable without changing the underlying system.

Thus, previous approaches in k-best reranking relied on selecting an oracle translation according to a error metric, e.g. BLEU or TER, as a stand in for the true reference translation.

We can define the structured perceptron as follows: The decision function is a arg max returning the best scoring structure (string and its derivation) according to a linear model $\mathbf{w}$:

$$(\hat{e}, \hat{h}) = \arg\max_{(e,h)} \langle \mathbf{w}, \phi(f, e, h) \rangle, \tag{4.1}$$

where $\phi(\cdot)$ is a joint feature representation, which may either be directly taken from the derivation, or be constructed using arbitrary features. Note that the arg max is approximated as a k-best list in k-best reranking.

In the structured perceptron algorithm, an error occurs if the returned translation $\hat{e}$ is not identical to the reference translation e^* (or the MT metric's oracle). If

[15] Correct in the sense of exactly resembling the reference translations.

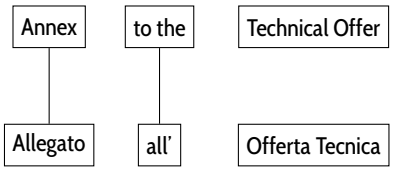

Figure 4.2: Sample output of the constrained search algorithm including two unaligned source and target words. Example adapted from [Bertoldi et al., 2014; Wäschle et al., 2013].

there is a mistake, an update of the weights is performed by:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \left(\phi(f, e^*, h^*) - \phi(f, \hat{e}, \hat{h})\right), \tag{4.2}$$

where e^* is the reference translation[16] and $\hat{h}$ its derivation. Note that in an external reranker we can also fully ignore the derivation, i.e. by using a feature mapping $\phi(f, e)$. Instead of a k-best BLEU oracle or limiting the update procedure to reachable reference translations as Liang et al. [2006*a*], we propose in this work the use of a robust variant of forced decoding as presented by Cettolo et al. [2010]: Given the complete set of translation options[17], a dynamic program finds an optimal, non-overlapping segmentation of the reference translation with a subset of the translation options. The criterion[18] of the algorithm is source- and target-side coverage. Note that the algorithm finds a set of translation options, regardless whether all parts can be covered — this means that the algorithm allows gaps in both source and target. By using this algorithm we can always[19] obtain a derivation of the reference translation in terms of the phrase-pairs of the underlying SMT system.

The approach taken here closely resembles the *bold* updating approach in the perceptron-based discriminative learning algorithm proposed by Liang et al. [2006*a*], who also use a structured perceptron. In contrast to our work, they discarded all examples that were not reachable by the decoder, i.e. where they could not obtain a derivation for the reference translation, which is why they were only able to use a maximum of 32% of their total available training data.

A further advantage of using the structured perceptron for reranking is the lack of hyperparameters[20], which eliminates the need for a development set. This

[16] Using BLEU or TER oracles as a stand-in for reference translation also enables the use of multiple references.
[17] Phrase-table entries for each source span in PBMT.
[18] The algorithm additionally takes a distortion penalty into account.
[19] Unless nothing can be covered by existing phrase-pairs.
[20] The initial weight vectors are $\mathbf{0}$, so there is no need for a learning rate.

renders k-best reranking as a portable, fast and simple method for learning user- and/or data-specific adaptation models.

4.5.1.1.1 Feature Representation

The joint feature representation for a pair of source f and target translations e in k-best reranking can be defined freely. For immediate adaptation and following our previous work on ranking for SMT we implement a rich sparse feature set, covering both generative models of the baseline SMT system:

- Phrase pairs: Every phrase-pair used in the derivation of the reference translation obtained by the constrained search method is used as a feature. This resembles a discriminative phrase-table. Feature values are derived from the size of the source span of the phrase-pair, to encourage the use of longer source spans.
- Target N-grams: All N-grams ($N = 1, \ldots, 4$) in the reference translation are used as individual features, using the respective N as the feature value, putting by default more weight on longer sequences. This actually constructs a discriminative language model [Akabe et al., 2014].

We use features with values derived from lengths instead of using simple (counting) binary features as we found a beneficial effect on BLEU scores in preliminary experiments. In contrast to Cesa-Bianchi et al. [2008] we obtained best results combining these feature templates, and by including the source in the phrase-pair features. An example of the result of constrained search is depicted in Figure 4.2: Translating from English into Italian, two phrase-pair features are extracted **Annex** $\rightarrow$ **Allegato** (with feature value 1.0) and **to the** $\rightarrow$ **all'** (feature value 2.0). Since the last tokens in source and target segments were not covered by the constrained search, no phrase-pair feature could be extracted[21]. N-gram features cover the full 4-gram **Allegato all' Offerta Tecnica**, and all included lower rank N-grams. This way non-aligned target sequences can be used in the features of the discriminative reranker[22].

These features only fire if they contain (for phrases on source- and target-sides) at least one content word. To this end we use language-specific lists of *stop words*.

4.6 Experiments

We evaluated the k-best reranking approach for online adaptation on three domains and an equal number of language pairs. All experiments use a static set of reference

[21] These phrase-pairs can be covered using a heuristic alignment, aligning non-ambiguous sequences in source and target, i.e. exact 1:1 correspondences, cf. [Bertoldi et al., 2014].

[22] If new phrase-pairs can be learned from data, these can also be included in the feature set of the reranker.

translations in lieu of post-edits. In essence this is an identical setup as the online tuning described in the previous chapter, yet there is only a single training epoch and, according to the online learning protocol, an output is produced for each example before training on it. This reflects the simulated PE setup [Hardt and Elming, 2010]. However, since we include real post-edited data obtained from a similar MT system, the evaluation should give realistic estimates of human-targeted translation metrics [Snover et al., 2006].

For all data setups we build phrase-based systems with the Moses toolkit [Koehn et al., 2007] using its default settings. Case-sensitive translation- and lexicalized reordering models are estimated on the parallel for each data setup. 5-gram language models are estimated using the *IRSTLM* [Federico et al., 2008] software using the target-side of each parallel data set. All language models are estimated including Kneser-Ney smoothing [Ney et al., 1994; Chen and Goodman, 1996]. The model further includes count-based penalties for word- and phrase applications as well as a distance-based distortion model. All features are tuned on held-out data with MERT using *Set-0–2* for each configuration. Note that these features and their weights are not used by the k-best reranker, so that k-best lists can be generated in a single step before experimentation. Development of the k-best reranker was performed on SET-0–2 of the English-to-Italian IT data.

For some experiments we employ the caching-based adaptation approaches presented by Bertoldi et al. [2013] and Wäschle et al. [2013] (cf. [Bertoldi et al., 2014]). Specifically we employ the markup-based (MARKUP) translation model adaptation based on the same algorithm as used for the reranker, as described in [Wäschle et al., 2013]. We use the model in its full configuration, i.e. adding new[23], known and full segments to the local model. Phrases are annotated greedily from left to right, without overlapping due to the used markup. Weights for each phrase-pair are estimated with relative frequency estimation.

Additionally we use the translation- (TM CACHE) and language model (N-GRAM CACHE) caches as described in [Bertoldi et al., 2013] and specifically in [Wäschle et al., 2013]. These models implement exponentially decaying caches for (non-overlapping) N-grams and phrase-pairs. Since these two models are associated to a features integrated into the log-linear model, their weights are estimated on SET-0 for each data configuration.

In all cases where we use the additional local models, the process of generating k-best translations is repeated for each setting including the respective adaptation method. While these methods may learn similar information about phrase-pairs as the reranker, it is important to note that the markup-based adaptive phrase-tables also have the ability to learn new rules by aligning unambiguous gaps of the constrained search system, which in turn are treated as features by the

[23] In this model, new phrases can be extracted from unambiguous gaps.

Task & Translation Direction	# Segments
IT English-to-Italian	1.2M
LEGAL English-to-Italian	2.3M
LEGAL Italian-to-English	2.3M
PATENT English-to-German	4.2M
PATENT German-to-English	4.2M

Table 4.1: Training data for building the phrase-based machine translation systems for the simulated post-editing experiments with k-best reranking. Table adapted from [Bertoldi et al., 2014].

discriminative reranker.

Note that we are using the exact same baseline- and adapted (for TM CACHE, N-GRAM CACHE, and MARKUP) systems as shown in [Wäschle et al., 2013] as starting points for our reranker.

Statistics for the training data for each data setup are depicted in Table 4.1.

4.6.1 IT Data

The IT data is a collection of English-Italian manuals for software products, consisting of freely available data from the *OPUS* corpora [Tiedemann and Nygaard, 2004] and commercial data. Translation direction is from English-to-Italian. The test data consists of eight consecutive documents from the commercial data. For data sets SET-6 and SET-7 there are four sets of references available (A-D), these are actual post-edits created by four independent translators with a static SMT system[24]. Note that the translators could choose which segments to translate first, resulting in different orders of the segments. This is why the repetition rates differ for all version of test sets SET-6 and SET-7.

Results for k-best reranking for each set are depicted as a scatter plot in Figure 4.3. The visualization shows BLEU and TER score differences to the respective baseline. The baseline (without any adaptation) scores are depicted in Table 4.2. By reranking the baseline system's outputs, improvements are achieved for almost all data sets, except for set SET-6D, where one translator apparently chose a non-optimal sequence of translations (for the reranker). For the same set, but with other translators' references, improvements up to 1 %BLEU score are achieved, and

[24]The system was similar, but not identical to the same one we used here for our experiments on reranking.

Test Set	# Segments	%RR Source	%RR Target	%BLEU Baseline	%TER Baseline
Set-0	420	13.3	12.0	25.6	53.7
Set-1	931	16.7	16.9	24.0	53.7
Set-2	375	14.1	12.1	23.0	54.9
Set-3	289	12.9	12.0	22.1	57.8
Set-4	1,000	16.0	15.8	21.4	56.0
Set-5	864	11.7	10.9	24.2	55.2
Set-6A	160	9.4	8.4	41.5	39.0
Set-6B	160	9.5	8.2	41.0	40.7
Set-6C	160	9.7	10.4	35.5	45.8
Set-6D	160	9.9	8.9	39.0	43.7
Set-7A	176	14.0	13.7	39.9	40.5
Set-7B	176	15.2	12.4	31.1	49.4
Set-7C	176	13.8	12.7	37.5	43.6
Set-7D	176	12.5	12.9	37.3	43.9

Table 4.2: Statistics for test sets used for simulated post-editing experiments for IT English-to-Italian system, along with scores for the 1-best results, as well as repetition rates (RR) for the source and target segments (cf. Section 4.6.4.1). Table adapted from [Bertoldi et al., 2014].

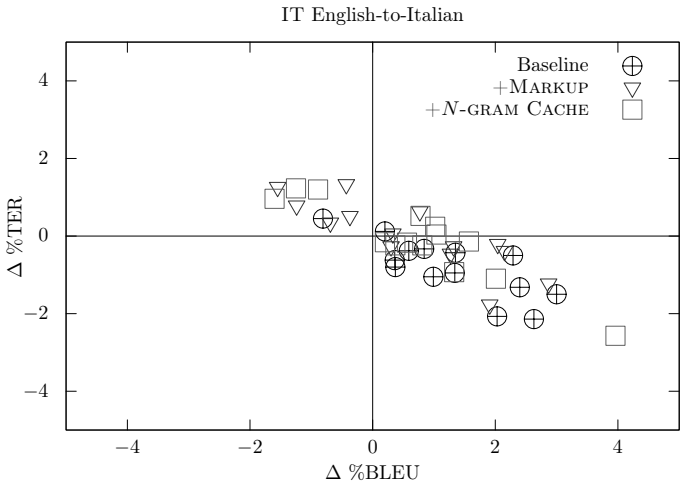

Figure 4.3: Results for the k-best reranking adaptation on English-to-Italian IT data: Positive results are located in the bottom right quadrant, with an increase in the BLEU score and a decrease in TER. Figure adapted from [Bertoldi et al., 2014].

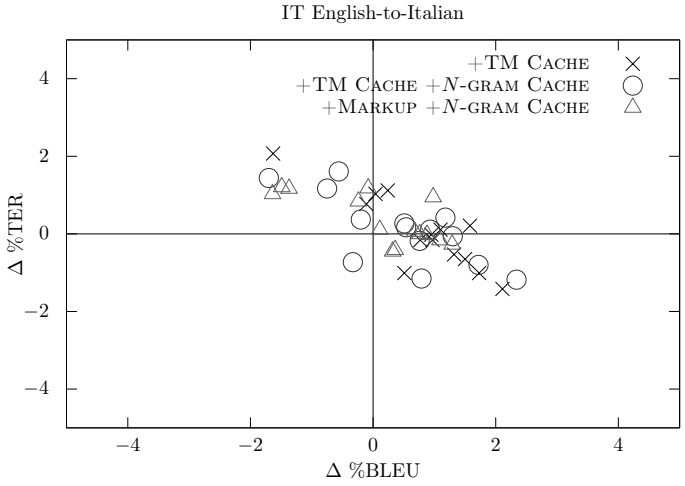

Figure 4.4: Second set of results for the k-best reranking adaptation on IT English-to-Italian data. Figure adapted from [Bertoldi et al., 2014].

overall a maximum of about +2.6 %BLEU and −2.1 %TER (Set-7A). Scoring with TER, two sets show deficient performance, one rather insignificant at +0.1 %TER (Set-0), the other being more pronounced (Set-6D at +0.5 %TER).

We also report on systems with applied adaptation by either the Markup local model or the N-gram Cache in Figure 4.3. We observe additional gains up to +2.1 %BLEU, but also reductions in quality up to −1.6 %BLEU when applying the reranker to a baseline system using the Markup adaptation. With the N-gram Cache we observe a similar behavior, although there are improvements up to +4 %BLEU and reduction in TER up to −2.6 %TER. This suggests a degree of non-additivity between the k-best reranking and these other local models.

We further tried reranking on top of the TM Cache and combinations of the translation model adaptation techniques and the N-gram Cache. Results are depicted in Figure 4.4. Again, we observe mixed results for these combination of adaptation methods.

4.6.2 Legal Data

For the Legal domain, training and test English-to-Italian and English-to-Spanish data is taken from the *JRC-Acquis* data [Steinberger et al., 2006]. For the test data a number of consecutive documents were extracted according to the *Eurovoc* subject classification (all data from selected subjects were excluded from the training data).

Test Set	# Segments	%RR Source	%RR Target	%BLEU Baseline	%TER Baseline
Set-0	551	11.8	12.5	36.5	49.2
Set-1	514	11.9	12.8	36.3	49.5
Set-2	571	18.2	16.7	37.8	43.7
Set-3A	91	9.4	7.5	46.4	37.8
Set-3B	91	8.5	7.9	49.7	33.1
Set-3C	91	8.8	9.0	32.6	44.4
Set-3D	91	9.7	7.4	49.0	35.0

Table 4.3: Statistics for test sets used for simulated post-editing experiments for the Legal English-to-Italian system, along with scores for the 1-best results. Table adapted from [Bertoldi et al., 2014].

Additionally, for English-to-Italian, a recent document was taken from the *EUR-Lex* platform and was translated by four translators (Set-3A–D).

On the English-to-Spanish data we explored how adaptation by k-best reranking performed using single documents with a low number of segments: Set-4–12 (Set-0 is the concatenation of these sets). Statistics for all Legal test sets are depicted in Tables 4.3 and 4.4.

The results for the English-to-Italian Legal experiments are depicted in Figure 4.5. On this domain and language pair, the results of the k-best reranker appear deficient in terms of both TER and BLEU. We suspect this is due to the bias towards higher N for the N-gram features and longer source phrases for the phrase-pair features. Reranking does not seem to be able to reliably improve over systems with translation model adaptation Markup and the N-gram Cache.

Following the English-to-Italian IT evaluation we also tried combinations of N-gram Cache and both of the translation model adaptation method in addition to the reranking. As for the previous results on this data, we do not observe much improvement when using the reranker on top of the other methods.

English-to-Spanish experiments are depicted in Figure 4.7. Although the same source segments were translated, results for the Spanish system are much improved by the reranking when used on top of the baseline system: up to −1.2 %TER reduction and +0.7 up to +1.4 %BLEU improvement for Set-0–2. A performance degradation occurs on one of the small documents which make up Set-0 (−0.3 %BLEU and +0.1 %TER). Notably, we observe improvements for all small data sets Set-3–11, with only two exceptions, where translation quality is reduced by a maximum of about −0.4 %BLEU and +0.1 %TER. This is evidence for a non-portability characteristic of the reranking models.

The results on top of the other adaptation methods look similarly: BLEU is not

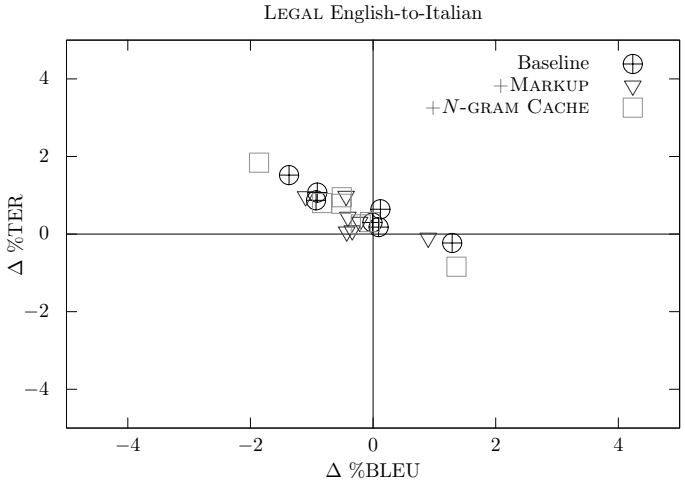

Figure 4.5: Results for the k-best reranking adaptation on Legal English-to-Italian data. Figure adapted from [Bertoldi et al., 2014].

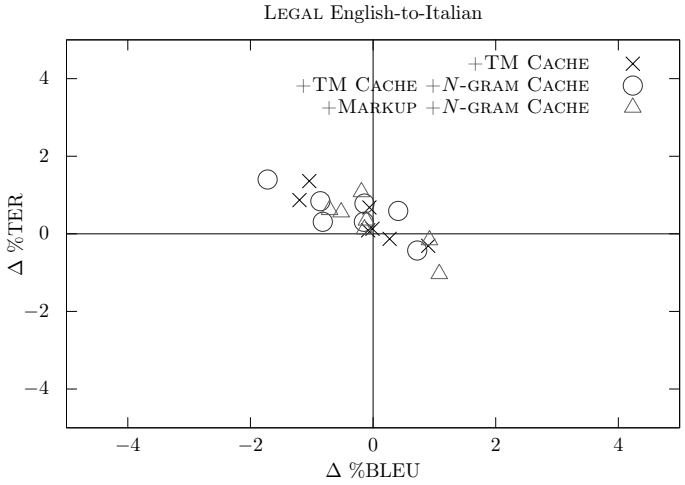

Figure 4.6: Second set of results for the k-best reranking adaptation on Legal English-to-Italian data. Figure adapted from [Bertoldi et al., 2014].

Test Set	# Segments	%RR Source	%RR Target	%BLEU baseline	%TER baseline
Set-0	551	11.8	14.8	42.4	43.2
Set-1	514	11.9	14.9	42.4	43.4
Set-2	571	18.2	18.1	42.6	40.3
Set-3	90	2.6	15.8	48.3	36.8
Set-4	30	13.5	14.6	34.7	50.2
Set-5	50	7.9	12.7	33.8	52.1
Set-6	40	18.4	19.2	38.3	48.3
Set-7	40	11.9	17.2	37.2	49.3
Set-8	60	11.2	13.8	40.4	44.0
Set-9	70	12.8	15.3	47.7	34.4
Set-10	70	11.7	12.7	42.8	40.2
Set-11	101	11.7	14.2	51.2	34.5

Table 4.4: Statistics for test sets used for simulated post-editing experiments for the Legal English-to-Spanish system, along with scores for the 1-best results. Set-0 is the concatenation of Set-3–11. Table adapted from [Bertoldi et al., 2014].

improved only in three of 24 cases, with a maximum degradation of −0.4. TER scores are higher in seven of 24 cases, with a maximum degradation of +0.7.

On this language pair and domain, reranking presents itself as a viable approach for improving translation quality. Compared to the English-to-Italian system we can observe, since Set-0-2 data sets are the same and also the used MT systems are similar, that the baseline translation quality seems to have an impact on adaptation performance — the baseline English-to-Spanish system performs about 5 to 6 %BLEU better on the same data sets (in opposing translation direction).

4.6.3 Patent Data

Lastly we experimented with German-English patent data extracted from the PatTR corpus [Wäschle and Riezler, 2012*b*], containing content from all patent sections (titles, abstracts, and descriptions).

For testing we used ten complete patent documents from section E ('Fixed Constructions'). The training data did not contain any data from this section. All documents were used in both translation directions. Statistics for the test sets are depicted in Table 4.5.

Results for German-to-English and English-to-German are depicted in Figures 4.8 and 4.9 respectively. For both translation directions, results with the k-

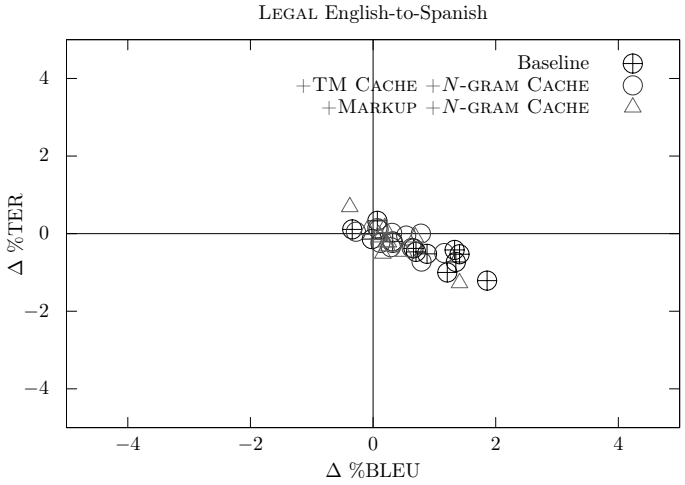

Figure 4.7: Results for the k-best reranking adaptation on Legal English-to-Spanish data. Figure adapted from [Bertoldi et al., 2014].

Test Set	# Segments	%RR Source	%RR Target	%BLEU Baseline	%TER Baseline
Set-0	318	15.7	8.9	22.6 / 34.8	60.1 / 45.0
Set-1	304	13.3	6.9	13.3 / 24.5	76.2 / 58.2
Set-2	222	21.1	15.6	20.8 / 26.1	67.3 / 61.8
Set-3	300	17.1	9.4	26.2 / 34.8	57.5 / 47.7
Set-4	227	19.8	13.8	17.2 / 24.1	66.6 / 63.0
Set-5	239	16.6	11.6	22.1 / 30.4	60.5 / 52.1
Set-6	232	17.7	11.9	16.6 / 26.3	68.1 / 57.7
Set-7	230	19.3	14.4	20.1 / 33.1	58.5 / 49.4
Set-8	225	16.9	10.0	26.6 / 36.2	53.9 / 49.0
Set-9	231	19.2	14.1	21.0 / 31.8	59.2 / 52.1

Table 4.5: Statistics for test sets used for simulated post-editing experiments for the Patent English-to-German / German-to-English systems, along with scores for the respective 1-best results. Repetition rates are depicted for the English-to-German direction. Table adapted from [Bertoldi et al., 2014].

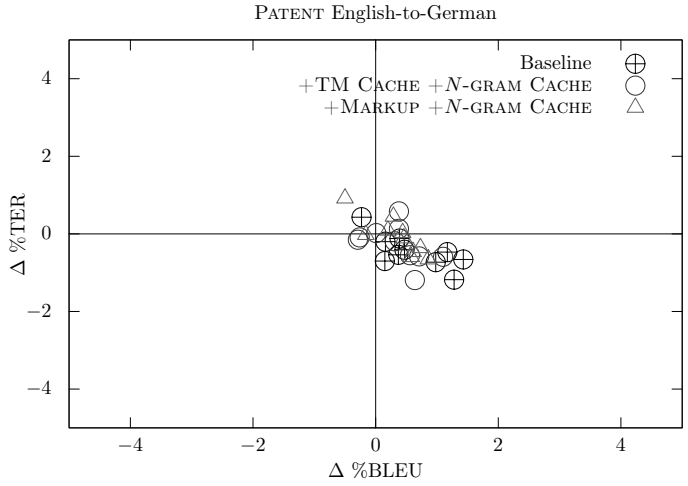

Figure 4.8: Results for the k-best reranking adaptation on PATENT English-to-German data. Figure adapted from [Bertoldi et al., 2014].

best reranker are improved for most of the documents and adaptation settings. Deficient or neutral behavior on both TER and BLEU is often observed adapting to SET-1, on both translation directions and all baselines. The improvements for German-to-English are strikingly better: While the maximum improvements on English-to-German are modest at about +1.4 %BLEU and −1.2 %TER, the gains for German-to-English are much higher at +3.5 %BLEU and −2.4 %TER. Note that the exact same data sets were used for both directions, which indicates that, since also the MT systems were similar, there is some language dependency. We also register that the baseline quality for the German-to-English system was largely better.

4.6.4 Analysis

For all data sets there were some documents where reranking could not improve the respective baseline results. Since sets SET-0–2 were used for tuning with MERT, improving over the tuned system is inevitably hard, even more so for an external tool, which cannot change the baseline system. Furthermore, since all reranking experiments use the **0** vector as starting weights, the initial predictions will be very weak, amplifying the difficulty improving on a data set that was used for tuning the baseline system.

