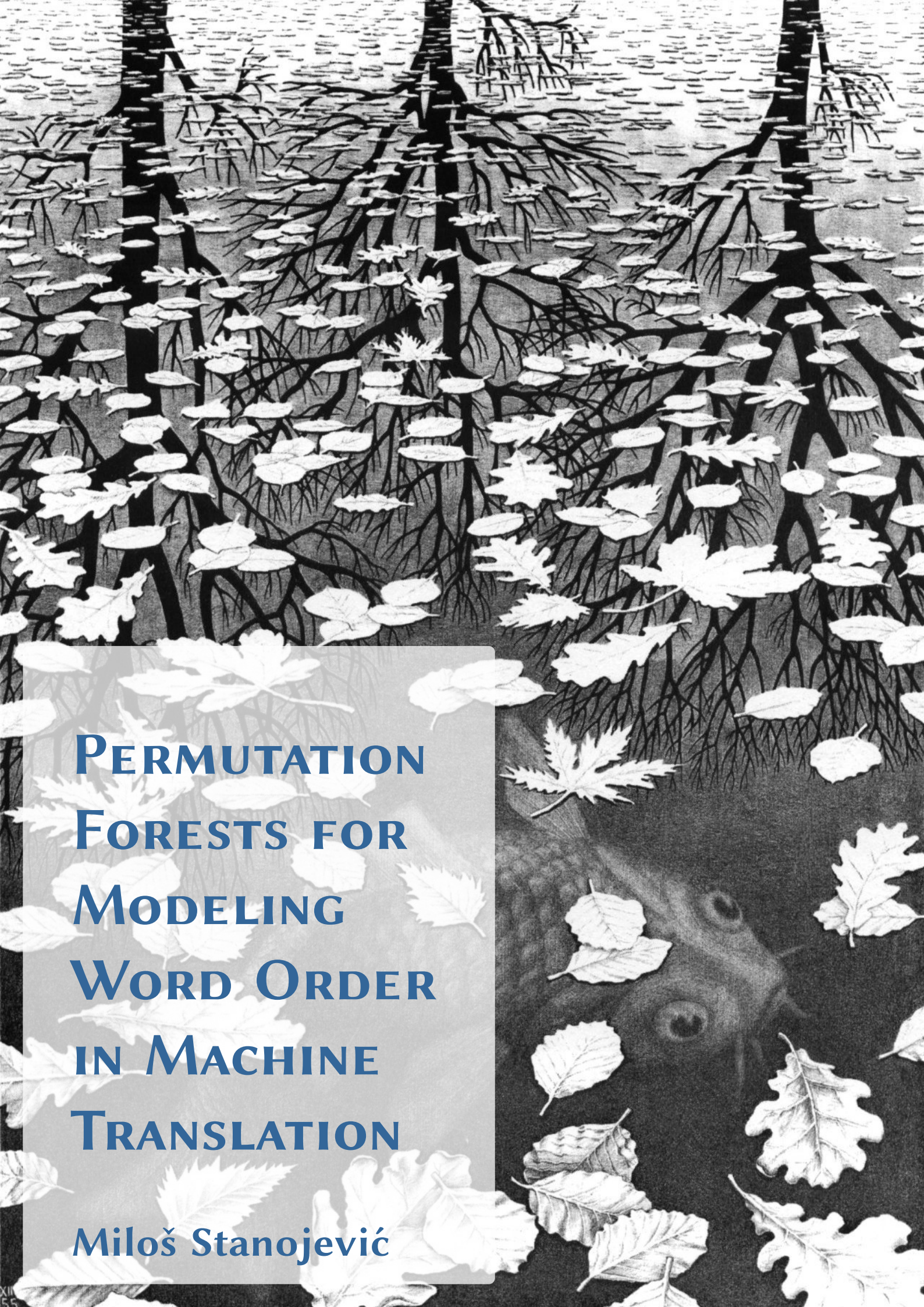


Permutation Forests for Modeling Word Order in Machine Translation

Miloš Stanojević

Permutation Forests for Modeling Word Order in Machine Translation

ILLC Dissertation Series DS-2017-09

INSTITUTE FOR LOGIC, LANGUAGE AND COMPUTATION

For further information about ILLC-publications, please contact

Institute for Logic, Language and Computation
Universiteit van Amsterdam
Science Park 107
1098 XG Amsterdam
phone: +31-20-525 6051
e-mail: illc@uva.nl
homepage: http://www.illc.uva.nl/

The investigations were supported by the Stichting voor de Technische Wetenschappen (STW) grant nr. 12271.

Permutation Forests for Modeling Word Order in Machine Translation

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het College voor Promoties ingestelde
commissie, in het openbaar te verdedigen in de Aula der Universiteit
op woensdag 13 december 2017, te 13.00 uur

door

Miloš Stanojević

geboren te Vranje, Servië

Promotiecommisie

Promotor:	Prof. Dr. K. Sima'an	Universiteit van Amsterdam
Co-promotor:	Dr. W. Ferreira Aziz	Universiteit van Amsterdam
Overige leden:	Prof. Dr. L.W.M. Bod	Universiteit van Amsterdam
	Dr. C. Monz	Universiteit van Amsterdam
	Dr. W.H. Zuidema	Universiteit van Amsterdam
	Prof. Dr. D. Gildea	University of Rochester
	Prof. Dr. W. Byrne	University of Cambridge

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

za Nanku

Contents

Acknowledgments

I am grateful to my promotor Khalil Sima'an for all the help and supervision during my PhD studies. His view that machine translation should be modeled with hierarchical structures is something that I shared with him. Because of that we agreed on many things, but when we disagreed it was even more interesting because then we had conversations in which I always learn something new. As my supervisor he gave me great freedom to develop my knowledge and interests beyond machine translation.

Khalil's MT group was filled with many people to whom one can turn for advice. You need help with probabilistic models? Sample Wilker. Something from linguistics is not clear? Don't worry, Sophie got you covered. Need to know about the latest results in the field? Jo knows it. The servers are crashing? That must be Hoang managing to meet yet another deadline. Need some help with coding things quickly? Amir and Bushra already finished it.

This is just the beginning of a long list of people who have surrounded me in ILLC, and from whom I learnt immensly. The opportunities to talk with Joost, Gideon, Jelle, Ivan, Tejaswini, Raquel, Rens, Dieuwke, Miguel, Andreas, Diego and Serhii were very rewarding and have exposed me to different areas and approaches to language processing. I would also like to thank Jenny and Tanja for always being helpful and for being tolerant of me missing many administration deadlines.

Raquel, Phong, Jo and Ke are my fellow chickens to whom I am grateful for all the fun, boredom, happiness, sadness, rainy days and (rare) non-rainy days that we spent together during these years. The days in Amsterdam with you were the high old time of my studies. I already miss going to the office every morning and seeing Raquel and Phong. Okay, that is not completely true because I rarely arrived to the office in the morning. And I don't miss Phong, since he is (luckily!) with me in Edinburgh —so we can keep alive our tradition of enjoying pubs and movies every weekend. But I do miss Raquel: her cheerfulness always brightened up the gloomy Amsterdam weather. To counterbalance Raquel's joy there is my friend Jo who with his serious face puts us back into normal. But don't let his serious face fool you: he is not serious at all. At any moment he can crack a good joke and make all chickens laugh. And how can I

not mention Ke who with his a unique philosophy of life always made us forget about problems and chill out.

Finally, I want to acknowledge my family in Serbia. Aco i Bibi, hvala vam što ste me uveli u svet tehnologije i što ste me uvek ohrabrivali da postignem više i studiram u inostranstvu. Vi ste verovali u moj uspeh više nego što sam ja sam verovao. Zahvalan sam i mojim roditeljima Goci i Boži i mom bratu Dušanu. Na vas sam uvek mogao da se oslonim za sve.

Amsterdam
November, 2017.

Miloš Stanojević

Chapter 1

Introduction

Machine Translation (MT) is the task of programming machines to automatically translate sentences from one natural language to another. It is by definition an engineering task, but just like many other engineering tasks it deals with a natural phenomenon, namely natural language. In this kind of engineering tasks one must consider both the insights that are coming from the scientific discipline that studies that natural phenomenon—in this case linguistics—and the constraints that come from the resources that are available at hand.

Take, for instance, an engineer who wants to build the wings of an airplane. She needs to take into consideration the laws of aerodynamics in order to decide of which material and shape should the wings be made, but she should also consider the constraints of the available time, tools and materials. Still, even with the very advanced knowledge of physics at hand, some aspects of the wing design are difficult to solve, and for this reason machine learning and optimization techniques are used to automatically search for better solutions than the ones that could be designed manually by an engineer (Martins et al., 2004; Sobieszczanski-Sobieski and Haftka, 1997).

The designer of MT systems faces similar challenges. The system should be designed with consideration to what the science of language identifies as the properties of language, but it should also satisfy the practical constraints imposed by computing power, available training data, required response time, etc. Often these requirements are in conflict: if one would implement the most advanced linguistic theory he would get a very slow system, and vice versa. The main challenge in designing MT systems is finding the right balance between all these requirements. And just like in aircraft design, the current state of knowledge that linguistics provides is not enough to design a good MT system, so for that reason machine learning techniques are used to fill in the gaps.

In this dissertation, I focus in modeling the reordering of words for machine translation. I have strived to develop reordering models that satisfy at least three properties that I consider crucial: (i) recursive compositionality, (ii) hierarchical structure and (iii) directly derived from data without any linguistic annotation. The first two properties

are motivated by the basic principles of how language works while the last one is motivated by practical constraints encountered in the data available for training statistical MT systems.

1.1 Why a Compositional and Hierarchical approach?

Perhaps the most relevant property of language is that, by combining its basic pieces (words and morphemes), it allows humans to express and unbounded number of ideas. As Chomsky (1965) puts it, "[...] an essential property of language is that it provides the means for expressing indefinitely many thoughts and for reacting appropriately in an indefinite range of new situations", or, more concisely expressed by von Humboldt (1836) "[Language] makes infinite use of finite means". This productive aspect of language use can be modeled with recursive functions that are a finite description of infinite sets of strings.

A recursive function is a function that can take its own output as input. For instance, a noun phrase (NP) can inside itself contain another noun phrase, which in turn can contain one more noun phrase and so on. Having a recursive description of this process allows us to build models of language which are finite (and therefore tractable), but also more compact and easier to learn from data.

The recursive functions used in modeling language are often represented in the form of a generative grammar containing phrase-structure rules. This grammar can be interpreted as a program that takes as input a sentence and outputs a yes/no answer about whether the sentence belongs to the language described by the grammar. The trace of the execution of that program is a *hierarchical* structure that essentially shows how words are combined together to form bigger structures (Chomsky, 1957).

The hierarchical structure shows how the problem of processing the sentence can be simplified by splitting the problem into sub-problems (constituents) that can be processed independently and then the results of solutions to the sub-problems are combined to solve the whole problem. These independence assumptions are helpful both for learning and for computational efficiency.

1.2 Why derived from data without linguistic annotation?

Since word order is primarily governed by syntax, a tempting choice would be to use syntactic trees as a structure for the reordering model. There are many linguistic theories that use syntactic trees as a basis for explaining the word order differences among languages, so it appears that using the syntactic trees should be a straightforward approach. This can be done in a so called syntactic-transfer based approach (Vauquois, 1968) in which the reordering process is divided in two phases. In the first phase a syntactic tree of the source sentence is obtained, and in the second phase the nodes of

the source tree are permuted so that the resulting tree would look just like the parse tree of the target sentence. Even though this might seem like a clear and well defined process, it suffers from many problems.

First, there exists a practical problem regarding the availability of syntactic parsers. Even though high quality parsers exist for English, for majority of other languages there are either no parsers at all or the parsers available are of considerably lower quality than the English parsers. English parsers are considered of high quality because they produce correct output on the Wall Street Journal texts, but it should not be neglected that very often MT systems get as input texts that are from completely different domains in which these parsers would not be as accurate.

The second problem is that we do not know *the right* structural analysis of natural language sentences. Many syntactic theories give structural descriptions of sentences that explain a great deal of language phenomena, but all of them are far from perfect: they either give descriptions that are too poor to be useful or too rich to be parsed efficiently. For instance, let us consider the two most prominent types of syntactic analysis used in natural language processing (NLP), namely projective constituency trees and projective dependency trees. In these two structures there is no information about agreement, wh-movement, control, nor about any language phenomena that go beyond context-free power. Thus, even though modeling these properties of language has been shown to be useful for MT systems (Chung and Gildea, 2010; Xiang et al., 2013), this information is often missing in the annotation.

The third problem is that the two-step approach outlined above involves mapping trees-to-trees, but often the trees that are used in different languages do not map to each other, as argued for dependency trees by Eisner (2003) and for constituency trees by Khalilov and Sima'an (2012). Let us take as an example a sentence which is a wh-question about its object. If the source language has wh-movement (e.g. English) the object will be extracted from the verb phrase. In case the target language does not have wh-movement (e.g. Japanese) the object will stay in the verb phrase. This causes that the direct mapping between the verb phrases of source and target language impossible [1]. A similar argument can be made for many other language phenomena, such as pro-drop.

One could imagine that some richer syntactic description that abstract away from these surface transformations could solve these problems. Minimalist Grammar derivation trees (Stabler, 2013) of the example with wh-question above would not have the mentioned problem because it allows discontinuous constituents. Empty (null) categories would also not present a problem because empty strings can also be represented in the Minimalist Grammar structure.

However, even these structures would not be enough to guarantee direct mapping between the source and target representations of parallel sentence pairs. Even with ideal syntactic representations, the data will contain sentences where translation is not done *word by word*, but instead more idiomatic or non-literal translation choices are

[1] Unless some non-local transformation is evoked like in Khalilov and Sima'an (2012)

made.

So the question then becomes: if none of the currently popular syntactic descriptions are able to capture all the translation reordering phenomena what should we use? The answer proposed in this thesis is to use hierarchical structures directly derived from translation data. Instead of presupposing some syntactic description and try to fit translation data to it, I propose to look at the problem in the opposite direction: taking translation data and from it deriving the hierarchical description that fits it.

1.3 How to derive such structures?

In order to derive these structures, we take a step back and observe what is available in the data that is used for training a typical statistical machine translation system. Usually we have only a large amount of parallel corpora (source sentences paired with their target translation) at our disposal. From these sentence pairs we can extract mappings of source words to target words using unsupervised alignment algorithms (Brown et al., 1993). This representation is illustrated with an example sentence pair in Figure 1.1a.

By adding some simplifications and some assumptions to this data, a richer compositional view of word order can be derived. Since the primary goal in this thesis is modeling of word order, we can simplify this representation by removing the target words and replacing them with their position indices as shown in Figure 1.1b. The next simplification step is to reduce many-to-many alignments to one-to-one alignments. As visible in Figure 1.1c this is a lossy reduction: we do not have information any more that the last three words of the English sentence translate to a single target word. However, this information is not crucial for the prediction of word order but only for the prediction of lexical translation, so this loss of information is acceptable. These simplifications allow us to treat the reordering problem as a problem of predicting a permutation of source sentence word positions into a target language word order. The permutation in the example sentence is given above the words of Figure 1.1c.

It is known that permutations can be decomposed into a hierarchical projective tree structure (Albert and Atkinson, 2005; Zhang and Gildea, 2007) called Permutation Trees (PETs). Figure 1.1d shows one of the possible permutation trees for the given permutation. In this tree a constituent corresponds in most cases to what would in phrase based systems be a phrase pair. Just like a phrase pair can inside itself contain another smaller phrase pair, constituents of this structure contain smaller sub-constituents. The labels on the constituents are called prime permutations (as an analogy to prime numbers that cannot be decomposed further) and signify how are sub-constituents (children) of the constituent reordered to get the target word order. A label $< 1, 2 >$ signifies that the first child should go to the first place and the second child to the second place and therefore implies no change in word order. The prime permutation $< 2, 1 >$ has the opposite effect: the order of children is inverted. PETs can model complex permutations that require more than binary branching nodes. These trees can be obtained by using efficient algorithms (Gildea et al., 2006; Zhang and Gildea, 2007;

Zhang et al., 2008).

The PET in Figure 1.1d is similar to a syntactic tree, but it is also different in some respects. For example, some types of constituency analysis would attach "the day after tomorrow" as a modifier to the verb phrase "wait for you". As we will see later this is just one of the many possible PETs, and some of the other PETs, in this example, are consistent with the mentioned syntactic analysis.

This tree representation is constructed on the source side, but it is to some extent bilingual: it contains information about compositionality of words in the source language and about its relation to the word order in the target language. The tree is built on the source side, but interestingly if it was built on the target side it would be very similar to a mirror image of the source side PET. The only thing that prevents a full mirror image is different number of words and alignments that are not one-to-one, but the reordering patterns will nevertheless to a large extent stay the same.

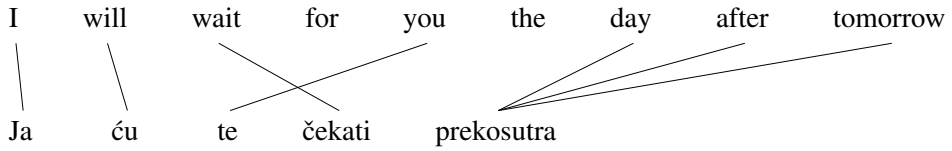
Zhang and Gildea (2007) describe efficient methods for extracting the canonical permutation tree from the given input. However, the canonical PET is not the only possible PET that describes the translation data. In principle, in the worst case the number of PETs per sentence can be exponentially big in sentence length. I call a set of these trees a *permutation forest* (PEF). Knowing which tree to use is not directly clear; in a sense, the most useful permutation tree is hidden in the forest.

1.4 Properties of PET-based models

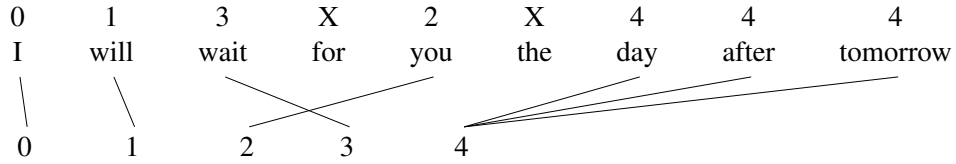
The approach of deriving trees directly from the data has several attractive properties. First, it is purely data driven and requires nothing more than a parallel corpus which makes this method applicable to a large number of languages. Second, it is able to fit any translation data—since the trees are directly derived from the data— and it directly explains surface translation compositionality present in the data.

This is not the first time that hierarchical structures derived from translation data are used for modeling translation phenomena. There are examples of inversion transduction grammar (ITG) (Wu, 1997), normalized decomposition trees (NDT) (Zhang et al., 2008) and hierarchical alignment trees (HAT) (Sima'an and Maillette de Buy Wenniger, 2011) all of which were inspirational for this work. What makes this work different from the mentioned ones is that a permutation tree (PET) structure is used instead that allows the model to handle non-binarizable reordering (compared to ITG), to have purely unlexicalized labels/non-terminals (compared to NDT) and to have simpler labels that allow easier grammar induction (compared to HAT).

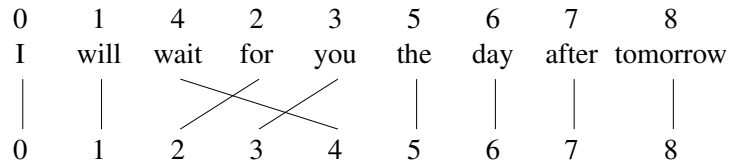
The usage of representations that are directly derived from data is by no means a silver bullet that solves all problems. All the models are based on some assumptions. The performance of these models depends on how justified these assumptions are. One of the important assumptions of the models used in this thesis is that the structure of the sentence is always a projective tree. This is clearly not the case for many of the world's languages, but is an assumption that is acceptable in comparison to other syn-



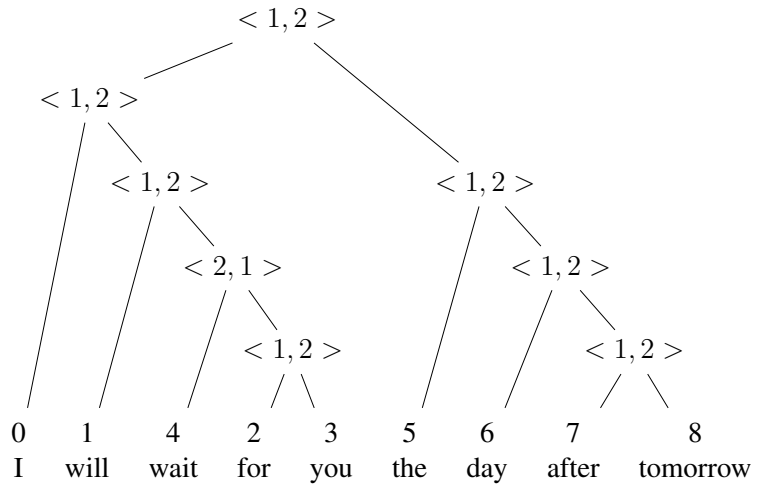
(a) Joint lexical and reordering view



(b) Alignment view



(c) Permutation view



(d) Hierarchical (ITG/PETs) view

Figure 1.1: Different views of word order mapping from English to Serbo-Croatian example sentence pair

tactic models used in NLP applications that are almost exclusively projective. Another potential problem of this approach is that it would in most circumstances require usage of unsupervised learning which is a much more difficult task than supervised learning used for standard syntactic models. Even if the structures derived directly from data would be in principle better, it might happen that for some languages they are difficult to learn and that syntactic parsers would give more reliable results.

In this thesis the whole forest of PETs is used for modeling the word order: both for prediction of word order (reordering) and for evaluation of word order. The modeling of word order is the task in which permutation trees are particularly well suited. As their name suggests, they are meant for giving a tree structure over permutations. These permutations can in principle come from any source. In the case of machine translation reordering, a permutation is a description about how words of the source sentence should be permuted so that they could be in the target language word order. The tree structure is especially important here because it shows the compositional structure that governs this reordering process. With permutation trees, any permutation can be decomposed into many smaller permutations that are combined into bigger permutations using a projective tree. By using this representation much of the existing knowledge about efficient learning and parsing algorithms can be reused for better machine translation reordering. In the coming chapters different machine learning models are proposed that are inducing the distributions over this latent representation of word order from parallel data alone in an attempt to improve machine translation reordering and evaluation.

1.5 Objective

The core objective of this work is to model word order in statistical machine translation using structures that are: (i) compositional, (ii) hierarchical, (iii) automatically induced from data. By using this type of structure, the expectation is that predicted word order would be improved, and by that the overall quality of machine translation output.

Several questions related to the modeling of word order will be addressed. First, is compositional treatment of word order more effective than *flat* treatment of it? Second, is the move from binary branching ITG to more powerful PET structures justified in terms of better prediction of word order? Third, should the model consider all possible derivations (PETs) of the observed word order or is picking a single arbitrary derivation that is consistent with the word order enough?

In addition to the improvement of translation quality, better modeling of word order should also improve evaluation of word order. So another goal pursued in this dissertation is the construction of an evaluation metric that is based on the same hierarchical structures as the preordering model. The expectation is that such a metric that evaluates word order in a compositional way would show higher correlation with human judgment of translation quality.

This metrics construction objective consists of two subgoals. The first subgoal is

the construction of a metric that evaluates word order only: this metric will in big part ignore lexical accuracy. The second subgoal is to incorporate this metric into a fullblown metric that evaluates not only word order but all other aspects of translation quality.

1.6 Contribution

The two main contributions of this dissertation are a preordering model and an evaluation metric based on permutation trees. Both of these will be presented briefly in this section.

The preordering model based on permutation trees is an unsupervised learning model that treats permutation trees as a latent variable. As its input the model takes parallel corpora and word alignments. From this input, during the training time, it constructs a chart of all possible permutation trees that are consistent with a given alignment i.e. all the PETs that predict the same reordering of the source sentence.

The goal of constructing the chart of possible PETs is to reduce the space of trees which should be considered as explanation of the reordering phenomena. Still, even this reduced space can have exponentially many trees and it is not directly observable which one of these trees explains the data in the best way. In order to discover that, I treat permutation trees as a latent variable and estimate the grammar parameters over all these trees using Inside-Outside algorithm (Lari and Young, 1990). Compared to previous work in latent variable PCFG induction (Matsuzaki et al., 2005; Prescher, 2005; Petrov et al., 2006; Saluja et al., 2014), here both the bracketing and the nonterminal label splits are treated as hidden variables. This training process produces a probabilistic grammar that can be used to parse new sentences and with that predict new permutation trees.

Given the predicted permutation tree getting the reordering is trivial with just a simple top-down traversal. However, we are often not interested in the permutation tree. We are interested in reordering and a permutation tree is just a vehicle that takes us there. Ideally, we would marginalize over all trees that are generating the same permutation and, instead of finding the most probable tree, use the most probable permutation. This solution is approximated by sampling from the resulting chart because the problem is known to be intractable (Sima'an, 1996). In addition to that, minimum-risk decoding is employed to optimize decision towards a reordering driven objective.

The experiments with this preordering model, other than improving translation accuracy, also give insights into some of the interesting questions about modeling word order. It is shown that using only one canonical PET instead of a full forest gives much worse results in predicting word order. Even in case only a single PET is used, depending on which PET is chosen (for instance left-branching or right-branching) the results in translation quality can be significantly different. It is also shown that the more powerful PETs are better for prediction of word order than binary branching ITG.

The contributions to the evaluation metrics are twofold. First, new evaluation met-

rics over PETs are introduced that improve the quality of evaluation of word order when compared to the non-hierarchical metrics. In these experiments some similar conclusions arise as in the preordering case: using PETs is beneficial and using all PETs instead of one single PET gives better results.

The second major contribution is an extension of this reordering metric to a full MT metric. In order to have a high accuracy evaluation metric it is necessary to train many different components of the metric for approximation of human judgment. New algorithms are proposed for training MT metrics both for sentence and corpus level judgment. Additionally, special training methods are proposed for training a robust metric that would be more difficult to *game*—an important property that would make tuning for this metric not diverge towards some very bad translations that are preferred by the metric.

The primary publications on which the chapters are based are the following:

- *Fitting Sentence Level Translation Evaluation with Many Dense Features* .
  In EMNLP (Stanojević and Sima'an, 2014a)

- *Evaluating Long Range Reordering with Permutation-Forests* .
  In SSST (Stanojević and Sima'an, 2014b)

- *Reordering Grammar Induction* .
  In EMNLP (Stanojević and Sima'an, 2015a)

- *Hierarchical Permutation Complexity for Word Order Evaluation* .
  In COLING (Stanojević and Sima'an, 2016)

- *Alternative Objective Functions for Training MT Evaluation Metrics* .
  In ACL (Stanojević and Sima'an, 2017)

The secondary publications on which only some of the sections are based are the following publications:

- *BEER: BEtter Evaluation as Ranking* .
  In WMT (Stanojević and Sima'an, 2014c)

- *BEER 1.1: ILLC UvA submission to metrics and tuning task* .
  In WMT (Stanojević and Sima'an, 2015c)

- *Evaluating MT systems with BEER* .
  In PBML (Stanojević and Sima'an, 2015b)

- *Examining the Relationship between Preordering and Word Order Freedom in Machine Translation* .
  In WMT (Daiber et al., 2016)

All the research and all implementations described in this dissertation were carried out by me, Miloš Stanojević. Khalil Sima'an guided me in the process of conducting research and has edited the papers which we co-authored. Wilker Aziz and Joachim Daiber also edited the papers in which they were co-authors. From co-authored papers I have included only the content that was directly my contribution.

1.7 Overview

This section contains the overview of the thesis chapters.

Chapter 2 presents a short introduction to statistical machine translation and permutation trees. It covers word alignment models, phrase based models, hierarchical translation models, discriminative models, evaluation of machine translation and finally treatment of word order mapping as a permutation modeling problem.

Chapter 3 presents a model for predicting target side word order by using PETs, published at EMNLP (Stanojević and Sima'an, 2015a). It starts with the presentation of the latent variable model for automatic induction of probabilistic reordering grammar from raw parallel data and word alignments. After that I present how that model can be used for predicting the least risky permutation tree in terms of reordering loss and experiments that show that this model improves translation quality. Additionally, an extension of this work presented at WMT (Daiber et al., 2016) is shown which propagates uncertainty of the preordering model to decoder by giving the decoder a lattice of best reorderings instead of a 1-best reordering.

Chapter 4 presents a model for evaluation of word order by using PETs, published at COLING (Stanojević and Sima'an, 2016) COLING and SSST (Stanojević and Sima'an, 2014b). It first presents a model which evaluates word order by simple counting of monotone (correct) nodes in the permutation tree. This non-recursive model over a recursive structure gives improved correlation with human judgment of translation quality compared to the standard baselines which do not take hierarchical structure of permutations into account. Following that section, a recursive metric over the recursive structure is proposed which also exhibits high correlation with human judgment.

Chapter 5 covers an extension of the word order only metric from Chapter 4 to a full evaluation metric that evaluates also lexical accuracy. An initial version of this metric named BEER was published at EMNLP (Stanojević and Sima'an, 2014a) and later modifications are distributed across several publications (Stanojević and Sima'an, 2014c, 2015c,b). BEER is an evaluation metric that is directly trained to approximate rankings of translations assigned by humans. The training algorithm for sentence level judgments is presented in the chapter together with the features that are used as indicators of translation quality. The most important of all these features are based on character n-grams which make BEER the most accurate evaluation metric on majority of languages on WMT shared metrics task (Macháček and Bojar, 2014; Stanojević et al., 2015b; Bojar et al., 2016) at the moment of writing.

Chapter 6 covers an extension of BEER from sentence level evaluation metric to a corpus level metric. A part of this chapter has been published at ACL (Stanojević and Sima'an, 2017) and part is novel and yet unpublished. Two methods for extending BEER to corpus level are proposed. First one is learning to scale the output of the sentence level BEER which gives a more accurate corpus level score when sentence level scores are averaged. Second method is to directly train for the good ranking of averaged corpus level scores. The chapter covers one more important topic of training MT metric to which not enough attention has been given by the community. Namely, the chapter contains description of methods for training evaluation metric to be more robust so that the tuning algorithm cannot *game* them. BEER avoids *gaming* by using self-training and symmetric scoring. It will be shown that these are important properties when it comes to usage of the metric as an objective function in training of SMT systems.

Chapter 7 presents the concluding remarks about the thesis and look into the possible extensions of this work.

Chapter 2

Background: Statistical Machine Translation

Machine Translation (MT) deals with automatically finding the translation of some *source language* sentence into its *translation equivalent* in the *target language*. Defining what translation equivalence exactly means is not a trivial task. Usually, what we want to get is some sort of equivalence in the meaning that the sentences express. However, the whole process of deriving the semantically equivalent sentences is not directly observable in the data. What we can see in the parallel data are pairs of sentences: one sentence in the source language and its translation equivalent in the target language, but the reason why one sentence is equivalent to the other is not directly visible. For instance, we do not know which words or phrases are equivalent (translations of each other). We also cannot directly observe in which order were these words or phrases combined nor which syntactic or semantic processes lead to that combination.

This is a problem of uncertainty due to missing information and ambiguity. Most machine translation systems address it with statistical models which put a probability distribution over the possible sentences in the target language that could be the correct translation. Formally speaking, given the source sentence s we want to find the translation t, in the set of all possible sentences in the target language T, with the highest probability under the model with parameters θ.

$$\hat{t} = \underset{t \in T}{\operatorname{argmax}}\ p_\theta(t|s) \tag{2.1}$$

The main part of defining a machine translation model consists of defining the set of target sentences T, the symbolic methods that will perform the argmax search and the statistical methods for the estimation of the model $p_\theta(t|s)$ (I will omit θ whenever convenient to simplify the notation). In the Section 2.1 the generative formulation of the probabilistic model $p_\theta(t|s)$ is presented, followed by Section 2.3 where the extension to discriminative model is shown.

This thesis tackles two important aspects of building MT system: MT evaluation and reordering. The background on MT evaluation is covered in the Section 2.4. Reordering model presented in the thesis relies on the view of word order mapping as a

permutation. That view and Permutation Trees as the data structure are presented in Section 2.5.

2.1 Generative Models

Generative models use Bayes' rule to decompose the probability model $p(t|s)$ to two components $p(s|t)$ and $p(t)$:

$$\hat{t} = \underset{t \in T}{\operatorname{argmax}}\ p(t|s) = \underset{t \in T}{\operatorname{argmax}}\ \frac{p(s|t)p(t)}{p(s)} = \underset{t \in T}{\operatorname{argmax}}\ p(s|t)p(t) \tag{2.2}$$

By doing this, the problem of modeling the translation probability is made easier by splitting the work between two models: the *language model* $p(t)$ will ensure that the output is a fluent sentence in the target language while the *translation model* $p(s|t)$ will ensure the adequacy of the lexical translations. Strictly speaking, the translation model $p(s|t)$ does not have to deal only with the lexical adequacy but it is an assumption that is often used because it simplifies modeling.

2.1.1 Language Model

Language models assign a probability to the sequence of words in the target language. The probability that is assigned to a sequence of words is computed sequentially from left to right by computing the probability of the current word given the history (the preceding words in the sentence):

$$p(w_1, w_2, \ldots, w_m) = p(w_1)p(w_2|w_1)\ldots p(w_m|w_1, w_2, \ldots, w_{m-1}) \tag{2.3}$$

Having history of arbitrary length is both computationally not feasible and practically not useful, because it leads to problems with sparsity. For that reason the language models that are in use in SMT are n-gram language models in which the size of the history is limited to n preceding words. These models use a Markov assumption: the probability of a word depends mostly on its immediately preceding words.

$$p(w_m|w_1, w_2, \ldots, w_{m-1}) \approx p(w_m|w_{m-n+1}, \ldots, w_{m-2}, w_{m-1}) \tag{2.4}$$

A naive method of estimating the probability for n-gram models would be based on Maximum Likelihood Estimation:

$$p(w_n|w_1, \ldots, w_{n-1}) = \frac{count(w_1, \ldots, w_{n-1}, w_n)}{\sum_w count(w_1, \ldots, w_{n-1}, w)} \tag{2.5}$$

The problem with this method for estimating the probability of a word is that some possible histories will never be observed, especially if the n-grams are of high order. This would cause the majority of n-grams, and therefore also sentences, to have probability zero. To reduce this effect, different *smoothing* techniques have been used.

Smoothing takes some probability mass from seen n-grams and distributes it over unseen n-grams. The simplest smoothing technique is "add one smoothing" in which the counts of all n-grams (both seen and unseen) get incremented by one:

$$p(w_n|w_1,\ldots,w_{n-1}) = \frac{count(w_1,\ldots,w_{n-1},w_n)+1}{\sum_w count(w_1,\ldots,w_{n-1},w)+1} \tag{2.6}$$

Adding exactly one is an arbitrary choice and often much better results can be achieved by adding some small number α that is tuned on a held out set:

$$p(w_n|w_1,\ldots,w_{n-1}) = \frac{count(w_1,\ldots,w_{n-1},w_n)+\alpha}{\sum_w count(w_1,\ldots,w_{n-1},w)+\alpha} \tag{2.7}$$

Another way to fight data sparsity of high order n-grams is to interpolate high order n-grams with the scores of lower order n-grams. For instance, for a three-gram language model the score would be computed in the following way:

$$\begin{aligned} p_I(w_3|w_1,w_2) = \; & \lambda_1 p_1(w_3) \\ & +\lambda_2 p_2(w_3|w_2) \\ & +\lambda_3 p_3(w_3|w_1,w_2) \end{aligned} \tag{2.8}$$

All the λ weights should sum to 1 and be between 0 and 1 in order to ensure that p_I is a proper probability distribution:

$$\begin{gathered} \forall \lambda_i : 0 \leq \lambda_i \leq 1 \\ \textstyle\sum_i \lambda_i = 1 \end{gathered} \tag{2.9}$$

The λ weights can be estimated via EM procedure to maximize the likelihood of the data under p_I model (Jelinek and Mercer, 1980).

The quality of language models is measured by how high probability they give to a test corpus. This is presented in the form of perplexity, which is a function of the likelihood of the data. Because longer corpora will have lower probability, the perplexity is computed per word.

$$PP(w_1,\ldots,w_n) = P(w_1,\ldots,w_n)^{-\frac{1}{n}} \tag{2.10}$$

Perplexity can be interpreted as the weighted average branching factor of a language. The branching factor refers to the number of possible words that can follow any word (Jurafsky and Martin, 2008).

2.1.2 Lexical Translation Model

In this section, a type of lexical translation models called IBM models (Brown et al., 1993) is presented. These models use individual words as their main units of translation. A more structured translation models, that are not necessarily generative, are

presented in Section 2.2 which consider bigger units of translation such as phrases or constituents in a tree.

As mentioned before, if we had the exact annotation in the data how the target translation was generated, then training the probabilistic model would be straightforward. However, the only data we have are sentence aligned parallel corpora. The word alignment (the mapping that shows which word on the source side *generated* which word on the target side) is a latent (or hidden) variable. The usual process of handling the latent variables is by first defining a *generative story* with the latent variables and then train its parameters using the expectation-maximization (EM) algorithm (Dempster et al., 1977).

Here the IBM1 generative model (Brown et al., 1993) is presented which treats word alignments as latent variables. The translation probability of a source sentence $\boldsymbol{s} = (s_1, \ldots, s_J)$ to a target sentence $\boldsymbol{t} = (t_1, \ldots, t_I)$ with an alignment of each target word t_j to a source word s_i according to the alignment function $a : j \rightarrow i$ is defined as:

$$p(\boldsymbol{s}, a|\boldsymbol{t}) = \frac{\epsilon}{(I+1)^J} \prod_{j=1}^{J} p(s_j|t_{a(j)}) \tag{2.11}$$

where $p(s|t)$ is the probability of word s being a translation of the word t and ϵ is the normalization constant so that $p(\boldsymbol{s}, a|\boldsymbol{t})$ is a proper distribution.

This is a very simplistic model that introduces many independence assumptions. The biggest assumption is that the translation of each word is independent of the translations of the other words in the sentence. The order of words also has no influence on the alignment. The model can even be seen as the alignment of two "bags of words". Modeling the sentence length is also simplistic, but that does not have a big influence if the model is used only for estimation of the alignments and not for decoding. In order to account for the words that have no counterpart on the other side of parallel corpus, i.e. they are unaligned, an artificial $NULL$ word is introduced. This is the reason that the denominator in Equation 2.11 is normalized for $I + 1$ instead of I words.

The probabilistic model that puts a distribution over different alignment configurations given the sentence pair $p(a|\boldsymbol{s}, \boldsymbol{t})$ can be written in the following form by application of the chain rule:

$$p(a|\boldsymbol{s}, \boldsymbol{t}) = \frac{p(\boldsymbol{s}, a|\boldsymbol{t})}{p(\boldsymbol{s}|\boldsymbol{t})} \tag{2.12}$$

The probability of source sentence $\boldsymbol{s}$ being a translation of the target sentence $\boldsymbol{t}$ is given by the probabilistic model $p(\boldsymbol{s}|\boldsymbol{t})$ that sums over all possible alignment configurations in order to marginalize alignment as a latent variable:

$$
\begin{aligned}
p(\boldsymbol{s}|\boldsymbol{t}) =& \sum_{a} p(\boldsymbol{s}, a|\boldsymbol{t}) \\
=& \sum_{a(1)=0}^{I} \cdots \sum_{a(J)=0}^{I} p(\boldsymbol{s}, a|\boldsymbol{t}) \\
=& \frac{p(J|I)}{(I+1)^J} \sum_{a(1)=0}^{I} \cdots \sum_{a(J)=0}^{I} \prod_{j=1}^{J} p(s_j|t_{a(j)}) \\
=& \frac{p(J|I)}{(I+1)^J} \prod_{j=1}^{J} \sum_{i=0}^{I} p(s_j|t_i)
\end{aligned} \tag{2.13}
$$

By inserting Equation 2.13 in Equation 2.12 we get:

$$
\begin{aligned}
p(a|\boldsymbol{s}, \boldsymbol{t}) =& \frac{p(\boldsymbol{s}, a|\boldsymbol{t})}{p(\boldsymbol{s}|\boldsymbol{t})} \\
=& \frac{\frac{p(J|I)}{(I+1)^J} \prod_{j=1}^{J} p(s_j|t_{a(j)})}{\frac{p(J|I)}{(I+1)^J} \prod_{j=1}^{J} \sum_{i=0}^{I} p(s_j|t_i)} \\
=& \prod_{j=1}^{J} \frac{p(s_j|t_{a(j)})}{\sum_{i=0}^{I} p(s_j|t_i)}
\end{aligned} \tag{2.14}
$$

Expected counts per sentence pair $(\boldsymbol{s}, \boldsymbol{t})$ are computed by counting the alignment links over all possible alignment configurations where each alignment configuration is weighted by its probability. Even though the number of alignment configurations is exponential, the expected counts can be computed efficiently by the use of dynamic programming:

$$
c(s|t; \boldsymbol{s}, \boldsymbol{t}) = \frac{t(s|t)}{\sum_{i=0}^{I} t(s|t_i)} \sum_{j=1}^{J} \delta(s, s_j) \sum_{i=1}^{I} \delta(t, t_i) \tag{2.15}
$$

The maximization step of the EM algorithm is the same as in the case where the alignments are observed except that here instead of normalizing with the observed alignment counts we normalize with the expected alignment counts:

$$
p(s|t; \boldsymbol{s}, \boldsymbol{t}) = \frac{\sum_{(\boldsymbol{s},\boldsymbol{t})} c(s|t; \boldsymbol{s}, \boldsymbol{t})}{\sum_{s} \sum_{(\boldsymbol{s},\boldsymbol{t})} c(s|t; \boldsymbol{s}, \boldsymbol{t})} \tag{2.16}
$$

Maximum likelihood estimation for IBM1 model is guaranteed to find the globally optimal solution because the objective function is convex. With all its naïve independence assumptions IBM1 is still useful in practice as the initial step to produce a good

starting point for more sophisticated alignment algorithms such as IBM2 which additionally models the absolute alignment positions of words to account for some reordering. Compared to Equation 2.11 of IBM1, in IBM2 we add the model $p(a(j)|j, I, J)$ to account for the alignment positions:

$$p(\boldsymbol{s}, a|\boldsymbol{t}) = \frac{p(J|I)}{I+1} \prod_{j=1}^{J} p(a(j)|j, I, J) \;\; p(s_j|t_{a(j)}) \tag{2.17}$$

The estimation is again done with EM. This model can further be extended with a fertility model (IBM3) and with modeling alignment positions with relative positions instead of absolute (IBM4).

All these alignment models are aligning only in one direction: each source word can be generated by only one target side word. Since this generates only 1-to-many alignments they would by definition be suboptimal since it is easy to see that in the actual translations there are many examples of many-to-many alignments. To account for this, alignment models are trained in both directions producing alignments in 1-to-many and many-to-1 form, which are afterwords combined in the process of *symmetrization* which produces many-to-many alignments. This is a process guided by heuristics whose performance varies among datasets. The simplest (and usually the worst) heuristic is the *intersection* heuristic. It always produces 1-to-1 alignments which misses many alignment links, but it has usually very high precision because all the alignment links that stay are confirmed by both models. The opposite heuristic is the *union* heuristic, which produces a high number of alignment links with high recall. In practice the heuristics that are used fall somewhere in between high precision of intersection heuristic and high recall of the union heuristic.

The most popular technique is *Grow-Diag-Final*. Its initial point is the high precision intersection of the alignments, followed by two steps Grow-Diag and Final. The Grow-Diag step adds to the intersection alignments, the alignments that are on the diagonal neighboring position with the existing alignments. The alignments that are added are naturally not in the intersected set but only in the union set. The Final step is applied only when Grow-Diag cannot be applied any more. In the Final step all the words that still do not have any alignments get their alignment from the union set (if there are any alignments for them in the union set).

2.2 Translation Equivalents

The historical development of translation models correlated with the complexity of the structures that were used for translation. The first successful statistical machine translation models were IBM models that used almost no structure. The next big breakthrough were the models that used phrases (contiguous sequence of words) as the units in translation (Zens et al., 2002; Koehn et al., 2003). The last big milestone in symbolic approaches to MT was a Hiero model (Chiang, 2005) that uses a hierarchical structure

over word and phrases. These models are reviewed next, in chronological order. This order is not only useful for didactic purposes, but it is also the order in which machine translation models are trained in practice: it starts with training of the word based models so the word alignments can be discovered, and after that initial process is finished, the alignments are used to train phrase based or hierarchical models.

2.2.1 Phrase Based Models

The obvious next step in the development of machine translation models after the word based models was to create models that translate bigger chunks of words. The typical example of why this is useful are idioms in which the meaning (and therefore also the translation) of the phrase is not the same as the composition of its parts.

Even though the need for phrase based models was obvious, the way how they should work was not. Initial attempts of training phrase-to-phrase alignment in the EM style similar to word alignment did not give big improvements (Marcu and Wong, 2002). The first models that made a big difference were models based on phrase extraction being done directly through word alignments by the usage of heuristics (Zens et al., 2002; Koehn et al., 2003).

The procedure is simple: first, the word alignment is extracted in both directions through the whole pipeline of IBM models, followed by symmetrization after which the phrase extraction is performed. In the phrase extraction, only the *consistent* phrase pairs are extracted. A phrase pair is consistent iff none of the words in the phrase pair are aligned to any word that is outside of the phrase pair and there must be at least one aligned word pair inside the phrase pair.

We can define consistency of the phrase pair $(\bar{s}, \bar{t})$ more formally in the following way:

$$
\begin{aligned}
(\bar{s}, \bar{t}) \text{ is consistent with } A \iff & \\
& \forall s_i \in \bar{s} : (s_i, t_j) \in A \implies t_j \in \bar{t} \\
& \forall t_j \in \bar{t} : (s_i, t_j) \in A \implies s_i \in \bar{s} \\
& \exists s_i \in \bar{s}, t_j \in \bar{t} : (s_i, t_j) \in A
\end{aligned} \tag{2.18}
$$

This procedure treats phrase alignments as directly observable so no latent variable technique is needed for estimation of the probabilities. Simple count and divide strategy suffices:

$$
p(\bar{t}|\bar{s}) = \frac{count(\bar{s}, \bar{t})}{\sum_{\bar{t}_i} count(\bar{s}, \bar{t}_i)} \tag{2.19}
$$

Ideally, the probability of the source sentence given the target translation should sum over all possible combinations of the phrase pairs that could generate it. In other words, phrase segmentation should be treated as a latent variable. Because this is

computationally intractable (Sima'an, 1996), the alternative solution is to pick one possible phrase segmentation and compute the probability of the sentence pair by the product of probabilities of its phrase pairs and of the probability for that ordering of phrase pairs:

$$p(\boldsymbol{s}|\boldsymbol{t}) = \prod_{i=1}^{I} p(\bar{s}_i|\bar{t}_i) d(\bar{t}_i, \bar{t}_{i-1}) \tag{2.20}$$

The *distortion model* $d(\bar{t}_i, \bar{t}_{i-1})$ models the probability of the current phrase $\bar{t}_i$ being translated right after the previous phrase $\bar{t}_{i-1}$. This model can be a very simple exponential penalty for the distance between these two phrases: the bigger jump is made during translation, the exponentially bigger will be the penalty. This model is called *distance based reordering model*:

$$d(\bar{t}_i, \bar{t}_{i-1}) = \alpha^{|start(\bar{t}_i) - end(\bar{t}_{i-1}) + 1|} \tag{2.21}$$

In order to make $d(\cdot, \cdot)$ a proper probability distribution α is chosen to be between 0 and 1. This model prefers monotone translation which might be good for some "simple" language pairs such as English-French and English-Spanish, but it does not work well for languages for which long distance reordering is needed such as English-Japanese or English-German. Another problem with the distance based reordering model is that it has the same score independently of the actual content of the phrases involved in reordering. For that reason some more sophisticated lexicalized reordering models were introduced out of which the most used one is MSD reordering model (Tillmann, 2004; Axelrod et al., 2005).

MSD reordering model puts a probability distribution over three possible orientations of phrase $\bar{t}_i$ versus the previously translated phrase $\bar{t}_{i-1}$:

$$orientation_i = \begin{cases} Monotone & \text{if } start(\bar{t}_i) - 1 = end(\bar{t}_{i-1}) \\ Swap & \text{if } start(\bar{t}_{i-1}) - 1 = end(\bar{t}_i) \\ Discontinuous & \text{otherwise} \end{cases} \tag{2.22}$$

Defined in this way, orientations of phrases are directly observable in the parallel data for which phrase pairs are extracted. The probability of each orientation can be computed simply by count and divide strategy:

$$p(orientation|\bar{s}, \bar{t}) = \frac{count(orientation, \bar{s}, \bar{t})}{\sum_o count(o, \bar{s}, \bar{t})} \tag{2.23}$$

There are many variations of the MSD reordering model mostly with respect to the different estimation techniques. For instance, Equation 2.23 can be improved by adding some smoothing or by conditioning on the bigger context (lexical content of previous phrases).

(1) $S \rightarrow \langle\ NP_1\ VP_2,\ NP_1\ VP_2\ \rangle$
(2) $VP \rightarrow \langle\ V_1$ 'you', V_1 'tebe' $\rangle$
(3) $VP \rightarrow \langle\ V_1$ 'you', 'te' $V_1\ \rangle$
(4) $NP \rightarrow \langle$ 'I', 'Ja' $\rangle$
(5) $V \rightarrow \langle$ 'see', 'vidim' $\rangle$

Figure 2.1: Example SCFG for translating English to Serbo-Croatian

MSD reordering models have been very successful in PBMT, but there was still space for improvement by introducing more structure to the phrase orientation definition. A problem that MSD models have is that a large majority of phrase orientations falls in the category of discontinuous which makes MSD model uninformative in those cases. To remedy this problem Galley and Manning (2008) introduce a *hierarchal reordering model* to phrase based MT systems.

The hierarchical reordering model builds a structure like Permutation Trees (PETs) (Zhang and Gildea, 2007) on which the phrase orientation can be seen more precisely. Many phrases that under MSD model look like discontinuous are actually just in the monotone or swap orientation and appear to be discontinuous only because of the phrase segmentation. Because the hierarchical reordering model groups phrases hierarchically, phrase segmentation will have a smaller negative effect. Hierarchical reordering models should not be confused with hierarchical phrase based models, because even though they do use similar hierarchical structures (permutation trees and normalized decomposition trees (NTDs) (Zhang et al., 2008) respectively) they do search in a completely different way. Decoding with Hiero models is done by a beam search constrained with the hierarchical structure, while decoding in phrase based models with hierarchical reordering models is a standard beam search that is not constrained by the hierarchical structure.

2.2.2 Hierarchical Phrase Based Models

Hierarchical Phrase Based Models (Chiang, 2005), often referred to as Hiero models, are models that are a type of Synchronous Context-Free Grammars (SCFG) or sometimes called Syntax-Directed Transduction Grammars (Lewis and Stearns, 1968; Aho and Ullman, 1969). Synchronous Context-Free Grammars differ from ordinary Context-Free Grammars by having two right-hand sides because, as their name suggests, they synchronously generate two strings. We can use them to simultaneously generate a string in the source language and another string, its translation equivalent, in the target language. Take for instance, an example synchronous grammar from Figure 2.1.

It contains some very simple rules such as rule (1) that generates subject and predicate in both English and Serbo-Croatian. Non-terminals on both right-hand sides must be the same and they are accompanied with indices that match them. Each pair of these

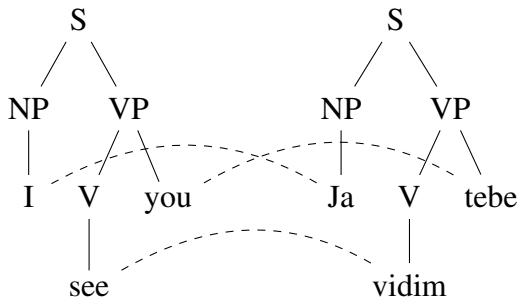


Figure 2.2: SCFG derivation for "Ja vidim tebe"

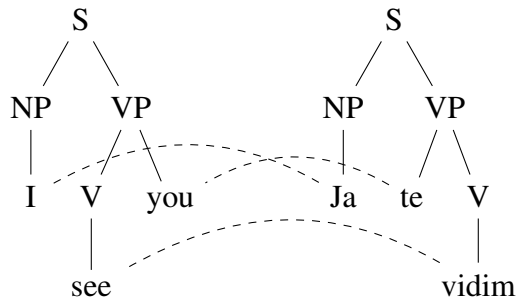


Figure 2.3: SCFG derivation for "Ja te vidim"

non-terminals whose indices match will be rewritten together. The production rule (1) is a completely abstract rule — it contains only non-terminals. Another type of rule are rules like (4) and (5) which are purely lexicalized. They can be interpreted as ordinary phrase pairs that map one string to another. The rules like (2) and (3) are interesting because they are a combination of abstract and lexical rules and because they are doing the reordering. The difference between these two rules can be seen as conditioning the reordering on the lexical choice: if "you" gets translated as "tebe" then the verb should come before the object just like in English (rule (2)), but if "you" gets translated as "te" then the verb should come after the object (rule (3)). The example synchronous derivations for these two possibilities can be seen in Figures 2.2 and 2.3.

This kind of syntactically motivated reordering would be very difficult to do in standard phrase based systems. That is why in the Hiero model the phrase based systems are extended to have "phrases with gaps". Gaps here signify non-terminals that could generate some other phrase pair (with or without a gap). The rule extraction starts with the standard phrase extraction as seen in the previous section. These phrase pairs create fully lexicalized rules as the rules (4) and (5) in the example grammar from Figure 2.1. After these initial rules are extracted, the rule set is extended recursively by searching through pairs of phrases and checking whether one phrase pair is contained in the other. If that is the case then a new rule is created in which the position in which subphrase is in the bigger phrase is replaced with the non-terminal X. If that is the case then a new rule is created in which the span occupied with the suphrase is

replaced with the non-terminal X. The recursive function for expanding the rule table is given bellow with P being the current set of phrase-pairs:

$$\begin{aligned}
\text{if } \; & (\bar{s}, \bar{t}) \in P \; \wedge \; (\bar{s}_{SUB}, \bar{t}_{SUB}) \in P \\
& \wedge \; \bar{s} = \bar{s}_{PRE} + \bar{s}_{SUB} + \bar{s}_{POST} \\
& \wedge \; \bar{t} = \bar{t}_{PRE} + \bar{t}_{SUB} + \bar{t}_{POST} \\
& \wedge \; \bar{s} \neq \bar{s}_{SUB} \wedge \bar{t} \neq \bar{t}_{SUB} \\
\text{then add } & (\bar{s}_{PRE} + X + \bar{s}_{POST}, \bar{t}_{PRE} + X + \bar{t}_{POST}) \text{ to } P
\end{aligned}$$

There are several reasons why Hiero is most often used with only one non-terminal label. One of them is that the non-terminal spans very often do not correspond to syntactic constituents so syntactic labels cannot be directly used for labeling Hiero non-terminals. Some approaches try to tackle this by the usage of CCG-like labels that can label even the spans that do not form complete constituents (Zollmann and Venugopal, 2006). The labels of these systems encode information like "this span is missing non-terminal Y on its right in order to form a complete non-terminal Z". These methods very often significantly increase the number of latent derivations and are very constrained in the sentences they accept compared to the Hiero with only one non-terminal. This can be resolved with a "soft matching" approach that uses non-terminal labels only as soft features that guide the parsing process (Chiang, 2010). Because syntactic annotation is very often not available on source or target sides approaches like the one from Maillette de Buy Wenniger and Sima'an (2015) are more promising since they label Hiero nodes with labels that do not require syntactic annotation, but are based purely on different structural configurations in which these non-terminals appear.

Since synchronous grammars can generate two strings in parallel, they can also be used to parse two strings in parallel. However, that is not the usual application of them in the context of machine translation. In machine translation we are given one string that needs to be parsed and we use only one side of the SCFG rules to parse that string. The derivation of that parse gives as a side effect the target side sentence that is the translation of the parsed sentence. For this any standard parsing algorithm such as CKY can be used (Cocke, 1969; Kasami, 1965; Younger, 1967).

Bilingual parsing can be useful in the case we want to induce the synchronous grammar directly from the parallel data. This has much higher polynomial complexity which is why it is not used in practice compared to the methods that are based on extraction of the grammar directly from alignments. The type of synchronous grammar that was used most frequently in MT is Inversion Transduction Grammar (ITG) (Wu, 1997) mostly because of the relatively low polynomial complexity of building the chart $O(n^6)$. ITG is a binary version of SCFG which does not have the same expressive power as arbitrary SCFG (Huang et al., 2009).

2.2.3 Continuous Models

In the recent years a different type of models that are based on neural networks became dominant in the machine translation community. Neural machine translation models (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015) have completely changed the whole architecture that was used for more than a decade. One of the ways to see this shift is as a shift from discrete representations to continuous representations that made it easier to incorporate many more advanced learning algorithms. It is difficult to say what is in this models exactly a translation unit because they consider the information from the whole source sentence while translating. This is largely possible because of type of representation that is used which is continuous.

The work done in this thesis is not based on neural models so they will not be described here. The reader is referred to some recent tutorials on this topic (Goldberg, 2015; Neubig, 2017).

2.3 Discriminative Models

The MT generative models have two main drawbacks. First is that the translation and language model make too many independence assumptions. A second drawback is that they optimize as an objective the likelihood function, which is often not the objective that is of interest. We want to optimize for the quality of translation and for that automatic evaluation metrics are much better indicator than likelihood.

Discriminative models for PBMT are an attempt to address both of these problems. The derivation of discriminative models starts with application of the $log(\cdot)$ function to the content of the argmax search. Since $log(\cdot)$ is a monotone function, its application will not change the result of argmax.

$$
\begin{aligned}
\hat{t} &= \underset{t \in T}{\mathrm{argmax}}\ p(t|s) \\
&= \underset{t \in T}{\mathrm{argmax}}\ \frac{p(s|t)p(t)}{p(s)} \\
&= \underset{t \in T}{\mathrm{argmax}}\ p(s|t)p(t) \\
&= \underset{t \in T}{\mathrm{argmax}}\ log(p(s|t)p(t)) \\
&= \underset{t \in T}{\mathrm{argmax}}\ log(p(s|t)) + log(p(t)) \\
&= \underset{t \in T}{\mathrm{argmax}}\ [1.0, 1.0] \begin{bmatrix} log(p(s|t)) \\ log(p(t)) \end{bmatrix}
\end{aligned} \tag{2.24}
$$

The last line shows that we can formulate generative models as a dot product between the vector that contains the logarithms of the translation and language model

probabilities and the vector of ones. We can call these two vectors vectors as *feature vector* and *weight vector*. The feature vector $\boldsymbol{\phi}$ contains indicators of the quality of a translation t for the source sentence s. For that reason sometimes instead of writing the feature vector $\boldsymbol{\phi}$ I will write the feature function $\phi(s,t)$. The weight vector $\boldsymbol{w}$ contains the importance of each feature indicator for the final translation score. The score of the translation is given by:

$$score(s,t) = \boldsymbol{w} \cdot \phi(s,t) \tag{2.25}$$

The decoder during search tries to find the translation that has the highest $score(s,t)$. This is not a probabilistic model anymore because the score does not have to be between 0 and 1. In theory there is no problem in making a discriminative model probabilistic. The reason for the model not being probabilistic in practice is that it is computationally costly to normalize the score over all possible translations. Giving up on probabilistic interpretation allows easier training for a good weight vector.

2.3.1 Feature Functions

This formulation of discriminative models is more flexible than the generative models, because now we can include arbitrary features into the model. The number of features in PBMT systems can go from a dozen of dense features that are complex models themselves, to millions of features where most of the features are simple sparse features.

Because discriminative models are not based on Bayes' rule, there is no restriction in using the translation model only in one direction $p(s|t)$, but also the other direction $p(t|s)$ can be added as a feature indicator to the model. The combination of these two models outperforms both the models based on direct translation and the generative models based on inverse direction (Koehn, 2010).

The problem that n-gram language models with high order have with data sparsity and require smoothing, also exists in the phrase based translation models. Very large phrases would be observed only a few times and their counts will not be very reliable. The effect of smoothing is achieved by introducing an additional feature in the discriminative model that does lexical weighting. Lexical translations have richer statistics which makes them more reliable.

The reordering models mentioned before as part of the translation model, in the discriminative models are individual features that get their own weight. All PBMT systems use distance based reordering model and some in addition to it have MSD reordering model. Having a discriminative model allows usage of several reordering models simultaneously.

Some feature functions, such as language models, have a strong preference for short translations. To prevent the translations getting too short, the discriminative model includes a *word penalty*: a feature that counts the words. If the weight of that feature

is set to a high value the translation system will prefer longer sentences, while if it is small it will prefer short sentences.

Translations can be formed by using different phrase segmentations. If longer phrases are used then bigger context could be exploited, but the statistics are less reliable. With shorter phrases the problem is the opposite: small context but more accurate statistics. To balance between the two, machine translation systems use *phrase penalty* which counts the number of phrases that are used. With high weight for the phrase penalty, the PBMT system will prefer many smaller phrases, while a small phrase penalty will give a preference for the longer phrases.

With the advance of learning algorithms for training the weights of the discriminative model, it became possible to train models with a very large number of features. These features are sparse features which means only small portion of the feature vector will have non-zero values. An example of a sparse feature are phrase pairs. Each phrase pair is a feature on its own and has its own weight. The difference with using phrase pairs in this way compared to the usage of phrase pair probability from the translation models is that in this case the weight assigned to the phrase pair is estimated jointly with the other components of the whole PBMT system and it is estimated for optimising the goal objective instead of likelihood.

2.3.2 Tuning Algorithms

Training the weights of a discriminative model is done by iteratively decoding and optimizing weights. Weights are optimized towards some desired evaluation metric. Ideally, the weights would be optimized in such a way that out of all possible translations, the translation that scores highest under some evaluation metric would also score highest under the model score. However, training for optimal weights over all possible translations is not feasible and that is why the optimization is done on an n-best list as an approximation to the full search space.

Algorithm 1 General scenario of SMT tuning

1: $\boldsymbol{w} \leftarrow initial\_parameters$
2: **repeat**
3: $\quad n\_best \leftarrow decode\_N\_best(s; \boldsymbol{w})$
4: $\quad \boldsymbol{w} \leftarrow optimize(n\_best, t)$
5: **until** convergence

Decoding after every optimization step (line 3) is necessary because the weights that are found are optimal only for the n-best list reranking. Reproducing the n-best list is a heuristic that tries to account for the difference between the n-best list and the search space of the decoder.

The optimization step (line 4) can be done in many different ways. The most popular algorithms are MERT (Och, 2003), kb-MIRA (Cherry and Foster, 2012) and PRO (Hopkins and May, 2011).

MERT

MERT (Och, 2003) optimizes the parameters one by one. Let us say that we are optimizing for a parameter w_c. The scoring function can be expressed as:

$$score(s,t) = \sum_i w_i \, \phi_i(s,t) = w_c \, \phi_c(s,t) \; + \; \sum_{i \neq c} w_i \, \phi_i(s,t) \tag{2.26}$$

Because the feature values do not change and all weights except weight w_c are not changing, we can treat $\sum_{i \neq c} w_i \phi_i$ as a constant u_c. This makes a scoring function a linear function with one variable.

$$score(s,t) = w_c \, \phi_c(s,t) \; + \; u_c(s,t) \tag{2.27}$$

Varying the value of w_c would change which translation gets on top of the n-best list. The point at which one translation gets higher score than the other is the point in which the two lines of these functions cross:

$$w_c \, \phi_c(s,t_1) \; + \; u_c(s,t_1) \; = \; w_c \, \phi_c(s,t_2) \; + \; u_c(s,t_2) \tag{2.28}$$

This is a simple linear equation with only one variable:

$$w_c = \frac{u_c(s,t_2) - u_c(s,t_1)}{\phi_c(s,t_1) - \phi_c(s,t_2)} \tag{2.29}$$

The points in which argmax changes are called threshold points. The first line with the best argmax is the one with the steepest line: with the smallest ϕ_c. The first threshold point is found by finding the smallest solution to Equation 2.29, where t_1 is the previous best translation and for t_2 all other translations in the n-best list are tried until the one that gives the highest model score is found. This procedure is repeated until all threshold points are found. Afterwords, all threshold points are tested over whole n-best list to see which one gives the highest score on the desired evaluation metric.

kb-MIRA

MERT's great advantage is that it can optimize any metric, even those that require evaluation on the corpus level, like BLEU does. The problem with MERT is that it can optimize only small number of parameters which limits its applicability to a dozen of dense features. If we want to use sparse features in PBMT some other tuning algorithm is needed. kb-MIRA (Cherry and Foster, 2012) is a large-margin learning algorithm that can tune a model with a large number of features. It is based on previous work on online large-margin modifications of a Perceptron algorithm (Crammer et al., 2006; Watanabe et al., 2007; Chiang et al., 2008).

MIRA tries to increase the margin between "hope" and "fear" translations. *Hope* translation is the translation which has high both the model score and the metric score compared to the reference translation:

$$t_{hope} = \underset{t \in T}{\operatorname{argmax}}\, score(s,t) + metric(t, t_{ref}) \tag{2.30}$$

Fear translation is the translation which has the highest potential for causing the mistake of our system: it has a high model score but a low metric score:

$$t_{fear} = \underset{t \in T}{\operatorname{argmax}}\, score(s,t) - metric(t, t_{ref}) \tag{2.31}$$

If the distance between the model scores of t_{hope} and t_{fear} is not as big as their distance in terms of metric score, then the update rule of MIRA is triggered which ensures that this is corrected. This update rule is similar to the Perceptron update rule except that the size of the update is dynamically changed to enforce the large margin. Since it is difficult to find the exact hope and fear translations as defined above because the search space of all possible sentences is intractable, an n-best list is used as an approximation of the search space in the case of kb-MIRA and lattice in the case of lattice-MIRA (Cherry and Foster, 2012).

The advantage of optimizing a large number of features compared to MERT comes with a price — the optimization is done on the sentence level and therefore requires a good sentence level metric. BLEU, the most used evaluation metric in machine translation, has a very low quality on the sentence level. Because of that usually the some modified versions of BLEU are used instead with the expectation that this modified version of BLEU would lead to the higher original BLEU on the test set.

2.4 Evaluation

The output of machine translation systems can be evaluated in two ways. First is by directly asking humans to assess the quality of the MT output. This is what MT developers want in the end. However, this process of evaluation is very slow and expensive compared to its alternative: automatic evaluation. Automatic evaluation metrics are just an approximation to the human judgment, but because of its speed very useful in the process of development of an MT system.

2.4.1 Human Evaluation

There are many ways how humans can judge the quality of MT systems. Early WMT shared tasks (Koehn and Monz, 2006) have used two criteria to judge MT systems: adequacy and fluency. The human judge is presented with 5 translations and for each one of them she/he needs to assign an integer between 1 and 5 for its fluency (independently of the reference translation) and for its adequacy (given the source sentence and its reference translation).

In the coming years this method was replaces by the *relative ranking* method (RR) (Callison-Burch et al., 2007). In relative ranking method, the human judges are asked

whether they prefer one translation over the other, given the reference translation made by a human translator. The judges are not told explicitly to evaluate any individual aspect of the translation system such as fluency or adequacy, but only to rank translations by their "quality" where they leave up to the judge to decide what quality means.

In WMT shared translation tasks since 2007, the main evaluation method is RR method in which the judges are presented 5 translations and, as before, they need to assign an integer between 1 and 5 but in this case the numbers have different meaning. The numbers do not represent the absolute measure of quality, but a "rank" of translation: rank 1 means that the translation is the best translation out of the 5 offered translations, while rank 5 means that this is the worst translation. Ties in rankings are allowed.

The reason why the judge is offered 5 translations to rank instead of just 2 is because judging ranks of 5 translations at the same time gives 10 pairwise rankings. Because this causes more cognitive effort and less reliable rankings some other translation tasks present only 2 translations to the judges (Braslavski et al., 2013).

In recent years there have been a few publications that recommend using a more modern version of absolute scoring of translation instead of relative scoring. Graham et al. (2013) propose usage of a continuous scale that can assign integer numbers between 1 and 100 which gives a much higher granularity to the judgment than the previous 5 integers scale. Together with large number of judges per sentence, the average of the these absolute scores can give more reliable judgments than relative ranking approach. This method of directly assigning the absolute adequacy score to the system, named *direct assessment* (DA), was used in parallel with RR method in human evaluation on WMT16 shared translation task (Bojar et al., 2016).

Human judges judge only individual sentences. The procedure for generalizing the judgments of individual sentences to the judgments of systems that generated these sentences is varies depending on the type of judgments that are collected. In the case of DA judgments the procedure is easy: we just need to take the average of the absolute scores assigned by human judges to the sentences produced by the system.

In case RR judgments there are several possible procedures that were changing over the years. Initial measures counted the number of wins of one system versus any other system that was compared against it. The formula for computing system's score is given bellow where $win(S_i, S_j)$ is the number of judgments in which system S_i was ranked as better than system S_j and the ties are ignored:

$$\text{score}(S_i) = \frac{\sum_{j,j\neq i} \text{win}(S_i, S_j)}{\sum_{j,j\neq i} \text{win}(S_i, S_j) + \text{win}(S_j, S_i)} \tag{2.32}$$

This method was used in the initial WMT tasks, but it was changed because it suffers from the problems of luck-of-the-draw — system that is less "lucky" might always get compared to the strong MT systems and unfairly get a low score.

To account for this a new scoring method called *expected wins* (Koehn, 2012) was used instead:

$$\text{score}(S_i) = \frac{1}{|\{S_j\}|} \sum_{j,j \neq i} \frac{\text{win}(S_i, S_j)}{\text{win}(S_i, S_j) + \text{win}(S_j, S_i)} \tag{2.33}$$

This method computes the expectation that the system in question S_i will have a win over any other system randomly drawn from the set of systems it was compared against.

Recent WMT tasks use more sophisticated measures such as TrueSkill (Sakaguchi et al., 2014), which give more reliable system scores (Bojar et al., 2014) but are much more complicated and computationally involved than Expected Wins scoring.

2.4.2 Automatic Evaluation Metrics

Automatic evaluation metrics are functions that evaluate the quality of the machine translation output given one or more reference translations. This function should return higher score for translations that have better quality than for translations with lower quality. What "quality" exactly means depends on the metrics definition. Some metrics are based on "edit distance"— how many operations are needed in order to repair the system output and make it a good translation. Some other metrics are based on overlap between system translation and reference translation under some properties. These properties that are checked for overlap can be word n-grams, syntactic treelets or even predicate-argument relations.

BLEU

The metric that is the *de facto* standard in machine translation evaluation is BLEU score (Papineni et al., 2002). It is a very simple metric based on precision over n-grams and created with the idea that many references could be used for evaluation. To define BLEU score we first define counting functions. Function $count_{ref}$ counts the maximal number of times an n-gram g appears in any available reference sentence:

$$\text{count}_{ref}(g) = \max_{r \in \text{refs}} \text{count}(g, r)) \tag{2.34}$$

Now we define a *clipped count* function that computes the number of occurrences of n-gram g that appears in both system and reference translations:

$$\text{count}_{\text{clip}}(g, \text{sys}) = \min(\text{count}(g,\ \text{sys}),\ \text{count}_{ref}(g)) \tag{2.35}$$

Precision of n-grams of order n is calculated by taking the ratio of n-grams from all the system translations n-grams that appear in the clipped counts:

$$p_n = \frac{\sum_{s \in \text{ syss}} \sum_{g \in \text{ngram}_n(s)} \text{count}_{\text{clip}}(g)}{\sum_{s \in \text{ syss}} \sum_{g \in \text{ngram}_n(s)} \text{count}(g, s)} \tag{2.36}$$

Here function $\mathrm{ngram}_n(\cdot)$ returns all the n-grams of length n that appear in the sentence that is given as an argument.

$BLEU_n$ score combines precisions of n-gram orders up to the order n by using the geometric mean. If the metric used only precision for its computation then it would be easy to "game" by, for example, a translation that contains only one n-gram that also appears in the reference translation. In that case the metric would give score 1 because all the precision scores will be 1. To account for this bias for short translations, BLEU uses the *brevity penalty* (BP) which applies only if system translation length c is longer than reference translation length r:

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases} \tag{2.37}$$

Now the formula for the $BLEU_n$ score can be given as:

$$BLEU_n = BP \sqrt[n]{p_1 p_2 \cdots p_n} \tag{2.38}$$

Because multiplying many precision scores can lead to underflow the score can be computed using the $\log$ trick:

$$BLEU_n = BP \ \exp\left(\sum_{i=1}^{n} \log \ p_i\right) \tag{2.39}$$

In most cases when papers report $BLEU$ score they refer to $BLEU_4$ score which works with 4-grams:

$$BLEU_4 = BP \sqrt[4]{p_1 p_2 p_3 p_4} \tag{2.40}$$

Even though BLEU score is used in almost every MT paper published in the last decade, some of its deficiencies are widely recognized. One of them is that it performs very badly on the sentence level. The reason for this is matching of big n-grams that on the sentence level often turns out to be 0 which when combined in a geometric mean causes BLEU score to also be 0.