We also observed that higher baseline translation quality had a positive effect

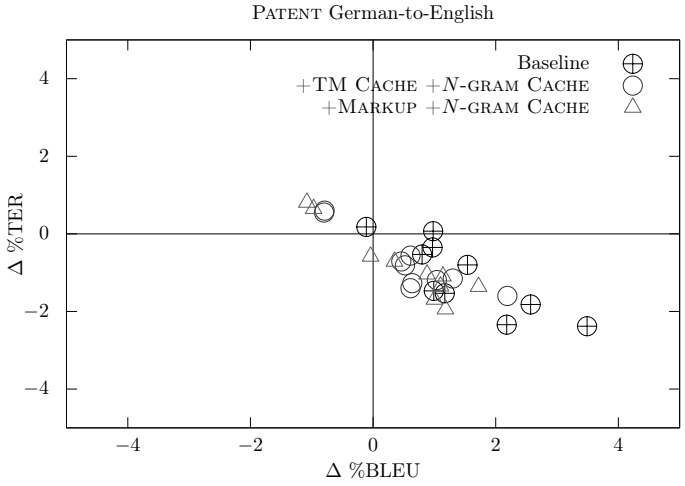

Figure 4.9: Results for the k-best reranking adaptation on PATENT German-to-English data. Figure adapted from [Bertoldi et al., 2014].

on adaptation results with reranking. However, adaptation by reranking on top of already adapted systems often does not lead to improved results. However, overall, as translation quality is often affected positively and the negative outcomes are not catastrophic, we conclude that the developed reranking method is a viable tool for adaptation. But we did observe some problems with portability of the models, which could be addressed e.g. by backward feature selection or similar techniques.

4.6.4.1 Repetition Rate

To further analyze cases where reranking lead to deteriorated results, we analyze the data in terms of the repetition rate [Bertoldi et al., 2013], which is defined as a geometric average over N-grams in evenly sized windows of consecutive tokens:

$$\mathrm{RR} = \left[\prod_{n=1}^{4} \left(\sum_{w \in W} \frac{|g_n(w)| - |g_n(w)|_1}{|g_n(w)|} \right) \right]^{1/4}, \tag{4.3}$$

where W is the set of windows within the data (we use a windows size of 1,000 tokens), $g_n(\cdot)$ returns all N-grams within a given window, $|\cdot|$ counts all N-gram types and $|\cdot|_1$ counts all types which occur only once (singletons).

Inspecting the source-side repetition rates and their averages, contrasting them with degenerate adaptation by reranking over the baseline system, we observe that

Data					Avg. RR
IT en-it	SET-0: 13.3	SET-6D: **9.9**			12.8
LEGAL en-it	SET-0: 11.8	SET-1: 11.9	SET-2: 18.2	SET-3A–D: **9.1** (Avg.)	11.2
LEGAL en-es	SET-3: **2.6**	SET-5: **7.9**	SET-8: **11.2**		12.0
PATENT en-de	SET-1: **13.3**				17.7
PATENT de-en	SET-0: **8.9**	SET-1: **6.9**			11.7

Table 4.6: Repetition rates versus average RR on sets with degraded translation quality through reranking (in terms of BLEU or TER).

most deficient results occur on below average repetition rates, see Table 4.6.

To further analyze this issue, we build an ordinary least squares regression model with the %BLEU and %TER deltas as dependent variables, predicted by the repetition rate for each document. While TER did not show any significant dependency on the repetition rate, a significant effect ($p < 0.03$) for %BLEU can be observed ($F(1, 51) = 5.550$, $R^2 = 0.098$). The improvement is about +1 %BLEU for each percentage point increase in RR.

Concerning the English-to-Italian LEGAL data, Bertoldi et al. [2014] and Bertoldi et al. [2013] also obtained mixed results for different adaptation methods using this data.

Wuebker et al. [2015*a*] confirm the connection between repetition rate and adaptation performance using patent data. Similar findings have been reported by Bertoldi et al. [2013], Bertoldi et al. [2014] and Cettolo et al. [2014*a*] for software manuals, legal documents and transcribed spoken language.

5 Learning Preferences from Human Interaction

> "Statistical inference is serious business."
> [Efron and Tibshirani, 1994]

As we established in the previous chapter, online adaptation is a viable approach in CAT to overcome the inherent issues of mismatches between the domain of data, the style of translators and the training data of MT systems, as well as their general shortcomings in modeling.

We argue however, that a more immediate, direct adaptation can be achieved by a richer interface between user and the MT system, or by using NMT to directly learn from user feedback without further approximation.

We therefore develop a new graphical user interface, allowing the edition of an SMT system's derivations. This enables the creation of rich feedback that can be used for adapting MT systems. We additionally develop a traditional editing user interface for PE with statistical- and neural MT backends to be used in translation user studies. We employ the interface for two studies, proving the effectiveness of both adaptation approaches.

First, related work in IAMT and user interfaces for CAT are discussed. We then describe our user interface design, and review evaluation in CAT. Then, our approach for direct adaptation of a traditional SMT system is described and evaluated. Finally, we present an effective adaptation approach for NMT systems, which is also evaluated in a real user study.

5.1 Interactive Machine Translation

First IAMT systems followed a pre-editing approach, helping the MT system in disambiguation with phenomena that it found unclear. This process is however not carried out prior to MT, but rather during translation, by interactions between a translator and the machine [Kay and Martins, 1970].

A more modern approach, similar to predictive typing [Darragh et al., 1990], is proposed by Foster et al. [1996] and Foster et al. [1997], and was later explored in the *TransType* projects for the word- [Langlais et al., 2000*a*] and phrase-based [Esteban et al., 2004; Casacuberta et al., 2009] SMT approaches.

While being an attractive concept, IAMT systems are difficult to implement: Both the MT system [Och et al., 2003*b*; Cubel et al., 2003; Ortiz-Martınez et al., 2016; Barrachina et al., 2009; Ortiz-Martínez et al., 2009; González-Rubio et al.,

2013] and the user interface[1] need special considerations. Due to the special interface, translators that want to use an IAMT system have to overcome a steep learning curve[2] [Alabau et al., 2012; Green et al., 2014*c*]. Furthermore, studies comparing IAMT approaches to traditional PE, show at best mixed for both time [Sanchis-Trilles et al., 2014; Green et al., 2014*c*; Underwood et al., 2014], as well as quality [Sanchis-Trilles et al., 2014; Green et al., 2014*c*; Underwood et al., 2014; Koehn, 2010] for the IAMT approach.

5.2 Immediate Adaptation from Post-Edits

As we have shown in Chapter 4, gains in translation quality are possible by exploiting feature representations of post-edited translations, given that the data set used is to some extent suitable[3] for adaptation. However, as shown by Wäschle et al. [2013] and Bertoldi et al. [2014], larger gains are possible by adapting the translation and language models directly. This finding is also confirmed by Denkowski et al. [2014*a*], who see greatest improvements by adaptation of the generative models, and not by adaptation of the log-linear weights.

Other approaches using k-best reranking[4], for online adaptation have shown mixed results: Cesa-Bianchi et al. [2008] only report an incrementally calculated sum of per-sentence BLEU scores on a test set of 1K sentences limited to 25 tokens per sentence, using target phrase- and word features, resulting in an average per-sentence improvement of about 0.07 %. Martínez-Gómez et al. [2011], report a maximum of 0.7 %BLEU improvement and 2.5 %TER improvement. Although these numbers are not comparable, these two data points in addition to our own work seem to suggest that the improvements possible by ranking or reranking are limited, at least when used in isolation.

In addition, different forms of IAMT were shown to be able to greatly improve translation outputs: Cheng et al. [2016] present a pick-revise framework, allowing a user to select parts of a translation deemed as correct, triggering a re-decoding step where the translation system is forced to use the selected parts as partial translations for the parts of the source to which the correct phrases are aligned[5]. While not presenting a model for adaptation using the feedback generated in this way, they present an evaluation via simulation which suggests that very large

[1] NMT systems can trivially support IAMT by *prefix decoding* [Knowles and Koehn, 2016; Peris et al., 2017*b*; Wuebker et al., 2016], but the user interface still has to be designed accordingly, and strict constraints concerning responsiveness have to be taken into account.

[2] Adopting the popular usage of the phrase "steep learning curve", i.e. being hard to learn.

[3] Suitable in terms of sufficient structure, e.g. as measured by repetition rates.

[4] Using oracle translations defined by per-sentence quality scores instead of the true reference translations.

[5] In their work, using a PBMT system, possible selections are constrained by the phrase segment given by the translation system, trivially allowing to use the alignment with each phrase for such partial forced decoding.

improvements are possible using the approach. The experiments also show that free-form corrections are superior to the left-to-right approach in their framework. A caveat of their approach is that since they rely on forced decoding, unreachable parts cannot be taken into account. Marie and Max [2015] previously presented a similar approach, also exploiting user feedback by selection of arbitrary parts of the translation hypotheses, but instead of using partial forced decoding, they employ adaptive phrase-tables and language models similar to the cache-based approaches of Bertoldi et al. [2013] and Wäschle et al. [2013]. However, instead of using a single cache, they propose "positive" and "negative" caches for discriminating correct from incorrect parts in re-decoding. The negative caches are filled with items not explicitly marked as correct by a user. They also show large gains in a simulation experiment. Gao et al. [2011] propose another approach for using adapted translation models by using a separate phrase-table, as well as Hardt and Elming [2010], which focus on an application in terms of post-editing. Domingo et al. [2016] present an evaluation of the general approach of Marie and Max [2015] focusing on post-editing effort. They also show that free-form selection improves over a left-to-right approach [Barrachina et al., 2009] in post-editing effort in simulated experiments.

Callison-Burch et al. [2004] present a user interface to fix incorrect alignments in the training data of a PBMT system, as well as means to leverage user corrections using a static word alignment model for re-alignment followed by a phrase-extraction step. Lastly, Sanchis-Trilles et al. [2008] show that information about correct parts of the hypothesis can be inferred by user behavior in the form of mouse actions, which can be successfully used for updating the hypothesis in an interactive translation environment.

In a different line of work, Albrecht et al. [2009] show, that by using a richer user interface [Albrecht, 2008], MT outputs can be post-edited more effectively by monolingual non-experts [Schwartz, 2014], helping with the ineffectiveness that was observed with monolingual PE [Nitzke, 2016; Mitchell et al., 2013]. Also, further methods have been tried for enabling monolingual PE, e.g. using additional aids inspired by IAMT [Koehn, 2010], or collaborative approaches [Hu, 2009; Hu et al., 2010; Yan et al., 2014].

Word alignments have been used for improving CAT systems: After presenting anecdotal evidence in [Schwartz et al., 2014], Schwartz et al. [2015] show that displaying word-alignment in the PE process may help post-editors when baseline machine translation quality is insufficient. Blain et al. [2012] use a pivotal word-alignment from machine translation output to post-edit inferred from a automatic string distance metric for learning corrections of the MT system.

Concerning specialized adaptation in IAMT, Foster et al. [2002*b*] show that in a

left-to-right IAMT prefix completion framework, the number of keystrokes that a translator has to perform can possibly[6] be significantly reduced by automatically maximizing the expected benefit of suggested completions. Wuebker et al. [2016] also show large gains (using a simulated evaluation) in terms of quality of suffix predictions for an IAMT approach when tuning towards specialized metrics such as *next word prediction accuracy* (WPA).

Inspired by these previous works on specialized IAMT and adaptation, we determined that a good CAT system with adaptation system would have the following features:

- Use PE for ease-of-use;
- Utilize MT systems that are able to output high quality translations hypotheses;
- Provide a rich representation of the post-edit that can be used for adaptation;
- For adaptation, adapt all parts of the model, with a focus on the generative models in SMT.

In the following we describe two approaches that satisfy these conditions.

5.2.1 Graphical Post-Editing Interface for Immediate Adaptation

User interfaces for PE are mostly constrained to the same basic setup, varying the positions of the text for the to-be translated source segment relative to the input field used for the translation.

Another possibly more important factor is introduced by additional aids provided to the translator, e.g. modules to access matches of a TM in addition or as an alternative to MT outputs, a concordance tool[7], or glossaries[8]. Access patterns to the to source material can also be designed, e.g. employing pagination, allowing or disallowing revision of finished translations, pre-translation of all source segments with an MT system (which has to be re-run after each finalized translation in an online adaptation setup), or enforcement of linear processing of the data.

Interfaces for prediction-based IAMT have to cope with additional complexities to display possible continuations of prefixes, and adhere to tight latency restrictions [Foster et al., 2002*a*; Green, 2014], while integration of MT output in a PE setup

[6] The evaluation is carried out by simulations.

[7] A concordance tool can query a corpus for occurrences of words, displaying typical contexts or usage examples.

[8] Possibly project-specific dictionaries, with additional information that can be used for disambiguation, similar to a termbase.

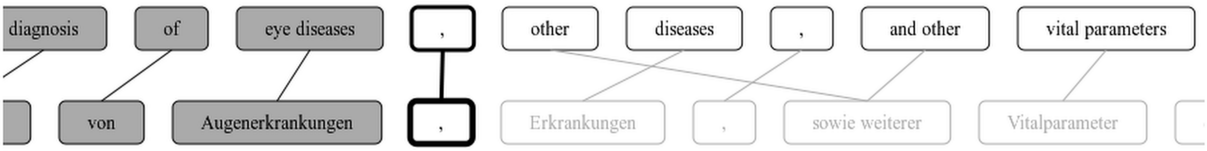

Figure 5.1: Partial view of an exemplary use of the graphical PE user interface. Figure adapted from [Simianer et al., 2016].

is straight-forward and does not impose such strong requirements to latency[9]. Examples for modern interfaces for research in CAT are described in [Denkowski et al., 2014*b*; Federico et al., 2014; Alabau et al., 2013*a*; Denkowski and Lavie, 2012; Aziz et al., 2012; Roturier et al., 2013].

For a first approach, we developed a graphical interface that enables extraction of specific corrections to the MT outputs which can be used for effective adaptation, without relying on approximations introduced by automatic alignment algorithms. An example of the graphical interface is depicted in Figure 5.1. When translating from English into German, the interface shows the source sentence on top, and the target translation on the bottom. The approach uses a phrase-based MT system. We visualize the phrase segmentation by boxes with rounded corners, phrase-alignments are shown as solid lines between the respective source and target phrases. The target can be freely edited, removal and creation of target phrases included. Alignment links between source and target can also be changed. Only the source segmentation is fixed in the interface[10]. All operations only apply for the row displaying the target phrases, but not the source row. Alignment links are added as well as removed by first selecting a source phrase and then a target phrase. The interface allows for arbitrary many-to-many alignments. The contents of the phrase are edited by the standard facilities of a text box, which in most cases also allows jumping between tokens by a keyboard shortcut and the positioning of the cursor by using a pointing device. To finalize the post-edition, all target phrases have to be explicitly marked as finished. For reference, the aligned source phrases are also marked as finished when all aligned targets were marked. Source and target phrases may remain unaligned, but a warning is issued before submission of the translation in such a case, allowing for cancellation of the submission.

In addition to the graphical interface, we also implemented a standard text-based

[9] There is no direct interaction between the user and the machine in PE which would impose such limits.

[10] We determined that allowing the edition of the source-side with the same degrees of freedom as the target-side would possibly overburden the users.

interface, replacing source and target rows by plain text fields. This allows for text editing (on the target side) by standard means provided by the operating system, which are the same as editing phrase contents in the graphical interface.

To the best of our knowledge, no such interface has been used for online adaptation in SMT before: There are tools for visualization of translation derivations for research purposes, which cannot be used for anything else besides inspection [DeNeefe et al., 2005; Weese and Callison-Burch, 2010]. Tools for research in PE [Koehn and Germann, 2014] and logging [Jakobsen, 1999; Carl, 2012] can also be useful for exploring the translation process, but were not used for adaptation. A similar interface to ours [Albrecht, 2008], has been shown to improve monolingual PE [Albrecht, 2008]. Alignment visualization is also utilized in evaluation of MT, providing further insights into the translation process [Girardi et al., 2014; Berka et al., 2012]. In other works (word-) alignment visualization is integrated in the interface, but serving no further practical purpose [Alabau et al., 2013*a*; Schwartz et al., 2014].

5.2.2 Phrase Alignment for Hierarchical Derivations

The phrase-alignment needed for the graphical interface would be trivial to achieve in a phrase-based system using flat, non-hierarchical phrases, since it is part of the output of the translation system. This is however not the case in the Hiero approach that we employ[11], as it allows rules with gaps, and derivations correspond to pairs of parse trees.

With Hiero and the constraints of allowing two non-terminals per rule, without allowing consecutive non-terminals, a total of 15 different rule shape configurations are possible on the target side. Out of the 15, nine allow to trivially produce a phrase-alignment since they consist of only a single consecutive group of non-terminal symbols which can be aligned using a single alignment link, disregarding the internal word-alignment.

Three shapes (six, when respecting the relative alignment of non-terminals) pose problems for the production of a phrase-alignment:

$$\mathsf{a\ X\ b\ X} \tag{5.1}$$

$$\mathsf{X\ a\ X\ b} \tag{5.2}$$

$$\mathsf{a\ X\ b\ X\ c} \tag{5.3}$$

[11] We use Hiero because of the good translation quality we obtained with the approach in our work on SMT optimization.

a and b being non-empty groups of terminal symbols. In these rules, since terminals contained in a, b or c can have arbitrary word alignments[12] to the source, it can be impossible to produce a valid one-to-one phrase-alignment. We resolve this issue by first establishing a source segmentation through heuristically grouping terminals by the hierarchical phrase boundaries, effectively disregarding the tree structure, to obtain an ordered set of source phrases. Then, for each target phrase, we find the source phrase with the maximum number of incoming word alignments, which is then taken to be the single aligned source phrase. If there is a draw, the first phrase is taken, when going from left to right. Using this heuristic, we can generate a single unambiguous phrase alignment from a hierarchical derivation. Also note, that this implies that larger phrase-pairs can be broken up in the process.

5.3 Adaptation

In this Section we describe how the graphical user interface can be exploited for adaptation of the translation model, describe the language model adaptation, and finally show how to use DTRAIN for online learning in this setting.

5.3.1 Translation Model Adaptation from User Edits

Allowing the user to directly edit the translation derivations of the SMT system enables the extraction of very fine-grained corrections. This can be straightforwardly implemented by generating the differential between the initial derivation and the final user-generated derivation. We always receive a full[13] phrase-based alignment since it is enforced by the user interface.

To this end, after post-editing, the current phrase-alignment (non-aligned phrases are prohibited) is mapped to a set of hierarchical translation rules, which are then compared to the rules used in the original derivation, and any rule that has changed is saved. The result of this procedure is a set of unweighted translation rules, which we refer to as EDIT.

In addition to the translation rules that are extracted from the edits, prior to translation, the source is checked for any tokens unknown to the MT system. We query the user to provide the system with suitable translations for all detected non-translatables. This check is done by comparing the input to the available rules in the corresponding per-sentence grammar. This results in a set of rules which we refer to as UNK.

To maximize the impact of the user edits, we further exploit the user-generated phrase-alignment to produce another set of rules, AUTO. We adapt the rule extraction algorithm of Chiang [2007] for this purpose: From the phrase-alignment,

[12] Note that this process could be carried out during the original phrase-extraction step.
[13] Given the user aligned all phrases.

we extract rules in the same way as in the original, word-based approach. Rules with a maximum of two non-terminals, and a maximum source length of eight words are extracted. We further forbid adjacent non-terminals on the source side, and the maximum seed size is three phrases. This results in a large set of additional rules.

During this process it can occur that rules are extracted that are already known, i.e. contained in per-sentence grammars — these rules are annotated by a new feature, denoting them as being seen in the adaptation process. Entirely new rules, only have a single feature, denoting them as new. Rules that fix non-translatables are also annotated with a separate feature. Note that the sets EDIT, UNK and AUTO are mutually disjoint.

Prior to the translation of a segment, the union of AUTO, UNK and EDIT is appended to the respective grammar.

5.3.2 Parameter Adaptation

Similar Denkowski et al. [2014*a*] we perform online updates of the SMT system's linear model after receiving a new training example. Instead of MIRA, we use our online pairwise ranking algorithm DTRAIN for this purpose. We employ the SPARSE feature set with our tuning algorithm, possibly adding millions of sparse features derived from translation rules, to be able to perform fine-grained model updates. Note that this applies also to the extended rule sets described before, since the additional rules also fire the respective features.

The general process is in principle identical to the regular online updates. Yet we need to make sure that there is no degenerate behavior combining the translation model adaptation and the online updates: Since all reference translations (the post-edits) are effectively reachable, due to the rule extraction, weights of rule-based features derived from the reference may be overestimated, especially from rules covering large parts of the reference, since less rule applications generally lead to higher model scores. We therefore only add the specifically extracted rules in EDIT to the grammar prior to an update, but not the larger set AUTO. The rules rules in UNK are added in any case, since they were used for the initial hypotheses used for post-editing.

5.3.3 Language Model Adaptation

Language model adaptation is carried out in the same way as proposed by Denkowski et al. [2014*a*], adding the post-edits to a hierarchical Bayesian trigram language model [Teh, 2006]. We update the language model prior to doing a parameter update.

5.3.4 Adaptation Scheme

The full adaptation scheme (see Figure 5.2) can be described as follows: After the client service requested a translation of a source segment, a remote service checks the source for non-translatables. If non-translatables are detected, translations for each non-translatable word are requested from the user via the local client. After this step, the remote service annotates the grammar as described, and the segment is translated and returned to the client. When the post-edition of the segment is finished, the server process receives a translation, alongside with a phrase-alignment and a set of rules that reflect the alterations done by the user (the EDIT set). The current grammar is then extended with the edited rules, and the adaptive language model is updated using the post-edit, followed by a parameter update, which includes another decoding step for generating a new k-best list including the new rules. Finally, and in any case prior to the translation of the next segment, the rule extraction process for generating the rules of AUTO is carried out with the current training example.

5.4 Implementation of an Online Adaptive Post-Editing System

Like other modern CAT interfaces, our approach is implemented in a client-server architecture, where the interface runs locally, but the MT engine, data storage and associated logic is implemented as a server component [Federico et al., 2014; Alabau et al., 2013*b*; Denkowski et al., 2014*b*]. On the one hand, due to its flexibility, this facilitates distribution of the system — on the other hand it is a technical necessity, since most MT systems cannot run properly on low-powered client machines.

The overall architecture for use with the graphical interface is depicted in Figure 5.2. The client and server processes are assumed to be constantly running. Once the server receives a client's translation request, the server first checks whether the payload transmitted with the request contains a finalized post-edit, which can be used for adaptation. If that is the case, a four stage update can be performed as described in Section 5.3. In the joint next step, a per-sentence grammar is either generated following Lopez [2007] or retrieved from storage. Since the source segment contains possibly non-translatable tokens, we intersect the source sides of the rules with the source sides of the per-sentence grammar before translation. If there are non-translatable words detected, the flow returns to the client and prompts the user for translations of each unknown token separately. After this step, the final grammar for the translation request can be assembled, which is then used in the decoding step alongside the current parameters. Since the user expects a naturally constructed sentence, i.e. without tokenization and with proper casing, a number of post-processing steps have to be carried out. These depend on the exact MT system, and typically contain a casing step and some form of de-tokenization[14].

[14]The exact procedure will be discussed when presenting the concrete models used in our

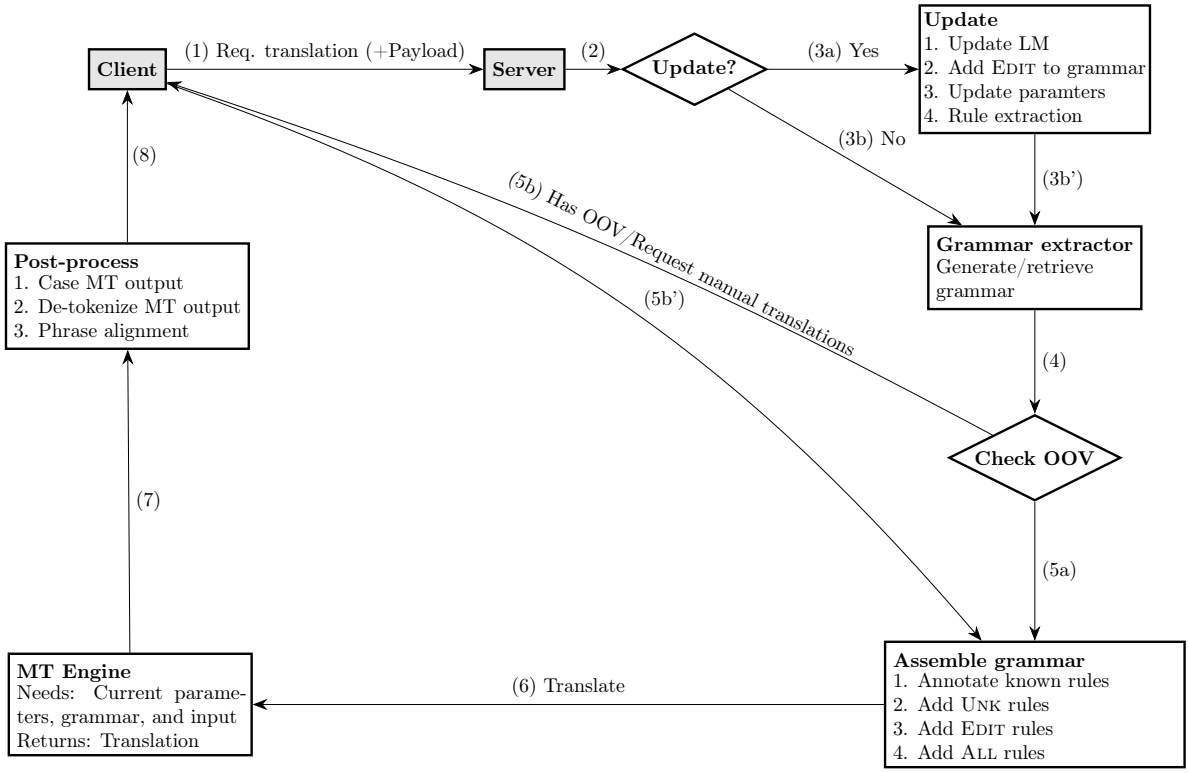

Figure 5.2: Server-client architecture of the online adaptive post-editing interface with statistical machine translation engine in a graph-based visualization with focus on server-side flow: Nodes with dark backgrounds represent abstract processes, diamond shaped nodes are junctions, and regular boxes represent processing steps. Figure adapted from [Simianer et al., 2016].

The last step is generating a phrase-alignment from within-rule word alignments, as described in Section 5.2.2.