The way BLEU could be modified to not assign zero scores is similar to the way language models are modified: smoothing of the n-gram counts. Smoothed BLEU score (sBLEU or BLEUs) (Lin and Och, 2004) applies add-one smoothing to the computation of precision scores:

$$p_n = \frac{\sum_{s \in \text{ syss}} \sum_{g \in \mathrm{ngram}_n(s)} \mathrm{count}_{\mathrm{clip}}(g) + 1}{\sum_{s \in \text{ syss}} \sum_{g \in \mathrm{ngram}_n(s)} \mathrm{count}(g, s) + 1} \tag{2.41}$$

This way BLEUs score is guaranteed not to give zero score to any input.

METEOR

METEOR (Lavie and Agarwal, 2007; Denkowski and Lavie, 2014) is the second most popular metric in machine translation research. It has brought many new ideas to evaluation; for instance, to have more flexible matching of words. Metrics like BLEU that use strict word matching unfairly punish the system that produces the translation with the same meaning as the reference translation but with the different words. In order to have more flexible matching of words METEOR uses paraphrase tables and WordNet, so that the system is allowed to translate some concept by using any of the allowed synonyms.

In addition to paraphrase tables and WordNet, another linguistic resource that METEOR uses are lists of function words. Distinguishing between function and content words can be important in evaluation, because we can expect that the human evaluator puts more importance on content words that bear the meaning of the sentence than on the function words that have very little (if any) semantic content.

Since there are many factors influencing the METEOR scores (exact matching, paraphrase matching, WordNet matching, function words, content words etc.) METEOR puts additional parameters that influence the importance of each of the factors. These parameters are tuned using hill-climbing methods for high correlation with human judgment.

The lexical overlap between system and reference translation in METEOR is computed over precision and recall of matched words. Bellow h_c refers to the content words in the hypothesis translation and h_f to hypothesis' function words. Variables r_c and r_f mean the same as h_c and h_f but for the reference translation. Functions $m_i(\cdot)$ return the number of matched words with the matcher i where matchers could be: exact matcher, stemming matcher, paraphrase matcher and WordNet matcher.

$$P = \frac{\sum_i w_i \cdot (\delta \cdot m_i(h_c) + (1-\delta) \cdot m_i(h_f))}{\delta \cdot |h_c| + (1-\delta) \cdot |h_f|} \tag{2.42}$$

$$R = \frac{\sum_i w_i \cdot (\delta \cdot m_i(r_c) + (1-\delta) \cdot m_i(r_f))}{\delta \cdot |r_c| + (1-\delta) \cdot |r_f|} \tag{2.43}$$

METEOR uses weights w_i to give different importance to different matchers, because most often one matcher is preferred over the other. Parameter δ controls for the different importance of content and function words.

Precision and recall are combined into F-score which is a harmonic mean, in this case weighted with parameter α for balancing the importance of precision and recall:

$$F = \frac{P \cdot R}{\alpha \cdot P + (1-\alpha) \cdot R} \tag{2.44}$$

This F score is purely lexical and contains no information about the correctness of the word order. If word order was completely correct then all the matched words would be contiguous and form one big matched chunk of words. If word order is not

correct there will be more than one of the matched smaller chunks. In the worst case there will be as many chunks as the number of matched words. METEOR uses that fact to compute the penalty for wrong word order:

$$Pen = \gamma \left(\frac{ch}{m} \right)^{\beta} \tag{2.45}$$

Here β and γ are parameters that try to find the right scaling and importance for the ordering component of METEOR. Final METEOR score is computed by combining the lexical score and the ordering penalty:

$$METEOR = (1 - Pen) \cdot F \tag{2.46}$$

METEOR is a sophisticated metric with very high correlation with human judgment both on sentence and corpus level. In spite of this fact, BLEU stays the most used metric mostly for its simplicity: it requires no training of any parameters, it is fast to compute and requires no linguistic resources such as WordNet, paraphrase tables or stemmers.

2.4.3 Meta-Evaluation

The quality of an evaluation metric is defined by how well it does its main task: approximation of human judgment. After collecting many human judgments of the quality of machine translation output, an evaluation metric is checked for its correlation with human judgment. This process of assessment of evaluation metrics is often called meta-evaluation.

Different types of human judgments require different type of correlations to be computed. In case of relative ranking judgments on the sentence level, the correlation with human judgment is computed via Kendall τ. Kendall τ correlation coefficient is based on computing how many pairs of translations are ranked the same way by the human and the metric. The pairs that are ranked the same are called concordant pairs and pairs that are ranked wrongly are called discordant. The Kendall τ correlation coefficient is computed in the following way:

$$\tau = \frac{|\,\text{concordant}\,| - |\,\text{discordant}\,|}{|\,\text{concordant}\,| + |\,\text{discordant}\,|} \tag{2.47}$$

This formula ignores ties. Ties are a tricky case because including them might cause an unwanted bias in the computation of the correlation scores. There were several modifications proposed by Macháček and Bojar (2014) that handle ties.

If, instead of relative ranking, we have absolute scores then the correlation with human judgment is computed with Pearson r correlation coefficient.

$$r = \frac{\sum_{i=1}^{n}(H_i - \bar{H})(M_i - \bar{M})}{\sqrt{\sum_{i=1}^{n}(H_i - \bar{H})^2}\sqrt{\sum_{i=1}^{n}(M_i - \bar{M})^2}} \tag{2.48}$$

$\bar{H}$ and $\bar{M}$ are means of human and metric scores respectively. H_i and M_i are scores given to system i by human judge and a metric respectively.

Meta-evaluation on the corpus level can be done in two ways. One of them is by comparing the rankings of systems extracted by some method (like Expected Wins or TrueSkill), to the rankings produced by the evaluation metric. For this case the Spearman's ρ correlation coefficient is used:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \tag{2.49}$$

Here d_i represents the difference in ranks of system i in human and metrics rankings, and n is a total number of systems that are being ranked. Spearman correlation is equivalent to Pearson correlation done on ranks instead of absolute values.

Recent metrics shared tasks, instead of Spearman's ρ correlation coefficient, use Pearson r because this correlation coefficient additionally shows whether metrics scores correspond to human scores by values, and not only by ranking.

2.5 Permutation Factorization

The core part of this thesis deals with modeling word order using hierarchical structures over permutations. In this section I present the background on treating word order mapping between two strings as a permutation and how permutations can be modeled using hierarchical structures.

2.5.1 Representing Word Order Mapping with Permutations

In machine translation we are usually trying to model a mapping between two sequences of words: the source side sentence and its translation on the target side, such as the one in Figure 2.4a. This mapping involves two main factors: lexical translation mapping and word order mapping. It is clear that these factors are interconnected and cannot be separated easily, but separating them has many advantages, the main one being that we will have a simpler task for which we can apply better learning models and better search algorithms.

If we are interested in modeling only the word order, then we can ignore the lexical translation on the target side and model only the mapping of source side words from their position in the source sentence to their position in the target sentence. This modeling can be done with alignments that do not have lexical part on the target side, as shown in Figure 2.4b where all the source words are annotated with the position they take on the target side. This example shows some of the main difficulties present in separating the word order from lexical translation. First, there are some words that do not have translation equivalent on the target side so predicting their target side position is not trivial. Second, several words can map to the same target position, as a result

of fertility—some words translate to more than one word. We can see both of these difficulties as arising from the fact that alignment is a many-to-many mapping.

Since the main focus in this thesis is on the word order, modeling fertility of a word can be factored out from the model. For instance, in the example in Figure 2.4b we are not interested exactly to which target position words *day*, *after* and *tomorrow* map, but only how are they ordered relative to each other. It is enough to know that they should not be reordered, but only stay as they are on the source side. So instead of modeling relation of source words to the target words, we can model relation of source words among themselves: we can express reordering only in terms of how should source words be reordered among themselves to match the target order as close as possible.

We can apply heuristics to convert many-to-many mapping to a bijective mapping that would be easier to model. By doing that the modeling problem is not any more a problem of modeling alignments but of modeling permutations as illustrated by Figure 2.4c. To get from alignment view of Figure 2.4b to the permutation view of Figure 2.4c we need i) a heuristic for reducing many to many alignments to one-to-one and ii) a heuristic for aligning unaligned words. These will be discussed in Section 2.5.2.

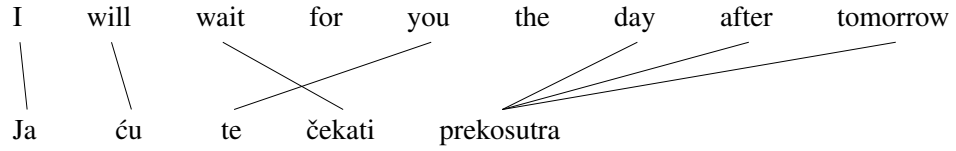
The permutations have an inner structure: majority of the permutations can be treated as permutations of smaller sub-permutations. This recursive decomposition of permutations results in a hierarchical view of permutations as in Figure 2.4d. This view will be discussed in Section 2.5.3.

2.5.2 Reducing Alignments to Permutations

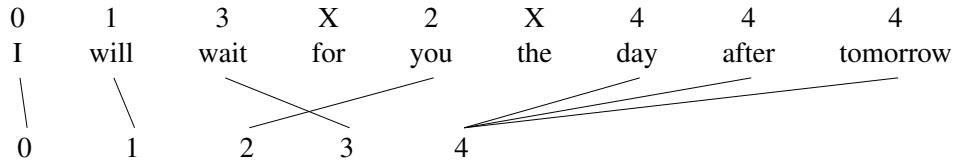
Modeling of word order as a permutation was done in many previous works both in the domain of word order prediction (Tromble and Eisner, 2009; Visweswariah et al., 2011; Bisazza and Federico, 2013) and in the domain of word order evaluation (Birch and Osborne, 2010; Isozaki et al., 2010a). Each of the works that modeled word order as a permutation made different choices for alignments and for the heuristics that reduce alignments to permutations.

In the context of word order prediction the most used alignment method is based on using existing alignment toolkits such as GIZA++ (Och and Ney, 2003) or Berkeley Aligner (Liang et al., 2006). Monolingual tasks such as word order evaluation either use specially designed alignment algorithms that are based on word matching or some more sophisticated techniques that exploit several sources of information (stemmers, paraphrase tables, WordNet) and use beam search to find the most likely alignment. An example of this method is the METEOR aligner (Lavie and Agarwal, 2007).

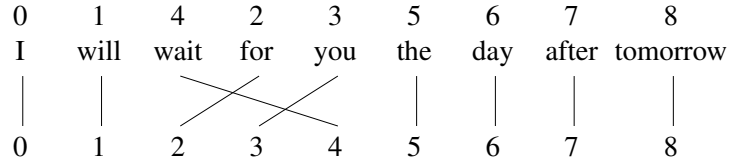
The usual process of reducing alignments goes trough three stages. In the first stage one-to-many alignments are reduced to one-to-one alignments by picking one alignment link that will stay and removing all the other links connected to the word under consideration. Many different choices can be made in this stage. One option is to pick the leftmost or the rightmost link, in this case the choice can be informed by the linguistic knowledge. For instance, if the target language is head-final then choosing the rightmost link might be a preferred choice. Another option is to keep the link with



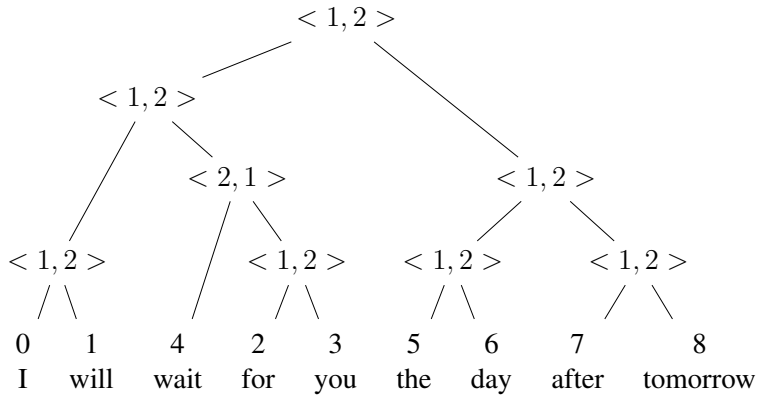
(a) Joint lexical and reordering view



(b) Alignment view



(c) Permutation view



(d) Hierarchical (ITG/PETs) view

Figure 2.4: Different views of word order mapping from English to Serbo-Croatian example sentence pair

the highest probability. This is possible only in the cases where the aligner provides probabilities together with the alignment links. This is in almost all cases specific to bilingual problems where tools like GIZA++ are used. One more technique that is specific to bilingual case is the "intersection" technique. The alignment tools based on IBM models produce one-to-many alignments in one direction and many-to-one in the other. The intersection of these two sets of alignment links gives one-to-one alignments.

After the first stage of reduction there will be no one-to-many alignments. In the second stage the unaligned words are given a "fake alignment". This is necessary because permutation is one-on-one mapping and therefore every word must have a target position even if that position is artificial. The most common heuristic for aligning the unaligned words is to assign them the same the target side word as the for their closest left (or right) aligned word. The choice of whether to look for the closest aligned word on left or right can again be linguistically informed. Most of the unaligned words are function words that do not exist in the target language. If the source language is head-initial, then the argument of the function words most probably comes to the right of the function word and in that case the heuristic should pick the closest aligned word to the right. If language is head-final the opposite holds.

After the second stage is finished only many-to-one alignments are left. To get the permutation, which is a one-to-one mapping, it is not necessary to explicitly transform many-to-one alignments to one-to-one. An alternative is to use any sorting algorithm that will sort source words by their target side alignment position. The mapping between the original order of source words and the sorted order of source words gives the permutation of the source words that is most similar to the target order. Since some words on the source side will be aligned to the same target side position it is necessary to use a *stable* sorting algorithm that will not reorder those words among themselves—they should keep the existing source ordering among themselves. The standard versions of Quick sort and Selection sort are not stable sorting algorithms, but many other sorting algorithms such as Merge sort, Insertion sort and Bubble sort are.

2.5.3 Permutation Trees

As mentioned before, permutations have inner structure that can be exploited for taking a hierarchical view of them. Take for instance the permutation in Figure 2.5a. This is the same permutation used earlier in Figure 2.4a. This permutation can be split in two sub-permutations where each sub-permutation forms a contiguous set of integers. There are many ways how these permutations can be split into two. One of the ways is shown in Figure 2.5b. The newly created node that connects the two sub-permutations is labeled with a *permutation of permutations*. The label $< 1, 2 >$ signifies that the first child goes to the first place and the second child to the second place.

This splitting can be continued recursively until all nodes are split to the smallest permutations that cannot be split any further. These smallest permutations are sometimes called *prime permutations* because of the analogy that can be made with prime

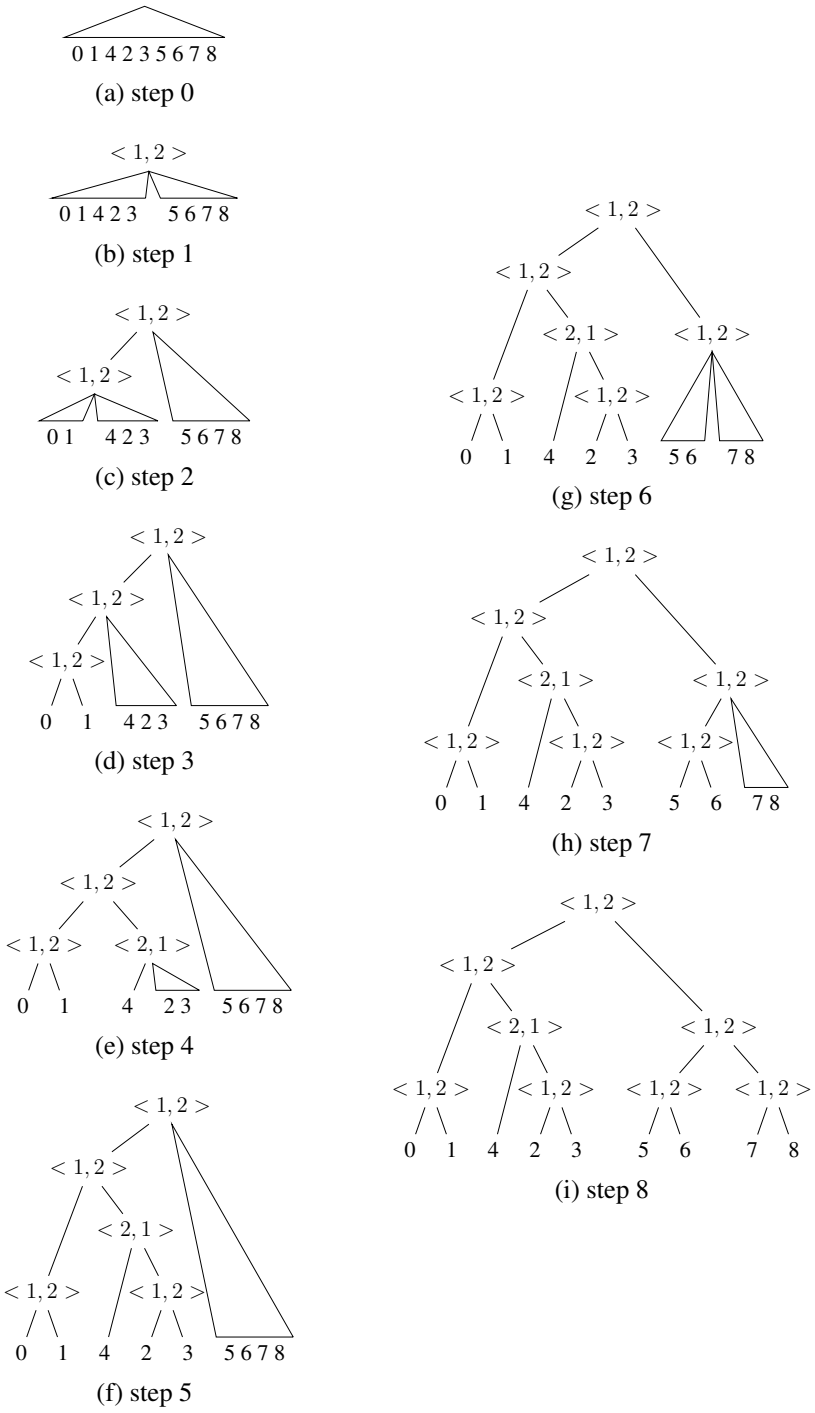


Figure 2.5: Step by step top-down permutation factorization example

numbers: just like natural numbers can be factorized into prime numbers, permutations can be factorized into prime permutations. Alternative names for *prime permutations* are *simple permutations* (because they cannot be simplified further) or *operators* (because they operate how the nodes in the permutation tree get reordered).

In the example that was shown in Figure 2.5 only two instances of prime permutations are used $< 1, 2 >$ and $< 2, 1 >$. When a permutation can be decomposed using only the binary permutations these permutations are called *binarizable permutations* or *ITG permutations* because they can be expressed with Inversion Transduction Grammars (ITG) (Wu, 1997).

However, not all permutations are binarizable permutations. For instance, a permutation 2413 cannot be expressed using smaller binary (nor any other) permutations. That makes it a prime permutation. But permutation does not need to be a prime permutation to be non-binarizable. For example, 5724136 can be decomposed but only with non-binary prime permutations. Even though ITG permutations can capture a large number of reordering patterns available in bilingual data (Huang et al., 2009; Maillette de Buy Wenniger and Sima'an, 2013), they cannot capture all the reordering phenomena (Wellington et al., 2006; Kaeshammer and Westburg, 2014) especially for languages with a more complex word order. For this reason an extension of ITG factorization is necessary in order to allow for non-binarizable permutations too. This extension was presented by Zhang and Gildea (2007) in the form of Permutation Trees (PETs). Permutation Trees work almost in the same way as trees based on ITG except that with PETs there is no restriction in the type of permutations that can be modeled.

An important property of PETs (including ITG) is that they can form many "spurious derivations". What is meant by that is that there can be exponentially many PETs that could generate the same permutation. PETs in Figure 2.5i, Figure 2.6a and Figure 2.6b have completely different structure: the first one is balanced, the second left branching and the third right branching, but even though they are so different they still generate the same permutation. In fact, for this permutation there is exactly 132 PETs that can generate it. In the following sections I present the algorithms that take a permutation as input and produce all the possible PETs as their output.

Shift-Reduce Permutation Parsing

Zhang and Gildea (2007) presented two algorithms for recognizing a canonical left branching PET out of a given permutation. The first algorithm is a simple Shift-Reduce algorithm that runs in $O(n^2)$ and the second is an optimal algorithm based on (Uno and Yagiura, 2000) that runs in $O(n)$. Here the Shift-Reduce algorithm is presented because it is of acceptable speed for majority of applications and much simpler to implement.

The algorithm is fully deterministic. It uses two data structures: a stack and a buffer. The algorithm starts with an empty stack and a buffer filled with the integers from the permutation that is being parsed. Two operations are applied iteratively: shift and reduce. Shift pops one element from the top of the buffer and pushes it on the stack.

Reduce tries to compose top k elements of a stack into a node in the permutation tree. The top k nodes can be composed only if the union of their leafs forms a contingent set of integers. It starts with $k = 2$ and increases it until it either succeeds to compose the node or it reaches the end of stack without succeeding to compose the node. If it successfully composes the node, that new node replaces the composed elements on the stack and a new trial for the reduction starts. If the end of the stack is reached without the success of reduction, then shift action is invoked. The algorithm is guaranteed to end with a left branching PET as the only element on the stack.

Permutation Forest

As mentioned before, a left branching PET is not necessarily the unique PET for a given permutation. There can be exponentially many PETs and if we are interested in modeling all of them it is necessary to encode them in a compact way. The method used in this thesis is to encode all the possible PETs in a chart data structure which can encode an exponential number of trees in a cubic space just like in the case of projective constituency trees. I will refer to this compact representation of all possible PETs that generate the same permutation as a Permutation Forest (PEF).

The main property of PETs that is necessary for constructing PEF is that multiple PETs are a result of a child-parent sequence of binary nodes with the same prime permutation. For instance, if we have a PET that has a sequence of $3 < 1, 2 >$ that fragment of a tree will have 4 nodes on a frontier and we can construct equivalent PETs by having all possible binary branching tree fragments that connect these 4 nodes, clearly with a condition that all the nodes are labeled with the same prime permutation $< 1, 2 >$. The number of these nodes is a Catalan number of the number of nodes on a frontier.

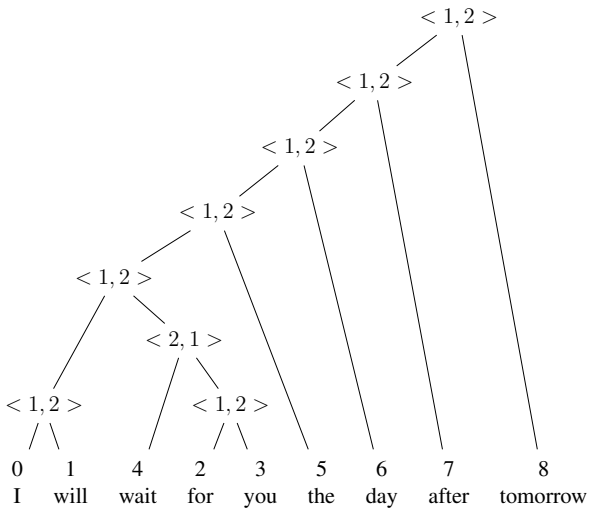
The number of all possible PETs can be computed directly from the canonical left-branching PET. Since multiple different PETs appear only in cases when there is a sequence of more than one node that is either $< 1, 2 >$ or $< 2, 1 >$ (Zhang et al., 2008), we can use these sequences to predict the number of PETs that could be built. Let X represent a set of sequences of the canonical derivation. The number of PETs is computed in the following way:

$$\#PETs = \prod_{x \in X} Cat(|x|) \tag{2.50}$$

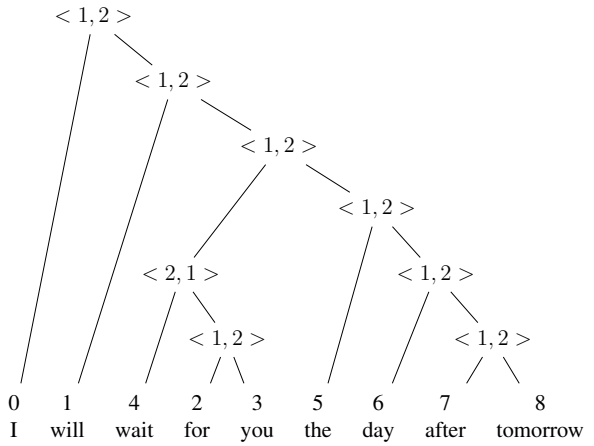
$$Cat(n) = \frac{1}{n+1}\binom{2n}{n} \tag{2.51}$$

where $Cat(\cdot)$ is a Catalan number.

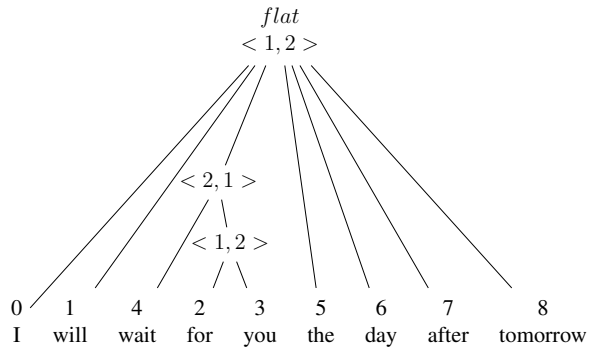
The number of all the possible PETs depends on all of the sequences of $< 1, 2 >$ and the sequences of $< 2, 1 >$ present in the tree. To have a compact view of these sequences it is useful to "flatten" them into a big node that has all their children in monotone order–$< 1, 2 >$ or inverted order–$< 2, 1 >$. For instance Figure 2.6c shows



(a) Left branching canonical PET



(b) Right branching canonical PET



(c) Flat PET

Figure 2.6: Different hierarchical views of the same permutation

a "flat PET" that is the same for both left branching PET in Figure 2.6a and right branching PET in Figure 2.6b.

PEF can be constructed deterministically from any single PET. The process of constructing PEF can go in three steps. First, a canonical left branching PET is extracted using an algorithm described in previous section. Second, the permutation tree is compressed in a more compact flat PET. Finally, from flat PET we extract all the edges that constitute the PEF chart. We do that by simply adding all non-flat nodes from flat PET to the chart and for the flat nodes we add all possible binary edges that could be extracted. This algorithm is based on the algorithm for extracting all possible phrase pairs from Normalized Decomposition Trees presented in (Zhang et al., 2008). For a given canonical permutation tree (which can be built in $O(n)$, this algorithm will construct the hyper-graph of all possible PETs in time that is linear to the number of edges of the hyper-graph: in the best case it will be in constant time (if there is only one edge in the whole hyper-graph because the whole permutation is a prime permutation) and in the worst case it will be $O(n^3)$ just like in the case of CFG parsing.

Chapter 3

Predicting Word Order with Permutation Trees

Predicting word order in machine translation is often left to the decoder which at the same time will try to search for both optimal word order and optimal lexical translation. Separating the two problems can often be beneficial because both the lexical and the reordering model would become more tractable and more advanced modeling techniques could be used. This can be accomplished by a preordering approach where a source sentence is first reordered into a target word order and afterwards gets translated word-by-word (or phrase-by-phrase) without any additional reordering.

Most of the current preordering models rely on syntactic parsing and work by permuting the children of a syntactic tree of the source sentence. This approach can work well only for source languages for which high quality supervised parsers (and therefore also treebanks) are available which presents a problem for the majority of languages. Approaches that do not assume syntactic trees frequently introduce an assumption of binary branching ITG trees. This assumption, as I will show later, presents a limiting factor for building effective preordering models.

In the process of predicting the target word order there is a lot of uncertainty: (1) What is the best tree to transduce a given source sentence? (2) What is the best transduction given the tree? (3) What is the best translation given the permutation? The best way to handle this ambiguity is to efficiently pack all the decisions into a forest or lattice and postpone the decision of the best reordering until the evidence for disambiguation becomes available.

This chapter presents a model that tries to address all of the mentioned problems of the existing reordering models. The chapter is based on two publications. The main publication on which the core of this chapter (from Sections 3.1 to 3.5) is based on is:

authors: Stanojević and Sima'an
title: *Reordering Grammar Induction*
venue: EMNLP 2015

All the research and all implementations were carried out by me. Khalil Sima'an guided me in shaping the key ideas about how Permutation Trees are an effective struc-

ture for the prediction of word order. Khalil Sima'an has also provided guidance and edited the publication.

Section 3.6 of this chapter presents a part of the following publication:

authors:	Daiber, Stanojević, Aziz, and Sima'an
title:	*Examining the Relationship between Preordering and Word Order Freedom in Machine Translation*
venue:	WMT 2016

I have included only the part about Reordering Grammar and of packing many different reorderings into a lattice. I have contributed with the implementation of the Reordering Grammar preordering model and of packing many reorderings into a lattice. I also wrote the sections of the paper that are included in this thesis. Joachim Daiber, Wilker Aziz and Khalil Sima'an helped in thinking about the problems covered in that section and also contributed a large part of the publication that is not included here.

Chapter Highlights

Problem Statement

- Current supervised preordering models rely on syntactic trees which for majority of languages are either unavailable or of low quality.
- Preordering locally on the nodes of a syntactic tree cannot derive all possible translation patters present in parallel corpora as shown by Eisner (2003) for dependency trees and by Khalilov and Sima'an (2012) for constituency trees. The correct reordering would be easier to predict on a tree structure in which reordering would be local to the nodes of the tree.
- Current unsupervised preordering models are constrained on ITG permutations which limits the space of reordering patterns that could be captured.
- Most preordering models do not consider the ambiguity present in the recognition of trees that are going to be transduced. They pick only one tree, usually the most probable one, and ignore the information present in the other less probable trees.

Research Question

- How to create a model that would not depend on syntactic parsers (i.e. is unsupervised), but at the same time not limited to ITG restrictions?
- How to consider all possible trees structures for the source sentence that could be used for reordering?

Research Contributions

- *Reordering Grammar* model is proposed which is based on Permutation Trees. The model is unsupervised because Permutation Trees are derived directly from the parallel corpora and word alignments. It treats both the bracketing and the labels of the brackets as latent variables.

- Permutation Trees are a superset of ITG and therefore *Reordering Grammar* can cover a larger space of reordering patterns than the previously proposed preordering models.

- Inference for the best permutation approximately marginalizes over all possible Permutation Trees that could derive it by using MC sampling. Even the low probability trees contribute to the final decision how to reorder the sentence.

- The final result of preordering is not one permutation but a distribution over possible permutations. The decision about which permutation should be used is left to the decoder which can make a more informed choice because it has also information about lexical translations.

3.1 Introduction

Preordering (Collins et al., 2005) aims at permuting the words of a source sentence s into a new order ś, hopefully close to a plausible target word order. Preordering is often used to bridge long distance reorderings (e.g., in Japanese- or German-English), before applying phrase-based models (Koehn et al., 2007). Preordering is often broken down into two steps: finding a suitable tree structure, and then finding a transduction function over it. A common approach is to use monolingual syntactic trees and focus on finding a transduction function of the sibling subtrees under the nodes (Lerner and Petrov, 2013; Xia and Mccord, 2004). The (direct correspondence) assumption underlying this approach is that permuting the siblings of nodes in a source syntactic tree can produce a plausible target order. An alternative approach creates reordering rules manually and then learns the right structure for applying these rules (Katz-Brown et al., 2011). Others attempt learning the transduction structure and the transduction function in two separate, consecutive steps (DeNero and Uszkoreit, 2011). *Reordering Grammar* model addresses the challenge of learning both the trees and the transduction functions jointly, in one fell swoop, from word-aligned parallel corpora.

Learning both trees and transductions jointly raises two questions. How to obtain suitable trees for the source sentence and how to learn a distribution over random variables specifically aimed at reordering in a hierarchical model? This work solves both challenges by using the factorizations of permutations into Permutation Trees (PETs)

(Zhang and Gildea, 2007). As it will be explained next, PETs can be crucial for exposing the hierarchical reordering patterns found in word-alignments.

The permutations in the training data are obtained by segmenting every word-aligned source-target pair into *minimal phrase pairs*; the resulting alignment between minimal phrases is written as a permutation (1:1 and onto) on the source side. Every permutation can be factorized into a *forest* of PETs (over the source sentences) which here is used as a *latent treebank* for training a Probabilistic Context-Free Grammar (PCFG) tailor made for preordering as explained next.

Figure 3.1 shows two alternative PETs for the same permutation over minimal phrases. The nodes have labels (like $P3142$) which stand for local permutations (called prime permutation) over the child nodes; for example, the root label $P3142$ stands for prime permutation $\langle 3, 1, 4, 2\rangle$, which says that the first child of the root becomes 3^{rd} on the target side, the second becomes 1^{st}, the third becomes 4^{th} and the fourth becomes 2^{nd}. The prime permutations are non-factorizable permutations like $\langle 1, 2\rangle$, $\langle 2, 1\rangle$ and $\langle 2, 4, 1, 3\rangle$, as described previously in Section 2.5.

PETs are suitable for learning preordering for two reasons. Firstly, PETs specify *exactly* the phrase pairs defined by the permutation. Secondly, every permutation is factorizable into prime permutations only (Albert and Atkinson, 2005). Therefore, PETs expose *maximal* sharing between different permutations in terms of both phrases and their reordering. These aspects of PETs can be advantageous for learning hierarchical reordering.

For learning preordering, first an *initial* PCFG is extracted from the latent treebank of PETs over the source sentences only. The nonterminal set of this PCFG is initialized to the *prime permutations* decorating the PET nodes. Subsequently these coarse labels are split in the same way as latent variable splitting is learned for treebank parsing (Matsuzaki et al., 2005; Prescher, 2005; Petrov et al., 2006; Saluja et al., 2014). Unlike treebank parsing, however, *Reordering Grammar* training treebank is latent because it consists of a whole forest of PETs per training instance ($\mathbf{s}$).

Learning the splits on a latent treebank of PETs results in a *Reordering PCFG* with which input source sentences are parsed into split-decorated trees, i.e., the labels are the splits of prime permutations. After parsing $\mathbf{s}$, the splits are mapped back on their initial prime permutations, and then retrieve a reordered version $\acute{\mathbf{s}}$ of $\mathbf{s}$. In this sense, the latent splits are *dedicated to reordering*.

Two technical difficulties appear that are alien to work on latent PCFGs in treebank parsing. First, as mentioned above, permutations may factorize into more than one PET (a forest) leading to a latent training treebank. All PETs for the same permutation share the same set of prime permutations but differ only in bracketing structure (Zhang and Gildea, 2007). And secondly, after a source string $\mathbf{s}$ is parsed, the goal is to find the best $\acute{\mathbf{s}}$, the permuted version of $\mathbf{s}$, and not the best derivation/PET of some $\acute{\mathbf{s}}$. Exact computation of best $\acute{\mathbf{s}}$ is in category of problems which are known to be NP-Complete (Sima'an, 1996). This problem is tackled by sampling from the chart. Additionally, the reordering loss function is integrated in the decision rule trough Minimum-Bayes Risk decoding approach. Kendall reordering score is used as a loss function, which is an

efficient measure over permutations (Birch and Osborne, 2011; Isozaki et al., 2010a).

In summary, this paper contributes:

- A novel latent hierarchical source reordering model working over all derivations of PETs
- A label splitting approach based on PCFGs over minimal phrases as terminals, learned from an ambiguous treebank, where the label splits start out from prime permutations.
- A fast Minimum Bayes Risk decoding over Kendall τ reordering score for selecting $\acute{\mathbf{s}}$.

The results for extensive experiments on the language pair with long distance reordering, namely English-Japanese, show that *Reordering Grammar* gives substantial improvements when used as preordering for phrase-based models, outperforming two existing baselines for this task.