5.4.1 Client Interface

The client receives an object containing all information necessary to display phrase-aligned source and target strings, or just the raw strings. A screenshot of the interface in text mode is depicted in Figure 5.3. A partial screenshot of the graphical editing model is shown in Figure 5.1, which replaces source and target fields. If the server process requests translations of unknown words, additional paired fields are displayed below the target inputs.

The graphical interface is controlled by means of keyboard and mouse. A full listing of keyboard navigation controls is shown in Table 5.1. It is implemented to mimic the behavior that the web-browser provides for standard text fields used in the text-based interface. The different modes can be accessed via additional keyboard shortcuts. The default mode is the navigation mode.

The full client-side application as depicted in Figure 5.3 has the following features: Areas marked with 1 and 2 are source- and target fields respectively, where the source field is not editable. Below the target field are four buttons: A press on the 'Help' button displays an on-screen help text, 'Pause' pauses the local timer, 'Reset' resets the post-edit to the initial state, 'Rate' (or 'Next') confirms the current user input and requests the next one. 3 is a rating feature which is not used in our current work, and 4 displays the overall progress, as well as all source segments.

5.4.2 Logging

The client interface logs all interactions with the graphical interface, or the text field in text-based mode. Keyboard or mouse actions outside these areas are not logged.

Upon receiving a translation, a timer starts, which can be paused by using a button in the interface, to account for longer unrelated breaks. The reset feature, while resetting all current progress, does not reset the timer. Translations of unknown words are not timed.

5.4.3 Efficiency

Although latency is not as much of an issue as in some IAMT approaches, the overall system needs to be efficient enough to not cause any long, unproductive pauses. The main source of latency in our approach is the adaptation process as described in Section 5.3.4, which is inherently not parallelizable and includes a

experiments

Common shortcuts	
Control + z	Undo
Control + r	Redo
Navigation mode	
←	Move cursor one phrase to the left
→	Move cursor one phrase to the right
Shift + i	Move cursor to the phrase farthest to the left
Shift + o	Move cursor to the phrase farthest to the right
Shift + d	Delete phrase at cursor's position
Shift + e	Edit contents of phrase at cursor's position
Move mode	
←	Move selected phrase one position to the left
→	Move selected phrase one position to the right
Enter	Accept current position and return to navigation mode
Mouse controls	
Click (on source phrase)	Enter, or leave alignment mode for this phrase
Click (on target phrase)	Align to currently source phrase if in alignment mode
Double Click (on target phrase)	Edit text

Table 5.1: Keyboard and mouse controls for the graphical post-editing interface.

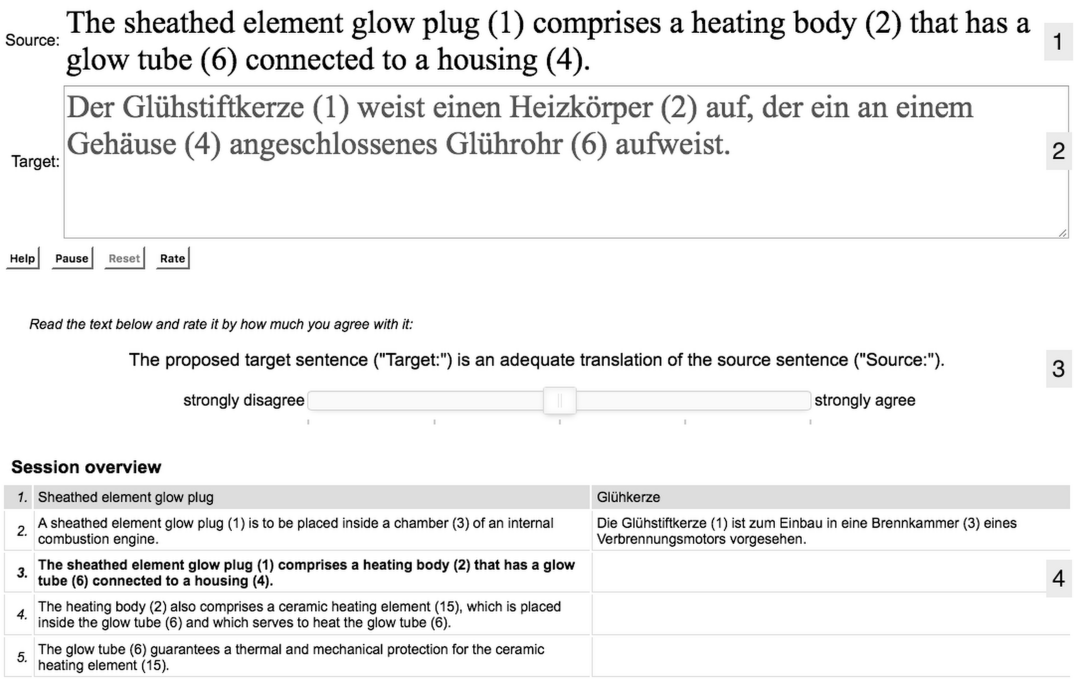

Figure 5.3: Post-editing user interface with text-based input. Screenshot adapted from from [Karimova et al., 2017].

number of expensive operations. Our model is geared towards not causing any pauses longer than 10 seconds between segments.

Experimenting with the suffix array-based translation model adaptation approach for Hiero of Denkowski et al. [2014*a*], we found that it can take several minutes[15] to produce a single grammar, which is unacceptable for our purposes. We thus we pre-generate all per-sentence grammars prior to translation.

5.5 Evaluation in Computer-Aided Translation

Evaluation in the CAT scenario adds a number of additional aspects to (machine) translation evaluation, being mostly confined to measuring quality by string- or semantic equivalence metrics.

After translation quality, the most important quantity to measure in the translation process is time, since it directly or indirectly affects cost of translation in the professional context. In terms of quality, a further aspect comes into play, since reference translations are naturally unavailable in a realistic setting, automatic quality estimation techniques [Specia et al., 2009; O'Brien, 2005] are needed if one does not want to carry out a manual evaluation. However, in arranged user studies, a ground truth is often attainable either by independently generated reference translations and/or by (bilingual) human evaluators which assess the correctness of produced translations.

There is also a more translator oriented perspective on evaluation in CAT — trying to measure technical and cognitive effort the translator has to invest in the translation process [Krings, 1997]. While technical effort can be readily measured by counting the number of observed user actions, such as keystrokes and mouse movements, measuring cognitive efforts is more complex, i.e. passively by making use of eye-tracking devices [Sekino, 2015; Sharmin et al., 2008], or by using the so-called read-aloud protocol [Krings, 1997], making the translator explain his thoughts on the translation, to gain insights into the cognitive process that is taking place while translating. Time is however a good indicator for cognitive effort [Koponen et al., 2012] (but not necessarily for technical effort), as is measuring the number and durations of pauses [Lacruz et al., 2014, 2012]. See [Lacruz et al., 2014] for an overview on pause-related measures for assessing cognitive effort.

For use in our own work we first discuss automatic evaluation of quality in CAT, then discuss how to efficiency and speed can be measured.

5.5.1 Measuring Speed

Time can be easily measured in CAT by including a timer which is active during translation. Different approaches can be followed for normalization, since raw time

[15] For suffix-array-based rule extraction for phrase-based models [Germann, 2015] efficient implementations have been described [Bertoldi et al., 2017].

is not a comparable quantity. Another aspect is how to handle pauses that occur in the translation process.

Normalization is preferably done using the length of the final translation or post-edit, since the source lengths can be considerably different depending on the language. This normalization by actually produced (or confirmed) characters or words is a realistic approach. In our work we normalize time by target characters (excluding spaces).

The total post-editing time can also be divided into a number of distinct phases, i.e. assessment-, editing- and reading time. See [Pinnis et al., 2016], for a more fine-grained analysis.

Another common approach to measure speed is throughput, which can be measured as words per hour, minute or working day.

Intuitively, post-editing time is positively correlated with source segment length [Popovic et al., 2014; Zaretskaya et al., 2016; Koponen, 2012].

5.5.2 Measuring Effort

In CAT, effort refers to any work that is done during the creation of a translation. It is an ambiguous term, since it can refer to technical effort, which is the actual amount, or to an estimate of the practical work that is done to create the final translation, or to cognitive effort, which can be described as the amount of mental processing that goes into the translation process. The cognitive effort can however be approximated by time measurements, see e.g. [Popovic et al., 2014].

Most commonly, technical effort is measured as a string edit distance, e.g. in post-editing between initial translation hypothesis and final post-edit. TER or human-targeted TER (HTER) as presented by [Snover et al., 2006] is a standard approach. However, since the original procedure is costly to carry out[16], the most common method is to simply calculated TER of the MT output against a single post-edit which created from just the same MT output. The same approach can be taken with the BLEU score, or sentence-wise BLEU for per-sentence measurements.

The previously described methods only measure technical effort indirectly, since they only consider the minimal amount of edits needed to arrive at the final translation.

In a user study (or by simulation with certain assumptions) a direct approach can be taken, by directly recording user actions, e.g. keystrokes and mouse actions. Barrachina et al. [2009] propose normalization over character in the post-edit, resulting in three metrics: keystroke ration (KSR[17]), mouse-action ratio (MAR), and keystroke and mouse-action ration (KSMR), which is just KSR + MAR.

[16] It involves an independent reference translation, as well as a number of different, independently generated post-edits.

[17] Sometimes also word stroke ration (WSR).

Koehn and Germann [2014] present another method to estimate technical effort – character or word provenance, which can measure which characters or words had to be actually typed and which ones were automatically proposed.

Finally, O'Brien [2011] proposes to measure cognitive effort through observation of translators using eye tracking. Measurements of fixation time and counts can give an estimate of the effort involved.

5.5.3 Measuring Quality

Final translation quality in CAT can be straight-forwardly measured if there are independently created and validated reference translations available. However, if these are not available, one has to resort to manual human judgments or to automatic quality estimation techniques.

5.6 User Studies

User studies are commonplace in CAT for industry and research applications to evaluate a setup in a realistic scenario, while having the ability to control the parameters of the environment. There are however many difficulties that must be considered when designing a user study: In comparative studies, where two or more setups are to be compared, the main problem arises from the non-repeatability of using the exact same variables in different conditions, i.e. it is not straight-forward to present the same translator the same text under two different conditions, which would be ideal for testing a hypothesis. But it is not expected to obtain an objective result under this specification. To overcome this inherent non-repeatability, one can either try to select translators that are similar to one another, or use texts with similar qualities, or experiment with introducing a very long pause between two experiments. These options are often not practical, since translators with similar backgrounds are often not available, similar but sufficiently differing text material might be impossible to find, and long breaks are often just unpractical or leave room for doubt in the results since it depends on the translators' memories whether the findings are valid or not.

Another aspect of the difficulty of user studies is the associated cost, since the results do not add any real value besides the gain in knowledge obtained by analyzing the results. Also, the studies are usually carried out in a single shot, leaving no room for error or variation in the experimental setup. A last point concerning the cost, is that user studies usually compare a system's performance at one point in time, ignoring further system development, e.g. improvements the baseline systems. When testing a new user interface, an additional problem arises as there is usually a learning curve involved when changing the interface, which has to be taken into account e.g. by having additional training sessions which also adds to the overall costs.

Concerning the used textual material, further issues arise when using publicly available corpora. Another important question to answer is whether the provided reference translations are verifiably of good quality, as it determines whether a comparison is insightful or not. Furthermore, when comparing against third party systems using publicly available corpora, it is important to verify whether or not the third party system has the proposed text material in their training data.

Whether to use a standardized workflow and work environments should also be defined when carrying out a user study. Using publicly available data, it is also important to make sure that the reference translations are not available in generally available search tools. A last point worth mentioning is the motivation of the study's participants, which may be suboptimal, given the nature of the testing environment.

Given these general observations, we review related published works on user studies in CAT.

5.6.1 User Studies in Computer-Aided Translation

User studies are common place in CAT, e.g. for comparing post-editing or IAMT with translation from scratch, or to traditional translation aids such as translation memories (Guerberof [2009]; Arenas [2008], Flournoy and Duran [2009], Federico et al. [2012] Zhechev [2012], Plitt and Masselot [2010], De Sousa et al. [2011], *inter-alia*). We present a brief review of related works in what follows.

Arenas [2008] and Guerberof [2009] present a study on PE versus translation from scratch and the use of a TM. Their experimental design includes nine professional translators, and a single document which is statically split into three approximately equal parts. These parts were translated either from scratch, with the help of a TM, or via post-editing by all nine translators. This way each part is translated three times under each translation condition. Standard statistical measures are reported per part, averaged over translators' outputs. This resembles a by-item analysis, where the items are the document splits.

Langlais et al. [2000*b*] report a by-subject analysis in terms of efficiency of the usage of an IAMT system.

In [Plitt and Masselot, 2010], the authors present a study on PE versus translation from scratch. For the study, a test set including data from three domains is to be translated into four target languages, with three distinct translators per language. Each translator receives at least six translation jobs, translating from scratch and post-editing data from each domain. In their analysis they opt for a by-subject design, reporting average improvements in words per hour per translator, comparing averages over different portions of text as well as from different domains.

Flournoy and Duran [2009] present a study on PE for two language pairs, comparing PE productivity results to an anecdotal baseline of 2,500 words per

working day (eight hours) of an "average" translator. Beaton and Contreras [2010] also provide anecdotal evidence of productivity improvements when using PE instead of translation from scratch.

Pouliquen et al. [2011] present a specialized user interface for patent translation, and present a user study reporting human judgments of resulting translation quality.

Federico et al. [2012] describe a field test involving twelve professional translators, translating data from two domains from English into German and Italian, where each translator only translates data from a single domain. The translated materials are only the same within each domain. Half of each document is translated without MT suggestions, the other half with MT suggestions (in addition to TM matches), which implies that each translation condition is observed on only 50% of the data, and the same 50% overall translators. The results are presented as a by-subject analysis, but without a comparison between domains.

In [Zhechev, 2012] a study on PE for nine language pairs is presented, with four translators per language pair. The translation conditions, PE and translation from scratch, are tested on pairs of similar documents, each translated either by PE or from scratch by each translator. Results are presented by language pair, which resembles a by-item analysis.

In a study comparing PE and translation from scratch, De Sousa et al. [2011] and Aziz et al. [2012] report results of eleven non-professional translators. The translators are divided into two groups, one group translates the first of two test sets by translating from scratch, and the second test set through PE, while the other group does the reverse. For PE four different MT systems are used, and the group that is post-editing translates with every system, translating the same set four times. Results are reported per MT system, comparing averages over test sets and translators, as well as per translation condition (PE and from scratch), which both can be characterized as a by-item analysis.

In one of the earliest studies comparing post-editing of MT to translation from scratch, Orr and Small [1967] report results for Russian-to-English on non-overlapping test sets. Much later, Macklovitch [2006] present a number of results comparing an IAMT approach to translation from scratch, reporting improvements in terms of raw words per hour measurements.

The authors of [Koehn, 2009] and [Koehn and Haddow, 2009] analyze the translation outputs in a French-to-English translation task of ten non-professional translators (five native speakers of French, and five native speakers of English), and compare five different translation conditions. A single shared test set is split into five blocks, and each block is translated under every translation condition, once by a native speaker of French and once by a native speaker of English. Results are reported by-subject, consequently comparing translations of different parts of the data.

Läubli et al. [2013] present a user study, which aims to evaluate the efficiency of PE in a realistic translation scenario, comparing traditional usage of a TM to PE.

Their study is carried out with six non-professional translators, with only have native language skills in the target language. The data used for the experiments is divided into four parts. Each translator translates all the data, but the assignment of which one is translated by PE or translation from scratch is random, while making sure that 50% of the documents a translator receives are post-editing tasks. Their analysis is done by comparing averaged measurements for post-editing and translation from scratch for each document, rendering it similar to a by-item design. They also report on a statistical analysis similar to [Green et al., 2013*a*], which accounts for by-item and by-subject variations in a single model.

Koehn and Germann [2014] present a user study which aims to compare different MT approaches and their impact on translation speed in a PE environment. In their experiments, four bilingual non-professional translators are presented with the output of four different MT systems for post-editing. A single test set is translated with all four systems, and each translator translates the full document once, seeing a randomly chosen MT output for each source segment to post-edit. This corresponds to a random split of the test data with a random assignment to translators per MT system. The variable of interest is the varying performance by using the different MT systems. Their results are either averaged over translators for each MT system, or per translator over the whole document including the different MT systems' suggestions. This setup can be considered as a by-subject analysis in both cases. The authors also provide results of significance tests for the comparison between MT systems, once global and once per translator.

The study of Gaspari et al. [2014] compares PE to translation from scratch, for two language pairs, and in four translation directions. For each translation direction, the authors selected a distinct test set, which was translated by two translators, one translating one half from scratch, and the other half by post-editing. The second translator does it in the reversed order of the first translator. Results are reported by language pair, i.e. by-item.

Sturgeon and Lee [2015] present a study comparing an IAMT approach to post-editing. Their study involves 24 non-professional translators, translating data from two distinct domains. For each domain a single test set is used, which is translated by twelve translators using the IAMT approach, and the other twelve translate using traditional PE. Their results are summarized in a by-item analysis, reporting averaged results for each test set and translation condition.

Bentivogli et al. [2016*b*] present a user study comparing several MT paradigms. In the study, offline generated outputs of all systems are post-edited, and HTER and BLEU scores (against independent references) for the MT outputs are reported.

Garcia [2011] present a total of three experiments, summarizing earlier results [Garcia, 2010]. All experiments use a distinct test set, which is split into two parts — one part translated from scratch, and the other part translated with via PE. Each part is translated under the same translation condition an equal number of times. The first two experiments each use translations of 14 translators in training,

28 in total. The third experiment employs 21 translators. The authors perform a simplified by-item analysis, reporting an averages over translators. They also perform a statistical hypothesis test for the independent variables translation speed and -quality.

Sanchis-Trilles et al. [2014] present a field test with nine translators, comparing three translation conditions (using various CAT aids). Three data sets are used in the study, which are further split into another three parts. Each translator gets to edit every part of each document with a different translation aid, such that every translator uses each aid three times, and every part of a document is translated with each of the aids three times by different translators. The authors present results averaged per translation condition, by-subject and -item.

Tatsumi [2010] presents a long study analyzing PE effort in English-to-Japanese translation. To this end, amongst other things, a number of multiple linear regression models are presented, including various source-side characteristics, number of edit operations, and translators as fixed effects. Post-editing time is used as main dependent variable. The analysis spans a single test set, all nine participants use PE for translation.[18]

Scheepers and Schulz [2016] report a user study on an IAMT approach with prefix completion. Four professional translators and two novice translators translate two sets of three documents, one set with suggestions from a monolingual language model, and the other set with suggestions from a NMT system. Their results are analyzed using a Bayesian linear regression model.

Castilho et al. [2017] present results of a user study on NMT versus SMT post-editing, comparing measures of technical effort for post-edits generated by the same translator for the same text.

Green et al. [2013*a*] present a user study, comparing PE to translation from scratch on data for three language pairs. The data for all language pairs consists of four paragraphs of coherent text (within each paragraph), and the assignment to translation condition is randomized per paragraph. 16 translators are employed per language pair, and all of them translate the same source segments. For analysis of the data, a linear mixed-effects model is used, with time and quality as dependent variables, and subjects as well as items (the source segments) as random effects, including the translation condition as fixed effect which acts as the primary independent variable. Green et al. [2014*b*] and Green et al. [2014*c*] present similar studies for contrasting PE to an IAMT approaches.

Our review shows that a wide range of experimental designs were used in CAT research, which are mostly restricted to either by-item *or* by-subjects analyses with the notable exceptions of [Tatsumi, 2010], [Scheepers and Schulz, 2016], [Green et al., 2013*a*], [Green et al., 2014*b*] and [Läubli et al., 2013], the latter three works

[18]The study also includes a brief comparison in terms of speed of post-editing to TM matches.

establishing linear mixed-effects models for analysis of CAT user studies. These statistical models enable to account for by-item *and* by-subject variance [Clark, 1973; Forster and Dickinson, 1976]. Otherwise, only Sanchis-Trilles et al. [2014]'s experimental design includes redundancy in translation condition by item and subject. Other studies report simple averages over items or subjects, instead of by-item and by-subject variance analyses [Baayen, 2008] or a mixed-effects analysis. The simpler methods provide no means to assert generalization beyond the sample at hand.

5.6.2 Studies on Adaptation

The majority of research on adaptation in CAT resorts to simulated feedback using offline generated references. This is due to the many issues that arise when carrying out a study with real user feedback, which can be infeasible when developing a new adaptation approach. While this is in general a valid approach, it has the downside that there is no clear relation between the suggested translations and the simulated post-edits. For example, the distinction between cognitive and technical effort is hard to assess with simulations.

After the first approach to adaptation of PBMT described by Nepveu et al. [2004], many adaptation approaches for the PBMT approach were developed [Martínez-Gómez et al., 2012; Ortiz-Martínez, 2016; Arun and Koehn, 2007; Hardt and Elming, 2010; Gao et al., 2011; Levenberg et al., 2010] (*inter-alia*), often focusing on tuning [Wuebker et al., 2015*a*; Mathur et al., 2013; Mathur and Cettolo, 2014; Cettolo et al., 2013]. In most of these works results on regular MT metrics or simulated experiments are reported. Some papers also show learning curves for the proposed methods [Mathur et al., 2014], *inter-alia*. Other approaches use reranking methods [Cesa-Bianchi et al., 2008; Martínez-Gómez et al., 2011] (*inter-alia*). Adaptation methods developed for hierarchical phrase-based systems are more rare [Denkowski et al., 2014*a*], *inter-alia*. Adaptation in NMT was also primarily explored with simulations [Turchi et al., 2017; Peris et al., 2017*a*].

Some studies with a focus on CAT also report results using real post-edits, generated from the same or similar system that is used during online adaptation [Bertoldi et al., 2014; Wäschle et al., 2013; Cettolo et al., 2013]. Most of these evaluations report MT metrics, comparing MT outputs to reference translations. Green et al. [2014*c*] also report on incremental re-tuning [Forcada et al., 2017] towards HTER [Denkowski and Lavie, 2010], using post-edits that were produced using a baseline system.

Most work on IAMT has also been confined to simulations, including systems using NMT [Knowles and Koehn, 2016; Peris et al., 2017*b*]. However, some studies do actually try to simulate user inputs to report on metrics such as KSMR [Ortiz-Martínez et al., 2010; López-Salcedo et al., 2012; Wuebker et al., 2016].

One of the first studies evaluating adaptation by a user study, is presented by Alabau et al. [2014], who seek to evaluate adaptation approaches compared to a non-adapted system in an IAMT setup. The study involves three professional translators and one non-professional translator. In a first evaluation, a single translator translates comparable source texts in different translation conditions. In the second evaluation, three professional translators translate the same text, two using an adaptive system, while the third one uses a system without adaptation. The results are reported as per-translator timing results, showing that in the second experiment less keystrokes were used when translators made use of the adaptive system. The first experiment shows improved throughput.

The online adaptation approach of Denkowski et al. [2014*a*] for the Hiero MT paradigm is evaluated in a user study reported in [Denkowski et al., 2014*b*][19], Five translators in training had to translate four texts, two by using an adapted system, and two by using a static baseline system, such that every text is translated more than once in any translation condition. Results are reported as aggregated HTER scores for the baseline and adapted system over full the full set. The authors of the study report a reduction in HTER of about two percentage points, as well as improved human judgment ratings when using the adapted system. In a study that used the same general approach, Lacruz et al. [2014], evaluate with five translators in training whether online adaptation can passively reduce cognitive effort, as measured by the number of pauses divided by the number of words[20], in addition to HTER. The study includes no comparison to a non-adapted system.

In a user study including batch and online adaptation presented by Bentivogli et al. [2016*a*], the authors propose an experimental setup that enables a by-item and by-subject analysis similar to e.g. Green et al. [2014*c*], including an analysis with linear mixed-effects models. Their approach is evaluated with 16 translators, each translating 820 segments in total in either of two language pairs. This includes 368 segments used for familiarization with the interface (the outputs are however used for adaptation of the later test systems for the adaptive case). In contrast to previous work on CAT evaluation, instead of testing translation conditions with the same translators on different data for each translation condition, each translator translates the same text twice, once for each translation condition. The authors propose a one month gap between the two experiments. Their results show significant improvements in post-editing effort, as measured in HTER, and also significant improvements of the quality of the original MT suggestions, as measured in BLEU, against independent reference translations.

[19] Also published in [Denkowski, 2015].
[20] Pauses to word ration (PWR).

5.6.3 Evaluation of User Studies

Evaluation of novel approaches in CAT suffers from the inherent non-repeatability of experiments, since the same translator can hardly be exposed to the same text under differing translation conditions to directly test the impact of an innovation on translation performance.

Additionally, large variability has been observed in both textual materials, as well as translators [Koponen, 2013; Koehn and Germann, 2014], which is commonly ignored when reporting (averaged) by-item or by-subject measurements, hiding the underlying variance in a single value. This variability is however a natural phenomenon, since every subset of text or translators can naturally only be small samples from very large distributions. This is why these factors — language material and translators — should be modeled as random effects, and not as fixed effects, e.g. estimating linear regression models. This is the classical *language-as-a-fixed-effect-fallacy* [Clark, 1973; Coleman, 1964], which, when ignored, can lead result in *Type I* errors, asserting statistical significance of differences when there is none (rejection of a true null hypothesis[21]).

A common remedy to this problem are mixed models as introduced by Laird and Ware [1982], combining fixed and random effects in a *mixed* model for improved variance analysis [Baayen et al., 2008; Judd et al., 2012].