3.2 PETs and the Hidden Treebank

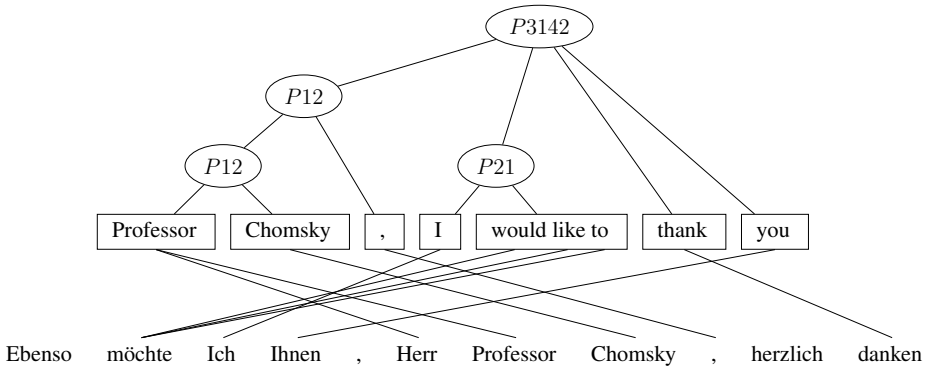
The aim is to learn a PCFG which will be used for parsing source sentences $\mathbf{s}$ into synchronous trees, from which a reordered source version $\acute{\mathbf{s}}$ can be obtained. Since PCFGs are non-synchronous grammars, the nonterminal labels will be used to encode reordering transductions, i.e., this PCFG is implicitly an SCFG. This can be done because both sides of the potential synchronous tree have constituents that bijectively map to each other and all these constituents that map to each other have the same yield (in terms of set, not sequence, of words that they generate).

Here, we have access only to a word-aligned parallel corpus, not a treebank. The following steps summarize the approach for acquiring a latent treebank and how it is used for learning a Reordering PCFG:

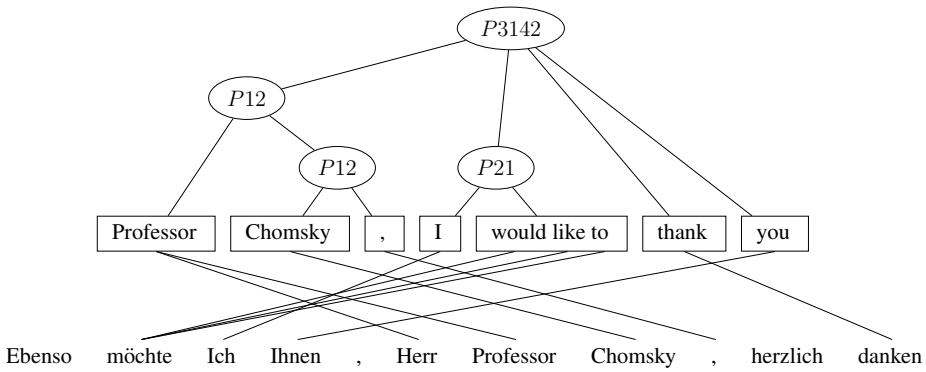
1. Obtain a permutation over minimal phrases from every word-alignment.
2. Obtain a latent treebank of PETs by factorizing the permutations.
3. Extract a PCFG from the PETs with initial nonterminals taken from the PETs.
4. Learn to split the initial nonterminals and estimate rule probabilities.

These steps are detailed in the next section. In this section an intuitive exposition of PETs, the latent treebank and the Reordering Grammar is presented.

Figure 3.1 shows examples of how PETs look like. Here instead of showing a permutation (a sequence of numbers) as a label, a more convenient notation is used that directly maps to permutations. For example, nonterminals $P12$, $P21$ and $P3142$ correspond respectively to reordering transducers $\langle 1, 2\rangle$, $\langle 2, 1\rangle$ and $\langle 3, 1, 4, 2\rangle$. A prime permutation on a source node μ is a transduction dictating how the children of μ are reordered at the target side, e.g., $P21$ inverts the child order. It should be pointed out that any similarity with ITG (Wu, 1997) is restricted to the fact that the straight and inverted operators of ITG are the binary case of prime permutations in PETs ($P12$ and $P21$). ITGs recognize only the binarizable permutations, which is a major restriction when used on data: there are many non-binarizable permutations in actual data (Wellington



(a) Canonical PET



(b) Alternative PET

Figure 3.1: Possible permutation trees for one English-German sentence pair

et al., 2006). In contrast, PETs are obtained by factorizing permutations obtained from the data, i.e., they exactly fit the range of prime permutations in the parallel corpus. In practice, the maximum arity limit is set to 5.

PCFG rules can be extracted from the PETs, e.g., $P21 \rightarrow P12\ P2413$. However, these rules are decorated with too coarse labels. A similar problem was encountered in non-lexicalized monolingual parsing, and one solution was to lexicalize the productions (Collins, 2003) using *head words*. But linguistic heads do not necessarily exist for PETs because constituents in PETs can also be non-syntactic constituents. For this reason, the choice is made for an alternative approach (Matsuzaki et al., 2005; Prescher, 2005; Petrov et al., 2006), which splits the nonterminals and softly percolates splits through the trees gradually fitting them to the training data. Splitting has a shadow side, however, because it leads to combinatorial explosion in grammar size.

Suppose for example node $P21$ could split into $P21_1$ and $P21_2$ and similarly $P2413$ splits into $P2413_1$ and 2413_2. This means that rule $P21 \rightarrow P12\ P2413$ will form eight new rules:

$$P21_1 \rightarrow P12_1\ P2413_1 \qquad P21_1 \rightarrow P12_1\ P2413_2$$
$$P21_1 \rightarrow P12_2\ P2413_1 \qquad P21_1 \rightarrow P12_2\ P2413_2$$
$$P21_2 \rightarrow P12_1\ P2413_1 \qquad P21_2 \rightarrow P12_1\ P2413_2$$
$$P21_2 \rightarrow P12_2\ P2413_1 \qquad P21_2 \rightarrow P12_2\ P2413_2$$

Should we want to split each nonterminal into 30 subcategories, then an n-*ary* rule will split into 30^{n+1} new rules, which is prohibitively large even for the binary rules where $n = 2$. To address this problem, a "unary trick" can be applied as in Figure 3.2. This introduces an independences assumption among the neighbouring nodes. The superscript on the nonterminals denotes the child position from left to right. For example $P21_1^2$ means that this node is a second child, and the mother nonterminal label is $P21_1$. For the running example rule, this gives the following rules:

$$P21_1 \rightarrow P21_1^1\ P21_1^2 \qquad P21_2 \rightarrow P21_2^1\ P21_2^2$$
$$P21_1^1 \rightarrow P12_1 \qquad P21_1^2 \rightarrow P2413_1$$
$$P21_1^1 \rightarrow P12_2 \qquad P21_1^2 \rightarrow P2413_2$$
$$P21_2^1 \rightarrow P12_1 \qquad P21_2^2 \rightarrow P2413_1$$
$$P21_2^1 \rightarrow P12_2 \qquad P21_2^2 \rightarrow P2413_2$$

The unary trick leads to substantial reduction in grammar size: for s splits of a rule with a children instead of getting s^{a+1} new rules we have only $s + s^2a$ new rules. To make it more concrete, let us consider example with a rule of arity 5 whose nonterminals gets split into 30 new states. Without unary trick there would be $30^6 = 729,000,000$ split-rules, but with the unary trick there is only $30 + 30^2 * 5 = 4530$ split rules. This is a result of a constraint that is introduced: after applying the unary trick, all nonterminals on an n-ary branching rule must be split simultaneously. The unary trick was used in early lexicalized parsing work (Carroll and Rooth, 1998), and in this model it is generalized to the latent variable grammar models. This split PCFG constitutes a *latent PCFG* because the splits cannot be read of a treebank. It must be learned from the latent treebank of PETs, as described next.

3.3 Details of Latent Reordering PCFG

Obtaining permutations Given a source sentence $\mathbf{s}$ and its alignment $\mathbf{a}$ to a target sentence $\mathbf{t}$ in the training corpus, $\langle \mathbf{s}, \mathbf{a}, \mathbf{t} \rangle$ is segmented into a sequence of *minimal phrases* $\mathbf{s}_m$ (maximal sequence) such that the reordering between these minimal phrases constitutes a permutation π_m.

Unaligned words In order to handle unaligned words PETs are extended to have specialized operators for them. An unaligned word is joined with a neighboring phrase to the left or the right, depending on the source language properties (e.g., whether the language is head-initial or head-final (Chomsky, 1970)). Since experiments are conducted with English as a source language which is head-initial, the unaligned words

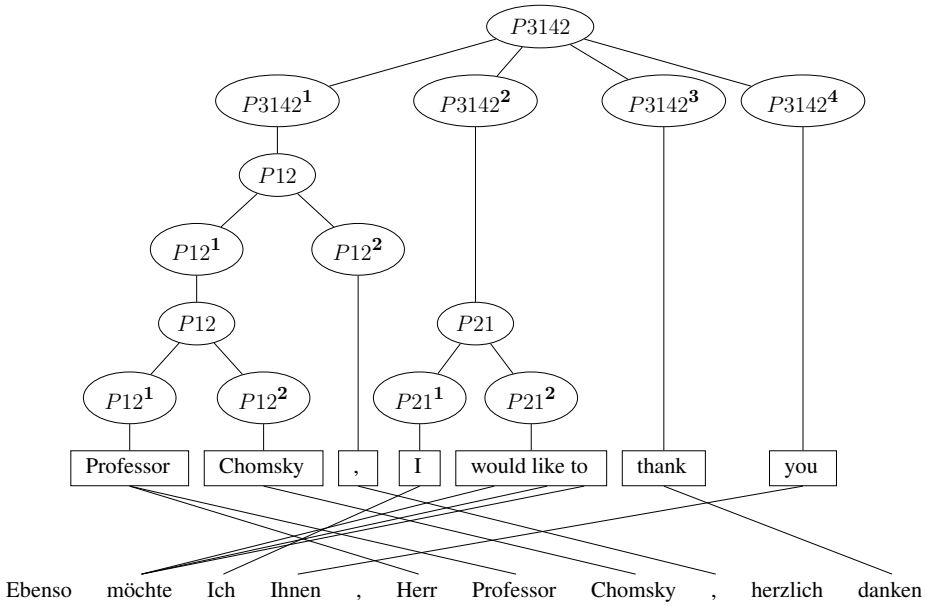


Figure 3.2: Permutation Tree with unary trick

are joined to phrases to their right. This modifies a PET by adding a new binary branching node μ (dominating the unaligned word and the phrase it is joined to) which is labeled with a dedicated nonterminal: $P01$ if the unaligned word joins to the right and $P10$ if it joins to the left.

3.3.1 Probability model

The permutation π_m is decomposed into a forest of permutation trees $PEF(\pi_m)$ in $O(n^3)$ where n is the length of the permutation, following algorithms algorithms described in previous chapter. Each PET $\mathbf{\Delta} \in PEF(\pi_m)$ is a different bracketing (differing in binary branching structure only). In the latent treebank the bracketing is a hidden variable and unsupervised learning is used to induce a distribution over the possible bracketings. The probability model starts from the joint probability of a sequence of minimal phrases $\mathbf{s}_m$ and a permutation π_m over it. This demands summing over all PETs $\mathbf{\Delta}$ in the forest $PEF(\pi_m)$, and for every PET also over all its label splits, which are given by the grammar derivations $\mathbf{d}$:

$$P(\mathbf{s}_m, \pi_m) = \sum_{\mathbf{\Delta} \in PEF(\pi_m)} \sum_{\mathbf{d} \in \mathbf{\Delta}} P(\mathbf{d}, \mathbf{s}_m) \tag{3.1}$$

The probability of a derivation $\mathbf{d}$ is a product of probabilities of all the rules $\mathbf{r}$ that build it:

$$P(\mathbf{s}_m, \pi_m) = \sum_{\mathbf{\Delta} \in PEF(\pi_m)} \sum_{\mathbf{d} \in \mathbf{\Delta}} \prod_{\mathbf{r} \in \mathbf{d}} P(\mathbf{r}) \tag{3.2}$$

As usual, the parameters of this model are the PCFG rule probabilities which are estimated from the latent treebank using expectation-maximization (EM) as explained next.

3.3.2 Learning Splits on the Latent Treebank

Training of the latent PCFG over the latent treebank is done with EM (Dempster et al., 1977) which estimates PCFG rule probabilities to maximize the likelihood of the parallel corpus instances. Computing expectations for EM is done efficiently using Inside-Outside (Lari and Young, 1990). As in other state splitting models (Matsuzaki et al., 2005), after splitting the non-terminals, the probability is distributed uniformly over the new rules, and small random noise is added to each new rule to break the symmetry. The non-terminals are split only once as in (Matsuzaki et al., 2005) (unlike (Petrov et al., 2006)). For estimating the distribution for unknown words all words that appear ≤ 3 times are replaced with the "UNKNOWN" token.

3.3.3 Inference

CKY+ (Chappelier and Rajman, 1998) is used to parse a source sentence $\mathbf{s}$ into a forest using the learned split PCFG. Unfortunately, computing the most-likely permutation (or alternatively $\acute{\mathbf{s}}$) as in

$$\operatorname*{argmax}_{\pi \in \Pi} \sum_{\boldsymbol{\Delta} \in PEF(\pi)} \sum_{\mathbf{d} \in \boldsymbol{\Delta}} P(\mathbf{d}, \pi_m)$$

from a lattice of permutations Π using a PCFG is NP-complete (Sima'an, 1996). Existing techniques, like variational decoding or Minimum-Bayes Risk (MBR), used for minimizing loss over trees as in (Petrov and Klein, 2007), are not directly applicable here because the aim is to find the best permutation not the best tree nor derivation. For that reason it is better to minimize the risk of a loss function over permutations by using the MBR decision rule (Kumar and Byrne, 2004):

$$\hat{\pi} = \operatorname*{argmin}_{\pi} \sum_{\pi_r} Loss(\pi, \pi_r) P(\pi_r) \tag{3.3}$$

The loss function that I used is Kendall τ (Birch and Osborne, 2011; Isozaki et al., 2010a) because of its simplicity which allows very attractive computational properties. Having exact computation of MBR decision with Kendall τ loss can be done in polynomial time $O(n^5)$, but since this is still computationally expensive, the solution is approximated by MC sampling from the chart. With ancestral sampling 10000 derivations are sampled from the chart and then the least risky permutation is found in terms of the loss function. Sampling is performed with respect to the true distribution by sampling edges recursively using their inside probabilities. An empirical distribution over permutations $P(\pi)$ is given by the relative frequency of π in the sample.

With large samples it is hard to efficiently compute expected Kendall τ loss for each sampled hypothesis. For a sentence of length k and a sample of size n the complexity of a naive algorithm is $O(n^2k^2)$. Computing Kendall τ alone takes $O(k^2)$. We can use the fact that Kendall τ decomposes as a linear function over all skip-bigrams b that could be built for any permutation of length k:

$$Kendall(\pi, \pi_r) = \sum_b \frac{1 - \delta(\pi, b)}{\frac{k(k-1)}{2}} \delta(\pi_r, b) \tag{3.4}$$

Here δ returns 1 if permutation π contains the skip bigram b, otherwise it returns 0. With this decomposition we can use the method from (DeNero et al., 2009) to efficiently compute the MBR hypothesis. Combining Equations 3.3 and 3.4 we get:

$$\hat{\pi} = \operatorname*{argmin}_{\pi} \sum_{\pi_r} \sum_b \frac{1 - \delta(\pi, b)}{\frac{k(k-1)}{2}} \delta(\pi_r, b) P(\pi_r) \tag{3.5}$$

The summation can be moved inside and we can reformulate the expected Kendall τ loss as expectation over the skip-bigrams of the permutation.

$$= \operatorname*{argmin}_{\pi} \sum_b (1 - \delta(\pi, b)) \Big[\textstyle\sum_{\pi_r} \delta(\pi_r, b) P(\pi_r) \Big] \tag{3.6}$$

$$= \operatorname*{argmin}_{\pi} \sum_b (1 - \delta(\pi, b)) \mathbb{E}_{P(\pi_r)} \delta(\pi_r, b) \tag{3.7}$$

$$= \operatorname*{argmax}_{\pi} \sum_b \delta(\pi, b) \mathbb{E}_{P(\pi_r)} \delta(\pi_r, b) \tag{3.8}$$

This means that only two passes over the data are needed: a first one to compute expectations over skip bigrams and a second one to compute expected loss of each sampled permutation. The time complexity is $O(nk^2)$ which is fast in practice.

3.4 Experiments

The experiments are conducted with three baselines:

- **Baseline A**: No preordering.
- **Baseline B**: Rule based preordering (Isozaki et al., 2010b), which first obtains an HPSG parse tree using Enju parser [1] and after that swaps the children by moving the syntactic head to the final position to account for different head orientation in English and Japanese.
- **Baseline C**: LADER (Neubig et al., 2012): latent variable preordering that is based on ITG and large-margin training with latent variables. LADER is ran in standard settings without any linguistic features (POS tags or syntactic trees).

[1] http://www.nactem.ac.uk/enju/

Four variants of the *Reordering Grammar* model are tested:

- **RG$_{\text{left}}$** - only canonical left branching PET
- **RG$_{\text{right}}$** - only canonical right branching PET
- **RG$_{\text{ITG-forest}}$** - all PETs up to arity 2 (ITG)
- **RG$_{\text{PET-forest}}$** - all PETs up to arity 5

Experiments are conducted on English-Japanese NTCIR-8 Patent Translation (PATMT) Task. Tuning is done on all NTCIR-7 development sets. Testing is done on the test set from NTCIR-9 from both directions. All used data was tokenized (English with Moses tokenizer and Japanese with KyTea [2]) and filtered for sentences between 4 and 50 words. A subset of this data is used for training the Reordering Grammar, obtained by filtering out sentences that have prime permutations of arity > 5, and for the ITG version arity > 2. Baseline C was trained on 600 sentences, just like in the original paper (Neubig et al., 2012), because training is prohibitively slow. Table 3.1 shows the sizes of data used.

corpus	#sents	#words source	#words target
train RG$_{\text{PET}}$	786k	21M	–
train RG$_{\text{ITG}}$	783k	21M	–
train LADER	600	15k	–
train translation	950k	25M	30M
tune translation	2k	55K	66K
test translation	3k	78K	93K

Table 3.1: Data stats

The Reordering Grammar was trained for 10 iterations of EM on *train RG* data. Binary non-terminals are split into 30 states and non-binary into 3 states. Training on this dataset takes 2 days and parsing tuning and testing set without any pruning takes 11 and 18 hours respectively on 32 CPUs. Figure 3.3 shows the perplexities after each iteration.

3.4.1 Intrinsic evaluation

First I conduct the intrinsic evaluation in which it can be observed how well does the preordering model predict the gold reorderings before translation. Gold reorderings are extracted from forced alignments on the test set–the alignments that are a result of applying the alignment model trained only on the training set. Gold reorderings for the test corpus are obtained by sorting words by their average target position and (unaligned words follow their right neighboring word). MGIZA++ was used for training the alignment model [3]. Kendall τ score was used for evaluation (note the difference with Section 3.3.3 where it was defined as a loss function).

[2] http://www.phontron.com/kytea/

[3] http://www.kyloo.net/software/doku.php/mgiza:overview

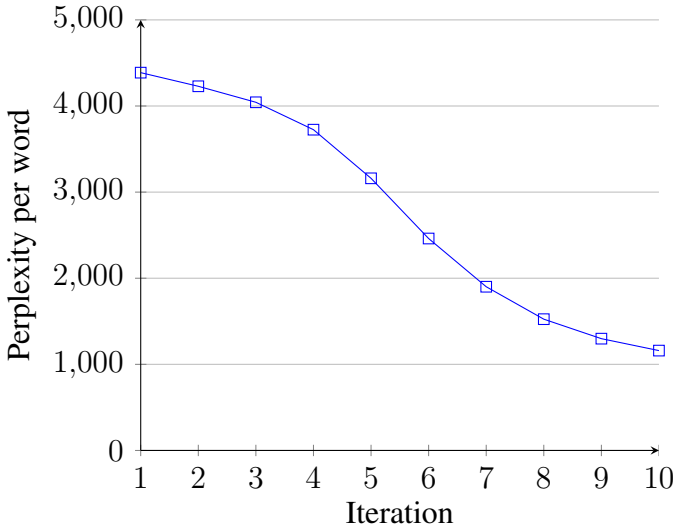


Figure 3.3: Perplexity on the training data

Table 3.2 shows that *Reordering Grammar* models outperform all baselines on this task. The only strange result here is that rule-based preordering obtains a lower score than no preordering, which might be an artifact of the Enju parser changing the tokenization of its input, so the Kendall τ of this system might not really reflect the real quality of the preordering. All other systems use the same tokenization.

	Kendall τ
A$^{No\ preordering}$	0.7655
B$^{Rule\ based}$	0.7567
C^{LADER}	0.8176
RG$_{\text{left-branching}}$	0.8201
RG$_{\text{right-branching}}$	0.8246
RG$_{\text{ITG-forest}}$	0.823
RG$_{\text{PET-forest}}$	0.8255

Table 3.2: Reordering prediction

3.4.2 Extrinsic evaluation in MT

The reordered output of all the mentioned baselines and versions of *Reordering Grammar* model are translated with phrase-based MT system (Koehn et al., 2007) (distortion limit set to 6 with distance based reordering model) that is trained on **gold preordering** of the training data pairs, which are the word re-aligned and then used for training the back-end MT system (Khalilov and Sima'an, 2011). Earlier work on preordering applies the preordering model to the training data to obtain a parallel corpus of *guessed* $\acute{\mathbf{s}} - \mathbf{t}$. This step is skipped in order to simplify training and save a good amount of

training time. This comes with a price of taking the risk of mismatch between the preordering and the back-end system. $\acute{\mathbf{s}} - \mathbf{t}$. The only exception is Baseline A which is trained on original $\mathbf{s} - \mathbf{t}$.

A 5-gram language model trained with KenLM [4] (Heafield et al., 2013) was used. Tuning was repeated three times with kb-mira (Cherry and Foster, 2012) to account for tuner instability and evaluated using Multeval [5] (Clark et al., 2011) for statistical significance on 3 metrics: BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2011) and TER (Snover et al., 2006). Additionally, RIBES scores (Isozaki et al., 2010a) are reported because RIBES concentrates on word order more than other metrics.

System	BLEU ↑	METEOR ↑	TER ↓	RIBES ↑
$A^{Nopreord.}$	27.8	48.9	59.2	68.29
$B^{Rule\,based}$	29.6	48.7	59.2	71.12
C^{LADER}	31.1	50.5	56.0	74.29
RG_{left}	31.2^{AB}	50.5^{AB}	56.3^{AB}_{C}	**74.45**
RG_{right}	31.4^{AB}	50.5^{AB}	56.3^{AB}_{C}	**75.29**
$RG_{ITG\text{-}forest}$	$\mathbf{31.6^{ABC}}$	$\mathbf{50.8^{ABC}}$	$\mathbf{55.7^{ABC}}$	**75.29**
$RG_{PET\text{-}forest}$	$\mathbf{32.0^{ABC}}$	$\mathbf{51.0^{ABC}}$	$\mathbf{55.7^{ABC}}$	**75.62**

Table 3.3: Comparison of different preordering models. Superscripts A, B and C signify if the system is significantly better ($p < 0.05$) than the respective baseline or significantly worse (in which case it is a subscript). Significance tests were not computed for RIBES. Score is bold if the system is significantly better than all the baselines.

Single or all PETs? In Table 3.3 we see that using *all PETs* during training makes a big impact on performance. Only the *all PETs* variants ($RG_{ITG\text{-}forest}$ and $RG_{PET\text{-}forest}$) significantly outperform all baselines. If we are to choose a *single PET* per training instance, then learning RG from only left-branching PETs (the one usually chosen in other work, e.g. (Saluja et al., 2014)) performs slightly worse than the right-branching PET. This is possibly because English is mostly right-branching. So even though both PETs describe the same reordering, RG_{right} captures reordering over English input better than RG_{left}.

All PETs or binary only? *$RG_{PET\text{-}forest}$ performs significantly better than $RG_{ITG\text{-}forest}$* ($p < 0.05$). Non-ITG reordering operators are predicted rarely (in only 99 sentences of the test set), but they make a difference, because these operators often appear high in the predicted PET. Furthermore, having these operators during training might allow for better fit to the data.

[4] http://kheafield.com/code/kenlm/

[5] https://github.com/jhclark/multeval

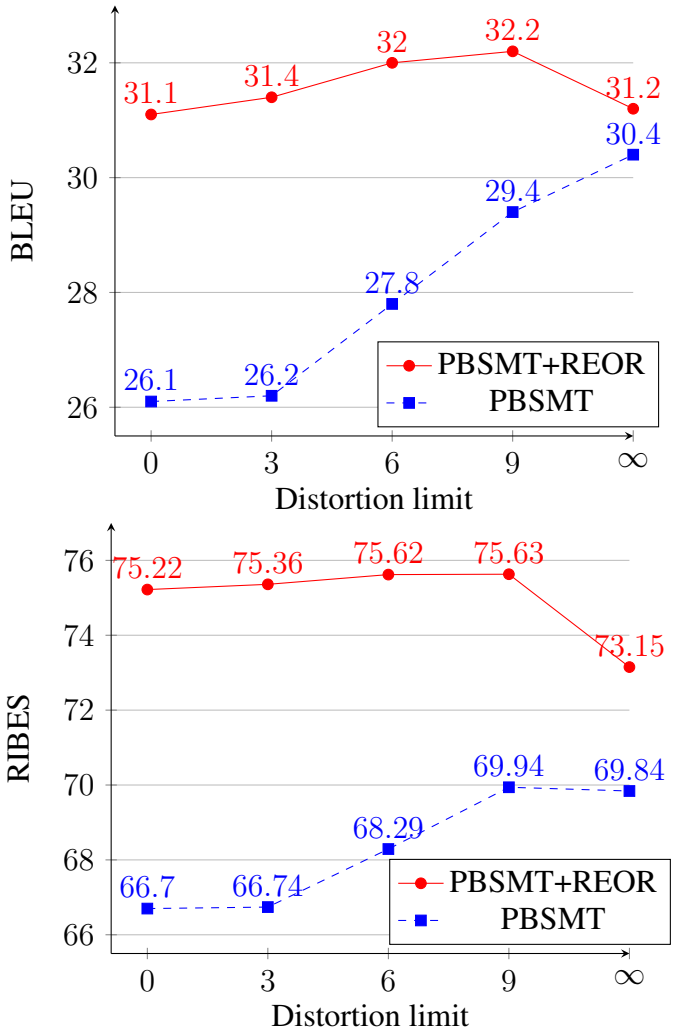


Figure 3.4: Distortion effect on BLEU and RIBES

How much reordering is resolved by the Reordering Grammar? Clearly, completely factorizing out the reordering from the translation process is impossible because reordering depends to a certain degree on target lexical choice. To quantify the contribution of Reordering Grammar, I tested decoding with different distortion limit values in the SMT system. The comparison is done with the phrase-based (PB) system with distance based cost function for reordering (Koehn et al., 2007) with and without preordering.

Would improvements stay if decoder has better search? Figure 3.4 shows that Reordering Grammar gives substantial performance improvements at all distortion limits (both BLEU and RIBES). $\text{RG}_{\text{PET-forest}}$ is less sensitive to changes in decoder distortion limit than standard PBSMT. The performance of $\text{RG}_{\text{PET-forest}}$ varies only by 1.1 BLEU points while standard PBSMT by 4.3 BLEU points. Some local reordering in

the decoder seems to help $\text{RG}_{\text{PET-forest}}$ but **large distortion limits seem to degrade the preordering choice**. This shows also that the improved performance of $\text{RG}_{\text{PET-forest}}$ is not only a result of efficiently exploring the full space of permutations, but also a result of improved scoring of permutations.

System	BLEU ↑	METEOR ↑	TER ↓	RIBES ↑
D^{PBMSD}	29.6	50.1	58.0	68.97
E^{Hiero}	32.6	52.1	54.5	74.12
$\text{RG}_{\text{PET-forest}}$+MSD	32.4^{D}	51.3^{D}_{E}	55.3^{D}_{E}	$\mathbf{75.72}$

Table 3.4: Comparison to MSD and Hiero

Does the improvement remain for a decoder with MSD reordering model? The $\text{RG}_{\text{PET-forest}}$ preordered model is compared against a decoder that uses the strong MSD model (Tillmann, 2004; Koehn et al., 2007). Table 3.4 shows that using Reordering Grammar as front-end to MSD reordering (full Moses) improves performance by 2.8 BLEU points. The improvement is confirmed by METEOR, TER and RIBES. The *Reordering Grammar* preordering model and MSD are complementary – the Reordering Grammar captures long distance reordering, while MSD possibly does better local reorderings, especially reorderings conditioned on the lexical part of translation units.

Interestingly, the MSD model (BLEU 29.6) improves over distance-based reordering (BLEU 27.8) by (BLEU 1.8), whereas the difference between these systems as back-ends to Reordering Grammar (respectively BLEU 32.4 and 32.0) is far smaller (0.4 BLEU). This suggests that a major share of reorderings can be handled well by preordering without conditioning on target lexical choice. Furthermore, this shows that $\text{RG}_{\text{PET-forest}}$ preordering is not very sensitive to the decoder's reordering model: both good and bad decoder give good results with preordering that comes from $\text{RG}_{\text{PET-forest}}$ model.

Comparison to a Hierarchical model (Hiero). Hierarchical preordering is not intended for a hierarchical model as Hiero (Chiang, 2005). Yet, here the preordering system (PB MSD+RG) is compared to Hiero for completeness, while we should keep in mind that Hiero's reordering model has access to much richer training data. These differences will be discussed shortly.

Table 3.4 shows that the difference in BLEU is not statistically significant, but there is more difference in METEOR and TER. RIBES, which concentrates more on reordering, prefers Reordering Grammar over Hiero. It is somewhat surprising that a preordering model combined with a phrase-based model succeeds to rival Hiero's performance on English-Japanese. Especially when looking at the differences between the two:

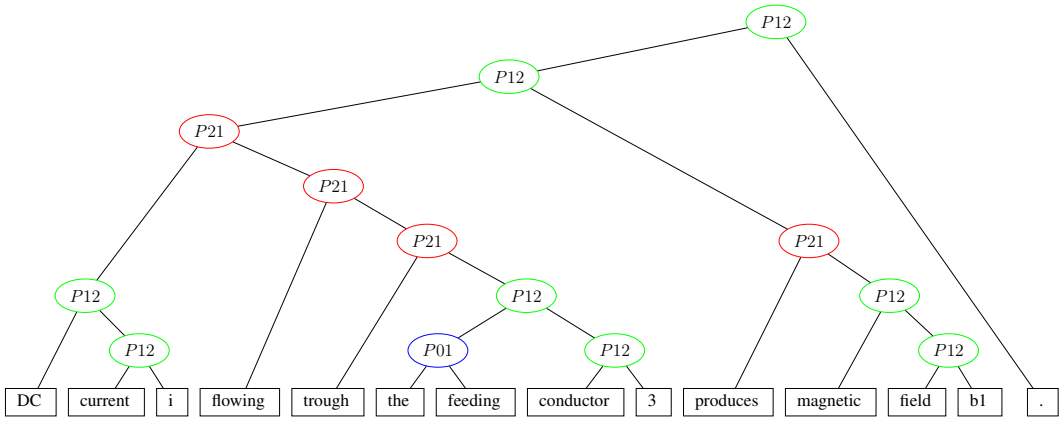


Figure 3.5: Example parser output for English sentence that predicts Japanese order

1. Reordering Grammar uses only minimal phrases, while Hiero uses composite (longer) phrases which encapsulate internal reorderings, but also non-contiguous phrases.
2. Hiero conditions its reordering on the lexical target side, whereas the Reordering Grammar does not (by definition).
3. Hiero uses a range of features, e.g., a language model, while Reordering Grammar is a mere generative PCFG.

The advantages of Hiero can be brought to bear upon Reordering Grammar by reformulating it as a discriminative model.

Do induced trees correspond to syntactic properties? Figure 3.5 shows an example PET output that demonstrates how *Reordering Grammar* model appears to have learned: (1) that the article "the" has no equivalent in Japanese, (2) that verbs go after their object, (3) to use postpositions instead of prepositions, and (4) to correctly group certain syntactic units, e.g. NPs and VPs.

3.5 Related work

The majority of work on preordering is based on syntactic parse trees, e.g., (Lerner and Petrov, 2013; Khalilov and Sima'an, 2011; Xia and Mccord, 2004). Here we concentrate on work that has common aspects with *Reordering Grammar* model. Neubig et al. (2012) trains a latent non-probabilistic discriminative model for preordering as an ITG-like grammar limited to binarizable permutations. Tromble and Eisner (2009) use ITG but do not train the grammar. They only use it to constrain the local search. DeNero and Uszkoreit (2011) present two separate consecutive steps for unsupervised induction of hierarchical structure (ITG) and the induction of a reordering function over it. In contrast, here the *Reordering Grammar* model learns both the structure and the reordering function simultaneously. Furthermore, at test time, the inference with

MBR over a measure of permutation (Kendall) allows exploiting both structure and reordering weights for inference, whereas test-time inference in (DeNero and Uszkoreit, 2011) is also a two step process – the parser forwards to the next stage the best parse.

Dyer and Resnik (2010) treat reordering as a latent variable and try to sum over all derivations that lead not only to the same reordering but also to the same translation. Although their work considers all permutations allowed by a given syntactic tree, it is also limited only to the derivations allowed by the syntactic tree.

Saers et al. (2012) induce synchronous grammar for translation by splitting the non-terminals, but unlike *Reordering Grammar* approach they split generic non-terminals and not operators. Their most expressive grammar covers only binarizable permutations. The decoder that uses this model does not try to sum over many derivations that have the same yield. They do not make independence assumption like the "unary trick" which is probably the reason they do not split more than 8 times. They do not compare their results to any other SMT system and test on a very small dataset.

Saluja et al. (2014) attempts inducing a refined Hiero grammar (latent synchronous CFG) from Normalized Decomposition Trees (NDT) (Zhang et al., 2008). While there are similarities with the *Reordering Grammar* , there are major differences. On the similarity side, NDTs are decomposing alignments in ways similar to PETs, and both Saluja's and *Reordering Grammar* models refine the labels on the nodes of these decompositions. However, the main differences between the two models are:

- the *Reordering Grammar* model is completely monolingual and unlexicalized (does not condition its reordering on the translation) in contrast with the Latent SCFG used in (Saluja et al., 2014),
- *Reordering Grammar* Latent PCFG label splits are defined as refinements of prime permutations, i.e., specifically designed for learning reordering, whereas (Saluja et al., 2014) aims at learning label splitting that helps predicting NDTs from source sentences,
- the *Reordering Grammar* model exploits **all PETs and all derivations**, both during training (latent treebank) and during inference. In (Saluja et al., 2014) only left branching NDT derivations are used for learning the model.
- The training data used by (Saluja et al., 2014) is about 60 times smaller in number of words than the data used here; the test set of (Saluja et al., 2014) also consists of far shorter sentences where reordering could be less crucial.

A related work with a similar intuition is presented in (Maillette de Buy Wenniger and Sima'an, 2014), where nodes of a tree structure similar to PETs are labeled with reordering patterns obtained by factorizing word alignments into Hierarchical Alignment Trees. These patterns are used for labeling the standard Hiero grammar. Unlike this work, the labels extracted by (Maillette de Buy Wenniger and Sima'an, 2014) are clustered manually into less than a dozen labels without the possibility of fitting the labels to the training data.

3.6 Propagating Uncertainty to the Decoder

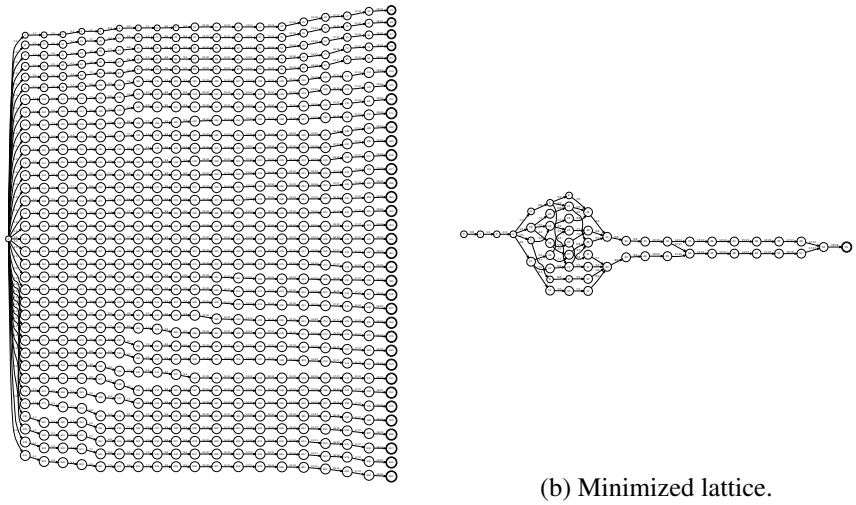
Preordering is tightly associated with the pipeline approach where we first predict one reordering and then MT decoder translates monotonically that one ordering of the source sentence. As any pipeline approach, even the best preordering models suffer from propagation of error: if the one best predicted reordering is not good then even the best decoder will not be able to recover from it and potentially introduce even bigger errors.

Preordering models can be seen as methods for narrowing down the search space of possible word orders. In the most extreme (and most frequent) case the reordering search space is narrowed to only one reordering. However, if we could somehow give to the decoder a bigger choice of possible reorderings then even if some of the reorderings are wrong as long as there is a good reordering in there the decoder will have a chance to get a good translation in the end.

One of the ways how the decoder's search space can be narrowed down to several reorderings is by making the decoder translate a lattice of words where each path in the lattice corresponds to a possible word order. A lattice can encode an exponential number of reorderings in a compact data structure. Compared to the translation of a single reordering this gives the MT decoder much more freedom to search during translation, but still this search will be only in the space that was licensed by the preordering models as a space that contains most promising reorderings. Most MT phrase based decoders implement the lattice translation algorithm described in (Dyer et al., 2008).

Now the question is how to extract a lattice from the weighted forest of Permutation Trees. Doing that exactly is impossible in polynomial time: a forest of Permutation Trees is essentially a hypergraph which is a strictly more powerful structure which can encode permutations that would require an exponentially big lattice. We can avoid conversion to lattice altogether and directly translate a hypergraph which has been done before (Dyer and Resnik, 2010) but this is a feature that is not present in most of the decoders and it is relatively slow. Instead we are going to settle for the relatively small n-best list of possible reorderings that will be given to the decoder.

This n-best list is extracted in a similar way to the way one-best preordering was extracted before: 10,000 derivations of permutation trees are sampled from the chart and from there we select n permutations with the lowest Kendall τ Bayes risk. In principle each of these permutations can be translated individually and then the highest scoring translation from all of them can be selected. However, that would be a very slow method of combining the n-best lists. In the experiments with Reordering Grammar 100-best permutations were used which would in this naive approach mean 100 times slower translation which is not acceptable. Many of these permutations share some ordering decisions which allows us to pack them efficiently into a lattice. By translating this packed lattice the decoder would not need to redo the work of translating shared parts among the permutations. The worst case (no sharing substructure among permutations) stays the same, but because in practice large number of permutations are similar the slowdown is roughly only 2 to 3 times.



(a) Linear form.

(b) Minimized lattice.

Figure 3.6: Example permutation lattice.

The process of conversion of an n-best list of reorderings into a lattice of reorderings consists of two steps. In the first step a big lattice is formed which contains n paths and each path corresponds to one entry in the n-best list. All paths start from the same *start* node and all end at different *final* nodes. An example of this lattice is shown in Figure 3.6a. In this second step this lattice is minimized using existing efficient algorithms for this task. In this work the OpenFST toolkit [6] was used for this purpose (Allauzen et al., 2007). An example of the minimized lattice is shown in Figure 3.6b. These figures give a good intuition about how much compact can lattices be as a representation of different reorderings.

		Translation	Word order
	DL	BLEU	Kendall τ
Baseline	6	29.65	44.87
Oracle order	6	34.22	56.23
	0	30.55	53.98
One-best	6	32.14[A]	49.68
Lattice	0	32.50[AB]	50.79

[A]Stat. significant against baseline. [B]Stat. significant against first-best.

Table 3.5: Lattice translation results for English–Japanese.

The experiments are conducted on the same English-Japanese dataset and with the same MSD decoding setting as in the previous sections. Table 3.5 shows the results

[6] `http://www.openfst.org/`

for different input and distortion limits. We can see here that the model that takes lattice as an input significantly outperforms both the no-preordering baseline and the one-best preordering even in the case where baselines and one-best preordering were allowed to use distortion limit of 6. The table also shows what would be the results if the optimal (oracle) word order was given to the decoder. Interestingly, even in that case it is beneficial to allow the decoder to do reordering. A likely explanation could be that there are alternative word orders that are easier for the decoder to translate which justifies even more an approach in which several reorderings would be given to the decoder.

3.7 Future extensions of PET preordering model

There are aspects of PET preordering model that could be improved in the future. One of them is making the model feature rich. Although LADER (Neubig et al., 2012) is a feature rich preordering model, it is not probabilistic. PET model can add support for features without losing the probabilistic interpretation by moving to a CRF based parsing model with latent variables (Petrov and Klein, 2008). These features do not need to be handcrafted; instead, they could be induced from data similar to Neural-CRF models (Durrett and Klein, 2015).

In addition to the learning model, the parsing algorithm could also be improved. PETs chart parsing has $O(n^3)$ time complexity, which is in some cases considered computationally expensive. Speed of the parser could be improved significantly by replacing chart-based parser with a transition based-parser, for example, like the parser of Nakagawa (2015) that runs in $O(n)$.

The possible line to explore would be how to integrate reordering predictions into modern MT translation systems based on neural networks (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015). These models are often based on *sequence-to-sequence* architecture where source sentence is treated as a sequence and is encoded into a vector representation. This representation is then decoded into a target sentence which is also treated as a sequence. In Section 3.6 I have presented a method for integration which instead of sequence of source words gives to decoder a lattice of possible reorderings of source sentence. This same intuition can be applied to neural models which could be *lattice-to-sequence* models instead of *sequence-to-sequence* (Su et al., 2017). Another even simpler way of integration was presented by Du and Way (2017) who show that integrating preordering predictions into neural MT decoders can be done by providing preordering predictions as an additional factor in a factor-based neural MT model (Sennrich and Haddow, 2016).

3.8 Conclusion

In this chapter a generative Reordering PCFG model learned from latent treebanks over PETs was presented. Reordering PCFG handles non-ITG reordering patterns (up to 5-ary branching) and it works with all PETs that factorize a permutation (rather than a single PET).

To the best of my knowledge this is the first time both extensions are shown to improve MT performance. The empirical results on English-Japanese show that (1) when used for preordering, the Reordering PCFG helps particularly with relieving the phrase-based model from long range reorderings, (2) combined with a state-of-the-art phrase model, Reordering PCFG shows performance not too different from Hiero, supporting the common wisdom of factorizing long range reordering outside the decoder, (3) Reordering PCFG generates derivations that seem to coincide well with linguistically-motivated reordering patterns for English-Japanese. Although recovering linguistically-motivated reordering patterns was not the goal of this work, it would be very interesting to explore more formally to which extent this happens and in which cases it happens (if some linguistic constructions are easier to automatically induce than others). This is left for the future work.

Finally, the method for propagating uncertainty of preordering to the decoder was presented that allows the decoder to disambiguate among several offered reorderings. This is beneficial because at the decoding time more information is available than at the preordering time so importance of different preordering predictions could more accurately estimated.

The experiments where permutation factorization into PETs was used showed favorable results. The next chapter will show that PETs can be used not only for prediction but also for evaluation of the word order.

Chapter 4

Evaluating Word Order with Permutation Trees

This chapter shows how permutation trees can be used for evaluation of word order. This is done by interpreting the word order of the system translation as a permutation of the words of the reference translation, and then measuring the distance between that permutation and the *ideal* monotone permutation. The distance measures over permutations can be defined in many ways. The previous measures, such as Kendall τ (Birch et al., 2010), Spearman ρ (Isozaki et al., 2010a), FuzzyScore (Talbot et al., 2011) etc., do not exploit the hierarchical structure inherently present in the permutation. This is where the content of this chapter differs from the earlier work. In the first part of the chapter, I present how simple (non-recursive) counts over a recursive tree structure, namely permutation trees (PETs), can be a good measure of word order. The second part of the chapter shows how a slightly more complex recursively defined measure over PETs can be used for the evaluation.

The first part of the chapter is based on the following publication:

authors: Stanojević and Sima'an
title: *Hierarchical Permutation Complexity for Word Order Evaluation*
venue: COLING 2016

All the research and all implementations incorporated in the thesis were carried out by me. Khalil Sima'an has provided guidance and edited the publication. Khalil Sima'an also provided tightness principle for the design of the evaluation metrics that is reported in the paper but is not included in this thesis.

The second part of the publication is based on the following publication:

authors: Stanojević and Sima'an
title: *Evaluating Word Order Recursively over Permutation-Forests*
venue: SSST 2014

All the research and all implementations were carried out by me. Khalil Sima'an has provided guidance and edited the publication.

Chapter Highlights

Problem Statement

- Existing evaluation metrics for evaluation of word order treat permutations as a flat sequence with no inner hierarchical structure. This view of permutation as a flat structure is too simplistic and prevents from extracting more abstract and more relevant reordering patterns present in the data.

Research Question

- Can a hierarchical view of permutations provide a better evaluation metric?
- If the answer to the previous question is positive, then what are the properties of this structure that should be considered in the evaluation? In other words, what and how should be counted in the hierarchical structure?

Research Contributions

- Permutation Trees (Zhang and Gildea, 2007) are proposed as a better view of the data, one that reveals more directly long distance reordering patterns.
- Non-recursive metrics are proposed that capture the degree of factorization of the permutation. Results show that the degree of factorization correlates with human judgment of translation quality.
- Recursive measures over permutation trees and permutation forests are proposed. Results show that, just like in the previous chapter on preordering, using a forest of trees is better than using a single arbitrary permutation tree for evaluating word order.

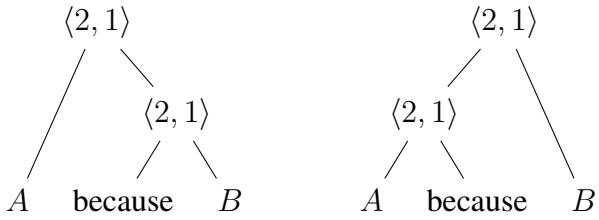
4.1 Introduction

Evaluating word order (also reordering) in MT is one of the main ingredients in automatic MT evaluation, e.g., (Papineni et al., 2002; Denkowski and Lavie, 2011). To monitor progress on evaluating reordering, recent work explores dedicated reordering evaluation metrics, cf. (Birch and Osborne, 2011; Isozaki et al., 2010a; Talbot et al., 2011). Existing work computes the correlation between the ranking of the outputs of different systems: one ranking of systems produced by evaluation metric and the other produced by human judgment , on e.g., the WMT evaluation data.

For evaluating reordering, it is necessary to word align system output with the corresponding reference translation. For convenience, a 1:1 alignment (a permutation) is induced between the words on both sides (Birch and Osborne, 2011), possibly leaving words unaligned on either side. The details about this conversion from many-to-many alignments to 1-to-1 mapping given by a permutation is shown in Section 2.5.2.

Existing work then concentrates on defining measures of reordering over permutations, cf. (Lapata, 2006; Birch and Osborne, 2011; Isozaki et al., 2010a; Talbot et al., 2011). Popular metrics over permutations are: Kendall's τ, Spearman ρ, Hamming distance, Ulam and Fuzzy score. These metrics treat a permutation as a flat sequence of integers or blocks, disregarding the possibility of hierarchical grouping into phrase-like units, making it difficult to measure long-range order divergence. As exemplified in previous chapters, by using permutation trees as the representation some very complex long range reordering can in fact be a simple operation over few nodes in a PET.

Isozaki et al. (2010a) argue that the conventional metrics cannot measure well long distance reordering between an English reference sentence "A because B" and a Japanese-English hypothesis translation "B because A", where A and B are blocks of any length. In this chapter the idea of factorizing permutations into permutation-trees (PETs) (Gildea et al., 2006) is explored and new tree-based reordering metrics are defined over them which aim at dealing precisely with this type of long range reorderings. For the Isozaki et al. (2010a) Japanese-English example, there are two PETs (when leaving A and B as encapsulated blocks):



PETs give exactly the right view for this example: they show that this wrong word order is a result of wrong ordering of big blocks and not of individual words. This is the same intuition that existed in earlier metrics, such as TER (Snover et al., 2006), but unlike those metrics PET based metrics exploit the compositional structure of the blocks: a block might contain other blocks that might contain some other blocks.

In this example, all versions of PET-based metrics interpolate the scores over the two inversion operators $\langle 2, 1 \rangle$ with the internal scores for A and B. If both A and B are large blocks, internally monotonically (also known as straight) aligned, then a PET based measure will not count every single reordering of a word in A or B, but will consider this case as block reordering. From a PET perspective, the distance of the reordering is far smaller than when looking at a flat permutation. But does this hierarchical view of reordering cohere better with human judgment than string-based metrics?

The example above also shows that a permutation may factorize into different

PETs, each corresponding to a different segmentation of a sentence pair into phrase-pairs. In previous chapters it was presented how all of these trees can be compactly encoded into permutation forests (PEFs).

This chapter explores evaluation of word order alone as the only aspect of translation and largely ignores the lexical part of translation. It is compared on the same ground with the other metrics that are evaluating word order only and shows higher correlation with human judgment. However, this is not the only application of these results. They can be incorporated as a word order component of a full evaluation metric as will be shown in Chapter 5.

This chapter will not present how word order can be viewed as a permutation distance since Section 2.5.2 shows that. In the following sections, first the baseline "flat" evaluation metrics are presented. That is followed by Section 4.3 where some simple metrics over permutation trees are presented and meta-evaluated. Section 4.4 presents more sophisticated recursive measures over permutation trees.

4.2 Baselines – Flat Metrics over Permutations

In (Birch and Osborne, 2010, 2011) Kendall τ and Hamming distance are combined with unigram BLEU (BLEU-1) leading to LRscore showing better correlation with human judgment than BLEU-4. Birch et al. (2010) additionally tests Ulam distance (longest common subsequence – LCS – normalized by the permutation length) and the square root of Kendall's τ. Isozaki et al. (2010a) present a similar approach to (Birch and Osborne, 2011) additionally testing Spearman ρ as a distance measure. Talbot et al. (2011) extract a reordering measure from METEOR (Denkowski and Lavie, 2011) dubbed *Fuzzy Reordering Score* and evaluates it on MT reordering quality.

Evaluation metrics usually have the interpretation – the higher the better – meaning that if permutation A gets higher score than some other permutation B then permutation A has a higher quality. Also, evaluation metrics are usually expected to give a score in the range $[0, 1]$. Yet, functions like Kendall's τ and Spearman's ρ are defined on the range $[-1, 1]$, while Hamming is a distance function which gives higher score to the worse (more distant) permutations. Hence, the existing permutation measures are not directly usable but should be adapted in order to satisfy the mentioned conditions without decreasing their discriminative power. Bellow are the adapted definitions of the *baseline metrics* that are used in the experiments.

A permutation over $[1..n]$ (subrange of the positive integers where $n > 1$) is a bijective function from $[1..n]$ to itself. To represent permutations I will use angle brackets as in $\langle 2, 4, 3, 1\rangle$. Given a permutation π over $[1..n]$, the notation π_i ($1 \leq i \leq n$) stands for the integer in the i^{th} position in π; $\pi(i)$ stands for the index of the position in π where integer i appears; and π_i^j stands for the (contiguous) sub-sequence of integers $\pi_i, \ldots \pi_j$.

The definitions of five commonly used metrics over permutations are shown in Figure 4.1. In these definitions, I use LCS to stand for Longest Common Subsequence,

and Kronecker $\delta[a]$ which is 1 if $(a == true)$ else zero, and $\text{ID}_1^n = \langle 1, \cdots, n \rangle$ which is the identity permutation over $[1..n]$.

They can be intuitively interpreted in the following way. Kendall τ measures the proportion of skip-bigrams of numbers that are matched between the observed and the ideal monotone permutation. Hamming distance measures the proportion of indices in the permutation that are at the same position as in the monotone permutation. Spearman ρ has similar intuition like Kendall τ but a different scoring function. Ulam distance is just a normalized longest common subsequence in order to make the score between 0 and 1. Fuzzy score is sometimes intuitively explained as a penalty for the number of jumps that eye needs to make in order to read the sentence in the correct order. The number of monotone chunks c is essentially also the number of eye jumps.

$$
\begin{aligned}
\text{KENDALL}(\pi) &= \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \delta[\pi(i) < \pi(j)]}{(n^2 - n)/2} \\
\text{HAMMING}(\pi) &= \frac{\sum_{i=1}^{n} \delta[\pi_i == i]}{n} \\
\text{SPEARMAN}(\pi) &= 1 - \frac{3 \sum_{i=1}^{n} (\pi_i - i)^2}{n(n^2 - 1)} \\
\text{ULAM}(\pi) &= \frac{LCS(\pi, \text{ID}_1^n) - 1}{n - 1} \\
\text{FUZZY}(\pi) &= 1 - \frac{c - 1}{n - 1}
\end{aligned}
$$

where c is # of monotone sub-permutations

Figure 4.1: Five commonly used metrics over permutations

$$
\begin{aligned}
|\text{PET}|(\pi) &= \frac{\text{COUNT}_{\text{node}}(\text{PET}(\pi)) - 1}{n - 2} \\
\text{MAX}_{|\text{Op}|}(\pi) &= 1 - \frac{\text{MaxOp}(\text{PET}(\pi)) - 2}{n - 2} \\
\#\text{PETS}(\pi) &= \frac{\text{COUNT}_{\text{pet}}(\text{PEF}(\pi)) - 1}{\text{COUNT}_{\text{pet}}(\text{PEF}(\text{ID}^n)) - 1}
\end{aligned}
$$

Figure 4.2: Summary of metrics: $\text{COUNT}_{\text{node}}$ is number of nodes in $PET(\pi)$; $\text{MaxOp}(\text{PET})$ is maximum operator length in PET; $\text{COUNT}_{\text{pet}}(\text{PEF})$ returns count of PETs in PEF.

4.3 Simple Metrics over Recursive Structures

All the previously defined permutation metrics are *flat* in the sense that they do not generalize over blocks that could be reordered jointly. Since permutation trees give exactly this view of permutation where permutation is recursively split into atomic blocks we can build many different metrics that would exploit the PET structure.

One of the simplest metrics that could be made over permutation trees is just counting the number of nodes in the permutation tree. The function $|\text{PET}|(\cdot)$ in Figure 4.2 represents the normalized version of this metric. The reason why we could expect this metric to work well is a simple intuition that simpler permutations (for example monotone permutations) should factorize more and that these simpler permutations should be preferred. Binarizable permutations (Wu, 1997) would be the simplest form while more complex PETs would be difficult to reorder which this metric reflects in its score.

Similar intuition is used in $\text{MAX}_{|\text{Op}|}(\cdot)$ metric except that this metric pays attention only to the *worst* node in the PET – the node which has the maximal number of children. The more children it has the more complicated it is to reorder that system translation into the reference word order.

Another measure of how much does the permutation factorize is not only in what kind of PET it factorizes but also in how many PETs it can factorize. For instance, the monotone permutation would factorize into a maximal number of PETs that is equivalent to the $Cat(|\pi| - 1)$ where $Cat(\cdot)$ is a Catalan number. Some permutations that that factorize into an ITG tree do not necessarily have a high number of trees that could explain them. If, for example, monotone and inverted nodes are intertwined in the tree then only a very small number of PETs can be built. The $\text{COUNT}_{\text{pet}}(PEF)$ function rewards permutations that factorize into more PETs. Even though this function is defined over permutation forests it is not necessary to build the forest to compute this function. The number of PETs can be predicted from a single *flat* PET as it was shown in Section 2.5.3.

4.3.1 Interpolation with Lexical Score

All the metrics discussed so far are similarity measures over permutations. Unfortunately, human judgments do not come in the form of judgments of permutations. Most of the human judgments that are available are rankings of the full machine translation outputs where human evaluators used their knowledge both of lexical and word order properties of language.

In order to compare word order metrics on this data, an interpolation of a reordering metric with the lexical component is introduced. All of the word order metrics are interpolated with the same lexical component in order to make comparison between them fair. Even though the final correlation with human judgment will depend both on lexical and ordering component, the difference between correlations will be mostly due to the ordering component because the lexical component is held constant as well as the interpolation weight.

If a human judge ranks translation A as better than translation B, and translation A has much higher lexical accuracy than translation B then we can expect that the lexical component will take more importance in making the ranking decision. This prevents ordering metrics to be compared on these examples. The lexical metric that is used is the bag of words F1 score of matched unigrams.

An additional problem, similar to the lexical matching problem, is that permutations might be too short if lexical matching fails. If for a 20 word long sentence only two words match and they happen to be in the correct order then ordering component would give a signal that this is a very good translation. To prevent this a brevity penalty is used. This brevity penalty is essentially the same as in BLEU score except that here instead of taking length of the system and the reference translation as its parameters, it takes the length of the system permutation and length of the reference. The ordering score, before it enters the interpolation, is multiplied with the brevity penalty as in the equation bellow.

$$\begin{aligned}\mathrm{SentScore}(\mathrm{ref},\mathrm{sys}) =& \alpha \times \mathrm{F1}(\mathrm{ref},\mathrm{sys}) + \\ & (1-\alpha) \times \mathrm{BP}(|\pi|, |\,\mathrm{ref}\,|) \times \mathrm{ordering}(\pi)\end{aligned} \tag{4.1}$$

The interpolation parameter was fixed $\alpha = 0.5$, weighing both lexical and reordering metrics equally, to avoid introducing preference for one over the other, but in principle this could be tuned on human rankings.

A system level score is computed by aggregating the sentence level scores. Sentence level scores are weighted by the reference length because the aim is the evaluation of long distance reordering so longer sentences should get more importance. The score is normalized in order to be between 0 and 1:

$$\mathrm{CorpScore}(\mathcal{S}) = \frac{\sum_{(\mathrm{ref},\mathrm{sys})\in\mathcal{C}_\mathcal{S}} |\,\mathrm{ref}\,| \times \mathrm{SentScore}(\mathrm{ref},\mathrm{sys})}{\sum_{(\mathrm{ref},\mathrm{sys})\in\mathcal{C}_\mathcal{S}} |\,\mathrm{ref}\,|} \tag{4.2}$$

4.3.2 Experimental Setting

Now that the evaluation metrics are defined over standard translation outputs, the same process of meta-evaluation can be used as in the WMT metric tasks (Macháček and Bojar, 2013). The meta-evaluation was conducted on the human judgments from WMT13 translation task (Bojar et al., 2013) over ten language pairs with a diverse set of MT systems.

All the MT systems were first ranked by human judgments using the ratio of the times they were judged to be better than some other system. Additionally, the systems got ranked by each of the tested evaluation metrics by using their system level score. The rankings produced by each evaluation metric are compared against the rankings produced by human judges in terms of Spearman rank correlation:

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)} \tag{4.3}$$

where $d_i = y_i - x_i$ represents a distance in ranks given by humans (y_i) and the metric (x_i) for system i.

In order to compute statistical significance bootstrap resampling was used with 1000 samples on which t-test was conducted. The metric pairs for which $p < 0.05$ are considered significantly different.

4.3.3 Results

The scores for translation into-English are shown in Table 4.2. Table 4.1 shows the results for the out-of-English direction. BLEU-Moses score is also included straight from WMT13 tables for an impression regarding a known full metric. Metrics RECPET and RECPEF will be explained in the next section. The permutation tree based metrics outperform the baselines on six language pairs (English into Czech/ Russian/ Spanish/ German, and out of Russian and German). But the baselines prevail on four language pairs (English-French, Czech-English, French-English and Spanish-English) out of which three do not have long distance reorderings which probably allowed baseline metrics to perform well.

For English-Russian and English-Czech, #PETs (bracketing freedom) is superior, likely because Russian and Czech allow freer order than English which is difficult for MT systems to capture. #PETs evaluates more the closeness of the blocks than their specific reordering. For instance completely monotone and completely inverted permutations are both equally good with respect to how close are the words. They differ only in the orientation of where the words end up but maybe that is not that important for languages with a freer word order. For these languages it is more important that predicates and arguments are close by then the order in which they appear.

English-Russian shows low correlations for all metrics (including BLEU), suggesting that either all systems participating are judged of lower quality, or that human judgments are less consistent.

Analysing results of eight metrics over ten language pairs is difficult. A more condensed view is shown in Table 4.3. Table 4.3 shows for each new metric $\mathcal{N}$ and baseline $\mathcal{B}$ a ratio $N/B/D$ where N is the number of language-pairs where statistically significant improvement by $\mathcal{N}$ over $\mathcal{B}$ is found, B is the reverse situation and D is the number of draws (insignificant difference).

Table 4.3 shows clearly that |PET| performs more often than not better than each of the baselines. $\text{MAX}_{|\text{Op}|}$ concerns factorizability and performs as well as FUZZY outperforming the other baselines. #PETs concerns bracketing freedom and performs worse than many baselines, suggesting that for most language pairs bracketing freedom, which does not always favor more factorization, is not sufficient.

These results exemplify that factorizing word order mismatch might have higher chance of correlating with human evaluation than the baselines. The permutation based metrics tested here are simple examples that illustrate the general usefulness of factorization of permutations in word order evaluation. More effective variants would do more justice to the complexity of primal permutations. Different metrics that were

Metric \ Target lang.	Czech	Russian	French	Spanish	German
HAMMING	0.868	0.511	0.911	0.806	0.851
KENDALL	0.849	0.511	0.907	0.844	0.918
SPEARMAN	0.852	0.508	0.907	0.848	0.915
FUZZY	0.854	0.498	**0.920**	0.818	0.897
ULAM	0.851	0.507	0.914	0.844	0.908
|PET|	0.853	0.515	0.907	**0.866**	0.923
#PETS	**0.879**	**0.538**	0.904	0.797	0.819
$\text{MAX}_{\text{\|Op\|}}$	0.849	0.513	0.907	0.864	**0.924**
RECPET	0.855	0.505	0.905	0.863	0.911
RECPEF	0.856	0.505	0.906	0.855	0.907
BLEU	*0.895*	*0.574*	*0.897*	*0.759*	*0.786*

Table 4.1: Correlation with human judgement out of English.

Metric \ Source lang.	Czech	Russian	French	Spanish	German
HAMMING	0.878	0.761	**0.984**	0.880	0.851
KENDALL	0.887	0.831	0.969	0.831	0.905
SPEARMAN	0.881	0.831	0.967	0.826	0.905
FUZZY	**0.931**	0.810	0.977	**0.889**	0.894
ULAM	0.909	0.830	0.974	0.860	0.895
|PET|	0.895	**0.839**	0.965	0.818	0.918
#PETS	0.878	0.698	0.959	0.883	0.786
$\text{MAX}_{\text{\|Op\|}}$	0.895	0.838	0.966	0.819	**0.921**
RECPET	0.872	0.834	0.963	0.826	0.902
RECPEF	0.878	0.834	0.968	0.831	0.896
BLEU	*0.936*	*0.651*	*0.993*	*0.879*	*0.902*

Table 4.2: Correlation with human judgement into English.

	HAMMING	KENDALL	SPEARMAN	FUZZY	ULAM
|PET|	**5/4/1**	**7/2/1**	**6/2/2**	**5/4/1**	**5/4/1**
$\text{MAX}_{\text{\|Op\|}}$	**5/4/1**	**5/2/3**	**6/2/2**	5/5/0	**5/4/1**
#PETS	2/6/2	3/7/0	3/7/0	2/8/0	3/7/0

Table 4.3: Pairs-wise comparison over 10 language pairs. In the triple $N/B/D$: N is number of language pairs where the new metric significantly outperforms the baseline, B is baseline outperforms new metric and D is the number of language pairs where the difference is insignificant (draw). Bold show the cases where $N > B$.

tested over permutation trees cover different dimensions of permutation complexity: factorizability and grouping. The results show that the importance of a dimension depends on the language pair, but that overall PETs can be an effective structure for designing evaluation metrics.

4.4 Recursive Metrics over Recursive Structures

Since PETs are recursive tree structures it comes natural to define functions, for example evaluation functions, over them in a recursive way. The evaluation functions defined in this section RECPET and RECPEF are inspired by scoring algorithms used in PCFG parsing. Namely, RECPET can be thought of as a scoring function that scores a single permutation tree recursively in a similar way to Viterbi algorithm[1], while RECPEF considers the whole permutation forest in determining the score in a similar fashion to Inside algorithm. The main goal of these experiments, other than creating a good evaluation metric, is to find out if using all PETs for computing the score is better than using a single canonical PET.

4.4.1 Formal Specification of Permutation Forests

Permutation Forests (PEFs) were defined in Section 2.5.3. The purpose of this section is only to define the notation that will be used for defining the recursive evaluation metrics over PETs.

A **permutation forest** (akin to a parse forest) $\mathcal{F}$ for permutation π (over $[1..n]$) is a data structure consisting of a subset of $\{[[i, j, \mathcal{I}_i^j, op_i^j]] \mid 0 \leq i \leq j \leq n\}$, where $\mathcal{I}_i^j$ is a (possibly empty) set of *inferences* (sets of split points) that as a fringe contains π_{i+1}^j and op_i^j is an operator (prime permutation) shared by all inferences of π_{i+1}^j. Given an inference $\eta \in \mathcal{I}_i^j$ with arity (number of children) a where $\eta = [l_1, \ldots, l_{a-1}]$, the notation η_x will be used to refer to split point l_x in η where $1 \leq x \leq (a-1)$, with the convenient boundary assumption that $l_0 = i$ and $l_a = j$.

Note the difference between this forest representation and the forest representation used in treebank parsing: here a non-terminal label (operator) is shared across all the inferences that have the same span, while in treebank parsing there may be many different labels for the same span. This is safe to do thanks to the properties of permutation trees that guarantee only one prime permutation per span (Zhang et al., 2008).

If π_{i+1}^j is a sub-permutation and it has arity $a \leq (j - (i+1))$, then each inference consists of a $a-1$-tuple $[l_1, \ldots, l_{a-1}]$, where for each $1 \leq x \leq (a-1)$, l_x is a *split point* which is given by the index of the last integer in the x^{th} sub-permutation in π. The permutation of the a sub-permutations (*children* of π_{i+1}^j) is stored in op_i^j and it is the same for all inferences of that span.

[1] Viterbi algorithm is used here only as an analogy. No Viterbi decoding is conducted in the computation of the score

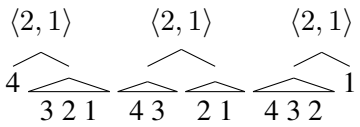


Figure 4.3: The factorizations of $\pi = \langle 4, 3, 2, 1 \rangle$.

As an example, consider the inferences for permutation $\pi = \langle 4, 3, 2, 1 \rangle$ (see Figure 4.3) which factorizes into pairs of sub-permutations ($a = 2$): a split point can be at positions with index $l_1 \in \{1, 2, 3\}$. Each of these split points (factorizations) of π will be represented as an *inference* for the *same root node* which covers the whole of π (placed in entry $[0, 4]$); the operator of the inference here consists of the permutation $\langle 2, 1 \rangle$ (swapping the two ranges covered by the children sub-permutations) and inference consists of $a - 1$ indexes $l_1, \ldots, l_{a-1}$ signifying the split points of π into sub-permutations: since $a = 2$ for π, then a single index $l_1 \in \{1, 2, 3\}$ is stored with every inference. For the factorization $((4, 3), (2, 1))$ the index $l_1 = 2$ signifying that the second position is a split point into $\langle 4, 3 \rangle$ (stored in entry $[0, 2]$) and $\langle 2, 1 \rangle$ (stored in entry $[2, 4]$). For the other factorizations of π similar inferences are stored in the permutation forest.

Recursive Evaluation Metric

I will start by describing the recursive evaluation functions first by defining the forest version because the single tree version can afterwords be interpreted as a special case of the forest version. Figure 4.4 shows the equations that define the metrics.

One of the intuitions that the metric should reflect is the "goodness" of the operators (prime permutation) on each node. For this a $\mathrm{opScore}$ function is defined which rewards the good prime permutation (monotone permutation $\langle 1, 2 \rangle$) and gives score 0 to all other prime permutations. Naturally, a more complex function could be defined, such as any of the *flat* metrics defined in previous sections, but here only this simple form is considered.

Function $\mathrm{opScore}$ scores only the operator. The node in the tree does not consist only of its label but also of its children – its subpermutations. To include this in the metric the function ϕ_{inf} is defined which scores inference (a branch in the PET) by taking the average of the ϕ_{span} scores (to be defined next) of the spans of its children. Only children with more than one word participate in this average because children with one word are not informative of the word order.

Function ϕ_{span} computes the score of the whole span by taking into consideration all inferences that participate in that span and the operator for the given span. If there are no inferences for the given span (for example if the span covers only one word) a score 1 is returned signifying that everything in that span is "in order". If the span as all of its children contains only words (if the arity of the operator is the same as the size of the span) then only the operator score should be returned because scores of the children are not informative. Finally, if all of these conditions are not fulfilled then the

function ϕ_{span} returns an interpolation of the operators score and of the average score of all the inferences in that span. The interpolation is controlled by the parameter β. If parameter β is set to a higher value then the nodes higher in the tree will have more importance.

Given these functions now it is easy to define RECPET and RECPEF. The forest score RECPEF of permutation π is a ϕ_{span} of the span that covers fully permutation and is applied over the full forest of all possible PETs for permutation π as given by function $PEF(\pi)$. The single PET score is computed in the same way except that the "forest" in this case contains only one tree: the canonical left-branching permutation tree.

$$\begin{aligned}
\text{RECPEF}(\pi) &= \phi_{span}(0, n, \text{PEF}(\pi)) \\
\text{RECPET}(\pi) &= \phi_{span}(0, n, \{\text{CanonicalPET}(\pi)\})
\end{aligned}$$

$$\phi_{span}(i, j, \mathcal{F}) = \begin{cases} \text{if } (\mathcal{I}_i^j == \emptyset) \text{ then } 1 \\ \text{else if } (\mathbf{a}(\pi_{i+1}^j) = j - i) \text{ then } \text{opScore}(op_i^j) \\ \text{else } \beta \times \text{opScore}(op_i^j) + (1 - \beta) \times \underbrace{\frac{\sum_{\eta \in \mathcal{I}_i^j} \phi_{inf}(\eta, \mathcal{F}, \mathbf{a}(\pi_{i+1}^j))}{|\mathcal{I}_i^j|}}_{\text{Avg. inference score over } \mathcal{I}_i^j} \end{cases}$$

$$\phi_{inf}(\eta, \mathcal{F}, a) = \underbrace{\frac{\sum_{x=1}^{a} \delta[\eta_x - \eta_{x-1} > 1] \times \phi_{span}(\eta_{(x-1)}, \eta_x, \mathcal{F})}{\sum_{x=1}^{a} \delta[\eta_x - \eta_{(x-1)} > 1]}}_{\text{Avg. score for non-terminal children}}$$

$$\text{opScore}(p) = \begin{cases} \text{if } (p == \langle 1, 2 \rangle) \text{ then } 1 \\ \text{else } 0 \end{cases}$$

Figure 4.4: RECPET and RECPEF definitions

4.4.2 Experimental Setting

The experimental setting here is similar to the one presented in Section 4.3.2 except that here correlation with human judgment was done on the sentence level. The data from WMT13 shared task (Macháček and Bojar, 2013) were used with the standard method for meta-evaluation on the sentence level using Kendall τ as described in (Callison-Burch et al., 2012). Kendall τ correlation for each metric is computed by first scoring all the translated sentences using the evaluation metric and then the ranks among the sentences as produced by the metric and the ranks as produced by human judges are compared. In case of tied scores for two sentences (either by human or by metric), the compared pair is excluded from meta-evaluation. The pairs in which human judge and

metric agree (concordant pairs) and pairs in which they disagree (discordant pairs) are combined using the following equation in order to compute the correlation coefficient:

$$\tau = \frac{|\text{concordant pairs}|-|\text{discordant pairs}|}{|\text{concordant pairs}|+|\text{discordant pairs}|} \tag{4.4}$$

Note that the formula for Kendall τ rank correlation coefficient that is used in meta-evaluation is different from the Kendall τ similarity function used for evaluating permutations. The values that it returns are in the range $[-1, 1]$, where -1 means that order is always opposite from the human judgment while the value 1 means that metric ranks the system translations in the same way as humans do.

4.4.3 Interpolation with the Lexical Score

The same baselines were used with the same method of transforming ordering metric into a full metric by incorporation of brevity penalty and lexical score as in Equation 4.1. The only difference in this case is that instead of F1 score as the lexical metric a unigram BLEU is used.

In order to run experiments two parameters need to be set. First of them is parameter α that regulates the importance of the lexical component in the full metric and is shared across all tested metrics including baselines. The value for that parameter is chosen to be 0.5 so it would not underestimate the lexical differences between translations ($\alpha \ll 0.5$) but also would not turn the whole metric into unigram BLEU ($\alpha \gg 0.5$). Note that for reordering evaluation it does not make sense to tune α because that would blur the individual contributions of reordering and adequacy during meta evaluation, which is confirmed by Figure 4.5 showing that $\alpha \gg 0.5$ leads to similar performance for all metrics on German-English.