Linear mixed-effects models, as used in our work, can be written similar to ordinary linear regression models:[22]

$$y_{ij} = b_0 + \sum_{k=1}^{p} b_k x_{ijk} + v_{i0} + \sum_{l=1}^{q} v_{il} z_{ijl} + \varepsilon_{ij}, \tag{5.4}$$

where $(y_{ij}) = \mathbf{Y} \in \mathbb{R}^{n \times m_i}$ is a response matrix for n subjects and m_i responses per subject[23] i, $b_0 \in \mathbb{R}$ is the global fixed intercept, and $v_{i0} \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma_0^2)$ are per-subject random intercepts. $\varepsilon_{ij} \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma_\varepsilon^2)$ are per-item and per-subject Gaussian[24] error terms. The p fixed-, and q random effects, measured per-item j and subject i, each have associated slopes. While the fixed slopes $b_k \in \mathbb{R}$ are shared between subjects and items, the random slopes $v_{il} \overset{\text{iid}}{\sim} \mathrm{N}(0, \sigma_l^2)$ are estimated per subject. Finally, $x_{ijk} \in \mathbb{R}$ and $z_{ijl} \in \mathbb{R}$ are the measurements for fixed- and random predictors respectively.

[21] Type II errors being the failure to reject a false null hypothesis, i.e. declining a significant difference.

[22] Presentation here largely based on [Helwig, 2017].

[23] Note that the number of measurements may be different per subject, which can be adjusted by padding with 0.

[24] Note that *generalized* linear mixed-effects models permit other families of distributions [Lo and Andrews, 2015].

This model can also be written more compactly in matrix form for each subject i:

$$\mathbf{y}_i = \mathbf{X}_i\mathbf{b} + \mathbf{Z}_i\mathbf{v}_i + \boldsymbol{\varepsilon}_i, \tag{5.5}$$

where $\mathbf{X}_i$, and $\mathbf{Z}_i$ are the fixed- and random effects design matrices respectively, $\mathbf{b}$ and $\mathbf{v}_i$ the fixed- and random-effects vectors, and $\boldsymbol{\varepsilon}_i$ the error vector. Per-subject responses are collected in the vector $\mathbf{y}_i$.

Linear mixed-effects models can be estimated using generalized least squares or maximum likelihood estimation (MLE). For our work, and also following Bentivogli et al. [2016*a*], we use the implementation described in [Bates et al., 2014] which uses MLE.

5.7 User Study with the Graphical Interface

We performed a user study to evaluate the effectiveness of the proposed adaptation scheme and the graphical interface. In this first experiment, we opted for a by-subject and by-item analysis, which we carry out using a linear mixed-effects model.

For the study, 19 students were recruited, six studying computer science and/or computational linguistics, and the remaining 13 being translators in training, all associated with the Heidelberg University, Germany. The group of participants had a diverse set of mother-tongues, nine students are German, the others are native Italian- (seven), Spanish-, Arabic- or Russian (each one) speakers. This data was collected anonymized and independently, which is why there is no mapping between mother tongue and field of study. All students are fluent in written and spoken German, according to a standardized test [Jung, 1995] (minimum level *DSH-2*), as required for being enrolled in the university.

The experiment was conducted in separate 90 minute sessions, and took place in a controlled environment in a computer pool. In each session at least one supervisor was present to provide an exam-like environment. Access to the internet was available, including online dictionaries and reference material. In some isolated cases, the time limit of 90 minutes was slightly extended in favor of finishing the assigned task.

5.7.1 Data Selection & Machine Translation Model

For this study we use patent data [Wäschle and Riezler, 2012*b*,*a*], since it allows to build SMT models from comparatively small data, which however provide sufficient baseline translation quality on in-domain data, which is important for our PE task, since we employ non-professional translators. Additionally, the data is diverse, covering a wide range of topics. However, since our participants are

Sheathed element glow plug

A sheathed element glow plug (1) is to be placed inside a chamber (3) of an internal combustion engine.

The sheathed element glow plug (1) comprises a heating body (2) that has a glow tube (6) connected to a housing (4).

Figure 5.4: Title (underlined) and excerpt containing two sentences of the abstract of patent WO-2007000372-A1.

Sheathed-element glow plug

A sheathed-element glow plug (1) serves for arrangement in a chamber of an internal combustion engine.

The sheathed-element glow plug comprises a heating body (2) which has a glow tube (5) and a heating coil (8) which is arranged in the glow tube (5).

Figure 5.5: Title and excerpt containing two sentences of the abstract of patent WO-2007031371-A1.

neither professional translators, nor familiar with patents, patent translation, or any subject matter, we restrict our experiments to patent titles and abstracts, since the other sections (descriptions and claims), can contain overly complicated segments which partially follow standardized language restrictions. Examples for this type of data are depicted in Figures 5.4 and 5.5.

To obtain a clean split between training and testing data, we first split the data by year, and then by its family patent identifier[25]. Practically, we select data from 1995 to 2005 for training, data from 2006 as development data, and data from the years 2007 and 2008 as testing data. We remove all data from development and training sets that share a family id from any patent used in the test data. We repeat this process for development and training data, removing data from training if it shares a family identifier with any patent used in the development data. The resulting training set contains about 350K segments.

In addition to the parallel data, there are about 16M target side segments available for language model training.

[25]This identifier groups related patents which are filed in different countries, e.g. patent applications by the same entity or for the same inventions.

The test data is split into groups of about five documents to obtain translation tasks of similar size for our experiments. The initial 23,048 documents (a document being an abstract with corresponding title) in the test data are further reduced by to 3,751 documents by selecting only those documents, where each source segment in the abstract has at most 45 tokens (in the fully pre-processed version). From these documents, 2,075 are discarded since they contain less than four total segments, resulting in the final set of 1,676 documents.

These documents are grouped into translation tasks of about 500 words[26] by the following procedure: First a full cosine similarity matrix over all documents is built, using a bag-of-words approach with *tf-idf* values [Sparck Jones, 1972] and applying stop word filtering [Singhal et al., 2001]. Each document (in random order) is then grouped with similar documents until the limit of 500 words is reached, with minimum and maximum similarities of 0.05 and 0.95, while preferring more similar documents. This process is executed until all documents are used, or the constraints cannot be satisfied anymore. Most groups contain three to five documents. This process is carried out to ensure that a translation task consists of similar sentences, to be able exploit similarities during training. Note that this process in entirely carried out on the source-side, which means a similar approach could be used in practice, e.g. assigning similar projects to the same translators in a corporate environment.

We built hierarchical phrase-based translation models for English-to-German from the training data using truecasing, where the first token of each segment is normalized to its most common form [Lita et al., 2003], to avoid having to employ a separate re-casing system. The translation model is built using all available parallel training data. A 5-gram language model is trained using all of the available German data. The adaptive Bayesian language model is trained with 5% subsample of the full monolingual data, since training time is otherwise infeasible, even when using only trigrams.

We use DTRAIN to tune the weights of the SPARSE feature set on the available development data. The development data contains 46,853 segments — we use a random sample of 2,000 segments for development testing, and the remaining 44,853 segments for tuning. We employ our ITERMIXSELSGD algorithm, with shards containing about 2,000 segments, in conjunction with the random re-sharding technique. We perform four tuning runs to account for optimizer instability. Additionally, since we want to prevent too harsh changes to common features, we implemented a distributed variant of the *ADADELTA* per-coordinate learning rate method Zeiler [2012]: After each epoch, each shard returns a vector of learning rates, as well as its current weight vector. While weights undergo the ℓ_1/ℓ_2 feature

[26] Taking a conservative words per hour value of about 333 words per hour as basis.

selection (selecting 100K features), the learning rates are simply averaged, and distributed together with the reduced weight vector for the next training epoch.

The ADADELTA method can be defined as follows for a single parameter k of the weights $\mathbf{w}$: After computing the gradient ∇, we first update its squared expectation (starting from 0 for $i = 0$) by accumulating the gradient:

$$\mathrm{E}[\nabla_k^2]_{(i)} = \rho \mathrm{E}[\nabla_k^2]_{(i-1)} + (1-\rho)\nabla_{k,(i)}^2, \tag{5.6}$$

$\nabla_{k,(i)}$ being the kth component of the ith gradient during training.
We then compute the actual update as:

$$\Delta \mathbf{w}_{k,(i)} \leftarrow -\frac{\sqrt{\mathrm{E}[\Delta \mathbf{w}_k^2]_{(i-1)} + \epsilon}}{\sqrt{\mathrm{E}[\nabla_k^2]_{(i)} + \epsilon}} \nabla_{k,(i)}, \tag{5.7}$$

and accumulate the actual updates:

$$\mathrm{E}[\Delta \mathbf{w}_k^2]_{(i)} \leftarrow \rho \mathrm{E}[\Delta \mathbf{w}_k^2]_{(i-1)} + (1-\rho)\Delta \mathbf{w}_{k,(i)}^2. \tag{5.8}$$

Finally, update the weights by:

$$\mathbf{w}_{k,(i+1)} \leftarrow \mathbf{w}_{k,(i)} + \Delta \mathbf{w}_{k,(i)}. \tag{5.9}$$

We employ the margin perceptron with a margin 1.0, and do not need to set a manual learning rate due to the use of ADADELTA[27]. Each run uses 16 epochs, and as a last step the weight vectors of all epochs are averaged, as well as the associated per-coordinate learning rates. Due to its size data is split into 42 shards. We obtain a mean score of 33.0 ±0.1 on the development test data, repeating the tuning procedure three times. For use in our experiments we use the best run which had a development testing score of 33.1. We also performed three MERT runs on the DENSE feature set using the same data, resulting in a mean score of 31.2 (maximum 31.3, standard deviation 0.1).

The initial weights and learning rates for the additional features[28] of the translation model are set as follows: Their weight is initially set to 1.0, and the learning rate is set to the mean rate over all features. The learning rates for the associated sparse features of the extended rule set, are set to the median rate for the respective feature group (rule identifiers, rule bigrams, and rule shapes). The weights and learning rates for the adaptive language model are learned in a downstream step by using DTRAIN on a smaller held-out data set, applying the simulated user feedback scheme.

[27] The hyperparameters of ADADELTA are set to the values suggested in [Zeiler, 2012]: $\rho = (0.95)$ for the decay rate and $\epsilon = 10^{-6}$.

[28] The additional features are: a feature for known rules, a feature for out-of-vocabulary fixing rules, and a feature for new rules.

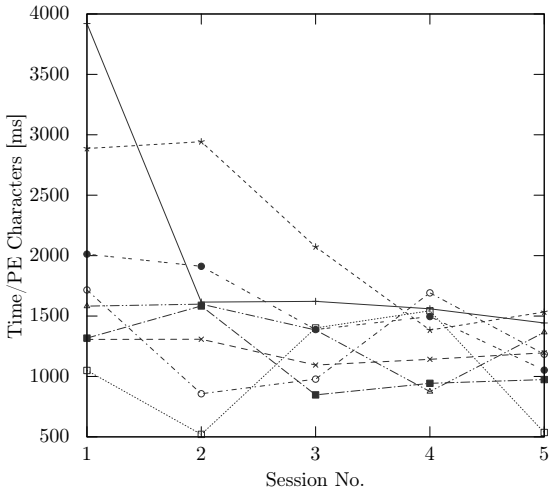

Figure 5.6: Learning curves for eight translators using the graphical interface in five consecutive sessions.

5.7.2 Experimental Design

In our experiment, we seek to compare an adaptive MT system providing initial translation suggestions for post-editing to a non-adaptive MT system for the same task.

We carry out five 90 minute sessions, each on a different day, using the graphical interface for post-editing. The first two sessions are reserved for familiarizing the users to the interface and translation material. A learning curve, illustrating the learning process of a subset of the translators is shown in Figure 5.6, clearly demonstrating a learning effect, which levels out after the second session. Note that, since each translator worked on different texts in each session, the data points in this plot are not directly comparable. In the following three sessions, we carry out a single experiment without adaptation for establishing a baseline, and two sessions with adaptation for comparison. The participants were not informed whether the MT system was adapting to their inputs or not. In each session, all translators receive a single translation task as described above. Within a session, each document is assigned to two users if possible, which means it is translated under the same translation condition. The supervisors made sure that participants working on the same document did not sit next to each other. Every participant starts in every session from the same baseline model.

Thus we were able to collect 978 per-sentence measurements, 312 without

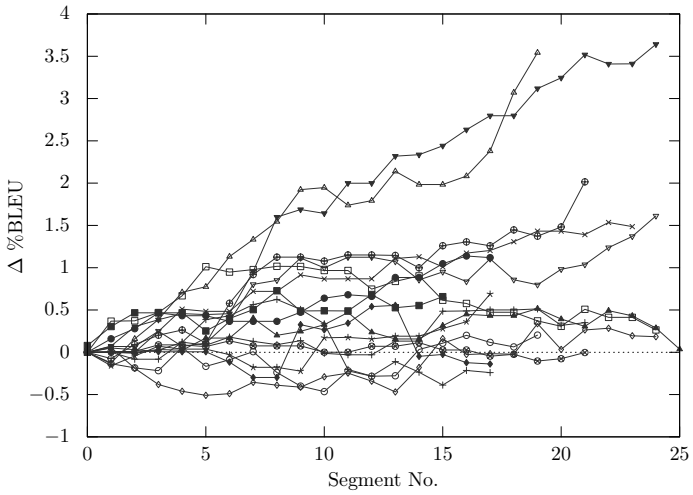

Figure 5.7: Cumulative difference in per-sentence %BLEU between adapted and the baseline system for both adaptive sessions.

adaptation and 666 with adaptation.

5.7.3 Adaptation: Sanity Check

Due to the time constraint of 90 minutes per session, it is not immediately clear that a system is able to learn something meaningful with the very little amount of the data it receives. To show that the adaptive system can learn effectively in this setup, we compare the MT outputs of the adapted systems to the outputs of the non-adapted vanilla baseline system for a single session, comparing human-targeted MT metrics sentence-wise and on the corpus-level.

In Figure 5.7, we show a moving sum of sentence-level differences in the (per-sentence) %BLEU scores of the adapted and the vanilla MT systems, calculating the score against the post-edit as reference, i.e. per-sentence HBLEU[29]. In the plot it is apparent that for most sessions, the curve grows in positive direction, which corresponds to an improvement over the non-adapted system. We additionally calculated the difference in corpus-level MT metrics, showing improvements in most cases. These results are depicted in Table 5.2. Again the repetition rates appear to be a reference point for adaptation performance, with one clear exception (session 4), which has a low rate (below average) but highly improved deltas.

[29]Now referred to as HBLEU+1.

Session #	Δ %BLEU	Δ %TER	%RR
1	+8.8	−3.0	26.9
2	+3.2	−3.7	25.5
3	−0.3	+2.7	14.8
4	+9.8	−7.0	12.7
5	+2.4	−0.4	54.9
6	+8.0	−2.0	28.0
7	+14.7	−7.4	26.7
8	+12.8	−4.8	37.0
9	−1.6	+2.0	20.1
10	+5.9	−3.3	27.9
11	−0.8	−0.7	26.5
12	+21.2	−16.0	26.7
13	+1.8	−3.9	27.9
14	+0.3	+3.3	27.3
15	+0.5	+0.6	31.1
16	+1.7	−0.0	31.0
Avg.	+5.5	−2.7	27.8

Table 5.2: Differences of adapted and vanilla systems in corpus-level MT metrics for all sessions using adaptive MT systems. Averages are depicted in the last row.

Response var.	est. Intercept	est. Δ	Significance
HTER [%]	51.6 ± 2.8	-5.3 ± 1.9	$p < 0.01$, $\chi^2(1) = 7.8741$
HBLEU+1 [%]	43.4 ± 3.0	$+6.8 \pm 2.0$	$p < 0.001$, $\chi^2(1) = 11.748$
norm. time [ms]	1319 ± 84	-118 ± 67	$p < 0.1$ $\chi^2(1) = 3.0688$
KSMR [%]	80.1 ± 11.5	$+7.5 \pm 8.0$	—
TER/MT [%]	70.0 ± 2.1	$+8.3 \pm 2.1$	$p < 0.001$ $\chi^2(1) = 15.344$
BLEU+1/MT [%]	31.1 ± 1.4	-3.8 ± 1.4	$p < 0.01$ $\chi^2(1) = 7.234$
TER/PE [%]	48.0 ± 3.7	$+5.1 \pm 2.3$	$p < 0.05$ $\chi^2(1) = 5.1258$
BLEU+1/PE [%]	45.6 ± 2.9	-2.2 ± 1.9	—

Table 5.3: Results of the estimated linear mixed-effects models with data from the experiments with the graphical user interface. Table adapted from [Simianer et al., 2016].

Given these results, we may conclude that the adaptation approach seems to be capable to learn quickly from user inputs, which is remarkable, since the maximum number of segments used for adaptation was 25 in this set of experiments.

5.7.4 Adaptation: Batch Analysis

Our main statistical analysis of the experiment is carried out by learning a linear mixed-effects model as described in Section 5.6.3. Apart from the translation condition, we use a binary indicator denoting segments that are translated by a native German speaker, and a source-length indicator with three equally sized levels[30] as fixed effects (independent variables). Random effects, with random intercepts, are identifiers for user and for source segment. The user id ranges over a number of segments, and is observed in both translation conditions. The segment identifier is however always observed in one translation condition only, either adaptive or non-adaptive, which is why we cannot implement a maximum random effects structure [Barr et al., 2013], as proposed by Green et al. [2014*c*], i.e. adding random slopes for all random effects.

Dependent variables (or response variables) include human-targeted segment-level MT metrics, HBLEU+1 and HTER, as well as KSMR and time, both normalized by total post-edited characters. We also include segment-level MT metrics with respect to the original reference translation for MT suggestions and final post-edits.

A summary of the results is depicted in Table 5.3, reporting estimated intercepts, slopes and p-values. Differences estimated for the response variables contrasting

[30] The levels are 1-15, 16-30, and 31-45.

non-adaptive to adaptive systems are given in the Δ column, along with intercepts and standard deviations. Significance is tested with likelihood ratio tests of the full model against the model without independent variable of interest, and p-values are reported, if $p \leq 0.1$. The estimates for HTER and HBLEU+1 confirm the findings of the previous section, showing a significant harmonization between translator and MT system for both BLEU and TER. User time per character is however only marginally affected by this improvement, as well as KSMR (which actually increases for the adapted condition), which could be due to the small sample size.

Interestingly, we find large, and significant reductions in translation quality of MT outputs and the final post-edits with respect to the reference translations. While this is an unfortunate result, it can be readily explained by the fact that the difficulty of the texts is very high, since patents cover very technical content, which usually requires specially trained translators which can comprehend the technical context. By adapting the MT to the outputs of our users, the initial suggestions are closer to what the user had in mind. We suspect however that the post-edits are not of high quality due to the discussed shortcomings in our setting. The participants furthermore could not see figures in the interface which were referenced in the text, which could have further complicated the translation.

By analyzing the fixed effects of the model with time as dependent variable, we can confirm a common finding in PE user studies — the time needed for post-editing significantly increases with source segment length (but only marginally: $p < 0.1$ $\chi^2(1) = 5.868$), resulting in an offset of +90 ms ($\pm$75), for lengths between 16 and 30, and +198 $\pm$82 for lengths ≥ 31.

The fixed effect for German mother tongue is non-surprisingly significant ($p < 0.01$ $\chi^2(1) = 7.0329$) with respect to translation time, +266 ms $\pm$92 per character for non-native translators.

5.7.5 Adaptation: Case Study

In this section we present a concrete example of the adaptation process, and how it enhances the post-editing experience.

In a translation task, a user was first confronted with a text from a patent filed as WO-2007059805-A1, starting with the title:

> Coating device comprising flowing coating material for smooth or structured surfaces

The MT system produced the following translation suggestion:

> Beschichtungsvorrichtung mit strömenden Beschichtungsmaterial zur glatten oder strukturierten Oberflächen

The term strömenden is an inept choice for the word flowing in the source, and also in the wrong case (genitive instead of possessive dative). The translator accordingly post-edits the suggestion to end up with:

> Beschichtungsvorrichtung mit fließfähigem Beschichtungsmaterial für glatte oder strukturierte Oberflächen

The participant translates flowing into fließfähigem, almost producing the reference translation:

> Beschichtungsanlage mit fließfähigem Beschichtungsmaterial für glatte oder strukturierte Oberflächen

The translation system could not produce the correct translation, since there was no rule translating flowing to fließfähigem. The adapted system however immediately picks up the rule "X → flowing | fließfähigem", and also associates a positive weight of about +0.5 to it, since the system could detect this particular change. The language model now also includes N-grams including this target term.

The translator is later also assigned to translate patent WO-2007059967-A1, where the title is:

> Coating device comprising a flowing coating material

Which is initially translated as:

> Beschichtungsvorrichtung mit einem fließfähigem Beschichtungsmaterial

This is almost correct, but fließfähigem needs to be corrected to the proper case: fließfähigen[31]. Accordingly the weight of the previously learned rule is adjusted to about +0.1.

This example shows that immediate adaptation can be useful, but also that the adaptation will possibly work best if all models are updated at once.

[31] In an NMT system based on sub-word units, there is a good chance that this will corrected automatically, since the original word could be split into several sub-words. Additionally, since more conditioning history is taken into account, phenomena like these can be handled effectively.

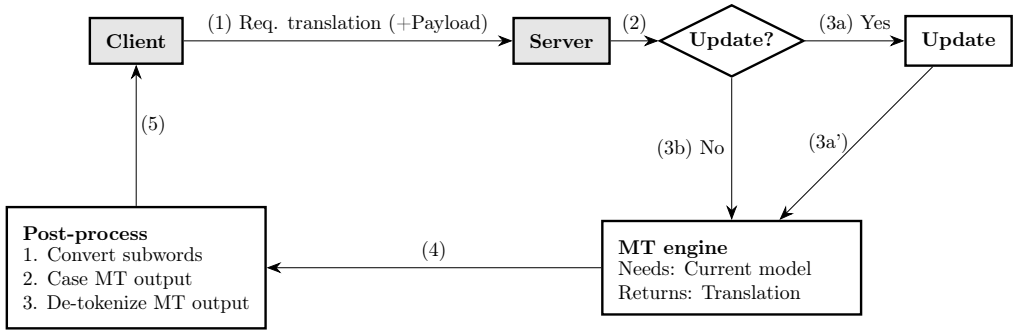

Figure 5.8: Simplified server-client architecture which is used for the NMT user study.

5.8 User Study with Neural Machine Translation

Neural machine translation, as described in 2.2.6, is ideally suited for use in CAT, and especially for online adaptation applications, for a number of reasons: First, since post-editing time is greatly affected by baseline translation quality, the significant improvement over SMT as reported in numerous publications [Luong and Manning, 2015; Bentivogli et al., 2016*b*; Wu et al., 2016; Bojar et al., 2017; Shterionov et al., 2017] (*inter-alia*), presents a valuable advantage over earlier approaches[32]. That the improvement in baseline quality carries over to improvements in CAT has been confirmed by Castilho et al. [2017] and Bentivogli et al. [2016*b*] for post-editing tasks, at least to some extent.

Neural machine translation can also be trivially extended to perform prefix-decoding, as used in prefix completion approaches in IAMT [Wuebker et al., 2016; Knowles and Koehn, 2016; Peris et al., 2017*b*; Scheepers and Schulz, 2016], which before required complex mapping in the SMT approaches, see e.g. [Ortiz-Martínez et al., 2009] or [Spence Green and Manning, 2014].

Lastly, online adaptation by online learning is also trivial to implement in NMT, since the original training method is often already using an online learning algorithm, e.g. mini-batch stochastic gradient descent. A further advantage is that only a single model has to be updated, instead log-linear-, reordering-, language- and translation models in separately. Several studies showed large improvements by this general approach [Turchi et al., 2017; Peris et al., 2017*a*]. Also, by using sub-words [Sennrich et al., 2015; Wu et al., 2016] as minimal translation units

[32] Although some report that with improved quality it becomes harder to spot errors, in turn hindering the post-editing process somewhat.

(including the full source- and target-side vocabularies), reachability is almost[33] certain, which implies that new vocabulary can be learned rapidly with simple online learning, without any modification of the learning algorithm.

In the following we present a user study that tests our approach of online adaptation of an NMT system to post-edited translations, with a similar design as the previously discussed study on immediate adaptation of a hierarchical phase-based machine translation system.

5.8.1 Adaptation by Fine-Tuning

Fine tuning of neural network models, similar to the tuning process in SMT, aims to adapt an existing model to a new domain or data set. Therefore, instead of randomly initializing the parameters of the model, the parameters are set to the result of a previous training process, possibly on a different data set. This approach is similar to transfer learning [Yosinski et al., 2014; Dauphin et al., 2012; Hinton and Salakhutdinov, 2006]. Fine-tuning can be readily applied to an encoder-decoder (cf. Section 2.2.6) based NMT system [Chu et al., 2017; Freitag and Al-Onaizan, 2016].

For the application of online adaptation to a stream of post-edits the scheme shown in Figure 14 can be directly used, with the only exception that instead of reranking, a single global model M_g is continuously adapted. In the adaptation experiments reported by Turchi et al. [2017], this is the most effective approach.

In our approach, online adaptation is performed as a single update step after a post-edit is received:

$$\theta \leftarrow \theta - \eta \nabla_{f,e}, \tag{5.10}$$

where $\nabla_{f,e}$ is the gradient with respect to the model's parameters θ for a mini-batch containing only the source and target sequences f and e. The target sequence is the sub-word representation of the respective post-edit.

While Peris et al. [2017*a*] use the same general approach, they employ a passive-aggressive update instead of plain gradient descent.

5.8.2 Experimental Design

For the user study, we employ the textual interface as shown in Figure 5.3, without modification, and also use the same general environment as in the previous study with the exception that internet access was restricted, only allowing access to a number of pre-defined websites in order to lower supervision overhead.

[33] Unknown characters still pose problems for systems using sub-words.

We conducted nine sessions in total, each about 90 minutes long. Some sessions took place in succession, in these cases we provided 20 minute breaks between sessions. Two of these sessions were used to familiarize the participants with the post-editing task and the user interface. The translation tasks distributed in each session are a random selection of groups of documents as described in Section 5.7.1, translating from English-to-German, following the previous study. We also distribute the data to two participants, but in contrast to the previous study, this time in different translation conditions, i.e. once with an adaptive model and once with a static model. The users did not know whether the system was adapting to their outputs or not. Participants working on the same documents could not sit next to each other.

29 master-level students of translation studies, associated with the Heidelberg University, Germany, were recruited for the study on a voluntary basis. We did not collect any additional personal data. None of the students took part in the previously presented study.