The PET and PEF measures have an additional parameter β that gives importance to the long distance errors that also needs to be tuned. On Figure 4.6 the effect of β can be observed on German-English for $\alpha = 0.5$. This figure shows that giving slightly higher weight $\beta = 0.6$ gives better results. This goes in line with the intuition that the higher nodes in the tree are more important. That is the reason why for further experiments on all language pairs the value for β is set to 0.6. The plot also shows that extreme values lead to bad metrics: $\beta = 0$ leads to only lowest nodes contributing to the score making the whole metric unable to cover long distance reordering, while $\beta = 1.0$ leads to only the top node of the PET being used which is usually too little information.

4.4.4 Results

The results are shown in Table 4.4 and Table 4.5. These scores could be much higher if a more sophisticated measure was used for the lexical part than unigram BLEU. However, this is not the issue here since the goal here is merely to compare different ways

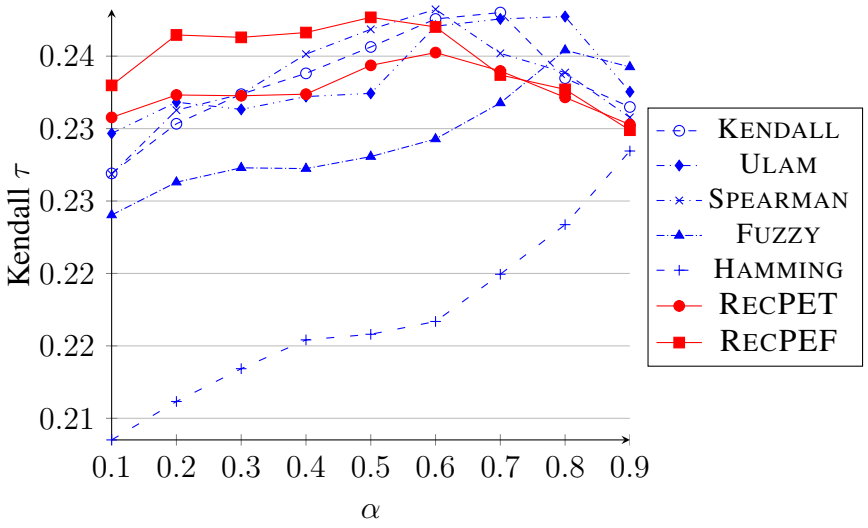


Figure 4.5: Effect of α on German-English evaluation for $\beta = 0.6$

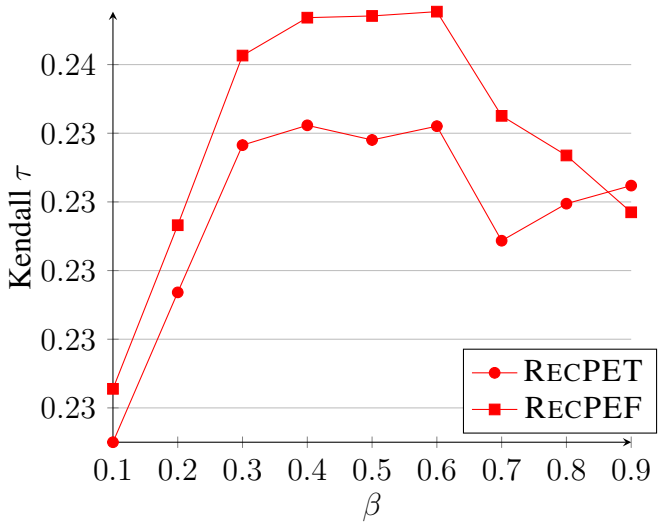


Figure 4.6: Effect of β on German-English evaluation for $\alpha = 0.5$

to evaluate word order. All metrics that are tested have the same lexical component, get the same permutation as their input and have the same value for α.

Metric \ Target lang.	Czech	Spanish	German	Russian	French
KENDALL	**0.16**	0.170	0.183	0.193	0.218
SPEARMAN	0.157	0.170	0.181	0.192	0.215
HAMMING	0.150	0.163	0.168	0.187	0.196
FUZZY	0.155	0.166	0.178	0.189	0.215
ULAM	0.159	0.170	0.181	0.189	**0.221**
RECPET	0.157	0.165	0.182	0.195	0.216
RECPEF	0.156	**0.173**	**0.185**	**0.196**	0.219

Table 4.4: Sentence level Kendall τ scores for translation out of English with $\alpha = 0.5$ and $\beta = 0.6$

Metric \ Source lang.	Czech	Spanish	German	Russian	French
KENDALL	0.196	**0.265**	0.235	0.173	0.223
SPEARMAN	0.199	**0.265**	0.236	0.173	0.222
HAMMING	0.172	0.239	0.215	0.157	0.206
FUZZY	0.184	0.263	0.228	0.169	0.216
ULAM	0.188	0.264	0.232	0.171	0.221
RECPET	0.200	0.264	0.234	0.174	0.221
RECPEF	**0.201**	**0.265**	**0.237**	**0.181**	**0.228**

Table 4.5: Sentence level Kendall τ scores for translation into English with $\alpha = 0.5$ and $\beta = 0.6$

Does hierarchical structure improve evaluation?

The results in Tables 4.4, 4.5 and 4.6 suggest that the PEFscore which uses hierarchy over permutations outperforms the string based permutation metrics in the majority of the language pairs.

Do PEFs help over one canonical PET?

From Figures 4.7 and 4.8 it is clear that using all permutation trees instead of only canonical ones makes the metric more stable in all language pairs. Not only that it makes results more stable but it also improves them in all cases except in English-Czech where both RECPET and RECPEF perform badly. The main reason why RECPEF outperforms RECPET is that it encodes all possible phrase segmentations

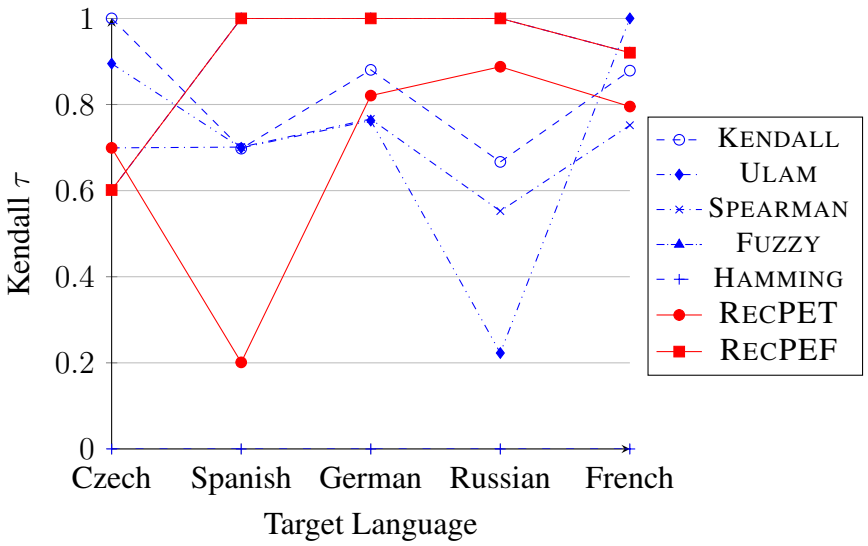


Figure 4.7: Plot of scaled Kendall τ correlation for translation from English

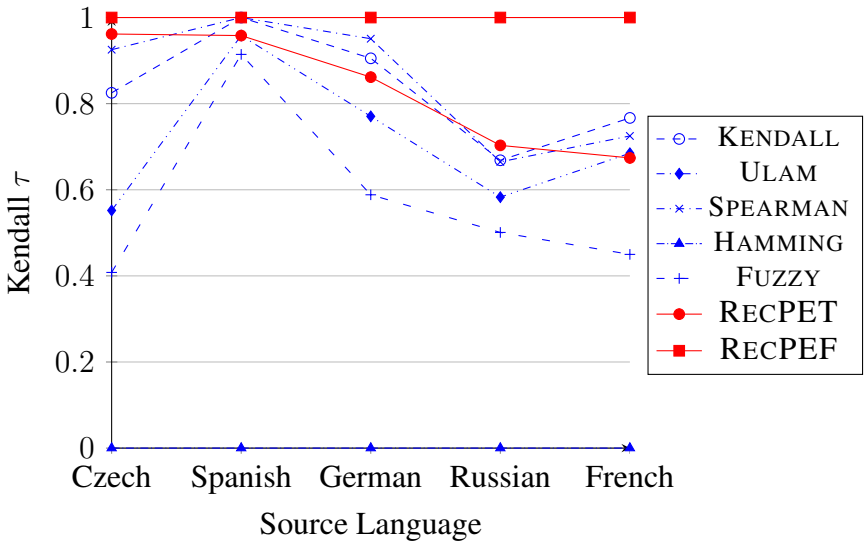


Figure 4.8: Plot of scaled Kendall τ correlation for translation into English

metric	avg rank	avg Kendall
RECPEF	1.6	0.2041
KENDALL	2.65	0.2016
SPEARMAN	3.4	0.201
RECPET	3.55	0.2008
ULAM	4	0.1996
FUZZY	5.8	0.1963
HAMMING	7	0.1853

Table 4.6: Average ranks and average Kendall scores for each tested metrics over all language pairs

of monotone and inverted sub-permutations. By giving the score that considers all segmentations, RECPEF also includes the right segmentation (the one perceived by human evaluators as the right segmentation), while RECPET gets the right segmentation only if the right segmentation is the canonical one.

Is improvement consistent over language pairs?

Table 4.6 shows average rank (metric's position after sorting all metrics by their correlation for each language pair) and average Kendall τ correlation coefficient over the ten language pairs. The table shows clearly that the RECPEF outperforms all other metrics. To make it more visible how metrics perform on the different language pairs, Figures 4.7 and 4.8 show Kendall τ correlation coefficient scaled between the best scoring metric for the given language (in most cases RECPEF) and the worst scoring metric (in all cases HAMMING score). Here it is easier to see that, except in English-Czech, RECPEF is consistently the best or second best (only in English-French) metric in all language pairs. RECPET is not stable and does not give equally good results in all language pairs. HAMMING distance is without exception the worst metric for evaluation since it is very strict about positioning of the words (it does not take relative ordering between words into account). KENDALL is the only string based metric that gives relatively good scores in all language pairs and in one (English-Czech) it is the best scoring metric.

4.5 Conclusions

This chapter leads to, in a sense, similar conclusions as the previous chapter but on a different task: instead of predicting word order here the word order was evaluated. The first main result is that factorization of permutations into PETs is a better way of modeling long distance reordering than a flat view of permutations. Additionally, using all permutations trees (permutation forest) yields empirically better and more stable results.

However, these metrics clearly model only one aspect of evaluation, namely word order, and very often we want a metric that would give a judgment of the translation quality overall. In the next chapter a complete evaluation metric is presented that has high correlation with human judgment and includes several indicators of translations quality including the metrics for word order presented in this chapter.

Chapter 5

Training a Sentence Level Metric

In this chapter new methods for training MT evaluation metrics are presented that are the result of work on the BEER evaluation metric for over three years. The ideal evaluation metric would need to satisfy at least the following conditions: good correlation with human judgment on the sentence level, good correlation on the corpus level, and the metric should not be easy to trick. This chapter covers the first part of these requirements: how to train a given metric for good correlation on the sentence level. Chapter 6 covers the second and third requirement for a good metric.

The main publication on which this chapter is based is:

authors:	Stanojević and Sima'an
title:	*Fitting Sentence Level Translation Evaluation with Many Dense Features*
venue:	EMNLP 2014

The chapter also draws conclusions from several system descriptions of BEER evaluation metric (Stanojević and Sima'an, 2014c, 2015b,c). All the research and all implementations were carried out by me. Khalil Sima'an has provided guidance and edited the mentioned publications.

Chapter Highlights

Problem Statement

- Current MT evaluation metrics have very low correlation with human judgment on the sentence level. They are either based on some heuristics or they are trained to approximate human judgment. Those that are trained give much better results but they also require some data that is not of the same type as the data on which they are going to be evaluated. For example, some metrics get trained on absolute judgments and get meta-evaluated on relative judgments.

Research Question

- What is the cause of the low correlation on the sentence level?
- Which information sources (feature indicators) to use for better correlation?
- How to combine many different sources of information about the translation quality?
- How to train directly for the target goal (i.e. relative ranking)?

Research Contributions

- Identification that the problem of low correlation on the sentence level is caused by usage of feature indicators that fire very rarely (i.e. sparse features). These features are replaced by dense features that fire often even on the short segments such as sentences or words.
- A training method for evaluation metrics that is based on the learning-to-rank framework (Herbrich et al., 1999; Li, 2011) which aims to train the metric to rank translations in the same order as humans do.

5.1 Introduction

Evaluation of an MT output involves comparing system translation with the reference translation on many levels: syntactic, semantic and sometimes even pragmatic. All these levels of comparison are complex cognitive tasks for which the state of knowledge is far from having a full grasp of how they work. That is why it is difficult to imagine that any simplistic metric that is based on count-and-divide strategy will give good performance.

In situations when it is not clear how to solve the problem directly, machine learning is often the right approach to pursue in order to solve the problem at least partially. By using machine learning methods we can approach the problem in the following way. First we say that our goal evaluation metric is a function in some large, but still constrained, space of possible functions. Second, we say that the goal function will be the function that is the best in that space of possible functions according to some criteria. Finally, because finding the goal function is difficult to do by hand, we apply automatic learning algorithms to find that function more efficiently.

In machine learning terminology the first step is called the definition of the model type and its capacity, the second step is the loss function and the final one is the training algorithm. In each of these steps there are many available options that condition the choice in the other steps.

Here all these steps are covered, most of which are explored during the work on the BEER evaluation metric (Stanojević and Sima'an, 2014a,c, 2015b,c). In Section 5.2 I give the main goals that were set for the design of BEER. In Section 5.3 we look at which feature indicators are the best for accomplishing the goal we have set. Section 5.4 covers learning algorithms for training MT evaluation metrics for high sentence level correlation.

BEER has been tested not only by in-house tests, but also by participation on WMT metrics and tuning shared tasks over the course of three years. The tables with the main results from these tasks can be found in appendix A.

5.2 BEER Design Goals

BEER was designed with the following goals:

1. **universality**: applicable to a variety of languages
2. **semantics**: has some notion of semantic equivalence
3. **syntax**: capable of measuring long distance reordering
4. **morphology**: capable of measuring sub-word units accuracy
5. **effectiveness**: has high correlation with human judgment on both sentence and corpus level
6. **efficiency**: fast evaluation suitable for tuning SMT systems
7. **non-gameability**: does not allow "tricking" or "gaming" the metric
8. **easy to distribute and use**

It is difficult to accomplish all these goals in one metric and BEER tries to find a reasonable balance between these goals. Different versions of BEER had different focus.

BEER 0.1 was focused on sentence level effectiveness, semantic equivalence (through paraphrasing), word order (through Permutation Trees and skip bi-grams), morphology (through character n-grams), and universality (no dependence on external tools).

BEER 1.0 was a small improvement on the ease of use.

BEER 1.1 added support for high quality corpus level scoring.

BEER 2.0 removed sentence length bias and improved significantly on efficiency.

BEER 3.0 improved on effectiveness by using better machine learning algorithms that optimise jointly for good correlation on both sentence and corpus level.

The content of this chapter is mainly derived from the main publication about BEER 1.0 (Stanojević and Sima'an, 2014a,c, 2015b,c). The improved training of BEER 2.0 and 3.0 which were published in (Stanojević and Sima'an, 2017) will be covered in Chapter 6.

5.3 Feature Indicators

The design of a model can be decomposed in two main questions. First, what are the sources of information (features) that will be used to indicate the translation quality. Second, how will these sources of information be combined to yield the final score for the translation.

The indicator functions should ideally correspond to all levels of language properties: morphology, syntax, semantics and pragmatics. Having indicators for all these levels is difficult. Hence, for that reason evaluation metrics either concentrate on one or few levels or they use some *proxy* indicators that could implicitly give indication for all these levels to some extent.

Take for instance n-gram matching from the BLEU score. If one four-gram is matched that indicates that at least for these four words it is likely that all levels of evaluation are mostly correct. The problem arises if the same four-gram from reference is not matched, but only *approximately* matched. For instance a translation system might have used different words that convey the same meaning. In that case simple n-gram matching would not be enough and some more explicit notion of semantics would be needed. Another case where approximate matching is needed is when chosen words are only slightly wrong, for instance some suffix is wrongly chosen. Here some explicit sub-word (potentially morphological) analysis would be needed.

Here I review some of the most effective feature indicators that work on different levels of evaluation. Many of them are used as the main ingredients of existing evaluation metrics.

5.3.1 Morphology Feature Indicators

Word Stems

Past evaluation on sub-word level was often based on using pre-existing morphological analysers or stemmers. For instance, METEOR (Denkowski and Lavie, 2014) uses the widely available Snowball stemmer[1]. Stemmers of this type are available for the major languages, but a large majority of morphologically rich languages do not have these tools widely available.

[1] `http://snowballstem.org/`

Even when these tools are available, they are not perfect. Very often they make errors that could further propagate through the system and in the end cause an inaccurate score. Furthermore, these tools still do not give the level of granularity on which we would want to evaluate. Stemmers split the word into its stem and its affixes which explains to some extent the inflectional morphology but, for instance, it does not show the derivational morphology.

Finally, there is a practical disadvantage of depending on external tools such as stemmers, taggers or parsers. In order to make a metric easy to use in practice it is important to make its distribution as easy as possible. When a metric depends on external tools, it either needs to distribute the tools with the metric—which is sometimes not possible because of licensing issues—or it has to require from the user to install the needed tool by herself, which makes the usage less convenient.

Character N-grams

All these disadvantages of stemming analysers were the main motivation for BEER to start using an alternative indicator of sub-word translation quality. The technique that is used in BEER is a simple n-gram matching on the level of characters. These n-grams can go from simple unigrams (how many characters are translated correctly, independently of their order) up to six-grams which almost match a complete word in most European languages.

Compared to the evaluation with word stems, this approach has several advantages. First, it is applicable to any language independently of the amount of the linguistic resources that are available. Second, it allows finer evaluation than stemmers because it allows looking inside the stems and inside the affixes. Third, by not having dependence on the external tools, distribution of the metric that uses characters is very light and easy for the end user.

Probably the most important property of character n-grams are their *dense counts* that are important for sentence level evaluation (Stanojević and Sima'an, 2014a). Sentence level evaluation often suffers from sparse counts: many indicators, especially those based on matching large word n-grams, tend to have zero count. These zero counts push the metric to have score close to zero which makes it uninformative of the actual quality. This problem is especially recognized for BLEU where n-gram precisions are combined using a geometric mean, so if any order of n-grams is not matched then the whole score becomes zero. For this reason many alternative "smoothed" versions of BLEU were proposed (Lin and Och, 2004; Chiang et al., 2008; Kumar et al., 2009; Chen and Cherry, 2014). Character level n-grams completely avoid this problem by always having dense counts. The scores reflect any small change in the string even if it is on the level of characters.

Character n-grams turned out to be an extremely effective method for evaluation of lexical matching. This was first shown on WMT14 (Macháček and Bojar, 2014) metrics task where BEER outperformed on average sentence level correlation all metrics in out-of-English direction and ended up second in into-English direction. This success

of BEER started a trend of character based metrics. The first metric to follow up was chrF3 on WMT15 (Popović, 2015) and after that CharacTer on WMT16 (Wang et al., 2016), both of which got very high accuracy on WMT metrics tasks. In Section 5.3.6 ablation tests are shown from (Stanojević and Sima'an, 2014a) which illustrate just how crucial character n-grams are for sentence level evaluation.

5.3.2 Syntax and Word Order Feature Indicators

Just like on the morphological level, here again we have distinction between the methods that are based on usage of explicit linguistic resources and on more general methods that are independent of linguistic resources. This distinction is even more important in modeling word order because reliable syntactic parsers are even more difficult to get by than stemmers.

Syntactic Trees

Syntactic trees contain useful information for checking the translation quality. For instance, we can check if the translation system has placed the object in the right place relative to the verb, or whether the verb has the right sub-categorization frame.

Early works on the usage of syntactic trees was based on constituency trees (Liu and Gildea, 2005), but at the present time the dependency trees are the dominant type of syntactic structure used in MT evaluation and NLP in general. There are several reasons for this. The most important one is probably that dependency annotation is more widely available across languages. The second reason is that the information we usually care about can be extracted more readily from dependency trees than from constituency trees.

Concretely, many metrics use syntactic trees to extract relations among words and check if all relations in the reference translation are covered in the system translation. Work that uses constituency trees has to use Collins style head rules to extract these relations, while in the dependency trees this is not necessary since head-dependent relation is directly observable.

Many metrics are based on the idea of *headword-chain*, the idea that started with HWCM metric (Liu and Gildea, 2005) and reappeared later in different forms in other evaluation metrics, such as RED (Yu et al., 2014) and BEER_Treepel (Stanojević and Sima'an, 2015c). HWCM is measuring overlap between chains of head words. These chains can be of different lengths and they do not contain the *type of relation*: arc label is completely ignored. BEER_Treepel is an extension of BEER that uses syntactic trees and tries to account for different importance of different arcs. The main idea is that the arcs such as those that represent verb's arguments (subject and object) are more important than arcs that connect, for instance, punctuation. In the section with experiments it can be seen that for English (the only language on which BEER_Treepel was tested), using syntactic trees makes a big difference compared to not using them.

Skip Bigrams and Permutation Distance

Standard word n-grams have a very local view of the word order that spans only n words. To alleviate this, several simple proposals were made for extending this method without introducing dependence on external resources. One of the simplest extensions is usage of skip bigrams—bigrams that can have many words in between them. This method was first incorporated in the metric called ROUGE-S (Lin and Och, 2004) with several variations that differ in the size of the skip that is allowed.

Skip bigrams measure word order globally because relations among all words will be counted independently of how far away they are. Skip bigrams are not limited by any local window like standard continuous n-grams are.

The idea of skip bigrams re-appeared in the MT evaluation literature in the form of LR-Score (Birch and Osborne, 2010) and RIBES (Isozaki et al., 2010a) metrics. The difference that these metrics introduced compared to ROUGE-S is that they formulate evaluation of word order as measurement of *permutation distance*. The word order of the system translation can be represented as a permutation of reference translation words. Ideally this permutation would be monotone i.e. sequence of number from 1 to n (size of the sentence). The evaluation is cast as distance measure between monotone permutation and the actual permutation. If Kendall τ is used as the permutation distance measure then it is effectively a measure based on skip bigrams. Other permutation distance metrics have also been tried such as Spearman's ρ, Hamming distance and Ulam distance.

Permutation Trees

In Chapter 4 we have seen that the above mentioned metrics are based on a *flat* interpretation of permutations and do not exploit the hierarchical structure inherently available in the permutations. An example of simple hierarchical structure in permutations are BTG/ITG trees (Wu, 1997), which decompose binarizable permutations into binary branching trees with small permutations on the tree nodes that permute the node's children. A problem of ITG is that it cannot cover a large number of permutations. A more powerful structure that can in principle cover all possible permutations in similar structures are Permutation Trees (PETs) (Zhang and Gildea, 2007). The main difference is that PETs, unlike ITG, allow arbitrary number of branches per node.

The effectiveness of the hierarchical view of permutations has already been tested in the task of evaluating word order alone (Stanojević and Sima'an, 2014b, 2016). Its effectiveness was the main reason to incorporate it in BEER for which it also showed bigger improvements than the "flat" measures.

5.3.3 Semantic Feature Indicators

Semantic representations come mainly in two forms: symbolic and continuous. It is possible to combine these two representations, but in practice metrics usually use only

one of the possible representations.

Semantic Roles

The most well known symbolic representations of meaning are semantic roles. In semantic roles the predicate-argument relations are explicitly shown. TINE (Rios et al., 2011) and MEANT (Lo et al., 2012) are some of the metrics that use semantic roles for evaluation. They are based on measuring the overlap between the semantic roles present in the reference and system translation.

There are many problems that metrics based on semantic roles face. First, semantic parsing fails on many sentences, especially on those that are malformed output of MT systems. To alleviate this problem TINE and MEANT back-off to simple lexical matching. These metrics for this same reason do not work well on the sentence level because for different sentences the suitable method of evaluation might be completely different ranging from purely lexical matching to purely semantic role matching. On the corpus level these metrics work well, but only for English and Chinese because these are the only languages for which they have semantic parsers available.

Paraphrases and Synonyms

An alternative symbolic semantic representation are paraphrase and synonym tables. They do not give semantic roles, but instead allow recognition of semantic equivalence between phrases that are different on the surface level. WordNet[2] is a manually built resource that expresses semantic relations among words, out of which the most important for evaluation is synonymy. WordNet was first used by METEOR (Lavie and Agarwal, 2007) for flexible matching between words of the system and reference translation.

Paraphrases are a cheap alternative to manually annotated semantic graphs like WordNet. Thanks to the pivoting method of Bannard and Callison-Burch (2005) it is possible to automatically extract paraphrases from just parallel corpora. Many metrics use paraphrases for flexible matching (Kauchak and Barzilay, 2006; Denkowski and Lavie, 2010; Barancikova, 2014; Marie and Apidianaki, 2015). Because extracting paraphrases is so easy many languages are supported with this semantic source of information.

Word Embeddings

Word embeddings are a method for representing the word meaning as a vector in a high dimensional continuous space. The main idea is that the words with similar meaning will be represented by the vectors that are close by, while words that are dissimilar will

[2] `http://wordnet.princeton.edu/`

have vectors that are far apart (Harris; Mikolov et al., 2013). This method for representing word, phrase and sentence meaning became very popular with the advances in deep learning methods for NLP.

Metrics based on neural networks often use the word vectors as the input representation (Gupta et al., 2015; Ma et al., 2016). There are also some extensions of non-neural metrics such as METEOR to enable them exploit the good aspects of word embeddings (Servan et al., 2016).

5.3.4 Normalizing the Counts

The previous sections covered the type of information that can be counted in the translation output. These counts need to be normalized in some way because without normalization its clear that longer translations will have higher number of counts, for example a higher number of n-grams matched.

The two main ways of combining the counts are precision and recall. They are given by the following equations:

$$P = \frac{|\{\text{units in system translation}\} \cap \{\text{units in reference translation}\}|}{|\{\text{units in system translation}\}|} \tag{5.1}$$

$$R = \frac{|\{\text{units in system translation}\} \cap \{\text{units in reference translation}\}|}{|\{\text{units in reference translation}\}|} \tag{5.2}$$

They can be computed for any type of "units" of translation. These units can be word n-grams, character n-grams, word skip bigrams or anything else that can be counted. The only difference between precision and recall is in the normalization that they use. Notice that if we would use only one type of normalizations it would be easy to "trick" or "game" the metric. For instance, if our only measure of quality is a precision of trigrams then outputting one out of many trigrams from the reference translation will be enough to get the perfect score which is not the metric behavior that is desirable.

To combine precision and recall into one number several types of means can be used. Arithmetic, geometric and harmonic mean are common choices. The most popular of these is probably the harmonic mean which is usually expressed as F-score. F-score can be customized to give different importance to precision and recall by weighting them using parameter β:

$$F_\beta = (1 + \beta^2)\frac{P \cdot R}{\beta^2 \cdot P + R} \tag{5.3}$$

However, the most frequent form of F-score is F_1 score which gives equal importance to precision and recall:

$$F_1 = \frac{2 \cdot P \cdot R}{P + R} \tag{5.4}$$

5.3.5 BEER Design Choices Regarding Feature Functions

Some of the feature choices used in different versions of BEER can be seen in Table 5.1. As mentioned before, BEER tries to satisfy several goals that were posed as desirable properties for a good evaluation metric. The goal to be universally applicable across languages prevents us from using any external tools that give rich annotations for the design of BEER, because these tools are available only for a restricted number of languages.

For morphology, BEER initially used Snowball word stemmer, but removed it in later versions because it was limiting universal applicability of BEER and it did not contribute much to the BEER quality since most of the heavy morphological work was done by character n-grams.

For word order, features based on Permutation Trees (PETs) were useful, but their computation took time that considerably slows down BEER. The slow down comes mostly from the PETs that require expensive computation of alignment between words in the reference and the system translation. Skip bigram are slightly less accurate, but much faster to compute and, similar to PETs, provide a global measure of word order. That is why the latest versions of BEER use only Skip Bigrams.

For semantic matching, BEER used to use paraphrase tables as a universally available resource. However, doing paraphrase matching/alignment required slow beam search[3], which prevents BEER from being easily usable in scenarios such as SMT tuning where it needs to be very fast. From BEER 2.0 the support for paraphrases was dropped.

To sum up, BEER from version 2.0 uses the following features:

- character n-grams of orders 1, 2, 3, 4 and 5
- word n-grams of orders 1, 2, 3 and 4
- skip bigrams of maximal skip size 2 and of unbounded size

For each of these types of units precision, recall, and three different F-scores were computed with β set to 1, 2 and 0.5.

One final feature deals with length-disbalance. If the length of the system and reference translation are a and b respectively then this feature is computed as $\frac{max(a,b)-min(a,b)}{min(a,b)}$. It is computed both for word and character length. It serves a similar purpose as brevity penalty in BLEU score except that this length-disbalance function is symmetric.

5.3.6 Empirical Comparison of Feature Indicators

Here I list results from (Stanojević and Sima'an, 2014a) and (Stanojević and Sima'an, 2014c) where empirical comparison of some of the above proposed features was originally presented. The method how features are combined is explained in more detail

[3] I have used METEOR aligner.

BEER version	semantics	word order	morphology	efficiency
0.1	paraphrase	word n-grams, skip bigrams, PETs	stems, char n-grams	slow
1.0	paraphrase	word n-grams, skip bigrams, PETs	stems, char n-grams	slow
1.1	paraphrase	word n-grams, skip bigrams, PETs	stems, char n-grams	slow
Treepel	paraphrase	word n-grams, skip bigrams, PETs, Dependencies	stems, char n-grams	slow
2.0 & 3.0	no	word n-grams, skip bigrams	char n-grams	fast

Table 5.1: Choices made in BEER feature indicators across versions

in Section 5.4. In this section, for the purposes of comparison, the only thing that matters is that only different feature indicators were used but the model is kept the same (except for parameter values that were retrained).

The analysis was conducted on the sentence level because it is a more difficult scenario. The main points of reference were METEOR 1.4 as the best sentence level metric at the time of testing and BEER with all the features included.

Paraphrasing consistently helps evaluation In Table 5.2 we can see how BEER performs with and without flexible matching with paraphrases. In all languages except Spanish paraphrasing helps the evaluation.

Character vs. Word Features The ablation test in Table 5.3 show that by far the most effective feature indicator are character n-grams. Without character n-gram features BEER drops to a very low quality metric. With character n-grams BEER overcomes every other evaluation metric on that dataset. Character n-grams are more effective than word n-grams, but BEER that does not use word n-grams also performs poorly so combining the two types of n-grams is a better strategy.

language pair	BEER 0.1 with paraphrases	BEER 0.1 without paraphrases	METEOR 1.4
en-cs	**0.194**	0.190	0.152
en-fr	0.257	0.250	**0.262**
en-de	**0.228**	0.217	0.180
en-es	0.227	**0.235**	0.201
cs-en	**0.215**	0.213	0.205
fr-en	**0.270**	0.254	0.249
de-en	**0.290**	0.271	0.273
es-en	**0.267**	0.249	0.247

Table 5.2: Kendall τ correleation on WMT12 data

metric	en-cs	en-fr	en-de	en-es	cs-en	fr-en	de-en	es-en	avg τ
BEER 1.0 without char features	0.124	0.178	0.168	0.149	0.121	0.17	0.179	0.078	0.146
BEER 1.0 without all word features	0.184	0.237	0.223	0.217	0.192	0.209	0.243	0.199	0.213
BEER 1.0 without all F-scores	0.197	0.243	0.219	0.22	0.177	0.227	0.254	0.211	0.219
METEOR 1.5	0.156	0.252	0.173	0.202	0.208	0.249	0.273	0.246	0.22
BEER 1.0 without PET features	0.202	0.248	0.243	0.225	0.198	0.249	0.268	0.234	0.233
BEER 1.0 without function words	0.2	0.245	0.231	0.227	0.189	0.268	0.267	0.253	0.235
BEER 1.0 without fluency features	0.201	0.248	0.236	0.223	0.202	0.257	0.283	0.243	0.237
BEER 1.0 without Kendall τ	0.205	0.246	0.244	0.227	0.202	0.257	0.282	0.248	0.239
BEER 1.0 full	0.206	0.245	0.244	0.23	0.198	0.263	0.283	0.245	0.239

Table 5.3: Ablation test on Kendall τ correlation with WMT12 data

Fluency vs. Adequacy Features The fluency features (PETs and Kendall τ) play a smaller role than adequacy features. It is possible that on WMT translation task that year many SMT systems that participated had rather similar reordering models, trained on similar data, which makes the fluency features not that discriminative relative to adequacy features. Perhaps in a different application, for example MT system tuning, the reordering features would be far more relevant because ignoring them would basically imply disregarding the importance of the reordering model in MT.

PETs vs. Kendall τ Despite the smaller role for reordering features a few observations can be made. Firstly, while PETs and Kendall seem to have similar effect on English-Foreign cases, in all four cases of Foreign-English PETs give better scores. It is possible that the quality of the permutations (induced between system output and reference) is better for English than for the other target languages. Discarding PET features has far larger impact than discarding Kendall. Most interestingly, for German-English it makes the difference in outperforming METEOR. In many cases discarding Kendall τ improves the BEER score, likely because it conflicts with the PET features that are being more effective.

F-score normalization is important Even though BEER already implicitly contains arithmetic average of precision and recall, without harmonic mean BEER does not perform well. This is an interesting result because F-score does not introduce a new type of information, but only combines the existing information (precision and recall) in a particular way that is not done by the linear interpolation of these measures. This, as will be discussed later, might have implications in making the decisions about what kind of model is necessary to be learned.

5.4 Training for Sentence Level Correlation

Given that the metric uses many features, it is necessary to choose the model family which will be used for combining these features. A desirable property of the model would be that it is easy to optimize and extend with new (possibly very large) feature set.

Some early trained metrics, such as METEOR (Lavie and Agarwal, 2007), had only few parameters to optimize but because of the complicated way of combining features the effective gradient based methods were not used to optimize the parameters. Instead, the hill climbing meta-heuristic was used to estimate the parameters of the model. This meta-heuristic limits the model development to only few parameters.

Metrics that used more sophisticated machine learning techniques can be clustered in three groups: regression based, classification based and learning-to-rank based metrics. All have different constraints on the type of human judgments that they can process.

The regression based metrics expect that the human judgments come in the form of absolute values (Albrecht and Hwa, 2008; Specia and Giménez, 2010; Song and Cohn, 2011; Gupta et al., 2015). When that is the case, it is indeed the easiest way to train an evaluation metric. However, the majority of human judgments come in the form of relative rankings where we do not have an absolute number assigned to a sentence but only information that one translation is better than the other. Because regression methods cannot exploit this big resource of relative ranking data they became less relevant in training MT metrics.

Metrics based on classification (Song and Cohn, 2011; Guzmán et al., 2015) try to answer the following question: given the reference translation and the translations of two MT systems, which of the two systems is better? For this task any binary classifier can be used. At the test time these metrics require not only one system and one reference translation but two system translations at the same time. This is a problem for several reasons. First, it is not a standard way of doing MT evaluation where we expect a metric to assign a score to a single translation system. Second, if we want to compare more than two systems then the number of comparisons grows quadratically. Finally, when we compare, for example, three systems A, B and C it is not guaranteed that we will not get contradicting scores in which $score(A) > score(B)$, $score(B) > score(C)$ and $score(C) > score(A)$.

Learning-to-rank based metrics (Ye et al., 2007; Duh, 2008; Stanojević and Sima'an, 2014a; Ma et al., 2016) try to have the best of both worlds: they are trained on pairwise judgments to assign higher score to better translation (just like classification methods), but at test time they are able to assign a single score to a system translation (like regression methods do). These good properties of learning-to-rank methods are the main motivation for using them to train the models of BEER.

5.4.1 Learning-to-Rank with a Linear Model

Learning-to-rank methods can be applied to many types of models (Li, 2011), but in practice the most used model is based on the ideas of Ranking SVM (Herbrich et al., 1999) because it is intuitive and very simple to implement.

Ranking SVM expects the model to be linear in the feature space. The score that the model assigns to the system translation t and the reference translation r with features $\phi(t, r)$ is given by:

$$score(t, r) = \mathbf{w} \cdot \phi(t, r) \tag{5.5}$$

where $\mathbf{w}$ is a weight vector that needs to be estimated.

The human judgment is given in a form: translation t_{good} is better than translation t_{bad} for reference r. Because of that we want the metric to satisfy the following condition:

$$score(t_{good}, r) > score(t_{bad}, r) \tag{5.6}$$

Since the model is linear we can easily derive the following:

$$\begin{aligned}
score(t_{good}, r) > score(t_{bad}, r) &\iff && (5.7)\\
\mathbf{w} \cdot \phi(t_{good}, r) > \mathbf{w} \cdot \phi(t_{bad}, r) &\iff && (5.8)\\
\mathbf{w} \cdot \phi(t_{good}, r) - \mathbf{w} \cdot \phi(t_{bad}, r) > 0 &\iff && (5.9)\\
\mathbf{w} \cdot (\phi(t_{good}, r) - \phi(t_{bad}, r)) > 0 &\iff && (5.10)\\
\mathbf{w} \cdot (\phi(t_{bad}, r) - \phi(t_{good}, r)) < 0 && && (5.11)
\end{aligned}$$

Equations 5.10 and 5.11 can serve as objectives for training the ranking model: $\phi(t_{good}, r) - \phi(t_{bad}, r)$ can be interpreted as features of the positive training instance and $\phi(t_{bad}, r) - \phi(t_{good}, r)$ as the features of the negative instance. Training for this objective is equivalent to training for minimising the Kendall τ pairwise ranking loss.

In principle any linear classifier can be used for estimating the parameters $\mathbf{w}$. Depending on the version, BEER has used both linear SVM and logistic regression. The motivation for both are related to corpus level scoring and are explain in Section 6.1.

5.4.2 Learning-to-Rank with a Non-Linear Model

Extending Ranking SVM from a simple linear model to a non-linear model via kernel trick is simple mathematically, but computationally very costly (Kuo et al., 2014). The simplest way to introduce non-linearity is by manually adding a non-linear transformations of the existing features. In the previous section, we have seen that F-score is crucial for good performance of BEER. F-score is an example of non-linear combinations of existing features (precision and recall) that gave significant boost to the BEER quality. F-score is a "hard-coded" non-linearity and yet it was very helpful. This suggests that there is a big space for improvement if we manage to find a way how to automatically learn non-linearities useful for MT evaluation.

A promising way of doing that is by using the current advances in learning with neural networks. Neural networks can be trained with stochastic gradient decent so they do not require a full pass over the data in order to make an update on the parameters, unlike Kernel Ranking SVM which needs to go over the data several times just to compute the kernel matrix. Neural networks can be trained for margin loss just like the Ranking SVM and additionally they can learn their own "kernel" unlike Ranking SVM. MaxSD (Ma et al., 2016) is a new evaluation metric that uses this property of neural networks to train non-linear LSTM based neural scoring model. However, training a neural network model directly for max-margin between the "good" and "bad" sentences is sometimes suboptimal as I will show in Chapter 6.

5.5 Conclusion

In this chapter the BEER evaluation metric was presented. In it the most important problems of sentence level evaluation were covered: choice of the dense features for which non-zero counts can be collected and the training method that trains the metric directly for the target (ranking) objective.

This metric exhibits very high correlation with human judgment. However, some problems are still left to be addressed. First, if the metric is going to be evaluated on the corpus level then training it on the sentence level, as done in this chapter, might be suboptimal. Second, the metric might be easy to game because the data on which it is trained might be biased toward long sentences. Both of these problems are addressed in the next chapter.

Chapter 6

Training a Corpus Level Metric

Metrics are not used only for the final evaluation of MT systems, but also for tuning the MT systems. As I will argue, the standard procedures for training MT metrics produce metrics that are not good for training MT systems. I propose new solutions how to overcome this problem. The first solution is based on the usage of semi-supervised training algorithms that help in decreasing the data bias present in the standard training data used for evaluation metrics. The second solution is based on more careful feature engineering that removes the sentence length bias problem usually present in MT metrics that have high correlation with human judgment.

This chapter is based on the following publication:

authors:	Stanojević and Sima'an
title:	*Alternative Objective Functions for Training MT Evaluation Metrics*
venue:	ACL 2017

All the research and all implementations were carried out by me. Khalil Sima'an has provided guidance and edited the mentioned publications.

Chapter Highlights

Problem Statement

- Trained evaluation metrics are without exception trained on the sentence level but they are frequently used on the corpus level. This training regime is suboptimal and might give bad correlation with human judgment on both sentence and corpus level.
- Evaluation metrics in general and trained evaluation metrics in particular are often easy to *game*. This limits their usage in tasks like tuning of MT systems where the tuning algorithm manages to *trick* the metric and exploit its bad properties.

Research Question

- How to directly train the MT metric to be good on the corpus level?
- How to train the MT metric to be at the same time good both on sentence and corpus level?
- How to prevent the metric gaming and unwanted biases?

Research Contributions

- Extension of the algorithms presented in the previous chapter to be able to train for corpus level correlation directly.
- An algorithm for training for both type of judgments (sentence and corpus level) at the same time by using a multi-task objective.
- Preventing metric gaming by accounting for the data bias with semi-supervised training methods.
- Feature engineering methods for preventing length bias that do not require special training regimes.

6.1 Training for Corpus Level Correlation

It might seem that the metric that is trained to be good on the sentence level would also be good on the corpus level, but that is not the case. Empirically it has been shown that many metrics that perform well on the sentence level do not perform well on the corpus level and vice versa. By training the model just to rank sentences, the model would not necessarily learn to *scale* the scores in a good way. The metric would just give higher score to better translation, but the difference in scores may not be proportional to the difference in quality. When the metric is judged by Kendall τ correlation on the sentence level this is not a real problem, but it becomes a problem in some other situations.

The first situation is when the metric needs to approximate some absolute human judgments such as Direct Assessment (DA) scores (Graham et al., 2013). When meta-evaluation is done on DA scores, the Pearson correlation is computed which reflects not only the correctness in ranking of MT systems but also the relative difference in scores that the metric gives to different systems. A metric trained only with standard learning-to-rank methods might perform badly on this measure because scaling is not a part of the loss function.

The second case is usage of the evaluation metric as the objective for training MT systems. The updates made to parameters during tuning of MT systems is often proportional to the magnitude of error as measured by the metric. If the metric does not give a good measure of the magnitude of error then tuning will not work well.

The third case where scaling might make a big difference is corpus level scoring. Trained metrics often compute corpus level score as average of sentence level scores. If these scores are not well scaled then the result of averaging is not reliable. I take the problem of corpus level scoring as the prime motivation for well scaled sentence level scores.

In the development of BEER three different approaches were used for constructing the corpus level metric over the years. All of them were successful in giving high correlation with human judgment.

6.1.1 Quasi-Probabilistic Method

This method was used in BEER 1.1 (Stanojević and Sima'an, 2015c) and does not require any training, but it puts constraints on the training model:

- the model must be linear
- the training instances must be extracted in the style of Ranking SVM
- the parameters must be optimized with logistic regression

If we take the model from the previous chapter that satisfies all these constraints, then we might ask the question: what does the probability of the logistic regression in this case mean? The features given to the logistic regression are subtraction of features of "good" and the features of "bad" hypotheses. This means that the probability output of logistic regression can be interpreted as a probability that the first translation A is better than the second translation B given the reference R.

$$p(A > B|R) = \sigma(\ \mathbf{w} \cdot (\phi(A, R) - \phi(B, R))\ ) \tag{6.1}$$

where $\sigma(\cdot)$ is a sigmoid function, $\mathbf{w}$ is a weight vector and ϕ is a feature function.

The model outputs probabilities which is a good property because we can assign some interpretation to the scores that it gives which is not the case with most models of evaluation metrics. However, this convenient scenario does not happen at the test time. At the test time we have only one system translation and the reference translation. There is no other system translation with which we can ask the logistic regression to assign the probability of one translation being better than the other. At test time the sentence level score in the old BEER 0.1, which did not apply any scaling method, is computed in the following way:

$$\text{score}(A, R) = \sigma(\mathbf{w} \cdot \phi(A, R)) \tag{6.2}$$

It is easy to show that:

$$\text{score}(A, R) > \text{score}(B, R) \iff p(A > B|R) > 0.5 \tag{6.3}$$

This scoring function does not influence the rankings of sentences but it makes scores scaled in a very bad way—most of the scores end up being close to 1 because the features are not subtracted at test time as they were at training time.

That is why in BEER 1.1, at test time, we ask the logistic regression another question: what is the probability that the system translation A given the reference translation R is better than the reference translation R given the reference translation R. This might seem a weird and useless question because it will always be the case that R is a better translation given R than A could be, but it is not a completely uninformative question. Depending on how certain the model is about R being better than A we can judge the quality of A. If R and A are virtually indistinguishable then the probability of one being better than the other would be roughly 0.5. If A is a very bad translation then the probability of A being better than R will tend to be 0. It is not guaranteed that the scores will always be between 0 and 0.5, but in most cases they will end up in that region. In order to make the scores in the standard metric scale between 0 and 1, we multiply the probability with 2. The equations that show how this sentence level score is computed is shown bellow.

$$\text{score}(A, R) = 2 \cdot p(A > R|R) = 2 \cdot \sigma(\ \mathbf{w} \times (\phi(A, R) - \phi(R, R))\ ) \tag{6.4}$$

The performance of this scaling can be observed on corpus level evaluation. The results on corpus level on WMT15 dataset is shown in Tables A.5 and A.6 in appendix A.

6.1.2 Separate Scaling Model

The problem of scaling the output of a classifier is not unique to MT evaluation. It has been faced before, for instance, in scaling the output of SVM classifiers. SVM classifiers produce only the distance from the separating hyperplane, but do not give a well calibrated output. To solve this the researches often used the approach of training a separate calibration model that scales the output of SVM. Two most well known approaches to scaling the output of arbitrary classifiers are Isotonic regression (Zadrozny and Elkan, 2001) and Platt scaling (Platt, 1999; Lin et al., 2007). In Platt scaling the logistic regression model is trained to transform the "raw" score into a probability-looking output:

$$\text{score}_{platt}(A, R) = \sigma(\ a \cdot \text{score}_{raw}(A, R) + b\ ) \tag{6.5}$$

There are only two parameters that need to be estimated: a and b. This is because it guarantees that the transformation function is monotone.

In BEER 2.0 we adopt the idea of training a separate scaling model that takes as its input the output of the unscaled "raw" model. Just like in Platt scaling, the BEER scaling model is a linear model with only two parameters a and b. Unlike Platt scaling, we do not estimate these parameters with logistic regression but with support vector regression that does not apply the sigmoid function in the end:

$$\text{score}_{scaled}(A, R) = a \cdot \text{score}_{raw}(A, R) + b \tag{6.6}$$

One important difference between Platt scaling and our setting is also that Platt scaling is meant for classification problems while we have a learning-to-rank type of problem. Therefore, to estimate our parameters a and b I apply a different technique of training the model to predict the "gold truth" about absolute translation performance. The "gold truth" that we use are Direct Assessment scores (DA). Even though the dataset that we used from (Graham and Liu, 2016)[1] is relatively small, it was big enough for estimating only two parameters.

This method of scaling the sentence level scores does not put any restrictions on the underlying "raw" scoring model. Compared to the Quasi-Probabilistic scaling model this method allows much more space for designing better models. A disadvantage of training the separate scaling model is that it requires a small amount of human judgments with absolute scores assigned by humans. This is a requirement that is difficult to satisfy for a large number of languages which means that a more universally applicable method is required. Another problem with this method is that it is a pipeline approach which could cause error propagation from the ranking model to the scaling model. The results that this method achieves can be seen on corpus level correlation results from WMT16 in Tables A.9 and A.10 in appendix A.

6.1.3 Learning-to-Rank Corpora

The main motivation for having a well scaled sentence level scores is getting corpus level score for which scaling of sentence level scores might have big effect: if sentence level scores are not well scaled some sentences will get more (or less) importance than what they deserve. If achieving the good corpus level scores is the main goal then maybe the best way to achieve that goal is to train the metric to directly optimise corpus level score. In experiments during the design of BEER 3.0 this method was explored and showed promising results.

The method that I employ is the same as on the sentence level: the model's parameters are trained to be good in ranking corpora in such a way to separate "good" corpora from "bad" corpora by some margin. On sentence level we usually use the standard size for the desired margin which is 1, but on the corpus level we can enforce a more informed margin size. On corpus level we want the margin between "good" and "bad" corpora to be at least as big as the margin between their human scores. That means that

[1] `https://github.com/ygraham/segment-mteval/raw/master/seg-mteval-data.tar.gz`

for each training instance (for each pair of corpora) we will have a different margin. The "margin scaling" was first introduced in Max-Margin Markov networks (Taskar et al., 2003). As human corpus level scores I use the scores computed by Expected Wins heuristic (Koehn, 2012).

Here, just like in the previous chapter, a simple linear model is used for scoring each sentence in the corpus. First we compute a "forward" score for all the sentences in the corpus:

$$\text{forward}(\phi) = \phi^T \mathbf{w} + b \tag{6.7}$$

where ϕ is the feature vector and parameters $\mathbf{w}$ and b are weight vector and a bias term that need to be estimated.

The $\text{forward}(\cdot)$ function can return any real number. To make sentence level score between 0 and 1 we apply a sigmoid function to the output of forward function:

$$\text{sentScore}(\phi) = \sigma(\ \text{forward}(\phi)\ ) \tag{6.8}$$

The corpus level score is computed as the average of sentence level scores:

$$\text{corpScore}(\mathbf{\Phi}) = \frac{1}{m} \sum_{i=1}^{m} \text{sentScore}(\ \phi_i\ ) \tag{6.9}$$

where $\mathbf{\Phi}$ is a matrix containing in each column ϕ_i a weight vector for sentence i in the corpus that is being evaluated.

The margins assigned by the model and by the humans are given with the following equations:

$$\Delta_{corp} = \text{corpScore}(\mathbf{\Phi_{cwin}}) - \text{corpScore}(\mathbf{\Phi_{clos}}) \tag{6.10}$$

$$\Delta_{human} = \text{expWins}(cwin) - \text{expWins}(clos) \tag{6.11}$$

Here expWins signifies the human score extracted with Expected Wins heuristic described in Section 2.4.1. Matrices $\mathbf{\Phi_{cwin}}$ and $\mathbf{\Phi_{clos}}$ are feature matrices for the "good" and "bad" corpora respectively.

The margin loss that is optimized is non-zero in all cases where the margin assigned by humans is bigger than margin assigned by the model:

$$Loss_{Corp} = \max(0, \Delta_{human} - \Delta corp) \tag{6.12}$$

This method of optimizing for corpus level score is better than both of the previous methods because it puts no constraints on the type of the model it optimizes and it does not require additional data. Also, it is not based on any heuristics, but instead optimizes in a principled way directly the objective in which we are interested.

6.2 Joint Training for Sentence and Corpus Level Correlation

It might seem that sentence and corpus level tasks are very similar but that is not the case. Empirically it has been shown that many metrics that perform well on the sentence level do not perform well on the corpus level and vice versa. This is especially true for the trained metrics because the majority (if not all) of the metrics are optimized for sentence level correlation.

Ideally we would have a metric that is good both on the sentence level and on the corpus level correlation, but given the previous performance of many evaluation metrics it comes as a question whether creating such a metric is possible. One might expect that if only one of the objectives is optimized (whether it is sentence or corpus level correlation) it would lead to good performance only on that measure. In the work on BEER 3.0 I have also tried to optimize for both objectives simultaneously by using a multi-task learning framework (Caruana, 1997).

In Figure 6.1 we can see an illustration of how the loss is computed. The left part of the computation graph shows how sentence level loss is computed in a similar way as described in subsection 5.4. Matrices $\mathbf{\Phi_{swin}}$ and $\mathbf{\Phi_{slos}}$ represent mini-batches of sentence comparisons where features in column i of each of the matrices belong to the winning and losing system in judgment i respectively.

The right part of the computation graph represents the computation of the corpus level loss in the same way as in subsection 6.1.3. The final topmost node represents an average of the sentence and corpus level loss. If this loss is optimized with gradient based techniques then the gradient will be such as to optimize parameters to be good for both objectives.

For the experiments a linear model is used, but this method can easily be extended to non-linear models. There are only two conditions that need to be satisfied. First is that the model must be differentiable with respect to the input features, which is a condition that almost any feed-forward neural network model satisfies. The second condition is that functions $\mathrm{forward}(\cdot)$ and $\mathrm{corpScore}(\cdot)$ must share parameters. Otherwise, there would be no effect of a joint objective.

The results for the three different objectives (sentence, corpus and joint objectives) on WMT16 relative ranking dataset are shown in Tables 6.2. There are some interesting conclusions that can be drawn from these results.

The corpus-objective is better than the sentence-objective for both corpus and sentence level relative ranking (RR) judgments on 5 out of 7 languages and also on average correlation.

Training for the joint objective improves even more on both levels of RR correlation and outperforms both single-objective models on average and on 4 out of 7 languages.

Making confident conclusions from these results is difficult because, to the best of my knowledge, there is no principled way of measuring statistical significance on the RR judgments. That is why I also tested on direct assessment (DA) judgments

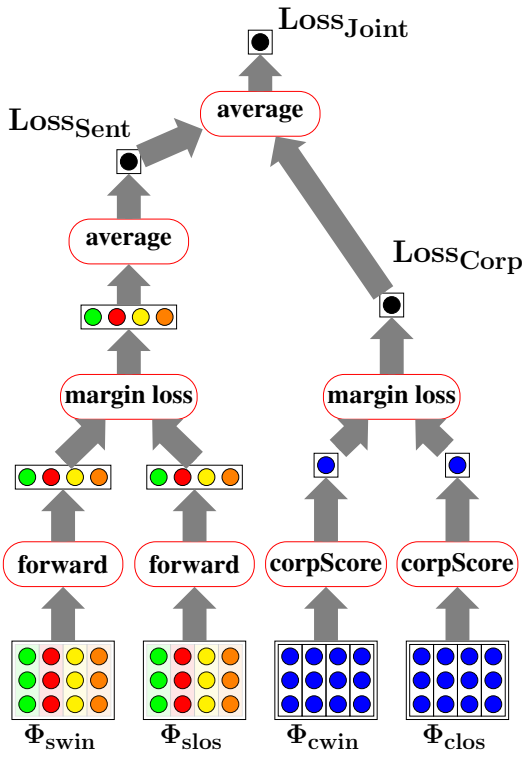


Figure 6.1: Computation graph for joint training of sentence and corpus objectives

available from WMT16. On DA we can measure statistical significance on the sentence level using Williams test (Graham et al., 2015) and on the corpus level using a combination of hybrid-supersampling and Williams test (Graham and Liu, 2016). The results of correlation with human judgment are for sentence and corpus level are shown in Table 6.1.

On DA judgments the results for the corpus level objective are completely different than on the RR judgments. On DA judgments the corpus-objective model is significantly outperformed on both levels and on all languages by both of the other objectives.

This shows that gambling on one objective function (being that sentence or corpus level objective) could give unpredictable results. This is precisely the motivation for creating the joint model with multi-objective training.

By choosing to jointly optimize both objectives we get a much more stable model that performs well both on DA and RR judgments and on both levels of judgment. On the DA sentence level, the joint model was *not* outperformed by any other model and on 3 out of 7 language pairs it significantly outperforms both alternative objectives. On the corpus level results are slightly mixed, but still the joint objective outperforms both other models on 4 out of 7 language pairs and also it gives higher correlation on average.

Table 6.3 summarizes all the choices made in the learning algorithms and models for BEER over the different versions.

Objective	en-ru	cs-en	de-en	fi-en	ro-en	ru-en	tr-en	Avg.
sent.	**0.911**$^{C}_{J}$	0.984^{C}	0.848^{C}	**0.956**$^{C}_{J}$	0.835^{C}	0.889^{C}	**0.971**$^{C}_{J}$	0.913
corpus	0.909	0.979	0.803	0.912	0.793	0.886	0.901	0.883
joint	0.911^{C}	**0.984**$^{C}_{S}$	**0.849**$^{C}_{S}$	0.955^{C}	**0.840**$^{C}_{S}$	**0.894**$^{C}_{S}$	0.965^{C}	**0.914**

(a) Corpus level

Objective	en-ru	cs-en	de-en	fi-en	ro-en	ru-en	tr-en	Avg.
sent.	0.666^{C}	0.648^{C}	0.493^{C}	0.461^{C}	0.507^{C}	0.554^{C}	0.580^{C}	0.558
corpus	0.563	0.568	0.391	0.364	0.377	0.431	0.458	0.450
joint	0.667^{C}	**0.663**$^{S}_{C}$	**0.502**$^{S}_{C}$	0.461^{C}	**0.528**$^{S}_{C}$	0.556^{C}	0.583^{C}	**0.566**

(b) Sentence level

Table 6.1: Direct Assessment (DA) Pearson r Correlation on WMT16 dataset. Super- and sub-scripts S, C and J signify that the model outperforms with statistical significance ($p < 0.05$) the model trained for sentence, corpus or joint objective respectively. Bold font marks that the system has outperformed both other models significantly.

Objective	en	cs	de	fi	ro	ru	tr	Avg.
sent.	**0.963**	0.977	0.737	0.938	0.922	0.905	0.937	0.912
corpus	0.944	0.982	**0.765**	0.940	0.917	**0.907**	**0.954**	0.916
joint	**0.963**	**0.983**	0.748	**0.951**	**0.933**	0.905	0.946	**0.918**

(a) Corpus level

Objective	en	cs	de	fi	ro	ru	tr	Avg.
sent.	0.347	0.405	0.345	0.304	0.293	0.382	0.304	0.340
corpus	0.337	**0.414**	0.349	**0.307**	0.292	0.385	**0.325**	0.344
joint	**0.350**	0.410	**0.356**	0.296	**0.299**	**0.396**	0.312	**0.346**

(b) Sentence level

Table 6.2: Relative Ranking (RR) correlation on WMT16 dataset

BEER version	scaling	RR sent. objective	RR corp. objective	DA sent objective	optimization
0.1	none	yes	no	no	single
1.0	quasi-probabilistic	yes	no	no	single
1.1	quasi-probabilistic	yes	no	no	single
2.0	separate scal. model	yes	no	yes	pipeline
3.0	jointly trained	yes	yes	no	joint

Table 6.3: Choices made in the learning models of BEER across versions

6.3 Preventing Unwanted Biases

One of the most important use cases of MT evaluation metrics is for optimization of MT systems. In SMT the most popular tuning algorithms either directly optimize for the desired evaluation metric (Och, 2003) or they use large-margin methods in which the metric determines the margin that should be enforced (Chiang et al., 2008; Cherry and Foster, 2012). In Neural MT, there is a work on incorporating metrics into the loss function by training for Minimum Risk objective (Shen et al., 2016).

One might expect that the metric that is good in correlation with human judgment would be a good metric to use for the optimization of MT systems. However, that does not seem to be the case. BLEU, even though it has a bad correlation with human judgment, still gives very good results for tuning compared to other metrics that have much higher correlation with human judgments (Cer et al., 2010; Stanojević et al., 2015a; Jawaid et al., 2016) .

I believe that the reason for this lies in the bias present in the two types of data on which the metrics are used: (1) the data used for meta-evaluation of metrics on the metrics tasks and (2) the data that appears in the search space of MT system while it is tuned. This data is completely different in many respects.

The first difference is the quality of translation. The data that is used in the metrics tasks are the best outputs of the best machine translation systems that participate in the translation task. This high quality data is completely different from the data that an MT decoder encounters in its search space which contains both high and low quality translations. A metric that is good in evaluation of high quality translations, might not be as good for evaluation of bad translations that appear in the search space, and that might lead to bad performance of the learning algorithm.

The second difference is the length of the translations that are preferred by humans. Lavie et al. (2004) showed that human judgments can be modeled much better with recall than with precision. This shows that humans prefer longer translations that would cover all the words from the reference and potentially have some wrong additional words, than to have a short translation that would not have all the words from the reference translation. An evaluation metric that would replicate this human preference would also prefer longer translation over short ones. This would give very good results

on the metrics tasks, but it would lead to a bad metric for tuning MT systems. A metric that rewards longer translations, if used for tuning, would lead MT system to produce pathologically long sentences (Nakov et al., 2013).

This problem has been observed early in METEOR which is a metric that combines precision and recall through F-score and usually puts higher weight for recall because it helps in correlation with human judgment. He and Way (2009) tackled the problem by manually setting equal weight for precision and recall which gave them very good results for tuning MT systems.

Even though this is an effective solution, it is not ideal. Firstly, because it removes only one type of bias – length bias, and there might be other biases in the data of which we are unaware of. The second problem with this solution is that it can be applied only on simple metrics such as METEOR. If a metric has a large number of features, which is the case with BEER, and if these features are combined in ways more complicated than F-score, then it would be very difficult to manually make precision and recall equally important.

Here I present two attempts to solve this problem that have been applied to BEER.

6.3.1 Semi-Supervised Training

The "data bias" problem is not specific to MT evaluation metrics. It appears in many NLP problems and has been studied extensively under the framework of domain adaptation (Søgaard, 2013). In domain adaptation it is often the case that we have a small amount of annotated data that is *out-of-domain* and a large amount of unannotated data that is *in-domain*.

The problem of metric bias can be framed in this way. Our annotated data is from a domain of long and high quality sentences, and we are interested in learning the metric that is good for a more general domain of sentences that could be of any length and any level of quality.

An ideal annotated in-domain dataset would be human judgments of sentences that are randomly sampled from the search space of the MT decoder. However such judgments do not exist. In this situation semi-supervised techniques can help. With semi-supervised algorithms we can exploit the labeled data as a supervised signal and unlabeled data to remove the bias of the labeled dataset.

In order to apply semi-supervised techniques we need to pick a learning algorithm that we are going to use and to find unlabeled data which would be closer to the target domain. As a learning algorithm I used the simplest semi-supervised learning algorithm called self-training (Abney, 2007; Søgaard, 2013). As unlabeled dataset I used pairs of translations uniformly sampled from n-best lists of a Moses based MT system that was translating a WMT12 test set. As common in MT tuning, the expectation here is that n-best list approximately represents the search space of the decoder and that the uniformly sampled pairs of translations in the n-best list are the pairs for which we would ideally like to have the correct ranking.

tuning metric	BLEU	MTR	BEER	Length
BEER	16.4	**28.4**	**10.2**	115.7
BLEU	**18.2**	28.1	10.1	103.0
BEER_no_bias	18.0	27.7	9.8	**99.7**

Table 6.4: Results for tuning with BEER, BLEU and self-trained BEER_no_bias

The algorithm works in the following way. First, we train BEER in the standard way on the annotated data (WMT human rankings). Afterwords, we use the trained BEER to assign ranks to randomly sampled pairs from n-best lists. This annotated rankings are added to the real human judgments and now BEER is trained again both on the human rankings and on the rankings produced by BEER itself. This can be done in several iterations until convergence, but I applied only one iteration since there was no improvement for more than one iteration.

To test whether this method helped in removing the bias for long translations in BEER, I tuned a Moses MT system by using both versions of BEER. The results are shown in Table 6.4. BEER_no_bias shows the results for tuning with self-trained BEER. The effect of data bias is easily visible in the last column of the table that shows the length of the system output compared to the length of the reference translation. Standard BEER which does not account for the length bias, prefers longer translations and has caused an MT system to produce sentences that have 15.7% more words than the reference translation. On the other hand BEER_no_bias that was trained with self-training has only 0.3% mismatch in the length with the reference. That is 10 times more precise match for the sentence length than BLEU which had 3% mismatch, and 52 times more precise than the original BEER.

Getting the length right was not rewarded well by the recall heavy metrics BEER and METEOR which gave the highest score to the system that produced the longest translation that was clearly wrong. BLEU gave more realistic scores in this case. This shows again that the metrics that behave well on the metrics task datasets (BEER and METEOR) when confronted with a different dataset that is out of their domain they are unable to make a correct judgment. With self-training we are able to remove that unwanted domain bias.

The MT system tuned with BEER_no_bias model participated in the shared tuning task on WMT15 and ended up second on Czech-English. The results are shown in Table 6.5. The self-trained BEER system ended up second by outperforming all the submitted systems except the system tuned with BLEU with which it ended up in the same cluster.

6.3.2 Symmetric MT Metric

The length bias in the model happens because the learning model, due to biased training data, managed to learn to give more importance to recall than to precision. If we

System Name	Human Score		BLEU
	Tuning-Only	All	
BLEU-MIRA-DENSE	0.153	-0.177	12.28
BEER-MIRA-DENSE	0.108	-0.188	12.05
BLEU-MERT-DENSE	0.087	-0.200	12.11
AFRL	0.070	-0.205	12.20
USAAR-TUNA	0.011	-0.220	12.16
DCU	-0.027	-0.256	11.44
METEOR-CMU	-0.101	-0.286	10.88
BLEU-MIRA-SPARSE	-0.150	-0.331	10.84
HKUST	-0.150	-0.331	10.99

Table 6.5: WMT15 Tuning Task results on Czech-English

would manage to "hide" from the learning model which feature is a feature that represents precision and which one represents recall, we would significantly decrease the effect of length bias.

To do that in BEER, we replace precision and recall with their maximums and minimums:

$$x = \min(precision, recall) \tag{6.13}$$

$$y = \max(precision, recall) \tag{6.14}$$

Now where we used precision and recall we will use their minimums and maximums. For example, F-score will be computed in the following way:

$$F_1 = \frac{2\ x\ y}{x + y} \tag{6.15}$$

Notice that minimum and maximum are symmetric functions, which makes all the functions that use only them also symmetric. If all features in our model follow this pattern then the whole metric is symmetric. Symmetry is a very desirable property because now the metric cannot give more importance to system translations that are longer than reference translation because it does not "know" any more which translation is system translation and which one is reference. The knowledge about which translation is reference and which one system is hidden in the design of features.

Compared to the previous method that is based on self-training, this is a much simpler method that does not require additional data nor any specialized training. However, it does not solve all the problems. Feature symmetrization is restricted only to the length bias, while self-training can potentially address other types of data bias.

Compared to the method used in METEOR of making precision and recall weight equal this method does seem similar. METEOR with weight 0.5 is indeed symmetric. Our method has an advantage over the METEOR's method in allowing for weights to

System Name	Human Score	BLEU
BLEU-MIRA	0.114	22.73
AFRL	0.095	22.90
NRC-NNBLEU	0.090	23.10
NRC-MEANT	0.073	22.60
BEER-PRO	0.032	22.46
BLEU-MERT	0.000	22.51

Table 6.6: WMT16 Tuning Task results on Czech-English

System Name	Human Score	BLEU
BLEU-MIRA	0.160	15.12
BEER-PRO	0.152	14.69
BLEU-MERT	0.151	14.93
AFRL2	0.139	14.84
AFRL1	0.136	15.02
DCU	0.134	14.34
FJFI-PSO	0.127	14.68
USAAR-HMM-MERT	-0.433	7.95
USAAR-HMM-MIRA	-1.133	0.82
USAAR-HMM	-1.327	0.20

Table 6.7: WMT16 Tuning Task results on English-Czech

still be tuned and keep the symmetric property, while METEOR has to keep one single weight setting.

This version of BEER participated in WMT16 shared tuning task (Jawaid et al., 2016) from which the results are shown in Tables 6.6 and 6.7. In case of English-Czech, just like the year before, BEER won the second place right after the system tuned for BLEU with MIRA. Unfortunately, the results on Czech-English do not look as good as on English-Czech. The only common pattern between these two results is that BEER tuning with PRO (Hopkins and May, 2011) works always better then BLEU tuning with MERT (Och, 2003), and BEER tuning with PRO works worse than BLEU tuning with kbMIRA (Cherry and Foster, 2012). Due to different optimization algorithms it cannot be concluded whether the difference in results comes from the different metrics or from different learning algorithms.

6.4 Conclusion

In the results that I have presented, it is visible that there are many aspects of training MT evaluation metrics that are important for the final performance, but are usually ignored during the metric design. Metric designers optimize their metric for correlation with human judgment on the sentence level and often ignore other tasks on which the metric could be used, for instance, corpus level evaluation and MT tuning. The training methods used for BEER are designed in such a way to account for all these use cases: we train jointly for sentence and corpus level, and use semi-supervised training to make BEER more domain-general and more difficult to game by tuning algorithms.

Several learning algorithms were presented for all aspects of metric design. Most of them gave very good results and made BEER one of the best performing metrics on the WMT shared metrics tasks over last three years.

Chapter 7

Conclusion

7.1 Summary

The main focus of this dissertation was on modeling word order in machine translation by using hierarchical tree structures that are directly derived from data. This approach has been tested in two tasks: prediction of target language word order and evaluation of word order quality of MT output.

In both of these tasks a large part of the method is the same. The main difference is in which pairs of sentences are modeled: in case of preordering the pairs of sentences that are used are source and target sentences from parallel corpora, while in the case of evaluation it is an MT system translation and a translation made by humans. After words of the two parallel sentences are aligned, a *permutation view* of that reordering is taken. With viewing the problem of reordering as a problem of modeling permutations we can use many existing insights that already exist about permutations.

The main insight on which this work is based is that permutations can be recursively decomposed into smallest atomic permutations called prime or simple permutations (Albert and Atkinson, 2005; Zhang and Gildea, 2007). This recursive decomposition can be interpreted as a permutation tree (PET) in which each node in the tree is labeled with a prime permutation that permutes the children of the node. An important property of this decomposition is that there can be exponentially many such decompositions that explain the same permutation.

The four main reasons why PETs are interesting for modeling word order are: (i) they require no linguistic annotation, (ii) they can form constituents even out of non-syntactic phrases, (iii) reordering in them is local to each node in the tree and (iv) the reordering of the whole permutation is derived in a compositional way. The first property is primarily a practical advantage since high quality parsers do not exist for the majority of natural languages. The second property is a modeling choice that can be argued for or against depending on the view of how compositionality in language works, but it is certainly a very convenient property from modeling perspective because all of the translation data can be covered in this way even if it does not completely fit

the syntactic analysis.

That latter two properties are crucial for efficient recognition and effective learning models whose task is to predict PETs on the new data. Having a compositional structure allows us to make independence assumptions that help the learning model in finding the best way to explain the reordering in the training data.

In Chapter 3 a preordering model was presented which is based on inducing a state-splitted PCFG for parsing permutation trees. As mentioned before, the number of PETs that can generate permutation can be even exponential in the terms of permutation size. The main difference among these PETs is the way in which they explain how words are combined in a compositional way. In that chapter it has been shown that these PETs can give significantly different results in terms of how well they predict the word order. For instance, for English a right branching PET gives much better results than the left branching PET. Possibly there are even better PETs somewhere in between the extremes of left and right branching PETs. Even if we knew the right heuristic for choosing the best performing PET for English, a completely different heuristic might be better for some other language.

To avoid making this arbitrary choices for the optimal PET a different approach was taken: all PETs are modeled as latent variables and the learning algorithm takes the role of promoting the one which explains the permutation in the best way. To make the algorithm fit the data better, the prime permutations (labels in the PET tree) were additionally split into different states that are also treated as latent variables. This method is very similar to state-splitting syntactic parsing models (Matsuzaki et al., 2005; Prescher, 2005; Petrov et al., 2006; Saluja et al., 2014) with the main difference being that here the splitting is not done over one tree but over the whole forest of trees.

The experimental results showed that using a whole forest of PETs instead of a single PET gave better results in the prediction of word order. Alternative to PET forest is a forest of ITG trees – the subset of PETs that can cover only binarizable permutations. In the experiments this version of the model showed good results but the more powerful PET forest still gave significantly better results.

Chapter 4 gave a similar view on modeling word order but in the context of evaluation of word order. Several metrics based on permutation trees were constructed and evaluated on how well do they approximate human judgment of the translation output. The main conclusions of that chapter are that having a hierarchical view of permutations gave better correlation with human judgment and that using a forest of permutation trees is better than relying on a single canonical permutation tree.

In Chapter 5 the evaluation metric was extended from the basic metric that evaluates only word order to a trained evaluation metric that evaluates both adequacy and fluency of the translation output. The design of the metric named BEER was based on defining features that are good for evaluation on the sentence level and on the algorithm that can use existing human judgments to learn how to combine the features in the right way to approximate human judgment of translation quality. The choices made in the feature extraction and the design of the learning algorithm made BEER the metric with the highest correlation with human judgment on the majority of languages on WMT

shared metrics task (Macháček and Bojar, 2014; Stanojević et al., 2015b; Bojar et al., 2016) at the moment of writing.

Chapter 6 goes further in the topic of training MT evaluation metrics by proposing training algorithms that address some aspects of training MT metrics that have been ignored by the community. Most of the MT evaluation metrics that use machine learning are trained only for the correct sentence level judgment, which contrasts with the usual setting in which MT metrics are used which is evaluation on the corpus level. In that chapter a new learning algorithm for training evaluation metric for corpus level judgment was presented.