5.8.2.1 Background Model

The adaptive NMT system is based on an implementation [Neubig, 2015] of the RNN-based encoder-decoder approach with attention mechanism [Bahdanau et al., 2014], based on the neural network toolkit described in [Neubig et al., 2017]. Instead of a feed-forward network as described in [Bahdanau et al., 2014], we use a simple dot product to calculate attention scores, as described in [Neubig, 2017, 2016], and employ attention feeding as described by [Luong and Manning, 2015]. As recurrent units, we use long short-term memory units (LSTM) [Hochreiter and Schmidhuber, 1997], as suggested by [Britz et al., 2017], with two layers for encoder and decoder, and a hidden size of 256. Source and target word embedding matrices have $128 \times 10,000$ parameters each, using separate vocabularies with 10K entries for both source and target language. For training we use the optimizer described by Kingma and Ba [2014] (*Adam*), with an initial learning rate of 10^{-3} and otherwise the algorithm's default parameters. For regularization we apply dropout [Srivastava et al., 2014] with probability 0.5.

We opted for a two-stage training, first training on a concatenation of of the previously used NC and EP corpora, which add up to about 2M segments. Training was performed until no further improvement could be observed on held-out data, selecting an earlier epoch for early stopping.

In a fine tuning step, we trained on the same 350K parallel segments as used in the previous study. From this truecased and tokenized data we also build vocabularies of sub-word units using the approach of [Sennrich et al., 2015]. Separate vocabularies are built for source and target languages with a limit of 10 K entries. Both training corpora use the same pre-processing.

The parameters for online adaptation are adjusted using the development data

Response var.	est. Intercept	est. Δ	Significance
HTER [%]	31.1 ± 1.1	-4.8 ± 0.9	$p < 0.001$ $\chi^2(1) = 18.902$
HBLEU+1 [%]	57.2 ± 1.2	$+5.9 \pm 1.1$	$p < 0.001$ $\chi^2(1) = 19.08$
norm. time [ms]	598 ± 26	-44 ± 29	—
KSMR [%]	44.0 ± 2.7	-7.8 ± 1.5	$p < 0.001$ $\chi^2(1) = 17.513$
TER/MT [%]	52.9 ± 0.6	-0.8 ± 0.4	$p < 0.1$ $\chi^2(1) = 3.2446$
BLEU+1/MT [%]	33.7 ± 0.5	$+1.5 \pm 0.3$	$p < 0.01$ $\chi^2(1) = 9.3528$
TER/PE [%]	49.6 ± 0.8	-0.5 ± 0.7	—
BLEU+1/PE [%]	36.9 ± 0.8	$+0.9 \pm 0.7$	—

Table 5.4: Results of the estimated linear mixed-effects model with data from the experiments with the neural MT system.

of the previous study. We halve dropout to 0.25, use SGD instead of Adam, and use a learning rate of 0.05. For inference we use a beam size of 10, a word penalty of 0.85 and an unknown penalty of 0.25 [Neubig, 2016].

Using the NMT system vastly reduces the complexity of the overall architecture as depicted in Figure 5.8.

5.8.3 Analysis

For this analysis we build linear mixed-effects models with maximum random effects structure following Barr et al. [2013]. The data consists of 3,212 per-sentence measurements for a total of 1,606 source segments. Each source segment is thus translated twice — once by a translator using an adaptive MT system, and once by a different translator using a non-adaptive system. The models incorporate a single fixed effect — the translation condition, which has the levels adaptive and non-adaptive. Random effects are the segment identifier, for which all levels are observed under both translation conditions, and user identifier, for which also all levels are observed under both translation conditions. For the maximum random effects structure the models include random slopes for translation condition, and random intercepts for both user identifier and segment identifier.

Results with models with varying response variables are depicted in Table 5.4. Differences estimated for the response variables contrasting non-adaptive to adaptive systems are given in the Δ column, along with intercepts and standard deviations. Significance is tested with likelihood ratio tests of the full model against the model without independent variable of interest, and p-values are reported, if

$p \leq 0.1$. As in the previous study, human-targeted MT metrics are greatly and significantly improved using adaptation. Time (normalized by characters of the final post-edit), is also again improved, but not significantly[34]. The measurement for KSMR is significantly better for the adapted system, which however did not translate to significant speed improvements. Machine translation evaluation metrics with respect to the independent reference translations give a mixed result — while the MT outputs seem to converge to the reference translation, the actual post-edits, which are used to update the system do not show the same behavior. We suspect that this is an artifact of a domain adaptation effect, since the post-edited documents are similar to each other within every session. The translators however, being non-professional and not trained at all in patent translation or subject matter, could not produce translations that are closer to the reference translations, but stay approximately at the same level. A similar effect to what we have already observed in the previous study.

Compared to the results of using the graphical interface, we find that this time there is no reduction in the quality of the post-edits with respect to the references using the NMT system and the textual interface. The MT outputs however are significantly improved in this respect. This confirms the findings of [Bentivogli et al., 2016*a*].

While NMT systems are certainly able to pick up new vocabulary at ease, see e.g. the examples in [Karimova et al., 2017] from the same data, other changes are harder to trace since the model is essentially a black box. Changes can be very subtle, and the responsible triggers may be unknown.

To exemplify this, consider the translation of patent WO-2007134473-A1, titled *Image recording means with local adaptive exposure control* (section H). In the non-adapted system there was the problem that translations were too short and repetitions occurred:

> Dies wird insbesondere bei Anwendungen, die eine hohe Beleuchtungsdynamik erfordern oder eine hohe Bewegungsdynamik erfordern, was eine hohe Bewegungsdynamik erfordert.

The reference being:

[34] We do however observe significant improvements for adaptation using raw time (85,336 ms $\pm 3,398$ $-7,280$ ms $\pm 3,154$, $p < 0.05$ $\chi^2(1) = 4.784$) following Läubli et al. [2013], as well as log-transformed raw time (-0.17416 ± 0.04674, $p < 0.001$ $\chi^2(1) = 10.85$) and log-transformed normalized time (-0.17969 ± 0.04944, $p < 0.01$ $\chi^2(1) = 10.588$), following Green et al. [2013*a*], as response variables. The response variables and the residuals of the respective models were however not normally distributed according to quantitative tests (repeated Shapiro-Wilk W tests [Shaphiro and Wilk, 1965; Royston, 1982]) and qualitative inspection with normal Q-Q plots, for any configuration. We choose to report normalized, non-transformed results for interpretability [Lo and Andrews, 2015], and comparability to the other normalized response variables used in this and the previous study.

> Dies ist insbesondere bei Applikationen hilfreich welche eine hohe Lichtdynamik erfordern oder welche eine hohe Bewegungsdynamik aufweisen wie zum Beispiel der Einsatz im Bereich Fahrerassistenzfunktionen für Automobile.

The adapted system produced however a much more reasonable translation, also getting a central part of the translation right (underlined):

> Dies wird insbesondere bei Anwendungen, die eine hohe Beleuchtungsdynamik erfordern oder hohe Bewegungsdynamik benötigen, wie z.B. im Feldfeld von Fahrerassistenzfunktionen für Kraftfahrzeuge verwendet.

The English source for this segment:

> This is helpful in particular in applications which require a high level of lighting dynamics or which have a high level of movement dynamics, such as for example use in the field of driver assistance functions for motor vehicles.

The peculiarity of this example is that neither the previous segments in the same patent, nor the other patent previously translated with the same model (US-20100003762-A1[35]), or the user generated post-edits would suggest this change. What is more, the term Fahrerassistenzfunktionen does not even occur in the in-domain training data, but is still reachable due to the use of sub-words as atomic translation units.

Another issue with NMT arises from the application of sub-word units: While using sub-words enables learning of new, previously unknown vocabulary, it has the side-effect that "hallucinated" words may appear in the target, cf. [Koehn and Knowles, 2017]. Translating into German, this can be problematic since long compound words need to be constructed on the target side, for example:

> Elektronenstärkigkeitserhöhung,

which is not a valid German word but could easily be mistaken for one.

[35] *Permanent chemical marker and identification of information in polymers*, classified under section C.

6 Summary & Outlook

> " [...] a 95% FAHQT system in the worst case produces a translated text that is analogous to a jar of cookies, only 5% of which are poisoned."
> [Carbonell and Tomita, 1985]

We have presented three distinct approaches for preference learning for MT:

First we turned to applying pairwise ranking for tuning SMT, developing a method for efficient, true online preference learning. To this end, we also presented an application of a matrix-norm based regularization method, which can be used for feature selection as well as multi-task learning. In extensive experimental evaluations we found among other things, that our method can be used to find the important features in vast amounts of training data, which also generalize to out-of-domain (with respect to training) data sets. We further determined that under some circumstances, we are able exploit commonalities found in related but diverse data sets, leading to improved results compared to random splits or simple accumulation of data sets. Another finding which is worth pointing out, is that significantly different variants of the proposed tuning algorithm as well as other, unrelated algorithms tend to perform similarly when a rigorous search for hyperparameters is performed. We also found indications that tuning the linear model of SMT systems possibly attained an upper limit, at least in our setting, since despite large training efforts, no or only little gains in translation quality could be achieved for a number of setups. An important aspect of our approach, which also delivers state-of-the art performance in SMT, is the use of sparse features, most notably rule identifiers which add a powerful fully discriminative translation model to SMT systems. Overall, with this approach we have found a method to train SMT models through preference learning that can generalize well.

Secondly, we turned to CAT, for which we developed a reranking approach, based on the same general ideas as our SMT tuning approach. However, the focus was different — we sought a method to learn more fine-grained preferences since the application demands a more specialized approach. The goal in this work was to directly learn preferences from post-edits to be able to effectively and promptly correct errors in the translation outputs of the MT system. We therefore developed a method to obtain representations of post-edits which can be directly used for learning, instead of inducing a ranking by utilizing an external scoring function as in the previous approach. We have shown the algorithm's effectiveness in a series of experiments, where we also showed that the method can improve over already otherwise adapted systems. We found however, that the models learned with this

method were possibly not portable without further processing. Again, as with the previous approach, we employed sparse features that include identifiers for the smallest translation units of the underlying SMT system.

In our next work, also an application for CAT, we first combined our developed tuning method with a graphical interface, that allowed to directly, without ranking and without other automatic induction methods, exploit literal user corrections to a SMT system. With this approach we tried the boldest method of learning from post-edits. In a user study we set-up and conducted, we found that the approach works very well, as it could quickly adapt (even within the range of short documents) to feedback and thus provided an improved user experience. Again, also with this approach, we used sparse features (the same as in our tuning approach) to have an expressive model which can learn very fine-grained preferences. In this case, due to the combination with the graphical interface, we could learn preferences almost in real-time and which were directly derived from user input. While the general approach of translating with a graphical user interface may not be a practical, we could still explore what *should* be possible in adaptation for MT. Lastly, we utilized a state-of-the-art NMT approach, and implemented a very efficient method of online adaptation. We found that overall system was vastly simpler than the one we had to build for SMT, while we were still able to directly learn from user inputs due to the combination of online learning and sub-word representations, which also allows to learn new vocabulary without another feedback loop. With the lessons learned from our first user study, we conducted a second study of larger scale, in which we found very encouraging results for NMT in CAT.

With the background of the results of our last user study, we believe that online adaptive NMT systems are the way forward in CAT. While tuning for SMT seems to have reached a limit, the improvements that can be had through NMT are tremendous. In this direction we think that multi-task and multi-domain learning systems could lead to further improvements for practical implementations of CAT systems. In this context it is also imperative to gain further insight on the adaptation process itself, and to explore the external factors that facilitate or prevent effective adaptation. Furthermore, we think that different types of loss functions, as exemplified in our SMT tuning approach, could have an large impact by improving baseline translation quality or adaptation performance.

Bibliography

Agarwal, A. and Lavie, A. *Meteor, M-BLEU and M-TER: Evaluation Metrics for High-Correlation with Human Rankings of Machine Translation Output.* In Proceedings WMT. 2008.

Airola, A., Pahikkala, T., and Salakoski, T. *Training Linear Ranking SVMs in Linearithmic Time Using Red-Black Trees.* In Pattern Recognition Letters, 2011.

Akabe, K., Neubig, G., Sakti, S., Toda, T., and Nakamura, S. *Discriminative Language Models as a Tool for Machine Translation Error Analysis.* In Proceedings of COLING. 2014.

Alabau, V., Bonk, R., Buck, C., Carl, M., Casacuberta, F., García-Martínez, M., González, J., Leiva, L., Mesalao, B., Ortiz, D. et al. *Advanced Computer Aided Translation with a Web-Based Workbench.* In Proceedings of the Workshop on Post-editing Technology and Practice. 2013*a*.

Alabau, V., Bonk, R., Buck, C., Carl, M., Casacuberta, F., Martínez, M. G., González, J., Koehn, P., Leiva, L., Mesa-Lao, B., Ortiz, D., Saint-Amand, H., Sanchis, G., and Tsoukala, C. *CASMACAT: An Open Source Workbench for Advanced Computer-Aided Translation.* In The Prague Bulletin of Mathematical Linguistics, 2013*b*.

Alabau, V., González-Rubio, J., Ortiz-Martínez, D., Casacuberta, F., Martinez, M. G., Mesa-Lao, B., Petersen, D. C., Dragsted, B., and Carl, M. *Integrating Online and Active Learning in a Computer-Assisted Translation Workbench.* In Proceedings of AMTA. 2014.

Alabau, V., Leiva, L. A., Ortiz-Martınez, D., and Casacuberta, F. *User Evaluation of Interactive Machine Translation Systems.* In Proceedings of EAMT. 2012.

Albrecht, J. *Visualizing and Correcting Machine Translation.* Tech. rep., Department of Computer Science, University of Pittsburgh, 2008.

Albrecht, J. S., Hwa, R., and Marai, G. E. *Correcting Automatic Translations Through Collaborations Between MT and Monolingual Target-Language Users.* In Proceedings of EACL. 2009.

Ando, R. K. and Zhang, T. *A Framework for Learning Predictive Structures From Multiple Tasks and Unlabeled Data.* In Machine Learning Research, 2005.

Arenas, A. G. *Productivity and Quality in the Post-Editing of Outputs From Translation Memories and Machine Translation.* In Localisation Focus, 2008.

Argyriou, A., Evgeniou, T., and Pontil, M. *Multi-Task Feature Learning.* In Proceedings of NIPS. 2007.

Argyriou, A., Evgeniou, T., and Pontil, M. *Convex Multi-Task Feature Learning.* In Machine Learning, 2008.

Arun, A., Dyer, C., Haddow, B., Blunsom, P., Lopez, A., and Koehn, P. *Monte Carlo Inference and Maximization for Phrase-Based Translation.* In Proceedings of CoNLL. 2009.

Arun, A. and Koehn, P. *Online Learning Methods For Discriminative Training of Phrase Based Statistical Machine Translation.* In Proceedings of MT Summit XI. 2007.

Auli, M., Galley, M., and Gao, J. *Large-Scale Expected BLEU Training of Phrase-Based Reordering Models.* In Proceedings of EMNLP. 2014.

Aziz, W., Castilho, S., and Specia, L. *PET: A Tool for Post-Editing and Assessing Machine Translation.* In Proceedings of LREC. 2012.

Baayen, R. H. *Analyzing Linguistic Data: A Practical Introduction to Statistics Using R.* Cambridge University Press, New York City, NY, USA, 2008.

Baayen, R. H., Davidson, D. J., and Bates, D. M. *Mixed-Effects Modeling with Crossed Random Effects for Subjects and Items.* In Journal of Memory and Language, 2008.

Bahdanau, D., Cho, K., and Bengio, Y. *Neural Machine Translation by Jointly Learning to Align and Translate.* In CoRR, 2014.

Bahl, L. R., Jelinek, F., and Mercer, R. L. *A Maximum Likelihood Approach to Continuous Speech Recognition.* In IEEE Transactions on Pattern Analysis and Machine Intelligence, 1983.

Baker, K. L., Franz, A. M., Jordan, P. W., Mitamura, T., and Nyberg, E. H. *Coping with Ambiguity in a Large-Scale Machine Translation System.* In Proceedings of COLING. 1994.

Balling, L. W. and Carl, M. *Post-Editing of Machine Translation: Processes and Applications.* Cambridge Scholars Publishing, 2014.

Baltescu, P. and Blunsom, P. *A Fast and Simple Online Synchronous Context Free Grammar Extractor.* In The Prague Bulletin of Mathematical Linguistics, 2014.

Banerjee, S. and Lavie, A. *METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments.* In Proceedings of ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization. 2005.

Bar-Hillel, Y. *The Present State of Research on Mechanical Translation.* In American Documentation, 1951.

Bar-Hillel, Y. *The Present Status of Automatic Translation of Languages.* In Advances in Computers, 1960.

Barr, D. J., Levy, R., Scheepers, C., and Tily, H. J. *Random Effects Structure for Confirmatory Hypothesis Testing: Keep It Maximal.* In Journal of Memory and Language, 2013.

Barrachina, S., Bender, O., Casacuberta, F., Civera, J., Cubel, E., Khadivi, S., Lagarda, A., Ney, H., Tomás, J., Vidal, E. et al. *Statistical Approaches to Computer-Assisted Translation.* In Computational Linguistics, 2009.

Bates, D., Mächler, M., Bolker, B., and Walker, S. *Fitting Linear Mixed-Effects Models Using lme4.* In arXiv preprint arXiv:1406.5823, 2014.

Bazrafshan, M., Chung, T., and Gildea, D. *Tuning as Linear Regression.* In Proceedings of NAACL. 2012.

Beaton, A. and Contreras, G. *Sharing the Continental Airlines and SDL Post-Editing Experience.* 2010.

Bentivogli, L., Bertoldi, N., Cettolo, M., Federico, M., Negri, M., and Turchi, M. *On the Evaluation of Adaptive Machine Translation for Human Post-Editing.* In IEEE/ACM TASLP, 2016*a*.

Bentivogli, L., Bisazza, A., Cettolo, M., and Federico, M. *Neural versus Phrase-Based Machine Translation Quality: A Case Study.* In arXiv preprint arXiv:1608.04631, 2016*b*.

Berg-Kirkpatrick, T., Burkett, D., and Klein, D. *An Empirical Investigation of Statistical Significance in NLP.* In Proceedings of EMNLP-CoNLL. 2012.

Berka, J., Bojar, O., Fishel, M., Popovic, M., and Zeman, D. *Automatic MT Error Analysis: Hjerson Helping Addicter.* In Proceedings of LREC. 2012.

Bernardini, S. *Think-Aloud Protocols in Translation Research: Achievements, Limits, Future Prospects.* In Target. International Journal of Translation Studies, 2001.

Bertoldi, N., Cattoni, R., Cettolo, M., Farajian, M., Federico, M., Caroselli, D., Mastrostefano, L., Rossi, A., Trombetti, M., Germann, U. et al. *MMT: New Open Source MT for the Translation Industry.* In Proceedings of EAMT. 2017.

Bertoldi, N., Cettolo, M., and Federico, M. *Cache-Based Online Adaptation for Machine Translation Enhanced Computer Assisted Translation.* In Proceedings of MT Summit XIV, 2013.

Bertoldi, N., Simianer, P., Cettolo, M., Wäschle, K., Federico, M., and Riezler, S. *Online Adaptation to Post-Edits for Phrase-Based Statistical Machine Translation.* In Machine Translation, 2014.

Biçici, E. and Yuret, D. *Instance Selection for Machine Translation Using Feature Decay Algorithms.* In Proceedings of WMT. 2011.

Bisazza, A. and Federico, M. *A Survey of Word Reordering in Statistical Machine Translation: Computational Models and Language Phenomena.* In Computational linguistics, 2016.

Blain, F., Schwenk, H., and Senellart, J. *Incremental Adaptation Using Translation Information and Post-Editing Analysis.* In Proceedings of IWSLT. 2012.

Blunsom, P., Cohn, T., and Osborne, M. *A Discriminative Latent Variable Model for Statistical Machine Translation.* In Proceedings of ACL-HLT. 2008.

Blunsom, P. and Osborne, M. *Probabilistic Inference for Machine Translation.* In Proceedings of EMNLP. 2008.

Bojar, O., Buck, C., Callison-Burch, C., Haddow, B., Koehn, P., Monz, C., Post, M., Saint-Amand, H., Soricut, R., and Specia, L., eds. *Proceedings of WMT.* 2013.

Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Leveling, J., Monz, C., Pecina, P., Post, M., Saint-Amand, H. et al. *Findings of the 2014 Workshop on Statistical Machine Translation.* In Proceedings of WMT. 2014*a*.

Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., Monz, C., Post, M., and Specia, L., eds. *Proceedings of WMT.* 2014*b*.

Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huang, S., Huck, M., Koehn, P., Liu, Q., Logacheva, V. et al. *Findings of the 2017 Conference on Machine Translation.* In Proceedings of WMT. 2017.

Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huck, M., Jimeno Yepes, A., Koehn, P., Logacheva, V., Monz, C., Negri, M., Neveol, A., Neves, M., Popel, M., Post, M., Rubino, R., Scarton, C., Specia, L., Turchi, M.,

Verspoor, K., and Zampieri, M. *Findings of the 2016 Conference on Machine Translation.* In Proceedings of WMT. 2016.

Bojar, O., Chatterjee, R., Federmann, C., Haddow, B., Hokamp, C., Huck, M., Logacheva, V., and Pecina, P., eds. *Proceedings of WMT.* 2015.

Bottou, L. *Stochastic Learning.* In Advanced Lectures on Machine Learning. Springer, Berlin, Germany, 2004.

Bousquet, O. and Bottou, L. *The Tradeoffs of Large Scale Learning.* In Proceedings of NIPS. 2008.

Braune, F., Fraser, A. M., III, H. D., and Tamchyna, A. *A Framework for Discriminative Rule Selection in Hierarchical Moses.* In Proceedings of WMT. 2016.

Britz, D., Goldie, A., Luong, M., and Le, Q. V. *Massive Exploration of Neural Machine Translation Architectures.* In CoRR, 2017.

Brown, P., Cocke, J., Pietra, S. D., Pietra, V. D., Jelinek, F., Mercer, R., and Roossin, P. *A Statistical Approach to Language Translation.* In Proceedings of COLING. 1988.

Brown, P. F., Chen, S. F., Della Pietra, S. A., Della Pietra, V. J., Kehler, A. S., and Mercer, R. L. *Automatic speech recognition in machine-aided translation.* In Computer Speech & Language, 1994.

Brown, P. F., Cocke, J., Pietra, S. A. D., Pietra, V. J. D., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. *A Statistical Approach to Machine Translation.* In Computational Linguistics, 1990.

Brown, P. F., Pietra, V. J. D., Pietra, S. A. D., and Mercer, R. L. *The Mathematics of Statistical Machine Translation: Parameter Estimation.* In Computational Linguistics, 1993.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. *Learning to Rank Using Gradient Descent.* In Proceedings of ICML. 2005.

Cai, Y., Sun, Y., Cheng, Y., Li, J., and Goodison, S. *Fast Implementation of L1 Regularized Learning Algorithms Using Gradient Descent Methods.* In Proceedings of SIAM International Conference on Data Mining. 2010.

Callison-Burch, C. *Fast, Cheap, and Creative: Evaluating Translation Quality Using Amazon's Mechanical Turk.* In Proceedings of EMNLP. 2009.

Callison-Burch, C., Bannard, C., and Schroeder, J. *Improved Statistical Translation Through Editing.* In Proceedings of EAMT. 2004.

Callison-Burch, C., Fordyce, C., Koehn, P., Monz, C., and Schroeder, J. *Further Meta-Evaluation of Machine Translation.* In Proceedings of WMT. 2008.

Callison-Burch, C., Koehn, P., Fordyce, C. S., and Monz, C., eds. *Proceedings of WMT.* 2007.

Callison-Burch, C., Koehn, P., Monz, C., and Zaidan, O. F. *Findings of the 2011 Workshop on Statistical Machine Translation.* In Proceedings of WMT. 2011*a*.

Callison-Burch, C., Koehn, P., Monz, C., and Zaidan, O. F., eds. *Proceedings of WMT.* 2011*b*.

Callison-Burch, C., Osborne, M., and Koehn, P. *Re-Evaluating the Role of Bleu in Machine Translation Research.* In Proceedings of EACL. 2006.

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., and Li, H. *Learning to Rank: From Pairwise Approach to Listwise Approach.* In Proceedings of ICML. 2007.

Carbonell, J. G. and Tomita, M. *New Approaches to Machine Translation.* In Proceedings of the Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages. 1985.

Carl, M. *Translog-II: A Program for Recording User Activity Data for Empirical Translation Process Research.* In Proceedings of LREC. 2012.

Carpenter, B. *Lazy Sparse Stochastic Gradient Descent for Regularized Multinomial Logistic Regression.* Tech. rep., Alias-i, Inc., 2008.

Carpuat, M., Daumé, H., Fraser, A., Quirk, C., Braune, F., Clifton, A. et al. *Domain Adaptation in Machine Translation.* In Johns Hopkins Summer Workshop Final Report. 2012.

Carpuat, M. and Simard, M. *The Trouble with SMT Consistency.* In Proceedings of WMT. 2012.

Carter, S. and Monz, C. *Discriminative Syntactic Reranking for Statistical Machine Translation.* In Proceedings of AMTA. 2010.

Caruana, R. *Multitask Learning: A Knowledge-Based Source of Inductive Bias.* In Proceedings of ICML. 1993.

Caruana, R. *Multitask Learning.* In Machine Learning, 1997.

Casacuberta, F., Civera, J., Cubel, E., Lagarda, A. L., Lapalme, G., Macklovitch, E., and Vidal, E. *Human Interaction for High-Quality Machine Translation.* In Communications of the ACM, 2009.

Casanellas, L. and Marg, L. *Assumptions, Expectations and Outliers in Post-Editing.* In Proceedings of the Workshop on Post-Editing Technology and Practice. 2014.

Castilho, S., Moorkens, J., Gaspari, F., Calixto, I., Tinsley, J., and Way, A. *Is Neural Machine Translation the New State of the Art?* In The Prague Bulletin of Mathematical Linguistics, 2017.

Cavallanti, G., Cesa-Bianchi, N., and Gentile, C. *Linear Algorithms for Online Multitask Classification.* In Journal of Machine Learning Research, 2010.

Cer, D., Jurafsky, D., and Manning, C. D. *Regularization and Search for Minimum Error Rate Training.* In Proceedings of WMT. 2008.

Cer, D., Manning, C. D., and Jurafsky, D. *The Best Lexical Metric for Phrase-Based Statistical MT System Optimization.* In Proceedings of NAACL-HLT. 2010.

Cesa-Bianchi, N. and Lugosi, G. *Prediction, Learning, and Games.* Cambridge University Press, New York City, NY, USA, 2006.

Cesa-Bianchi, N., Reverberi, G., and Szedmak, S. *Online Learning Algorithms for Computer-Assisted Translation.* Tech. rep., SMART Project, 2008.

Cettolo, M., Bertoldi, N., and Federico, M. *Methods for Smoothing the Optimizer Instability in SMT.* 2011.

Cettolo, M., Bertoldi, N., and Federico, M. *The Repetition Rate of Text as a Predictor of the Effectiveness of Machine Translation Adaptation.* In Proceedings of AMTA. 2014*a*.

Cettolo, M., Federico, M., and Bertoldi, N. *Mining Parallel Fragments From Comparable Texts.* In Proceedings of IWSLT. 2010.

Cettolo, M., Girardi, C., and Federico, M. *WIT*^{3}*: Web Inventory of Transcribed and Translated Talks.* In Proceedings of EAMT. 2012.

Cettolo, M., Niehues, J., Stüker, S., Bentivogli, L., Cattoni, R., and Federico, M. *The IWSLT 2015 Evaluation Campaign.* In Proceedings of IWSLT, 2015.

Cettolo, M., Niehues, J., Stüker, S., Bentivogli, L., and Federico, M. *Report on the 11th IWSLT Evaluation Campaign, IWSLT 2014.* In Proceedings of IWSLT. 2014*b*.

Cettolo, M., Servan, C., Bertoldi, N., Federico, M., Barrault, L., and Schwenk, H. *Issues in Incremental Adaptation of Statistical MT From Human Post-Edits.* In Proceedings of the Workshop on Post-editing Technology and Practice. 2013.

Chapelle, O., Do, C. B., Teo, C. H., Le, Q. V., and Smola, A. J. *Tighter Bounds for Structured Estimation.* In Proceedings of NIPS. 2009.

Chapelle, O., Shivaswamy, P., Vadrevu, S., Weinberger, K., Zhang, Y., and Tseng, B. *Boosted Multi-Task Learning.* In Machine Learning, 2011.

Chappelier, J.-C. and Rajman, M. *A Generalized CYK Algorithm for Parsing Stochastic CFG.* In Proceedings of TAPD. 1998.

Chen, H., Huang, S., Chiang, D., Dai, X.-Y., , and Chen, J. *Top-Rank Enhanced Listwise Optimization for Statistical Machine Translation.* In Proceedings of CoNLL. 2017.

Chen, S. F. and Goodman, J. *An Empirical Study of Smoothing Techniques for Language Modeling.* In Proceedings of ACL. 1996.

Cheng, S., Huang, S., Chen, H., Dai, X., and Chen, J. *PRIMT: A Pick-Revise Framework for Interactive Machine Translation.* In Proceedings of NAACL-HLT. 2016.

Cherry, C. and Foster, G. *Batch Tuning Strategies for Statistical Machine Translation.* In Proceedings of NAACL-HLT. 2012.

Chiang, D. *A Hierarchical Phrase-Based Model for Statistical Machine Translation.* In Proceedings of ACL. 2005.

Chiang, D. *Hierarchical Phrase-Based Translation.* In Computational Linguistics, 2007.

Chiang, D. *Hope and Fear for Discriminative Training of Statistical Translation Models.* In Journal of Machine Learning Research, 2012.

Chiang, D., Knight, K., and Wang, W. *11,001 New Features for Statistical Machine Translation.* In Proceedings of NAACL-HLT. 2009.

Chiang, D., Marton, Y., and Resnik, P. *Online Large-Margin Training of Syntactic and Structural Translation Features.* In Proceedings of EMNLP. 2008.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. *Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation.* In arXiv preprint arXiv:1406.1078, 2014.

Chrisman, L. *Learning Recursive Distributed Representations for Holistic Computation.* In Connection Science, 1991.

Chu, C., Dabre, R., and Kurohashi, S. *An Empirical Comparison of Simple Domain Adaptation Methods for Neural Machine Translation.* In arXiv preprint arXiv:1701.03214, 2017.

Chung, T. and Galley, M. *Direct Error Rate Minimization for Statistical Machine Translation.* In Proceedings of WMT. 2012.

Church, K. W. and Gale, W. A. *Probability Scoring for Spelling Correction.* In Statistics and Computing, 1991.

Clark, H. H. *The Language-as-Fixed-Effect Fallacy: A Critique of Language Statistics in Psychological Research.* In Journal of Verbal Learning and Verbal Behavior, 1973.

Clark, J. H., Dyer, C., Lavie, A., and Smith, N. A. *Better Hypothesis Testing for Statistical Machine Translation: Controlling for Optimizer Instability.* In Proceedings of ACL-HLT. 2011.

Cohen, W. W., Schapire, R. E., and Singer, Y. *Learning to Order Things.* In Proceedings of NIPS. 1998.

Coleman, E. B. *Generalizing to a Language Population.* In Psychological Reports, 1964.

Collins, M. *Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms.* In Proceedings of EMNLP. 2002.

Collins, M. and Duffy, N. *New Ranking Algorithms for Parsing and Tagging: Kernels Over Discrete Structures, and the Voted Perceptron.* In Proceedings of ACL. 2002.

Collins, M. and Koo, T. *Discriminative Reranking for Natural Language Parsing.* In Computational Linguistics, 2005.

Collobert, R. and Bengio, S. *Links Between Perceptrons, MLPs and SVMs.* In Proceedings of ICML. 2004.

Collobert, R., Sinz, F., Weston, J., and Bottou, L. *Trading Convexity for Scalability.* In Proceedings of ICML. 2006.

Collobert, R. and Weston, J. *A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning.* In Proceedings of ICML. 2008.

Cortes, C. and Vapnik, V. *Support-Vector Networks.* In Machine Learning, 1995.

Coughlin, D. *Correlating Automated and Human Assessments of Machine Translation Quality.* In Proceedings of MT Summit IX. 2003.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y. *Online Passive-Aggressive Algorithms.* In Journal of Machine Learning Research, 2006.

Crammer, K. and Singer, Y. *Ultraconservative Online Algorithms for Multiclass Problems.* In Journal of Machine Learning Research, 2003.

Cubel, E., González, J., Lagarda, A., Casacuberta, F., Juan, A., and Vidal, E. *Adapting Finite-State Translation to the TransType2 Project.* In Proceedings of EAMT/CLAW. 2003.

Cui, L., Chen, X., Zhang, D., Liu, S., Li, M., and Zhou, M. *Multi-Domain Adaptation for SMT Using Multi-Task Learning.* In Proceedings of EMNLP. 2013.

Cuong, H. and Sima'an, K. *A Survey of Domain Adaptation for Statistical Machine Translation.* In Machine Translation, 2018.

Darragh, J. J., Witten, I. H., and James, M. L. *The Reactive Keyboard: A Predictive Typing Aid.* In Computer, 1990.

Daumé, H., III. *Practical Structured Learning Techniques for Natural Language Processing.* Ph.D. thesis, University of Southern California, 2006.

Daumé, H., III. *Frustratingly Easy Domain Adaptation.* In CoRR, 2009.

Daumé, H., III, Langford, J., and Marcu, D. *Search-Based Structured Prediction.* In Machine Learning, 2009.

Daumé III, H. *Notes on CG and LM-BFGS Optimization of Logistic Regression*, 2004.

Dauphin, G. M. Y., Glorot, X., Rifai, S., Bengio, Y., Goodfellow, I., Lavoie, E., Muller, X., Desjardins, G., Warde-Farley, D., Vincent, P. et al. *Unsupervised and Transfer Learning Challenge: A Deep Learning Approach.* In Proceedings of ICML Workshop on Unsupervised and Transfer Learning. 2012.

De Sousa, S. C., Aziz, W., and Specia, L. *Assessing the Post-Editing Effort for Automatic and Semi-Automatic Translations of DVD Subtitles.* In Proceedings of RANLP. 2011.

Dean, J., Corrado, G. S., Monga, R., Chen, K., Devin, M., Le, Q. V., Mao, M. Z., Ranzato, M., Senior, A., Tucker, P., Yang, K., and Ng, A. Y. *Large Scale Distributed Deep Networks.* In Proceedings of NIPS. 2012.

Dean, J. and Ghemawat, S. *MapReduce: Simplified Data Processing on Large Clusters.* In Communications of ACM, 2008.

Dempster, A. P., Laird, N. M., and Rubin, D. B. *Maximum Likelihood From Incomplete Data via the EM Algorithm.* In Journal of the Royal Statistical Society. Series B (Methodological), 1977.

DeNeefe, S., Knight, K., and Chan, H. H. *Interactively Exploring a Machine Translation Model.* In Proceedings of ACL (Interactive Poster and Demonstration Sessions). 2005.

DeNero, J., Gillick, D., Zhang, J., and Klein, D. *Why Generative Phrase Models Underperform Surface Heuristics.* In Proceedings of WMT. 2006.

Denkowski, M. *Machine Translation for Human Translators.* Ph.D. thesis, Carnegie Mellon University, 2015.

Denkowski, M. and Lavie, A. *Choosing the Right Evaluation for Machine Translation: An Examination of Annotator and Automatic Metric Performance on Human Judgment Tasks.* In Proceedings of AMTA. 2010.

Denkowski, M. and Lavie, A. *TransCenter: Web-Based Translation Research Suite.* In Proceedings of the Workshop on Post-editing Technology and Practice. 2012.

Denkowski, M., Lavie, A., and Dyer, C. *Learning From Post-Editing: Online Model Adaptation for Statistical Machine Translation.* In Proceedings of EACL, 2014*a*.

Denkowski, M., Lavie, A., Lacruz, I., and Dyer, C. *Real Time Adaptive Machine Translation for Post-Editing with cdec and TransCenter.* In Proceedings of EACL Workshop on Humans and Computer-Assisted Translation. 2014*b*.

Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R., and Makhoul, J. *Fast and Robust Neural Network Joint Models for Statistical Machine Translation.* In Proceedings of ACL. 2014.

Doddington, G. *Automatic Evaluation of Machine Translation Quality Using N-gram Co-Occurrence Statistics.* In Proceedings of HLT. 2002.

Domingo, M., Peris, A., and Casacuberta, F. *Interactive-Predictive Translation Based on Multiple Word-Segments.* In Baltic Journal of Modern Computing, 2016.

Dostert, L. E. *The Georgetown-IBM Experiment.* In Machine Translation of Languages, 1955.

Draper, N. and Smith, H. *Applied Regression Analysis.* Wiley, New York [u.a.], 1966.

Dredze, M., Kulesza, A., and Crammer, K. *Multi-Domain Learning by Confidence-Weighted Parameter Combination.* In Machine Learning, 2010.

Dreyer, M. and Dong, Y. *APRO: All-Pairs Ranking Optimization for MT Tuning.* In Proceedings of NAACL-HLT. 2015.

Duchi, J., Hazan, E., and Singer, Y. *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization.* Tech. Rep. UCB/EECS-2010-24, EECS Department, University of California, Berkeley, 2010.

Duh, K., Sudoh, K., Tsukada, H., Isozaki, H., and Nagata, M. *N-Best Reranking by Multitask Learning.* In Proceedings of WMT/MetricsMATR. 2010.

Dyer, C. *Using a Maximum Entropy Model to Build Segmentation Lattices for MT.* In Proceedings of NAACL-HLT. 2009.

Dyer, C. *Two Monolingual Parses Are Better Than One (Synchronous Parse).* In Proceedings of NAACL-HLT. 2010*a*.

Dyer, C. *Minimum Error Rate Training and the Convex Hull Semiring.* In CoRR, 2013.

Dyer, C., Chahuneau, V., and Smith, N. A. *A Simple, Fast, and Effective Reparameterization of IBM Model 2.* In Proceedings of ACL. 2013.

Dyer, C., Gimpel, K., Clark, J. H., and Smith, N. A. *The CMU-ARK German-English Translation System.* In Proceedings of WMT. 2011.

Dyer, C., Muresan, S., and Resnik, P. *Generalizing Word Lattice Translation.* Tech. rep., Maryland University, College Park, 2008.

Dyer, C., Weese, J., Setiawan, H., Lopez, A., Ture, F., Eidelman, V., Ganitkevitch, J., Blunsom, P., and Resnik, P. *cdec: A Decoder, Alignment, and Learning Framework for Finite-State and Context-Free Translation Models.* In Proceedings of ACL (System Demonstrations). 2010.

Dyer, C. J. *A Formal Model of Ambiguity and Its Applications in Machine Translation.* Ph.D. thesis, 2010*b*.

Earley, J. *An Efficient Context-Free Parsing Algorithm.* In Communications of the ACM, 1970.

Efron, B. *Bootstrap Methods: Another Look at the Jackknife.* In Breakthroughs in Statistics. Springer, Berlin, Germany, 1992.

Efron, B. and Tibshirani, R. J. *An Introduction to the Bootstrap.* CRC press, Boca Raton, Florida, 1994.

Eidelman, V. *Optimization Strategies for Online Large-Margin Learning in Machine Translation.* In Proceedings of WMT. 2012.

Eidelman, V., Marton, Y., and Resnik, P. *Online Relative Margin Maximization for Statistical Machine Translation.* In Proceedings of ACL. 2013*a*.

Eidelman, V., Wu, K., Ture, F., Resnik, P., and Lin, J. *Mr. MIRA: Open-Source Large-Margin Structured Learning on MapReduce.* In Proceedings of ACL (System Demonstrations). 2013*b*.

Eidelman, V., Wu, K., Türe, F., Resnik, P., and Lin, J. J. *Towards Efficient Large-Scale Feature-Rich Statistical Machine Translation.* In Proceedings of WMT. 2013*c*.

Elman, J. L. *Finding Structure in Time.* In Cognitive Science, 1990.

Erdmann, G. and Gwinnup, J. *Drem: The AFRL Submission to the WMT15 Tuning Task.* In Proceedings of WMT. 2015.

España Bonet, C., Enache, R., Slaski, A., Ranta, A., Màrquez Villodre, L., and González Bermúdez, M. *Patent Translation within the MOLTO Project.* In Proceedings of MT Summit XIII. 2011.

Esteban, J., Lorenzo, J., Valderrábanos, A. S., and Lapalme, G. *TransType2: An Innovative Computer-Assisted Translation System.* In Proceedings of ACL (Interactive Poster and Demonstration Sessions). 2004.

Estrella, P. S., Hamon, O., and Popescu-Belis, A. *How Much Data Is Needed for Reliable MT Evaluation? Using Bootstrapping to Study Human and Automatic Metrics.* In Proceedings of MT Summit XI. 2007.

Evgeniou, T. and Pontil, M. *Regularized Multi–Task Learning.* In Proceedings of SIGKDD. 2004.

Federico, M., Bertoldi, N., and Cettolo, M. *IRSTLM: An Open Source Toolkit for Handling Large Scale Language Models.* In Proceedings of Interspeech. 2008.

Federico, M., Bertoldi, N., Cettolo, M., Negri, M., Turchi, M., Trombetti, M., Cattelan, A., Farina, A., Lupinetti, D., Martines, A., Massidda, A., Schwenk, H., Barrault, L., Blain, F., Koehn, P., Buck, C., and Germann, U. *The Matecat Tool.* In Proceedings of COLING. 2014.

Federico, M., Cattelan, A., and Trombetti, M. *Measuring User Productivity in Machine Translation Enhanced Computer Assisted Translation.* In Proceedings of AMTA. 2012.

Finkel, J. R. and Manning, C. D. *Hierarchical Bayesian Domain Adaptation.* In Proceedings of NAACL-HLT. 2009.

Flanigan, J., Dyer, C., and Carbonell, J. G. *Large-Scale Discriminative Training for Statistical Machine Translation Using Held-Out Line Search.* In Proceedings of NAACL-HLT. 2013.

Flournoy, R. and Duran, C. *Machine Translation and Document Localization at Adobe: From Pilot to Production.* In Proceedings of MT Summit XII, 2009.

Forcada, M. L., Sánchez-Martínez, F., Esplà-Gomis, M., and Specia, L. *Towards Optimizing MT for Post-Editing Effort: Can BLEU Still Be Useful?* In The Prague Bulletin of Mathematical Linguistics, 2017.

Forster, K. I. and Dickinson, R. G. *More on the Language-as-Fixed-Effect Fallacy: Monte Carlo Estimates of Error Rates for F1, F2, F', and Min F'.* In Journal of Verbal Learning and Verbal Behavior, 1976.

Foster, G. *TransType Project Description.* Tech. rep., RALI-University of Montreal, 1998.

Foster, G., Isabelle, P., and Plamondon, P. *Word Completion: A First Step Toward Target-Text Mediated IMT.* In Proceedings of COLING. 1996.

Foster, G., Isabelle, P., and Plamondon, P. *Target-Text Mediated Interactive Machine Translation.* In Machine Translation, 1997.

Foster, G. and Kuhn, R. *Stabilizing Minimum Error Rate Training.* In Proceedings of WMT. 2009.

Foster, G., Langlais, P., and Lapalme, G. *TransType: Text Prediction for Translators.* In Proceedings of ICHLT. 2002*a*.

Foster, G., Langlais, P., and Lapalme, G. *User-Friendly Text Prediction for Translators.* In Proceedings of the EMNLP. 2002*b*.

Freitag, M. and Al-Onaizan, Y. *Fast Domain Adaptation for Neural Machine Translation.* In arXiv preprint arXiv:1612.06897, 2016.

Freund, Y. and Schapire, R. E. *Large Margin Classification Using the Perceptron Algorithm.* In Machine Learning, 1999.

Fuji, M., Fujita, A., Utiyama, M., Sumita, E., and Matsumoto, Y. *Patent Claim Translation Based on Sublanguage-Specific Sentence Structure.* In Proceedings of MT Summit XV, 2015.

Fürnkranz, J. and Hüllermeier, E. *Pairwise preference learning and ranking.* In Proceedings of ECML. 2003.

Fürnkranz, J. and Hüllermeier, E. *Preference Learning: An Introduction.* In Preference Learning. Springer, 2010.

Gale, W. A., Church, K. W., and Yarowsky, D. *One Sense Per Discourse.* In Proceedings of the Workshop on Speech and Natural Language. 1992.

Galley, M. and Manning, C. D. *Accurate Non-Hierarchical Phrase-Based Translation.* In Proceedings of NAACL-HLT. 2010.

Galley, M. and Quirk, C. *Optimal Search for Minimum Error Rate Training.* In Proceedings of EMNLP. 2011.

Galley, M., Quirk, C., Cherry, C., and Toutanova, K. *Regularized Minimum Error Rate Training.* In Proceedings of EMNLP. 2013.

Ganitkevitch, J., Cao, Y., Weese, J., Post, M., and Callison-Burch, C. *Joshua 4.0: Packing, PRO, and Paraphrases.* In Proceedings of WMT. 2012.

Gao, Q., Lewis, W., Quirk, C., and Hwang, M.-Y. *Incremental Training and Intentional Over-Fitting of Word Alignment.* In Proceedings of MT Summit XIII, 2011.

Gao, Q. and Vogel, S. *Parallel Implementations of Word Alignment Tool.* In Software Engineering, Testing, and Quality Assurance for Natural Language Processing. 2008.

Garcia, I. *Is Machine Translation Ready Yet?* In Target. International Journal of Translation Studies, 2010.

Garcia, I. *Translating by Post-Editing: Is It the Way Forward?* In Machine Translation, 2011.

Gaspari, F., Toral, A., Naskar, S. K., Groves, D., and Way, A. *Perception vs Reality: Measuring Machine Translation Post-Editing Productivity.* In Proceedings of the Workshop on Post-editing Technology and Practice. 2014.

Gentzsch, W. *Sun Grid Engine: Towards Creating a Compute Power Grid.* In Proceedings of First IEEE/ACM International Symposium on Cluster Computing and the Grid. 2001.

Germann, U. *Sampling Phrase Tables for the Moses Statistical Machine Translation System.* In The Prague Bulletin of Mathematical Linguistics, 2015.

Germann, U., Jahr, M., Knight, K., Marcu, D., and Yamada, K. *Fast Decoding and Optimal Decoding for Machine Translation.* In Proceedings of ACL. 2001.

Gimpel, K. and Smith, N. A. *Addendum to Structured Ramp Loss Minimization for Machine Translation.* Tech. rep., Language Technologies Institute, Carnegie Mellon University, USA, 2012*a*.

Gimpel, K. and Smith, N. A. *Structured Ramp Loss Minimization for Machine Translation.* In Proceedings of NAACL-HLT. 2012*b*.

Girardi, C., Bentivogli, L., Farajian, M. A., and Federico, M. *MT-EQuAl: A Toolkit for Human Assessment of Machine Translation Output.* In Proceedings of COLING (Demos). 2014.

Giraud-Carrier, C. *A Note on the Utility of Incremental Learning.* In Ai Communications, 2000.

González-Rubio, J., Ortiz-Martínez, D., Benedí, J.-M., and Casacuberta, F. *Interactive Machine Translation Using Hierarchical Translation Models.* In Proceedings of EMNLP. 2013.

Goodman, J. *Semiring Parsing.* In Computational Linguistics, 1999.

Goto, I., Chow, K.-P., Lu, B., Sumita, E., and Tsou, B. K. *Overview of the Patent Machine Translation Task at the NTCIR-10 Workshop.* In Proceedings of NTCIR. 2013.

Graham, Y., Baldwin, T., Moffat, A., and Zobel, J. *Continuous Measurement Scales in Human Evaluation of Machine Translation.* In Proceedings of LAW@ ACL. 2013*a*.

Graham, Y., Baldwin, T., Moffat, A., and Zobel, J. *Continuous Measurement Scales in Human Evaluation of Machine Translation.* In Proceedings of the 7th Linguistic Annotation Workshop & Interoperability with Discourse. 2013*b*.

Graham, Y., Baldwin, T., Moffat, A., and Zobel, J. *Can Machine Translation Systems Be Evaluated by the Crowd Alone.* In Natural Language Engineering, 2017.

Graham, Y., Mathur, N., and Baldwin, T. *Randomized Significance Tests in Machine Translation.* In Proceedings of WMT. 2014.

Green, S. *Mixed-Initiative Natural Language Translation.* Ph.D. thesis, Stanford University, 2014.

Green, S., Cer, D., and Manning, C. *An Empirical Comparison of Features and Tuning for Phrase-Based Machine Translation.* In Proceedings of WMT. 2014*a*.

Green, S., Chuang, J., Heer, J., and Manning, C. D. *Predictive Translation Memory: A Mixed-Initiative System for Human Language Translation.* In Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology. 2014*b*.

Green, S., Heer, J., and Manning, C. D. *The Efficacy of Human Post-Editing for Language Translation.* In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 2013*a*.

Green, S., Wang, S. I., Cer, D. M., and Manning, C. D. *Fast and Adaptive Online Training of Feature-Rich Translation Models.* In Proceedings of ACL. 2013*b*.

Green, S., Wang, S. I., Chuang, J., Heer, J., Schuster, S., and Manning, C. D. *Human Effort and Machine Learnability in Computer Aided Translation.* In Proceedings of EMNLP. 2014*c*.

Guerberof, A. *Productivity and Quality in MT Post-Editing.* In Proceedings of MT Summit XII: Workshop Beyond Translation Memories: New Tools for Translators MT. 2009.

Guerberof, A. *What Do Professional Translators Think About Post-Editing?* In The Journal of Specialised Translation, 2013.

Haddow, B. *Applying Pairwise Ranked Optimisation to Improve the Interpolation of Translation Models.* In Proceedings of NAACL-HLT. 2013.

Haddow, B., Arun, A., and Koehn, P. *SampleRank Training for Phrase-Based Machine Translation.* In Proceedings of WMT. 2011.

Hajlaoui, N., Kolovratnik, D., Väyrynen, J., Steinberger, R., and Varga, D. *DCEP-Digital Corpus of the European Parliament.* In Proceedings of LREC. 2014.

Han, A. L.-F. and Wong, D. F. *Machine Translation Evaluation: A Survey.* In arXiv preprint arXiv:1605.04515, 2016.

Hardt, D. and Elming, J. *Incremental Re-Training for Post-Editing SMT.* In Proceedings of AMTA. 2010.

Hasan, S., Zens, R., and Ney, H. *Are Very Large N-Best Lists Useful for SMT?* In Proceedings of NAACL-HLT (Short Papers). 2007.

Hasler, E., Blunsom, P., Koehn, P., and Haddow, B. *Dynamic Topic Adaptation for Phrase-Based MT.* In Proceedings of EACL. 2014.

Hasler, E. and Haddow, B. *Sparse Lexicalised Features and Topic Adaptation for SMT.* In Proceedings of IWSLT. 2012.

Hasler, E., Haddow, B., and Koehn, P. *Margin Infused Relaxed Algorithm for Moses.* In The Prague Bulletin of Mathematical Linguistics, 2011.

He, X. and Deng, L. *Maximum Expected Bleu Training of Phrase and Lexicon Translation Models.* In Proceedings of ACL. 2012.

Heafield, K. *KenLM: Faster and Smaller Language Model Queries.* In Proceedings of EMNLP. 2011.

Heafield, K., Pouzyrevsky, I., Clark, J. H., and Koehn, P. *Scalable Modified Kneser-Ney Language Model Estimation.* In Proceedings of ACL. 2013.

Hearst, M. A. *TextTiling: Segmenting Text Into Multi-Paragraph Subtopic Passages.* In Computational Linguistics, 1997.

Helwig, N. E. *Linear Mixed-Effects Regression.* 2017.

Herbrich, R., Graepel, T., and Obermayer, K. *Support Vector Learning for Ordinal Regression.* In Proceedings of ICANN. 1999.

Hieber, F. and Riezler, S. *Bag-of-Words Forced Decoding for Cross-Lingual Information Retrieval.* In Proceedings of NAACL-HLT. 2015.

Hinton, G. E. and Salakhutdinov, R. R. *Reducing the Dimensionality of Data with Neural Networks.* In Science, 2006.

Hochreiter, S. and Schmidhuber, J. *Long Short-Term Memory.* In Neural Computation, 1997.

Hoerl, A. E. and Kennard, R. W. *Ridge Regression: Biased Estimation for Nonorthogonal Problems.* In Technometrics, 1970.

Hopkins, M. and Langmead, G. *SCFG Decoding without Binarization.* In Proceedings of EMNLP. 2010.

Hopkins, M. and May, J. *Tuning as Ranking.* In Proceedings of EMNLP. 2011.

Hu, C. *Collaborative Translation by Monolingual Users.* In Proceedings of CHI. 2009.

Hu, C., Bederson, B. B., and Resnik, P. *Translation by Iterative Collaboration Between Monolingual Users.* In Proceedings of Graphics Interface. 2010.

Huang, L. *Advanced Dynamic Programming in Semiring and Hypergraph Frameworks.* In Proceedings of COLING. 2008.

Huang, L. and Chiang, D. *Better k-Best Parsing*. In Proceedings of IWPT. 2005.

Huang, L. and Chiang, D. *Forest Rescoring: Faster Decoding with Integrated Language Models*. In Proceedings of ACL. 2007.

Huang, L., Fayong, S., and Guo, Y. *Structured Perceptron with Inexact Search*. In Proceedings of NAACL-HLT. 2012.

Hutchins, J. *Looking Back to 1952: The First MT Conference*. In Theoretical and Methodogical Issues in Machine Translation, 1997.

Hutchins, J. *Machine Translation: A Concise History*. In Computer Aided Translation: Theory and Practice, 2007.

Hutchins, W. J. and Somers, H. L. *An Introduction to Machine Translation*, vol. 362. Academic Press London, 1992.

IWSLT. *International Workshop on Spoken Language Translation, IWSLT*. ISCA Speech, 2004.

IWSLT. *International Workshop on Spoken Language Translation, IWSLT*. ISCA Speech, 2013.

IWSLT. *International Workshop on Spoken Language Translation, IWSLT*. ISCA Speech, 2015.

Jääskeläinen, R. *Think-Aloud Protocol*. In Handbook of Translation Studies, 2010.

Jakobsen, A. L. *Logging Target Text Production with Translog*. In Copenhagen Studies in Language, 1999.

Jehl, L., Simianer, P., Hitschler, J., and Riezler, S. *The Heidelberg University English-German Translation System for IWSLT 2015*. In Proceedings of IWSLT. Da Nang, Vietnam, 2015.

Jiang, X., Hu, Y., and Li, H. *A Ranking Approach to Keyphrase Extraction*. In Proceedings of SIGIR. 2009.

Joachims, T. *Optimizing Search Engines Using Clickthrough Data*. In Proceedings of SIGKDD. 2002.

Joachims, T. *Training Linear SVMs in Linear Time*. In Proceedings of SIGKDD. 2006.

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Y., Chen, Z., Thorat, N., Viégas, F. B., Wattenberg, M., Corrado, G., Hughes, M., and Dean, J. *Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation*. In CoRR, 2016.

Johnson, R., King, M., and des Tombe, L. *Eurotra: A Multilingual System Under Development.* In Computational Linguistics, 1985.

Judd, C. M., Westfall, J., and Kenny, D. A. *Treating Stimuli as a Random Factor in Social Psychology: A New and Comprehensive Solution to a Pervasive but Largely Ignored Problem.* In Journal of Personality and Social Psychology, 2012.

Jung, L. *Deutsche Sprachprüfung Für Den Hochschulzugang Ausländischer Studienbewerber (DSH)*, vol. 1. Hueber, 1995.

Kalchbrenner, N. and Blunsom, P. *Recurrent Continuous Translation Models.* In Proceedings of EMNLP. 2013.

Karimova, S., Simianer, P., and Riezler, S. *Offline Extraction of Overlapping Phrases for Hierarchical Phrase-Based Translation.* In Proceedings of IWSLT. Lake Tahoe, USA, 2014.

Karimova, S., Simianer, P., and Riezler, S. *A User-Study on Online Adaptation of Neural Machine Translation to Human Post-Edits.* In arXiv preprint arXiv:1712.04853, 2017.

Kay, M. *The Proper Place of Men and Machines in Language Translation.* In Machine Translation, 1997.

Kay, M. and Martins, G. R. *The MIND System.* Tech. rep., RAND Corporation, 1970.

Kendall, M. G. *A New Measure of Rank Correlation.* In Biometrika, 1938.

Kernighan, M. D., Church, K. W., and Gale, W. A. *A Spelling Correction Program Based on a Noisy Channel Model.* In Proceedings of COLING. 1990.

Kim, H.-W. and Kankanhalli, A. *Investigating User Resistance to Information Systems Implementation: A Status Quo Bias Perspective.* In MIS quarterly, 2009.

Kingma, D. P. and Ba, J. *Adam: A Method for Stochastic Optimization.* In CoRR, 2014.

Klein, D. and Manning, C. D. *Parsing and Hypergraphs.* In New Developments in Parsing Technology, 2004.

Kneser, R. and Ney, H. *Improved Backing-Off for M-gram Language Modeling.* In Proceedings of ICASSP. 1995.

Knight, K. *Decoding Complexity in Word-Replacement Translation Models.* In Computational Linguistics, 1999.

Knowles, R. and Koehn, P. *Neural Interactive Translation Prediction.* In Proceedings of AMTA. 2016.

Koehn, P. *Pharaoh: A Beam Search Decoder for Phrase-Based Statistical Machine Translation Models.* In Machine Translation: From Real Users to Research, 2004*a*.

Koehn, P. *Statistical Significance Tests for Machine Translation Evaluation.* In Proceedings of EMNLP. 2004*b*.

Koehn, P. *Europarl: A Parallel Corpus for Statistical Machine Translation.* In Proceedings of MT Summit X. 2005.

Koehn, P. *A Process Study of Computer-Aided Translation.* In Machine Translation, 2009.

Koehn, P. *Enabling Monolingual Translators: Post-Editing vs. Options.* In Proceedings of NAACL-HLT. 2010.

Koehn, P. and Germann, U. *The Impact of Machine Translation Quality on Human Post-Editing.* In Proceedings of the Workshop on Humans and Computer-Assisted Translation. 2014.

Koehn, P. and Haddow, B. *Interactive Assistance to Human Translators Using Statistical Machine Translation Methods.* In Proceedings of MT Summit XII. 2009.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. *Moses: Open Source Toolkit for Statistical Machine Translation.* In Proceedings of ACL (Interactive Poster and Demonstration Sessions). 2007.

Koehn, P. and Knight, K. *Empirical Methods for Compound Splitting.* In Proceedings of EACL. 2003.

Koehn, P. and Knowles, R. *Six Challenges for Neural Machine Translation.* In arXiv preprint arXiv:1706.03872, 2017.

Koehn, P. and Monz, C. *Manual and Automatic Evaluation of Machine Translation Between European Languages.* In Proceedings of WMT. 2006*a*.

Koehn, P. and Monz, C. *Proceedings on the Workshop on Statistical Machine Translation.* In Proceedings of WMT. 2006*b*.

Koehn, P., Och, F. J., and Marcu, D. *Statistical Phrase-Based Translation.* In Proceedings of NAACL-HLT. 2003.

Koehn, P. and Schroeder, J. *Experiments in Domain Adaptation for Statistical Machine Translation.* In Proceedings of WMT. 2007.

Koglin, A. *An Empirical Investigation of Cognitive Effort Required to Post-Edit Machine Translated Metaphors Compared to the Translation of Metaphors.* In Translation & Interpreting, 2015.

Kohavi, R. and John, G. H. *Wrappers for Feature Subset Selection.* In Artificial Intelligence, 1997.

Koponen, M. *Assessing Machine Translation Quality with Error Analysis.* In Electronic Proceedings of the KäTu Symposium on Translation and Interpreting Studies. 2010.

Koponen, M. *Comparing Human Perceptions of Post-Editing Effort with Post-Editing Operations.* In Proceedings of WMT. 2012.

Koponen, M. *This Translation Is Not Too Bad: An Analysis of Post-Editor Choices in a Machine Translation Post-Editing Task.* In Proceedings of the Workshop on Post-editing Technology and Practice. 2013.

Koponen, M. *Is Machine Translation Post-Editing Worth the Effort? A Survey of Research Into Post-Editing and Effort.* In The Journal of Specialised Translation, 2016.

Koponen, M., Aziz, W., Ramos, L., and Specia, L. *Post-Editing Time as a Measure of Cognitive Effort.* In , 2012.

Krings, H. P. *Texte Reparieren: Empirische Untersuchungen Zum Prozeß Der Nachredaktion Von Maschinen-Übersetzungen.* Narr, 1997.

Krings, H. P. *Repairing Texts: Empirical Investigations of Machine Translation Post-Editing Processes*, vol. 5. Kent State University Press, 2001.

Kudo, T. *MeCab: Yet Another Part-of-Speech and Morphological Analyzer.* Tech. rep., 2005.

Kumar, S., Macherey, W., Dyer, C., and Och, F. *Efficient Minimum Error Rate Training and Minimum Bayes-Risk Decoding for Translation Hypergraphs and Lattices.* In Proceedings of ACL-IJCNLP. 2009.

Lacruz, I., Denkowski, M., and Lavie, A. *Cognitive Demand and Cognitive Effort in Post-Editing.* In Proceedings of AMTA. 2014.

Lacruz, I., Shreve, G. M., and Angelone, E. *Average Pause Ratio as an Indicator of Cognitive Effort in Post-Editing: A Case Study.* In Proceedings of the Workshop on Post-editing Technology and Practice. 2012.

Lagoudaki, E. *Translation Editing Environments.* In MT Summit XII: Workshop on Beyond Translation Memories. 2009.

Laird, N. M. and Ware, J. H. *Random-Effects Models for Longitudinal Data.* In Biometrics, 1982.

Lal, T., Chapelle, O., Weston, J., and Elisseeff, A. *Embedded Methods.* In Feature Extraction: Foundations and Applications. Springer, Berlin, Germany, 2006.

Lambert, P. and Banchs, R. *Tuning Machine Translation Parameters with SPSA.* In Proceedings of IWSLT. 2006.

Langlais, P., Foster, G., and Lapalme, G. *TransType: A Computer-Aided Translation Typing System.* In Proceedings of NAACL-ANLP Workshop on Embedded Machine Translation Systems. 2000*a*.

Langlais, P., Sauvé, S., Foster, G. F., Macklovitch, E., and Lapalme, G. *Evaluation of TransType, a Computer-Aided Translation Typing System: A Comparison of a Theoretical- and a User-Oriented Evaluation Procedures.* In Proceedings of LREC. 2000*b*.

Läubli, S., Fishel, M., Massey, G., Ehrensberger-Dow, M., and Volk, M. *Assessing Post-Editing Efficiency in a Realistic Translation Environment.* In Proceedings of the Workshop on Post-editing Technology and Practice. 2013.

Lavie, A., Sagae, K., and Jayaraman, S. *The Significance of Recall in Automatic Metrics for MT Evaluation.* In Proceedings of AMTA. 2004.

LDC. *Linguistic Data Annotation Specification: Assessment of Fluency and Adequacy in Translations.* Tech. rep., Linguistic Data Consortium, 2005.

Le, Q. V. and Smola, A. J. *Direct Optimization of Ranking Measures.* In CoRR, 2007.

Lee, C.-P. and Lin, C.-J. *Large-Scale Linear Ranksvm.* In Neural Computation, 2014.

Leusch, G., Matusov, E., and Ney, H. *Complexity of Finding the BLEU-optimal Hypothesis in a Confusion Network.* In Proceedings of EMNLP. 2008.

Levenberg, A., Callison-Burch, C., and Osborne, M. *Stream-Based Translation Models for Statistical Machine Translation.* In Proceedings of NAACL-HLT. 2010.

Levenberg, A. and Osborne, M. *Stream-Based Randomised Language Models for SMT.* In Proceedings of EMNLP. 2009.

Levenshtein, V. I. *Binary Codes Capable of Correcting Deletions, Insertions, and Reversals.* In Soviet Physics Doklady. 1966.

Li, H. *Learning to Rank for Information Retrieval and Natural Language Processing.* Morgan & Claypool Publishers, 2011*a*.

Li, H. *A Short Introduction to Learning to Rank.* In IEICE Transactions, 2011*b*.

Liang, P., Bouchard-Côté, A., Klein, D., and Taskar, B. *An End-to-End Discriminative Approach to Machine Translation.* In Proceedings of COLING/ACL. 2006*a*.

Liang, P. and Klein, D. *Online EM for Unsupervised Models.* In Proceedings of NAACL HLT. 2009.

Liang, P., Taskar, B., and Klein, D. *Alignment by Agreement.* In Proceedings of NAACL-HLT. 2006*b*.

Lin, C.-Y. and Och, F. J. *ORANGE: A Method for Evaluating Automatic Evaluation Metrics for Machine Translation.* In Proceedings of COLING. 2004.

Lita, L. V., Ittycheriah, A., Roukos, S., and Kambhatla, N. *Truecasing.* In Proceedings of ACL. 2003.

Liu, L. and Huang, L. *Search-Aware Tuning for Machine Translation.* In Proceedings of EMNLP. 2014.

Liu, L., Zhao, T., Watanabe, T., and Sumita, E. *Tuning SMT with a Large Number of Features via Online Feature Grouping.* In Proceedings of IJCNLP. 2013.

Liu, T.-Y. *Learning to Rank for Information Retrieval.* In Foundations and Trends in Information Retrieval, 2009.

Lo, S. and Andrews, S. *To transform or not to transform: Using generalized linear mixed models to analyse reaction time data.* In Frontiers in Psychology, 2015.

Lopez, A. *Hierarchical Phrase-Based Translation with Suffix Arrays.* In Proceedings of EMNLP-CoNLL. 2007.

Lopez, A. *Translation as Weighted Deduction.* In Proceedings of EACL. 2009.

López-Salcedo, F.-J., Sanchis-Trilles, G., and Casacuberta, F. *Online Learning of Log-Linear Weights in Interactive Machine Translation.* In Proeecings of IberSPEECH. 2012.

Luce, R. D. *Individual Choice Behavior a Theoretical Analysis.* John Wiley and sons, 1959.

Luong, M.-T. and Manning, C. D. *Stanford Neural Machine Translation Systems for Spoken Language Domains.* In Proceedings of IWSLT. 2015.

Macherey, W., Och, F. J., Thayer, I., and Uszkoreit, J. *Lattice-Based Minimum Error Rate Training for Statistical Machine Translation.* In Proceedings of EMNLP. 2008.

Macklovitch, E. *TransType2: The Last Word.* In Proceedings of LREC. 2006.

Mairal, J. and Yu, B. *Complexity Analysis of the Lasso Regularization Path.* In arXiv preprint arXiv:1205.0079, 2012.

Manning, C. D., Raghavan, P., and Schütze, H. *Introduction to Information Retrieval.* Cambridge University Press, New York City, NY, USA, 2008.

Marie, B. and Max, A. *Touch-Based Pre-Post-Editing of Machine Translation Output.* In Proceedings of EMNLP. 2015.

Martínez-Gómez, P., Sanchis-Trilles, G., and Casacuberta, F. *Online Learning via Dynamic Reranking for Computer Assisted Translation.* In Computational Linguistics and Intelligent Text Processing, 2011.

Martínez-Gómez, P., Sanchis-Trilles, G., and Casacuberta, F. *Online Adaptation Strategies for Statistical Machine Translation in Post-Editing Scenarios.* In Pattern Recognition, 2012.

Martins, A. F. T., Gimpel, K., Smith, N. A., Xing, E. P., Figueiredo, M. A. T., and Aguiar, P. M. Q. *Aggressive Online Learning of Structured Classifiers.* Tech. rep., School of Computer Science, Carnegie Mellon University, Pittsburgh PA, USA, 2010.

Martins, A. F. T., Smith, N. A., Aguiar, P. M. Q., and Figueiredo, M. A. T. *Structured Sparsity in Structured Prediction.* In Proceedings of EMNLP. 2011.

Marton, Y. and Resnik, P. *Soft Syntactic Constraints for Hierarchical Phrased-Based Translation.* In Proceedings of ACL. 2008.

Mathur, P. and Cettolo, M. *Optimized MT Online Learning in Computer Assisted Translation.* In Workshop on Interactive and Adaptive Machine Translation. 2014.

Mathur, P., Cettolo, M., Federico, M., and de Souza, J. G. *Online Multi-User Adaptive Statistical Machine Translation.* In Proceedings of AMTA. 2014.

Mathur, P., Cettolo, M., Federico, M., and Kessler, F.-F. B. *Online Learning Approaches in Computer Assisted Translation.* In Proceedings of WMT. 2013.

Mays, E., Damerau, F. J., and Mercer, R. L. *Context Based Spelling Correction.* In Information Processing & Management, 1991.

McDonald, R., Hall, K., and Mann, G. *Distributed Training Strategies for the Structured Perceptron.* In Proceedings of NAACL-HLT. 2010.

McDonald, R., Pereira, F., Ribarov, K., and Hajič, J. *Non-Projective Dependency Parsing Using Spanning Tree Algorithms.* In Proceedings of EMNLP. 2005.

Metzler, D. and Kanungo, T. *Machine Learned Sentence Selection Strategies for Query-Biased Summarization.* In SIGIR Learning to Rank Workshop. 2008.

Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J., and Khudanpur, S. *Recurrent Neural Network Based Language Model.* In Proceedings of Interspeech. 2010.

Mirkin, S. and Cancedda, N. *Assessing Quick Update Methods of Statistical Translation Models.* In Proceedings of IWSLT. 2013.

Mitchell, L., Roturier, J., and O'Brien, S. *Community-Based Post-Editing of Machine-Translated Content: Monolingual vs. Bilingual.* In Proceedings of MT Summit XIV. 2013.

Mizumoto, T. and Matsumoto, Y. *Discriminative Reranking for Grammatical Error Correction with Statistical Machine Translation.* In Proceedings of NAACL-HLT. 2016.

Mladenić, D. *Feature Selection for Dimensionality Reduction.* In Proceedings of Subspace, Latent Structure and Feature Selection: Statistical and Optimization Perspectives Workshop. 2006.

Moore, R. C. and Quirk, C. *Random Restarts in Minimum Error Rate Training for Statistical Machine Translation.* In Proceedings of COLING. 2008.

Moorkens, J. and O'Brien, S. *Post-Editing Evaluations: Trade-Offs Between Novice and Professional Participants.* In Proceedings of EAMT, 2015.

Mozer, M. C. *A Focused Back-Propagation Algorithm for Temporal Pattern Recognition.* In Complex Systems, 1989.

Nakazawa, T., Mino, H., Ding, C., Goto, I., Neubig, G., Kurohashi, S., and Sumita, E. *Overview of the 3rd Workshop on Asian Translation.* In Proceedings of the Workshop on Asian Translation, 2016.

Nakazawa, T., Mino, H., Goto, I., Kurohashi, S., and Sumita, E. *Overview of the 1st Workshop on Asian Translation.* In Proceedings of the Workshop on Asian Translation. 2014.

Nakov, P., Guzman, F., and Vogel, S. *Optimizing for Sentence-Level BLEU+1 Yields Short Translations.* In Proceedings of COLING. 2012.

Nakov, P., Guzmán, F., and Vogel, S. *A Tale About PRO and Monsters.* In Proceedingsof ACL. 2013.

Nepveu, L., Lapalme, G., Langlais, P., and Foster, G. *Adaptive Language and Translation Models for Interactive Machine Translation.* In Proceedings of EMNLP. 2004.

Neubig, G. *Lamtram: A Toolkit for Language and Translation Modeling Using Neural Networks.* Tech. rep., Carnegie Mellon University Pittsburgh, USA, 2015.

Neubig, G. *Lexicons and Minimum Risk Training for Neural Machine Translation: NAIST-CMU at Wat2016.* In arXiv preprint arXiv:1610.06542, 2016.

Neubig, G. *Neural Machine Translation and Sequence-to-Sequence Models: A Tutorial.* In arXiv preprint arXiv:1703.01619, 2017.

Neubig, G., Dyer, C., Goldberg, Y., Matthews, A., Ammar, W., Anastasopoulos, A., Ballesteros, M., Chiang, D., Clothiaux, D., Cohn, T. et al. *DyNet: The Dynamic Neural Network Toolkit.* In arXiv preprint arXiv:1701.03980, 2017.

Ney, H., Essen, U., and Kneser, R. *On Structuring Probabilistic Dependences in Stochastic Language Modelling.* In Computer Speech & Language, 1994.

Niehues, J., DO, Q.-K., Allauzen, A., and Waibel, A. *ListNet-Based MT Rescoring.* In Proceedings of WMT. 2015.

Nitzke, J. *Monolingual Post-Editing: An Exploratory Study on Research Behaviour and Target Text Quality.* In Eyetracking and Applied Linguistics, 2016.

Obozinski, G., Taskar, B., and Jordan, M. I. *Joint Covariate Selection and Joint Subspace Selection for Multiple Classification Problems.* In Statistics and Computing, 2010.

O'Brien, S. and Moorkens, J. *Towards Intelligent Post-Editing Interfaces.* In Proceedings of the XXth FIT World Congress. 2014.

Och, F. J. *Minimum Error Rate Training in Statistical Machine Translation.* In Proceedings of ACL. 2003.

Och, F. J., Gildea, D., Khudanpur, S., Sarkar, A., Yamada, A., Kenji Fraser, Kumar, S., Shen, D., Libin Smith, Eng, K., Jain, V., Jin, Z., and Radev, D. *Syntax for Statistical Machine Translation: Final Report of John Hopkins 2003 Summer Workshop.* Tech. rep., John Hopkins University, Baltimore, USA, 2003*a*.

Och, F. J., Gildea, D., Khudanpur, S., Sarkar, A., Yamada, K., Fraser, A., Kumar, S., Shen, L., Smith, D., Eng, K. et al. *A Smorgasbord of Features for Statistical Machine Translation.* In Proceedings of NAACL-HLT. 2004.

Och, F. J. and Ney, H. *Discriminative Training and Maximum Entropy Models for Statistical Machine Translation.* In Proceedings of ACL. 2002.

Och, F. J. and Ney, H. *A Systematic Comparison of Various Statistical Alignment Models.* In Computational Linguistics, 2003.

Och, F. J., Tillmann, C., Ney, H. et al. *Improved Alignment Models for Statistical Machine Translation.* In Proceedings of the Joint SIGDAT Conference On Empirical Methods in Natural Language Processing and Very Large Corpora. 1999.

Och, F. J., Ueffing, N., and Ney, H. *An Efficient A* Search Algorithm for Statistical Machine Translation.* In Proceedings of the Workshop on Data-Driven Methods in Machine Translation. 2001.

Och, F. J., Zens, R., and Ney, H. *Efficient Search for Interactive Statistical Machine Translation.* In Proceedings of EACL. 2003*b*.

Orr, D. B. and Small, V. H. *Comprehensibility of Machine-Aided Translations of Russian Scientific Documents.* In Mechanical Translation & Computational Linguistics, 1967.

Orsnes, B., Music, B., and Maegaard, B. *PaTrans: A Patent Translation System.* In Proceedings of COLING. 1996.

Ortiz-Martínez, D. *Online Learning for Statistical Machine Translation.* In Computational Linguistics, 2016.

Ortiz-Martınez, D., Benedı, J. M., and Casacuberta, F. *Beyond Prefix-Based Interactive Translation Prediction.* In Proceedings of CoNLL, 2016.

Ortiz-Martínez, D., García-Varea, I., and Casacuberta, F. *Interactive Machine Translation Based on Partial Statistical Phrase-Based Alignments.* In Proceedings of RANLP. 2009.

Ortiz-Martínez, D., García-Varea, I., and Casacuberta, F. *Online Learning for Interactive Statistical Machine Translation.* In Proceedings of NAACL-HLT. 2010.

Ortiz-Martínez, D., Leiva, L. A., Alabau, V., García-Varea, I., and Casacuberta, F. *An Interactive Machine Translation System with Online Learning.* In Proceedings of ACL-HLT (Systems Demonstrations). 2011.

O'Brien, S. *Methodologies for Measuring the Correlations Between Post-Editing Effort and Machine Translatability.* In Machine Translation, 2005.

O'Brien, S. *Towards Predicting Post-Editing Productivity.* In Machine Translation, 2011.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. *BLEU: A Method for Automatic Evaluation of Machine Translation.* In Proceedings of COLING. 2002.

Pareek, H. and Ravikumar, P. *A Representation Theory for Ranking Functions.* In Proceedings of NIPS. 2014.

Parker, R., Graff, D., Kong, J., Chen, K., and Maeda, K. *English Gigaword Fifth Edition.* Tech. rep., Linguistic Data Consortium, Philadelphia, USA, 2011.

Pearson, K. *Note on Regression and Inheritance in the Case of Two Parents.* In Proceedings of the Royal Society of London, 1895.

Peris, Á., Cebrián, L., and Casacuberta, F. *Online Learning for Neural Machine Translation Post-Editing.* In CoRR, 2017*a*.

Peris, Á., Domingo, M., and Casacuberta, F. *Interactive Neural Machine Translation.* In Computer Speech & Language, 2017*b*.

Perkins, S., Lacker, K., and Theiler, J. *Grafting: Fast, Incremental Feature Selection by Gradient Descent in Function Space.* In Machine Learning Research, 2003.

Pierce, J. R. and Carroll, J. B. *Language and Machines: Computers in Translation and Linguistics.* National Academy of Sciences/National Research Council, 1966.

Pinnis, M., Kalnins, R., Skadins, R., and Skadina, I. *What Can We Really Learn From Post-Editing.* In Proceedings of AMTA, 2016.

Plackett, R. L. *The Analysis of Permutations.* In Applied Statistics, 1975.

Plitt, M. and Masselot, F. *A Productivity Test of Statistical Machine Translation Post-Editing in a Typical Localisation Context.* In Prague Bulleting of Mathematical Linguistics, 2010.

Popovic, M., Lommel, A., Burchardt, A., Avramidis, E., and Uszkoreit, H. *Relations Between Different Types of Post-Editing Operations, Cognitive Effort and Temporal Effort.* In Proceedings of EAMT. 2014.

Pouliquen, B., Mazenc, C., and Iorio, A. *Tapta: A User-Driven Translation System for Patent Documents Based on Domain-Aware Statistical Machine Translation.* In Proceedings of EAMT. 2011.

Powell, M. J. *An Efficient Method for Finding the Minimum of a Function of Several Variables without Calculating Derivatives.* In The Computer Journal, 1964.

Quattoni, A., Carreras, X., Collins, M., and Darrell, T. *An Efficient Projection for $l_{1,\infty}$ Regularization.* In Proceedings of ICML. 2009.

Quenouille, M. H. *Notes on Bias in Estimation.* In Biometrika, 1956.

Rafferty, A. N. and Manning, C. D. *Parsing Three German Treebanks: Lexicalized and Unlexicalized Baselines.* In Proceedings of the Workshop on Parsing German. 2008.

Reifler, E. *The First Conference on Mechanical Translation.* In Mechanical Translation, 1954.

Riedel, S. and Clarke, J. *Revisiting Optimal Decoding for Machine Translation IBM Model 4.* In Proceedings of NAACL-HLT (Short Papers). 2009.

Riezler, S. and Maxwell, J. T. *On Some Pitfalls in Automatic Evaluation and Significance Testing for MT.* In Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization. 2005.

Rosenblatt, F. *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain.* In Psychological Review, 1958.

Rosti, A.-V. I., Zhang, B., Matsoukas, S., and Schwartz, R. *BBN System Description for WMT10 System Combination Task.* In Proceedings of WMT/MetricsMATR. 2010.

Roturier, J., Mitchell, L., Silva, D., and Park, B. B. *The ACCEPT Post-Editing Environment: A Flexible and Customisable Online Tool to Perform and Analyse Machine Translation Post-Editing.* In Proceedings of the Workshop on Post-editing Technology and Practice. 2013.

Royston, J. P. *An extension of Shapiro and Wilk's W test for normality to large samples.* In Applied Statistics, 1982.

Rumelhart, D. E., Hinton, G. E., Williams, R. J. et al. *Learning Representations by Back-Propagating Errors.* In Cognitive Modeling, 1988.

Ryan, J. P. *SYSTRAN: A Machine Translation System to Meet User Needs.* In Machine Translation Summit. 1989.

Sakaguchi, K., Post, M., and Van Durme, B. *Efficient Elicitation of Annotations for Human Evaluation of Machine Translation.* In Proceedings of WMT. 2014.

Sakai, I. *Syntax in Universal Translation.* Her Magesty's Stationary Office, 1962.

Salton, G., Wong, A., and Yang, C. S. *A Vector Space Model for Automatic Indexing.* In Communications of ACM, 1975.

Sanchez-Torron, M. and Koehn, P. *Machine Translation Quality and Post-Editor Productivity.* In Proceedings of AMTA, 2016.

Sanchis-Trilles, G., Alabau, V., Buck, C., Carl, M., Casacuberta, F., García-Martínez, M., Germann, U., González-Rubio, J., Hill, R. L., Koehn, P. et al. *Interactive Translation Prediction vs. Conventional Post-Editing in Practice: A Study with the CasMaCat Workbench.* In Machine Translation, 2014.

Sanchis-Trilles, G., González, M.-T., Casacuberta, F., Vidal, E., and Civera, J. *Introducing Additional Input Information Into Interactive Machine Translation Systems.* In Machine Learning for Multimodal Interaction, 2008.

Sankaran, B., Razmara, M., Farzindar, A., Khreich, W., Popowich, F., and Sarkar, A. *Domain Adaptation Techniques for Machine Translation and Their Evaluation in a Real-World Setting.* In Proceedings of Advances in Artificial Intelligence. 2012.

Scheepers, T. and Schulz, P. *Interactive Neural Translation Assistance for Human Translators.* 2016.

Schmidt, M., Fung, G., and Rosales, R. *Fast Optimization Methods for L1 Regularization: A Comparative Study and Two New Approaches.* In Proceedings of ECML. 2007.

Schmidt, M., Fung, G., and Rosaless, R. *Optimization Methods for L1-Regularization.* Tech. rep., University of British Columbia, Canada, 2009.

Schuster, M. and Nakajima, K. *Japanese and Korean Voice Search.* In Proceedings of ICASSP. 2012.

Schuster, M. and Paliwal, K. K. *Bidirectional Recurrent Neural Networks.* In IEEE Transactions on Signal Processing, 1997.

Schwartz, L. *Monolingual Post-Editing by a Domain Expert Is Highly Effective for Translation Triage.* In Proceedings of the Workshop on Post-editing Technology and Practice. 2014.

Schwartz, L., Anderson, T., Gwinnup, J., and Young, K. *Machine Translation and Monolingual Postediting: The AFRL WMT-14 System.* In Proceedings of WMT. 2014.

Schwartz, L., Lacruz, I., and Bystrova, T. *Effects of Word Alignment Visualization on Post-Editing Quality & Speed.* In Proceedings of MT Summit XV. 2015.

Sculley, D. *Combined Regression and Ranking.* In Proceedings of SIGKDD. 2010.

Sekino, K. *An Investigation of the Relevance-Theoretical Approach to Cognitive Effort in Translation and the Post-Editing Process.* In Translation & Interpreting, 2015.

Sennrich, R. *A CYK+ Variant for SCFG Decoding without a Dot Chart.* In Proceedings of SSST@ EMNLP. 2014.

Sennrich, R., Haddow, B., and Birch, A. *Neural Machine Translation of Rare Words with Subword Units.* In arXiv preprint arXiv:1508.07909, 2015.

Sennrich, R., Haddow, B., and Birch, A. *Edinburgh Neural Machine Translation Systems for WMT 16.* In Proceedings of WMT. 2016.

Serrano, N., Andres-Ferrer, J., and Casacuberta, F. *On a Kernel Regression Approach to Machine Translation.* In Pattern Recognition and Image Analysis, 2009.

Setiawan, H. and Zhou, B. *Discriminative Training of 150 Million Translation Parameters and Its Application to Pruning.* In Proceedings of NAACL-HLT. 2013.

Shannon, C. *A Mathematical Theory of Communication.* In Bell System Technical Journal, 1948.

Shaphiro, S. S. and Wilk, M. B. *An Analysis of Variance Test for Normality.* In Biometrika, 1965.

Sharmin, S., Špakov, O., Räihä, K.-J., and Jakobsen, A. L. *Effects of Time Pressure and Text Complexity on Translators' Fixations.* In Proceedings of the Symposium on Eye Tracking Research & Applications. 2008.

Shen, L. and Joshi, A. K. *Flexible Margin Selection for Reranking with Full Pairwise Samples.* In Proceedings of IJCNLP. 2004.

Shen, L. and Joshi, A. K. *Ranking and Reranking with Perceptron.* In Machine Learning, 2005.

Shen, L., Sarkar, A., and Och, F. J. *Discriminative Reranking for Machine Translation.* In Proceedings of NAACL-HLT. 2004.

Shen, S., Cheng, Y., He, Z., He, W., Wu, H., Sun, M., and Liu, Y. *Minimum Risk Training for Neural Machine Translation.* In CoRR, 2015.

Shterionov, D., Casanellas, P. N. L., Superbo, R., and O'Dowd, T. *Empirical Evaluation of NMT and PBSMT Quality for Large-Scale Translation Production.* In Proceedings of EAMT. 2017.

Simard, M. and Foster, G. *Pepr: Post-Edit Propagation Using Phrase-Based Statistical Machine Translation.* In Proceedings of MT Summit XIV, 2013.

Simard, M., Goutte, C., and Isabelle, P. *Statistical Phrase-Based Post-Editing.* 2007.

Simianer, P. *Tuning SMT on the Training Set.* Master's thesis, University of Heidelberg, 2012.

Simianer, P., Jehl, L., and Riezler, S. *The Heidelberg University Machine Translation Systems for IWSLT2013.* In International Workshop on Spoken Language Translation. Heidelberg, Germany, 2013*a*.

Simianer, P., Karimova, S., and Riezler, S. *A Post-Editing Interface for Immediate Adaptation in Statistical Machine Translation.* In Proceedings of COLING (System Demonstrations). Osaka, Japan, 2016.

Simianer, P. and Riezler, S. *Multi-Task Learning for Improved Discriminative Training in SMT.* In Proceedings of WMT. Sofia, Bulgaria, 2013.

Simianer, P., Riezler, S., and Dyer, C. *Joint Feature Selection in Distributed Stochastic Learning for Large-Scale Discriminative Training in SMT.* In Proceedings of ACL. Jeju, Korea, 2012.

Simianer, P., Stupperich, G., Jehl, L., Wäschle, K., Sokolov, A., and Riezler, S. *The HDU Discriminative SMT System for Constrained Data PatentMT at NTCIR10.* In Proceedings of NTCIR. Tokyo, Japan, 2013*b*.

Simianer, P., Wäschle, K., and Riezler, S. *Multi-Task Minimum Error Rate Training for SMT.* In The Prague Bulletin of Mathematical Linguistics, 2011.

Singhal, A. et al. *Modern Information Retrieval: A Brief Overview.* In IEEE Data Eng. Bull., 2001.

Skadiņš, R., Puriņš, M., Skadiņa, I., and Vasiļjevs, A. *Evaluation of SMT in Localization to Under-Resourced Inflected.* In Proceedings of EAMT. 2011.

Skiena, S. *Dijkstra's Algorithm.* In Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica, 1990.

Smith, D. A. and Eisner, J. *Minimum Risk Annealing for Training Log-Linear Models.* In Proceedings of COLING/ACL. 2006.

Smith, J. R., Saint-Amand, H., Plamada, M., Koehn, P., Callison-Burch, C., and Lopez, A. *Dirt Cheap Web-Scale Parallel Text from the Common Crawl.* In Proceedings of ACL. 2013.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. *A Study of Translation Edit Rate with Targeted Human Annotation.* In Proceedings of AMTA. 2006.

Snover, M., Madnani, N., Dorr, B. J., and Schwartz, R. *Fluency, Adequacy, or HTER?: Exploring Different Human Judgments with a Tunable MT Metric.* In Proceedings of WMT. 2009.

Søgaard, A., Johannsen, A., Plank, B., Hovy, D., and Alonso, H. M. *What's in a p-Value in NLP?* In Proceedings of CoNLL. 2014.

Sokolov, A., Wisniewski, G., and Yvon, F. *Computing Lattice BLEU Oracle Scores for Machine Translation.* In Proceedings of EACL. 2012*a*.

Sokolov, A., Wisniewski, G., and Yvon, F. *Non-Linear N-Best Reranking with Few Features.* In Proceedings of AMTA. San Diego, CA, 2012*b*.

Sokolov, A., Wisniewski, G., and Yvon, F. *Lattice BLEU Oracles in Machine Translation.* In ACM Transactions on Speech and Language Processing (TSLP), 2013.

Sokolov, A. and Yvon, F. *Minimum Error Rate Semiring.* In Proceedings of EAMT. 2011.

Somers, H. *Computers and Translation: A Translator's Guide*, vol. 35. John Benjamins Publishing, 2003.

Sparck Jones, K. *A Statistical Interpretation of Term Specificity and its Application in Retrieval.* In Journal of documentation, 1972.

Spearman, C. *The Proof and Measurement of Association Between Two Things.* In The American Journal of Psychology, 1904.

Specia, L., Turchi, M., Cancedda, N., Dymetman, M., and Cristianini, N. *Estimating the Sentence-Level Quality of Machine Translation Systems.* In Proceedings of EAMT. 2009.

Spence Green, D. C. and Manning, C. D. *Phrasal: A Toolkit for New Directions in Statistical Machine Translation.* In Proceedings of WMT. 2014.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. *Dropout: A Simple Way to Prevent Neural Networks From Overfitting.* In Journal of Machine Learning Research, 2014.

Steinberger, R., Pouliquen, B., Widiger, A., Ignat, C., Erjavec, T., Tufis, D., and Varga, D. *The JRC-Acquis: A Multilingual Aligned Parallel Corpus with 20+ Languages.* In CoRR, 2006.

Stolcke, A. *SRILM - An Extensible Language Modeling Toolkit.* In Proceedings of Interspeech. 2002.

Sturgeon, D. and Lee, J. S. *Translation Quality and Effort: Options versus Post-Editing.* In Proceedings of ICNLP. 2015.

Stymne, S. and Ahrenberg, L. *On the Practice of Error Analysis for Machine Translation Evaluation.* In Proceedings of LREC. 2012.

Su, K.-Y., Wu, M.-W., and Chang, J.-S. *A New Quantitative Quality Measure for Machine Translation Systems.* In Proceedings of COLING. 1992.

Sundermeyer, M., Schlüter, R., and Ney, H. *On the Estimation of Discount Parameters for Language Model Smoothing.* In Proceedings of Interspeech. 2011.

Surdeanu, M., Ciaramita, M., and Zaragoza, H. *Learning to Rank Answers on Large Online QA Collections.* In Proceedings of ACL. 2008.

Sutskever, I., Vinyals, O., and Le, Q. V. *Sequence to Sequence Learning with Neural Networks.* In Proceedings of NIPS. 2014.

Tan, L., Dehdari, J., and van Genabith, J. *An Awkward Disparity Between BLEU/RIBES Scores and Human Judgements in Machine Translation.* In Proceedings of the Workshop on Asian Translation. 2015.

Tan, M., Xia, T., Wang, S., and Zhou, B. *A Corpus Level MIRA Tuning Strategy for Machine Translation.* In Proceedings of EMNLP. 2013.

Tatsumi, M. *Correlation Between Automatic Evaluation Metric Scores, Post-Editing Speed, and Some Other Factors.* In Proceedings of MT Summit XII, 2009.

Tatsumi, M. *Post-Editing Machine Translated Text in a Commercial Setting: Observation and Statistical Analysis.* Ph.D. thesis, Dublin City University, 2010.

Teh, Y. W. *A Hierarchical Bayesian Language Model Based on Pitman-Yor Processes.* In Proceedings of ACL. 2006.

Thrun, S. and O'Sullivan, J. *Discovering Structure in Multiple Learning Tasks: The TC Algorithm.* In Proceedings of ICML. 1996.

Tian, X. *Learning to Rank Algorithms and their Application in Machine Translation.* Ph.D. thesis, Wright State University, Ohio, USA, 2015.

Tibshirani, R. *Regression Shrinkage and Selection via the Lasso.* In Journal of the Royal Statistical Society, Series B, 1994.

Tiedemann, J. and Nygaard, L. *The OPUS Corpus - Parallel and Free: http://logos.uio.no/opus.* In Proceedings of LREC. European Language Resources Association, 2004.

Tillmann, C. and Zhang, T. *A Localized Prediction Model for Statistical Machine Translation.* In Proceedings of ACL. 2005.

Tillmann, C. and Zhang, T. *A Discriminative Global Training Algorithm for Statistical MT.* In Proceedings of COLING. 2006.

Tinsley, J., Way, A., and Sheridan, P. *PLuTO: MT for Online Patent Translation.* In Proceedings of AMTA. 2010.

Tromble, R., Kumar, S., Och, F., and Macherey, W. *Lattice Minimum Bayes-Risk Decoding for Statistical Machine Translation.* In Proceedings of EMNLP. 2008.

Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. *Support Vector Machine Learning for Interdependent and Structured Output Spaces.* In Proceedings of ICML. 2004.

Tsujii, J.-i. *Future Directions of Machine Translation.* In Proceedings of COLING. 1986.

Tsuruoka, Y., Tsujii, J., and Ananiadou, S. *Stochastic Gradient Descent Training for L1-Regularized Log-Linear Models with Cumulative Penalty.* In Proceedings of ACL-IJCNLP. 2009.

Turchi, M., Negri, M., Farajian, M. A., and Federico, M. *Continuous Learning From Human Post-Edits for Neural Machine Translation.* In The Prague Bulletin of Mathematical Linguistics, 2017.

Turian, J. P., Shea, L., and Melamed, I. D. *Evaluation of Machine Translation and Its Evaluation.* Tech. rep., New York University, New York, USA, 2006.

Underwood, N. L., Mesa-Lao, B., García-Martínez, M., Carl, M., Alabau, V., González-Rubio, J., Leiva, L. A., Sanchis-Trilles, G., Ortíz-Martínez, D., and Casacuberta, F. *Evaluating the Effects of Interactivity in a Post-Editing Workbench.* In Proceedings of LREC. 2014.

Unicode Staff, C. *The Unicode Standard: Worldwide Character Encoding.* Addison-Wesley Longman Publishing Co., Inc., 1991.

Uszkoreit, H. *Survey of Machine Translation Evaluation.* Tech. rep., EuroMatrix Project, Germany, 2007.

Utiyama, M. and Isahara, H. *A Japanese-English Patent Parallel Corpus.* In Proceedings of MT Summit XI. 2007.

van der Wees, M., Bisazza, A., Weerkamp, W., and Monz, C. *What's in a Domain? Analyzing Genre and Topic Differences in Statistical Machine Translation.* In Proceedings of ACL (Short Papers). 2015.

Vapnik, V. *Estimation of Dependences Based on Empirical Data.* Springer Series in Statistics. Springer New York, Inc., Secaucus, NJ, USA, 1982.

Vauquois, B. *Structures Profondes Et Traduction Automatique. Le Système Du CETA.* In Revue Roumaine De Linguistique, 1968.

Vauquois, B. *La Traduction Automatique À Grenoble*, vol. 24. Dunod, 1975.

Velldal, E. and Oepen, S. *Maximum Entropy Models for Realization Ranking.* In In Proceedings of MT Summit X. 2005.

Vilar, D., Stein, D., and Ney, H. *Analysing Soft Syntax Features and Heuristics for Hierarchical Phrase Based Machine Translation.* In Proceedings of IWSLT. 2008.

Vilar, D., Xu, J., d'Haro, L. F., and Ney, H. *Error Analysis of Statistical Machine Translation Output.* In Proceedings of LREC. 2006.

Viterbi, A. *Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm.* In IEEE Transactions on Information Theory, 1967.

Wagner, E. *Rapid Post-Editing of Systran.* In Tools for the Trade, Translating and the Computer, 1985.

Wallis, J. *Interactive Translation vs Pre-Translation in the Context of Translation Memory Systems: Investigating the Effects of Translation Method on Productivity, Quality and Translator Satisfaction.* Ph.D. thesis, University of Ottawa (Canada), 2006.

Wang, Y.-Y. and Waibel, A. *Decoding Algorithm in Statistical Machine Translation.* In Proceedings of EACL. 1997.

Wang, Z., Shawe-Taylor, J., and Szedmak, S. *Kernel Regression Based Machine Translation.* In Proceedings of NAACL-HLT (Short Papers). 2007.

Wäschle, K. and Riezler, S. *Analyzing Parallelism and Domain Similarities in the MAREC Patent Corpus.* In Multidisciplinary Information Retrieval, 2012*a*.

Wäschle, K. and Riezler, S. *Structural and Topical Dimensions in Multi-Task Patent Translation.* In Proceedings of EACL. 2012*b*.

Watanabe, T. *Optimized Online Rank Learning for Machine Translation.* In Proceedings of NAACL-HLT. 2012.

Watanabe, T., Suzuki, J., Sudoh, K., Tsukada, H., and Isozaki, H. *Larger Feature Set Approach for Machine Translation in IWSLT 2007.* In Proceedings of IWSLT. 2007*a*.

Watanabe, T., Suzuki, J., Tsukada, H., and Isozaki, H. *NTT Statistical Machine Translation for IWSLT 2006.* In Proceedings of IWSLT. 2006.

Watanabe, T., Suzuki, J., Tsukada, H., and Isozaki, H. *Online Large-Margin Training for Statistical Machine Translation.* In In Proceedings of EMNLP. 2007*b*.

Weaver, W. *Translation.* In Machine Translation of Languages. MIT Press, Cambridge, MA, USA, 1949.

Weese, J. and Callison-Burch, C. *Visualizing Data Structures in Parsing-Based Machine Translation.* In The Prague Bulletin of Mathematical Linguistics, 2010.

Wellington, B., Turian, J., and Melamed, D. *Toward Purely Discriminative Training for Tree-Structured Translation Models.* In Learning Machine Translation. MIT Press, Cambridge, MA, USA, 2009.

White, J., O'Connell, T., and O'Mara, F. *The ARPA MT Evaluation Methodologies: Evolution, Lessons, and Future Approaches.* In Proceedings of AMTA. 1994.

Wick, M., Rohanimanesh, K., Bellare, K., Culotta, A., and McCallum, A. *SampleRank: Training Factor Graphs with Atomic Gradients.* In Proceedings of ICML. 2011.

Williams, P., Sennrich, R., Post, M., and Koehn, P. *Syntax-Based Statistical Machine Translation.* In Synthesis Lectures on Human Language Technologies, 2016.

WIPO. *WIPO Patent Drafting Manual.* 2014.

Wisniewski, G. and Yvon, F. *Fast Large-Margin Learning for Statistical Machine Translation.* In Proceedings of CICLing. 2013.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.* In CoRR, 2016.

Wuebker, J., Green, S., and DeNero, J. *Hierarchical Incremental Adaptation for Statistical Machine Translation.* In Proceedings of EMNLP. 2015*a*.

Wuebker, J., Green, S., DeNero, J., Hasan, S., and Luong, M.-T. *Models and Inference for Prefix-Constrained Machine Translation.* In Proceedings of ACL. 2016.

Wuebker, J., Hwang, M.-Y., and Quirk, C. *Leave-One-Out Phrase Model Training for Large-Scale Deployment.* In Proceedings of WMT. 2012.

Wuebker, J., Muehr, S., Lehnen, P., Peitz, S., and Ney, H. *A Comparison of Update Strategies for Large-Scale Maximum Expected BLEU Training.* In Proceedings of NAACL-HLT. 2015*b*.

Wäschle, K., Simianer, P., Bertoldi, N., Riezler, S., and Federico, M. *Generative and Discriminative Methods for Online Adaptation in SMT.* In Proceedings of MT Summit XIV. Nice, France, 2013.

Xia, F., Liu, T.-Y., Wang, J., Zhang, W., and Li, H. *Listwise Approach to Learning to Rank: Theory and Algorithm.* In Proceedings of ICML. 2008.

Xiao, X., Liu, Y., Liu, Q., and Lin, S. *Fast Generation of Translation Forest for Large-Scale SMT Discriminative Training.* In Proceedings of EMNLP. 2011.

Xu, J. and Li, H. *AdaRank: A Boosting Algorithm for Information Retrieval.* In Proceedings of SIGIR. 2007.

Yan, R., Gao, M., Pavlick, E., and Callison-Burch, C. *Are Two Heads Better Than One? Crowdsourced Translation via a Two-Step Collaboration of Non-Professional Translators and Editors.* In Proceedings of ACL. 2014.

Yoo, A. B., Jette, M. A., and Grondona, M. *Slurm: Simple Linux Utility for Resource Management.* In Workshop on Job Scheduling Strategies for Parallel Processing. 2003.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. *How Transferable Are Features in Deep Neural Networks?* In Proceedings of NIPS. 2014.

Yu, H., Huang, L., Mi, H., and Zhao, K. *Max-Violation Perceptron and Forced Decoding for Scalable MT Training.* In Proceedings of EMNLP. 2013.

Yuan, M. and Lin, Y. *Model Selection and Estimation in Regression with Grouped Variables.* In Journal of the Royal Statistical Society: Series B (Statistical Methodology), 2006.

Yue, Y., Finley, T., Radlinski, F., and Joachims, T. *A Support Vector Method for Optimizing Average Precision.* In Proceedings of SIGIR. 2007.

Zaretskaya, A., Vela, M., Corpas Pastor, G., and Seghiri, M. *Measuring Post-Editing Time and Effort for Different Types of Machine Translation Errors.* In New Voices in Translation Studies, 2016.

Zeiler, M. D. *ADADELTA: An Adaptive Learning Rate Method.* In CoRR, 2012.

Zhang, D., Sun, L., and Li, W. *A Structured Prediction Approach for Statistical Machine Translation.* In Proceedings of IJCNLP. 2008.

Zhang, H., Huang, L., Zhao, K., and McDonald, R. *Online Learning for Inexact Hypergraph Search.* In Proceedings of EMNLP. 2013.

Zhang, M., Liu, Y., Luan, H., and Sun, M. *Listwise Ranking Functions for Statistical Machine Translation.* In IEEE/ACM Transactions Audio, Speech and Language Processing, 2016.

Zhao, K. and Huang, L. *Minibatch and Parallelization for Online Large Margin Structured Learning.* In Proceedings of NAACL-HLT. 2013.

Zhao, K., Huang, L., Mi, H., and Ittycheriah, A. *Hierarchical MT Training Using Max-Violation Perceptron.* In Proceedings of ACL (Short Papers). 2014.

Zhao, P., Rocha, G., and Yu, B. *The Composite Absolute Penalties Family for Grouped and Hierarchical Variable Selection.* In The Annals of Statistics, 2009.

Zhechev, V. *Machine Translation Infrastructure and Post-Editing Performance at Autodesk.* In Proceedings of the Workshop on Post-editing Technology and Practice. 2012.

Zinkevich, M., Weimer, M., Li, L., and Smola, A. J. *Parallelized Stochastic Gradient Descent.* In Proceedings of NIPS. 2010.

Zollmann, A. and Sima'an, K. *A Consistent and Efficient Estimator for Data-Oriented Parsing.* In Journal of Automata, Languages and Combinatorics, 2005.

Zollmann, A. and Venugopal, A. *Syntax Augmented Machine Translation via Chart Parsing.* In Proceedings of WMT. 2006.