Ideally, an MT metric should be good both for sentence and corpus level objective. This has been addressed by a multi-task learning objective. The chapter also addressed how the metric should be trained in order to be good for usage of training of MT systems. The data bias present in available human judgments is identified as the main problem in making trained MT metrics good for training MT systems. This problem has been addressed by the usage of semi-supervised training methods that allow usage of sentences that are not labeled with human judgment as part of the training data. By being able to incorporate data from different domains a data bias problem can be significantly reduced.

7.2 PETs and the Future of Machine Translation

During the four years work on this thesis the field of machine translation has changed radically. Neural machine translation models (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015) have completely changed the whole architecture that was used for more than a decade. One of the ways to see this shift is as a shift from discrete representations to continuous representations that made it easier to incorporate many more advanced learning algorithms. Most of the neural systems that gave impressive results on the translation tasks use almost no symbolic structure except for the trivial sequential structure.

This raises the question if all the work that has been done in symbolic hierarchical structures, including this thesis, is useful in the new world of neural MT. I believe that the answer to this question is positive. The answer is not based only on majority of linguistic theories that give symbolic description of language but also on recent research that shows how syntactic structures can be used to inform or constrain neural models and give them the right learning bias (Eriguchi et al., 2016; Eriguchi et al., 2017; Stahlberg et al., 2016, 2017; Bastings et al., 2017; Aharoni and Goldberg, 2017). One could imagine the same techniques being used to bring permutation trees to the neural translation models.

It is difficult to judge what makes neural models so much better than SMT models because they have almost every component designed in a different way. In my opinion, what makes the biggest difference are not representations that are used by neural system, but instead the powerful non-linear models that are trained end-to-end.

The results of this thesis show that hierarchical compositional modeling word order is beneficial for good performance of MT systems. This, I believe, holds for all aspects of human language. If hierarchical structures, such as permutation trees, get combined with new modeling techniques provided by recursive neural architectures then we could get a new type of MT systems that would bring the best of both worlds.

Appendix A

Results from the WMT Metrics Shared Tasks

This appendix presents the results from the metrics shared tasks where BEER participated. On WMT14 (Macháček and Bojar, 2014), we participated with BEER version 0.1 only on the sentence level. The results from WMT14 are shown in Tables A.1 and A.2.

On WMT15 (Stanojević et al., 2015b), we participated with BEER 1.1 on both sentence and corpus level. The results for sentence level are shown in Tables A.4 and A.3. The results for the corpus level correlation are shown in Tables A.5 and A.6.

On WMT16 (Bojar et al., 2016), we participated with BEER 2.0 on both sentence and corpus level with both DA and RR types of judgments. The results for sentence level correlation are shown in Tables A.7 and A.8, while the results for the corpus level are shown in Tables A.9 and A.10.

Direction	en-fr	en-de	en-hi	en-cs	en-ru
BEER	**.295**	**.258**	.250	**.344**	**.440**
METEOR	.278	.233	.264	.318	.427
AMBER	.261	.224	**.286**	.302	.397
BLEU-NRC	.257	.193	.234	.297	.391
APAC	.255	.201	.203	.292	.388

Table A.1: Sentence level Kendall τ correlation on WMT14 out of English.

Direction	fr-en	de-en	hi-en	cs-en	ru-en
DISCOTK-PARTY-TUNED	**.433**	**.381**	.434	**.328**	**.364**
BEER	.417	.337	**.438**	.284	.337
REDCOMBSENT	.406	.338	.417	.284	.343
REDCOMBSYSSENT	.408	.338	.416	.282	.343
METEOR	.406	.334	.420	.282	.337

Table A.2: Sentence level Kendall τ correlation on WMT14 into English.

Direction	fr-en	fi-en	de-en	cs-en	ru-en	Average
DPMFCOMB	.367	**.406**	**.424**	**.465**	**.358**	**.404**
BEER_TREEPEL	.358	.399	.386	.435	.352	.386
RATATOUILLE	**.367**	.384	.380	.442	.336	.382
BEER	.359	.392	.376	.417	.336	.376
METEOR-WSD	.347	.376	.360	.416	.331	.366
CHRF	.350	.378	.366	.407	.322	.365
DPMF	.344	.368	.363	.413	.320	.362
CHRF3	.345	.361	.360	.409	.317	.359
LEBLEU-OPTIMIZED	.349	.346	.346	.400	.316	.351
LEBLEU-DEFAULT	.343	.342	.341	.394	.317	.347
TOTAL-BS	$-.305$	$-.277$	$-.287$	$-.357$	$-.263$	$-.298$

Table A.3: Sentence level Kendall τ correlation on WMT15 into English.

Direction	en-fr	en-fi	en-de	en-cs	en-ru	Average
BEER	.323	**.361**	**.355**	**.410**	**.415**	**.373**
CHRF3	.309	.357	.345	.408	.398	.363
RATATOUILLE	**.340**	.300	.337	.406	.408	.358
LEBLEU-DEFAULT	.321	.354	.345	.385	.386	.358
LEBLEU-OPTIMIZED	.325	.344	.345	.383	.385	.356
CHRF	.317	.346	.315	.407	.387	.355
METEOR-WSD	.316	.270	.287	.363	.373	.322
TOTAL-BS	$-.269$	$-.205$	$-.231$	$-.324$	$-.332$	$-.273$
DPMF	.308	n/a	.289	n/a	n/a	.298
PARMESAN	n/a	n/a	n/a	.089	n/a	.089

Table A.4: Sentence level Kendall τ correlation on WMT15 out of English.

Correlation coefficient	Pearson Correlation Coefficient					
Direction	**fr-en**	**fi-en**	**de-en**	**cs-en**	**ru-en**	**Average**
DPMFCOMB	.995	.951	**.949**	**.992**	.871	**.952**
RATATOUILLE	.989	.899	.942	.963	**.941**	.947
DPMF	**.997**	.939	.929	.986	.868	.944
METEOR-WSD	.982	.944	.914	.981	.857	.936
CHRF3	.979	.893	.921	.969	.915	.935
BEER_TREEPEL	.981	**.957**	.905	.985	.846	.935
BEER	.979	.952	.903	.975	.848	.931
CHRF	.997	.942	.884	.982	.830	.927
LEBLEU-OPTIMIZED	.989	.895	.856	.970	.918	.925
LEBLEU-DEFAULT	.960	.895	.856	.946	.912	.914

Table A.5: Corpus level correlation on WMT15 into English.

Correlation coefficient	Pearson Correlation Coefficient					
Metric	**en-fr**	**en-fi**	**en-de**	**en-cs**	**en-ru**	**Average**
CHRF3	.949	**.813**	.784	**.976**	.913	**.887**
BEER	.970	.729	.811	.951	**.942**	.880
LEBLEU-OPTIMIZED	.949	.727	**.896**	.944	.867	.877
LEBLEU-DEFAULT	.949	.760	.827	.946	.849	.866
RATATOUILLE	.962	.675	.777	.953	.869	.847
CHRF	.949	.771	.572	.968	.871	.826
METEOR-WSD	.961	.663	.495	.941	.839	.780
BS	−.977	.334	−.615	−.947	−.791	−.600
DPMF	**.973**	n/a	.584	n/a	n/a	.778

Table A.6: Corpus level correlation on WMT15 out of English.

Direction	**cs-en**		**de-en**		**fi-en**		**ro-en**		**ru-en**		**tr-en**	
Human	RR	DA	RR	DA	RR	DA	RR	DA	RR	DA	RR	DA
DPMFCOMB	.388	**.713**	.420	**.584**	.481	**.598**	.383	.627	.420	**.615**	.401	**.663**
METRICS-F	.345	**.696**	.421	**.601**	.447	.557	.388	**.662**	.412	**.618**	.424	**.649**
COBALT-F.	.336	.671	.415	**.591**	.433	.554	.361	.639	.397	**.618**	.423	.627
UPF-COBA.	.359	.652	.387	.550	.436	.490	.356	.616	.394	.556	.379	.626
BEER	.342	.661	.371	.462	.416	.471	.331	.551	.376	.533	.372	.545
MPEDA	.331	.644	.375	.538	.425	.513	.339	.587	.387	.545	.335	.616
CHRF2	.341	.658	.358	.457	.418	.469	.344	.581	.383	.534	.346	.556
CHRF3	.343	.660	.351	.455	.421	.472	.341	.582	.382	.535	.345	.555
CHRF1	.323	.644	.372	.454	.410	.452	.339	.570	.379	.522	.345	.551
UOW-REVAL	.261	.577	.329	.528	.376	.471	.313	.547	.314	.528	.342	.531
WORDF3	.299	.599	.293	.447	.377	.473	.304	.525	.343	.504	.287	.536
WORDF2	.297	.596	.296	.445	.378	.471	.300	.522	.341	.503	.283	.537
WORDF1	.290	.585	.293	.435	.369	.464	.293	.508	.336	.497	.275	.535
SENTBLEU	.284	.557	.265	.448	.368	.484	.272	.499	.330	.502	.245	.532
DTED	.201	.394	.130	.254	.209	.361	.144	.329	.201	.375	.142	.267

Table A.7: Sentence level correlation on WMT16 into English.

Direction	**en-cs**		**en-de**		**en-fi**		**en-ro**		**en-ru**		**en-tr**	
Human	RR	DA	RR	DA	RR	DA	RR	DA	RR	DA	RR	DA
BEER	.422	-	.333	-	.364	-	.307	-	.405	**.666**	.337	-
CHRF2	.420	-	.329	-	.374	-	.304	-	.406	**.661**	.330	-
CHRF3	.421	-	.327	-	.380	-	.304	-	.400	**.661**	.326	-
CHRF1	.402	-	.320	-	.350	-	.305	-	.389	.642	.320	-
MPEDA	.393	-	.274	-	.342	-	.238	-	.372	**.645**	.255	-
WORDF2	.373	-	.247	-	.313	-	.250	-	.358	.580	.218	-
WORDF3	.373	-	.247	-	.314	-	.245	-	.359	.582	.216	-
WORDF1	.369	-	.245	-	.311	-	.248	-	.351	.573	.209	-
SENTBLEU	.359	-	.236	-	.306	-	.233	-	.328	.550	.222	-

Table A.8: Sentence level correlation on WMT16 out of English.

Direction	**cs-en**		**de-en**		**fi-en**		**ro-en**		**ru-en**		**tr-en**	
Human	RR	DA	RR	DA	RR	DA	RR	DA	RR	DA	RR	DA
MPEDA	**.996**	**.993**	**.956**	.937	**.967**	**.976**	**.938**	**.932**	**.986**	.929	**.972**	**.982**
UOW.REVAL	**.993**	**.986**	**.949**	**.985**	**.958**	**.970**	**.919**	**.957**	**.990**	**.976**	**.977**	**.958**
BEER	**.996**	**.990**	.949	.879	**.964**	**.972**	**.908**	.852	**.986**	.901	**.981**	**.982**
CHRF1	.993	.986	.934	.868	**.974**	**.980**	.903	.865	**.984**	.898	**.973**	.961
CHRF2	**.992**	.989	.952	.893	**.957**	**.967**	.913	.886	**.985**	.918	.937	.933
CHRF3	.991	.989	**.958**	.902	.946	.958	.915	**.892**	**.981**	.923	.918	.917
CHARACTER	**.997**	**.995**	**.985**	.929	.921	.927	**.970**	**.883**	.955	**.930**	.799	.827
MTEVALNIST	.988	.978	.887	.801	.924	.929	.834	.807	.966	.854	**.952**	**.938**
MTEVALBLEU	**.992**	**.989**	.905	.808	.858	.864	.899	.840	.962	.837	.899	.895
MOSESCDER	**.995**	**.988**	.927	.827	.846	.860	.925	.800	.968	.855	.836	.826
MOSESTER	**.983**	**.969**	.926	.834	.852	.846	.900	.793	.962	.847	.805	.788
WORDF2	**.991**	**.985**	.897	.786	.790	.806	.905	.815	.955	.831	.807	.787
WORDF3	**.991**	**.985**	.898	.787	.786	.803	.909	.818	.955	.833	.803	.786
WORDF1	**.992**	**.984**	.894	.780	.796	.808	.890	.804	.954	.825	.806	.776
MOSESPER	.981	.970	.843	.730	.770	.767	.791	.748	**.974**	.887	**.947**	.940
MOSESBLEU	**.991**	**.983**	.880	.757	.752	.759	.878	.793	.950	.817	.765	.739
MOSESWER	**.982**	**.967**	.926	.822	.773	.768	.895	.762	.958	.837	.680	.651

Table A.9: Corpus level correlation on WMT16 into English.

Direction	**en-cs**		**en-de**		**en-fi**		**en-ro**		**en-ru**		**en-tr**	
Human	RR	DA	RR	DA	RR	DA	RR	DA	RR	DA	RR	DA
CHARACTER	**.947**	-	**.915**	-	**.933**	-	**.959**	-	**.954**	**.966**	**.930**	-
BEER	**.973**	-	.732	-	.940	-	**.947**	-	.906	.922	**.956**	-
CHRF2	.954	-	.725	-	**.974**	-	.828	-	.930	**.955**	**.940**	-
CHRF3	.954	-	.745	-	**.974**	-	.818	-	**.936**	**.960**	**.916**	-
MOSESCDER	**.968**	-	.779	-	.910	-	**.952**	-	.874	.874	.791	-
CHRF1	.955	-	.645	-	.931	-	.858	-	.901	.928	**.938**	-
WORDF3	**.964**	-	.768	-	.901	-	**.931**	-	.836	.840	.714	-
WORDF2	**.964**	-	.766	-	.899	-	**.933**	-	.836	.840	.715	-
WORDF1	**.964**	-	.756	-	.888	-	**.937**	-	.836	.839	.711	-
MPEDA	**.964**	-	.684	-	**.944**	-	.786	-	.856	.866	.860	-
MOSESBLEU	**.968**	-	.784	-	.857	-	**.944**	-	.820	.820	.693	-
MTEVALBLEU	**.968**	-	.752	-	.868	-	.897	-	.835	.838	.745	-
MTEVALNIST	**.975**	-	.625	-	.886	-	.882	-	.890	.897	.788	-
MOSESTER	.940	-	.742	-	.863	-	.906	-	.882	.879	.644	-
MOSESWER	.935	-	.771	-	.855	-	**.912**	-	.882	.876	.570	-
MOSESPER	**.974**	-	.681	-	.700	-	**.944**	-	.857	.854	.641	-

Table A.10: Corpus level correlation on WMT16 out of English.

Bibliography

Steven Abney. *Semisupervised Learning for Computational Linguistics*. Chapman & Hall/CRC, 1st edition, 2007. ISBN 1584885599, 9781584885597.

Roee Aharoni and Yoav Goldberg. Towards String-to-Tree Neural Machine Translation. *ArXiv e-prints*, April 2017.

Alfred V. Aho and Jeffrey D. Ullman. Syntax directed translations and the pushdown assembler. *J. Comput. Syst. Sci.*, 3(1):37–56, February 1969. ISSN 0022-0000. doi: 10.1016/S0022-0000(69)80006-1. URL http://dx.doi.org/10.1016/S0022-0000(69)80006-1.

Michael H. Albert and Mike D. Atkinson. Simple Permutations and Pattern Restricted Permutations. *Discrete Mathematics*, 300(1-3):1–15, 2005.

Joshua S. Albrecht and Rebecca Hwa. Regression for machine translation evaluation at the sentence level. *Machine Translation*, 22(1):1, 2008. ISSN 1573-0573. doi: 10.1007/s10590-008-9046-1. URL http://dx.doi.org/10.1007/s10590-008-9046-1.

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. OpenFst: a general and efficient weighted finite-state transducer library. In *Proc. CIAA*, 2007.

Amittai Axelrod, Ra Birch Mayne, Chris Callison-burch, Miles Osborne, and David Talbot. Edinburgh system description for the 2005 iwslt speech translation evaluation. In *In Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, 2005.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*, 2015.

Colin Bannard and Chris Callison-Burch. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 597–604. Association for Computational Linguistics, 2005.

Petra Barancikova. Parmesan: Meteor without paraphrases with paraphrased references. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 355–361, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics.

Joost Bastings, Ivan Titov, Wilker Aziz, Diego Marcheggiani, and Khalil Sima'an. Graph Convolutional Encoders for Syntax-aware Neural Machine Translation. *ArXiv e-prints*, April 2017.

Alexandra Birch and Miles Osborne. LRscore for Evaluating Lexical and Reordering Quality in MT. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 327–332, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W10-1749`.

Alexandra Birch and Miles Osborne. Reordering Metrics for MT. In *Proceedings of the Association for Computational Linguistics*, Portland, Oregon, USA, 2011. Association for Computational Linguistics.

Alexandra Birch, Miles Osborne, and Phil Blunsom. Metrics for MT evaluation: evaluating reordering. *Machine Translation*, pages 1–12, 2010. ISSN 0922-6567.

Arianna Bisazza and Marcello Federico. Dynamically shaping the reordering search space of phrase-based statistical machine translation. *Transactions of ACL*, 2013.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W14/W14-3302`.

Ondřej Bojar, Yvette Graham, Amir Kamran, and Miloš Stanojević. Results of the wmt16 metrics shared task. In *Proceedings of the First Conference on Machine Translation*, pages 199–231, Berlin, Germany, August 2016. Association for

Computational Linguistics. URL http://www.aclweb.org/anthology/W/W16/W16-2302.

Pavel Braslavski, Alexander Beloborodov, Maxim Khalilov, and Serge Sharoff. English-to-Russian MT evaluation campaign. In *Proceedings of the* 51^{st} *ACL*, pages 262–267, Sofia, Bulgaria, August 2013.

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computional Linguistics*, 19(2):263–311, June 1993. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=972470.972474.

Chris Callison-Burch, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. (meta-) evaluation of machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 136–158, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Chris Callison-Burch, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. Findings of the 2012 Workshop on Statistical Machine Translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada, June 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W12-3102.

Glenn Carroll and Mats Rooth. Valence Induction with a Head-Lexicalized PCFG. In *In Proceedings of Third Conference on Empirical Methods in Natural Language Processing*, 1998.

Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, July 1997. ISSN 0885-6125. doi: 10.1023/A:1007379606734. URL http://dx.doi.org/10.1023/A:1007379606734.

Daniel Cer, Christopher D. Manning, and Daniel Jurafsky. The best lexical metric for phrase-based statistical mt system optimization. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 555–563, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. ISBN 1-932432-65-5.

Jean-Cédric Chappelier and Martin Rajman. A Generalized CYK Algorithm for Parsing Stochastic CFG. In *Proceedings of the First Workshop on Tabulation in Parsing and Deduction*, pages 133–137, 1998. URL ftp://ftp.inria.fr/INRIA/Projects/Atoll/TAPD98/chappelier.ps.gz.

Boxing Chen and Colin Cherry. A systematic comparison of smoothing techniques for sentence-level bleu. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 362–367, Baltimore, Maryland, USA, June 2014. Association for

Computational Linguistics. URL http://www.aclweb.org/anthology/W/W14/W14-3346.

Colin Cherry and George Foster. Batch Tuning Strategies for Statistical Machine Translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 427–436, 2012. ISBN 978-1-937284-20-6. URL http://dl.acm.org/citation.cfm?id=2382029.2382089.

David Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 263–270, 2005.

David Chiang. Learning to translate with source and target syntax. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1443–1452, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

David Chiang, Yuval Marton, and Philip Resnik. Online Large-margin Training of Syntactic and Structural Translation Features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 224–233, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1613715.1613747.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1179.

Noam Chomsky. *Syntactic Structures*. Mouton, The Hague, 1957.

Noam Chomsky. *Aspects of a Theory of Syntax*. MIT Press, Cambridge, Mass., 1965.

Noam Chomsky. Remarks on Nominalization. In Roderick A. Jacobs and Peter S. Rosenbaum, editors, *Readings in English Transformational Grammar*, pages 184–221. Ginn, Boston, 1970.

Tagyoung Chung and Daniel Gildea. Effects of empty categories on machine translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 636–645, Cambridge, MA, October 2010. Association for Computational Linguistics.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. Better Hypothesis Testing for Statistical Machine Translation: Controlling for Optimizer Instability.

In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 176–181, 2011. ISBN 978-1-932432-88-6.

John Cocke. *Programming Languages and Their Compilers: Preliminary Notes*. Courant Institute of Mathematical Sciences, New York University, 1969. ISBN B0007F4UOA.

Michael Collins. Head-Driven Statistical Models for Natural Language Parsing. *Comput. Linguist.*, 29(4):589–637, December 2003. ISSN 0891-2017.

Michael Collins, Philipp Koehn, and Ivona Kučerová. Clause Restructuring for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 531–540, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219906. URL `http://dx.doi.org/10.3115/1219840.1219906`.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *J. Mach. Learn. Res.*, 7:551–585, December 2006. ISSN 1532-4435.

Joachim Daiber, Miloš Stanojević, Wilker Aziz, and Khalil Sima'an. Examining the relationship between preordering and word order freedom in machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 118–130, Berlin, Germany, August 2016. Association for Computational Linguistics.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38, 1977.

John DeNero and Jakob Uszkoreit. Inducing Sentence Structure from Parallel Corpora for Reordering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 193–203, 2011. ISBN 978-1-937284-11-4.

John DeNero, David Chiang, and Kevin Knight. Fast Consensus Decoding over Translation Forests. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 567–575, 2009. ISBN 978-1-932432-46-6. URL `http://dl.acm.org/citation.cfm?id=1690219.1690226`.

Michael Denkowski and Alon Lavie. Extending the METEOR Machine Translation Evaluation Metric to the Phrase Level. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 250–253, 2010.

Michael Denkowski and Alon Lavie. Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems. In *Proceedings of the EMNLP 2011 Workshop on Statistical Machine Translation*, 2011.

Michael Denkowski and Alon Lavie. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*, 2014.

Jinhua Du and Andy Way. Pre-reordering for neural machine translation: Helpful or harmful? *The Prague Bulletin of Mathematical Linguistics*, 108(1):171–182, 2017.

Kevin Duh. Ranking vs. Regression in Machine Translation Evaluation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, StatMT '08, pages 191–194, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. ISBN 978-1-932432-09-1. URL http://dl.acm.org/citation.cfm?id=1626394.1626425.

Greg Durrett and Dan Klein. Neural crf parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 302–312, Beijing, China, July 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-1030.

Chris Dyer and Philip Resnik. Context-free Reordering, Finite-state Translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 858–866, 2010. ISBN 1-932432-65-5. URL http://dl.acm.org/citation.cfm?id=1857999.1858127.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. Generalizing word lattice translation. In *Proceedings of ACL-08: HLT*, pages 1012–1020, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P08/P08-1115.

Jason Eisner. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 2*, ACL '03, pages 205–208, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. ISBN 0-111-456789. doi: 10.3115/1075178.1075217.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. Tree-to-sequence attentional neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 823–833, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P16-1078.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. Learning to Parse and Translate Improves Neural Machine Translation. *ArXiv e-prints*, February 2017.

Michel Galley and Christopher D. Manning. A simple and effective hierarchical phrase reordering model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 848–856, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1613715.1613824`.

Daniel Gildea, Giorgio Satta, and Hao Zhang. Factoring Synchronous Grammars by Sorting. In *ACL*, 2006.

Yoav Goldberg. A Primer on Neural Network Models for Natural Language Processing. *ArXiv e-prints*, October 2015.

Yvette Graham and Qun Liu. Achieving accurate conclusions in evaluation of automatic machine translation metrics. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, CA, 2016. Association for Computational Linguistics.

Yvette Graham, Timothy Baldwin, Alistair Moffat, and Justin Zobel. Continuous measurement scales in human evaluation of machine translation. In *Proceedings of the 7th Linguistic Annotation Workshop & Interoperability with Discourse*, pages 33–41, Sofia, Bulgaria, 2013. Association for Computational Linguistics.

Yvette Graham, Nitika Mathur, and Timothy Baldwin. Accurate evaluation of segment-level machine translation metrics. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies*, Denver, Colorado, 2015.

Rohit Gupta, Constantin Orasan, and Josef van Genabith. Reval: A simple and effective machine translation evaluation metric based on recurrent neural networks. In *EMNLP*, 2015.

Francisco Guzmán, Shafiq Joty, Lluís Màrquez, and Preslav Nakov. Pairwise neural machine translation evaluation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and The 7th International Joint Conference of the Asian Federation of Natural Language Processing (ACL'15)*, pages 805–814, Beijing, China, July 2015. Association for Computational Linguistics.

Zellig Harris. Distributional structure. *Word*, 10(23):146–162, 1954.

Yifan He and Andy Way. Improving the objective function in minimum error rate training. *Proceedings of the Twelfth Machine Translation Summit*, pages 238–245, 2009.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. Scalable Modified Kneser-Ney Language Model Estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria, August 2013. URL http://kheafield.com/professional/edinburgh/estimate_paper.pdf.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. Support Vector Learning for Ordinal Regression. In *In International Conference on Artificial Neural Networks*, pages 97–102, 1999.

Mark Hopkins and Jonathan May. Tuning as Ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D11-1125.

Liang Huang, Hao Zhang, Daniel Gildea, and Kevin Knight. Binarization of synchronous context-free grammars. *Comput. Linguist.*, 35(4):559–595, December 2009. ISSN 0891-2017. doi: 10.1162/coli.2009.35.4.35406.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. Automatic Evaluation of Translation Quality for Distant Language Pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 944–952, Stroudsburg, PA, USA, 2010a. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1870658.1870750.

Hideki Isozaki, Katsuhito Sudoh, Hajime Tsukada, and Kevin Duh. Head Finalization: A Simple Reordering Rule for SOV Languages. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, WMT '10, pages 244–251, 2010b. ISBN 978-1-932432-71-8.

Bushra Jawaid, Amir Kamran, Miloš Stanojević, and Ondřej Bojar. Results of the wmt16 tuning shared task. In *Proceedings of the First Conference on Machine Translation*, pages 232–238, Berlin, Germany, August 2016. Association for Computational Linguistics.

Fred Jelinek and Robert L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In Edzard S. Gelsema and Laveen N. Kanal, editors, *Proceedings, Workshop on Pattern Recognition in Practice*, pages 381–397. North Holland, Amsterdam, 1980.

Dan Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, Englewood Cliffs, NJ, 2nd edition, 2008.

Miriam Kaeshammer and Anika Westburg. On complex word alignment configurations. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, may 2014. European Language Resources Association (ELRA). ISBN 978-2-9517408-8-4.

Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. Seattle, October 2013. Association for Computational Linguistics.

Tadao Kasami. An efficient recognition and syntax-analysis algorithm for context-free languages. Technical report, Air Force Cambridge Research Lab, 1965.

Jason Katz-Brown, Slav Petrov, Ryan McDonald, Franz Och, David Talbot, Hiroshi Ichikawa, Masakazu Seno, and Hideto Kazawa. Training a Parser for Machine Translation Reordering. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 183–192, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D11-1017`.

David Kauchak and Regina Barzilay. Paraphrasing for automatic evaluation. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 455–462, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. doi: 10.3115/1220835.1220893.

Maxim Khalilov and Khalil Sima'an. Context-Sensitive Syntactic Source-Reordering by Statistical Transduction. In *IJCNLP*, pages 38–46, 2011.

Maxim Khalilov and Khalil Sima'an. Statistical translation after source reordering: Oracles, context-aware models, and empirical analysis. *Natural Language Engineering*, 18(4):491519, 2012. doi: 10.1017/S1351324912000162.

Philipp Koehn. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition, 2010. ISBN 0521874157, 9780521874151.

Philipp Koehn. Simulating human judgment in machine translation evaluation campaigns. In *Proceedings of International Workshop on Spoken Language Translation*, 2012.

Philipp Koehn and Christof Monz. Manual and automatic evaluation of machine translation between european languages. In *Proceedings of the Workshop on Statistical Machine Translation*, StatMT '06, pages 102–121, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1654650.1654666`.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics, 2003.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1557769.1557821`.

Shankar Kumar and William Byrne. Minimum Bayes-Risk Decoding for Statistical Machine Translation. In *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*, 2004.

Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Och. Efficient minimum error rate training and minimum bayes-risk decoding for translation hypergraphs and lattices. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, ACL '09, pages 163–171, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-45-9.

Tzu-Ming Kuo, Ching-Pei Lee, and Chih-Jen Lin. *Large-scale Kernel RankSVM*, pages 812–820. SIAM, 2014.

Mirella Lapata. Automatic Evaluation of Information Ordering: Kendall's Tau. *Computational Linguistics*, 32(4):471–484, 2006.

Karim Lari and Steve J. Young. The Estimation of Stochastic Context-Free Grammars using the Inside-Outside Algorithm. *Computer Speech and Language*, 4:35–56, 1990.

Alon Lavie and Abhaya Agarwal. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 228–231, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

Alon Lavie, Kenji Sagae, and Shyamsundar Jayaraman. *The Significance of Recall in Automatic Metrics for MT Evaluation*, pages 134–143. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-30194-3.

Uri Lerner and Slav Petrov. Source-Side Classifier Preordering for Machine Translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 513–523. Association for Computational Linguistics, 2013.

Philip M. Lewis, II and Richard E. Stearns. Syntax-directed transduction. *J. ACM*, 15 (3):465–488, July 1968. ISSN 0004-5411. doi: 10.1145/321466.321477.

Hang Li. *Learning to Rank for Information Retrieval and Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2011.

Percy Liang, Ben Taskar, and Dan Klein. Alignment by Agreement. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 104–111, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. doi: 10.3115/1220835.1220849. URL http://dx.doi.org/10.3115/1220835.1220849.

Chin-Yew Lin and Franz Josef Och. ORANGE: A Method for Evaluating Automatic Evaluation Metrics for Machine Translation. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics. doi: 10.3115/1220355.1220427. URL http://dx.doi.org/10.3115/1220355.1220427.

Hsuan-Tien Lin, Chih-Jen Lin, and Ruby C. Weng. A note on platt's probabilistic outputs for support vector machines. *Mach. Learn.*, 68(3):267–276, October 2007. ISSN 0885-6125. doi: 10.1007/s10994-007-5018-6. URL http://dx.doi.org/10.1007/s10994-007-5018-6.

Ding Liu and Daniel Gildea. Syntactic features for evaluation of machine translation. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 25–32, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

Chi-kiu Lo, Anand Karthik Tumuluru, and Dekai Wu. Fully automatic semantic mt evaluation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, WMT '12, pages 243–252, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

Qingsong Ma, Fandong Meng, Daqi Zheng, Mingxuan Wang, Yvette Graham, Wenbin Jiang, and Qun Liu. Maxsd: A neural machine translation evaluation metric optimized by maximizing similarity distance. In Chin-Yew Lin, Nianwen

Xue, Dongyan Zhao, Xuanjing Huang, and Yansong Feng, editors, *Natural Language Understanding and Intelligent Applications: 5th CCF Conference on Natural Language Processing and Chinese Computing and 24th International Conference on Computer Processing of Oriental Languages*, pages 153–161, Kunming, China, 2016. Springer International Publishing. ISBN 978-3-319-50496-4. doi: 10.1007/978-3-319-50496-4_13.

Matouš Macháček and Ondřej Bojar. Results of the wmt14 metrics shared task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 293–301, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W14/W14-3336.

Matouš Macháček and Ondřej Bojar. Results of the WMT13 Metrics Shared Task. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 45–51, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

Gideon Maillette de Buy Wenniger and Khalil Sima'an. A formal characterization of parsing word alignments by synchronous grammars with empirical evidence to the itg hypothesis. In *Proceedings of the Seventh Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 58–67, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

Gideon Maillette de Buy Wenniger and Khalil Sima'an. Bilingual Markov Reordering Labels for Hierarchical SMT. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 138–147, Doha, Qatar, October 2014.

Gideon Maillette de Buy Wenniger and Khalil Sima'an. Labeling hierarchical phrase-based models without linguistic resources. *Machine Translation*, 29(3):225–265, 2015. ISSN 1573-0573.

Daniel Marcu and William Wong. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 133–139, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1118693.1118711. URL http://dx.doi.org/10.3115/1118693.1118711.

Benjamin Marie and Marianna Apidianaki. Alignment-based sense selection in meteor and the ratatouille recipe. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 385–391, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

Joaquim RRA Martins, Juan J Alonso, and James J Reuther. High-fidelity aerostructural design optimization of a supersonic business jet. *Journal of Aircraft*, 41(3): 523–530, 2004.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. Probabilistic CFG with Latent Annotations. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 75–82, 2005. doi: 10.3115/1219840.1219850. URL http://dx.doi.org/10.3115/1219840.1219850.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

Tetsuji Nakagawa. Efficient top-down btg parsing for machine translation preordering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 208–218, Beijing, China, July 2015. Association for Computational Linguistics.

Preslav Nakov, Francisco Guzmán, and Stephan Vogel. A Tale about PRO and Monsters. In *ACL (2)*, pages 12–17. The Association for Computer Linguistics, 2013. ISBN 978-1-937284-51-0.

Graham Neubig. Neural Machine Translation and Sequence-to-sequence Models: A Tutorial. *ArXiv e-prints*, March 2017.

Graham Neubig, Taro Watanabe, and Shinsuke Mori. Inducing a discriminative parser to optimize machine translation reordering. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 843–853, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

Franz Josef Och. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 160–167, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1075096.1075117. URL http://dx.doi.org/10.3115/1075096.1075117.

Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL http://dx.doi.org/10.3115/1073083.1073135.

Slav Petrov and Dan Klein. Improved Inference for Unlexicalized Parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the*

Association for Computational Linguistics; Proceedings of the Main Conference, pages 404–411, Rochester, New York, April 2007.

Slav Petrov and Dan Klein. Discriminative log-linear grammars with latent variables. In J. C. Platt, D. Koller, Y. Singer, and S.Editors Roweis, editors, *Advances in Neural Information Processing Systems 20 (NIPS)*, pages 1153–1160. MIT Press, 2008.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia, July 2006. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P06/P06-1055`.

John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pages 61–74. MIT Press, 1999.

Maja Popović. chrf: character n-gram f-score for automatic mt evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL `http://aclweb.org/anthology/W15-3049`.

Detlef Prescher. Inducing Head-Driven PCFGs with Latent Heads: Refining a Tree-Bank Grammar for Parsing. In *In ECML05*, 2005.

Miguel Rios, Wilker Aziz, and Lucia Specia. Tine: A metric to assess mt adequacy. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 116–122, Edinburgh, Scotland, July 2011. Association for Computational Linguistics.

Markus Saers, Karteek Addanki, and Dekai Wu. From Finite-State to Inversion Transductions: Toward Unsupervised Bilingual Grammar Induction. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8-15 December 2012, Mumbai, India*, pages 2325–2340, 2012.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. Efficient elicitation of annotations for human evaluation of machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 1–11, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W14-3301`.

Avneesh Saluja, Chris Dyer, and Shay B. Cohen. Latent-Variable Synchronous CFGs for Hierarchical Translation. *Proceedings of EMNLP*, 2014.

Rico Sennrich and Barry Haddow. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation*, pages 83–91, Berlin, Germany, August 2016. Association for Computational Linguistics.

Christophe Servan, Alexandre Berard, Zied Elloumi, Hervé Blanchon, and Laurent Besacier. Word2vec vs dbnary: Augmenting METEOR using vector representations or lexical resources? In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*, pages 1159–1168, 2016.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1692, Berlin, Germany, August 2016. Association for Computational Linguistics.

Khalil Sima'an. Computational complexity of probabilistic disambiguation by means of tree grammar. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, pages 1175–1180, 1996.

Khalil Sima'an and Gideon Maillette de Buy Wenniger. Hierarchical translation equivalence over word alignments, 2011.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231, 2006.

J. Sobieszczanski-Sobieski and R. T. Haftka. Multidisciplinary aerospace design optimization: survey of recent developments. *Structural optimization*, 14(1):1–23, Aug 1997. ISSN 1615-1488.

Anders Søgaard. *Semi-Supervised Learning and Domain Adaptation in Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2013. ISBN 9781608459865.

Xingyi Song and Trevor Cohn. Regression and Ranking based Optimisation for Sentence Level MT Evaluation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 123–129, Edinburgh, Scotland, July 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W11-2113`.

Lucia Specia and Jesús Giménez. Combining Confidence Estimation and Reference-based Metrics for Segment-level MT Evaluation. In *Ninth Conference of the Association for Machine Translation in the Americas*, AMTA-2010, Denver, Colorado, 2010. URL `http://www.mt-archive.info/AMTA-2010-Specia.pdf`.

Edward P. Stabler. Two models of minimalist, incremental syntactic analysis. *Topics in Cognitive Science*, 5(3):611–633, 2013. ISSN 1756-8765. doi: 10.1111/tops.12031.

Felix Stahlberg, Eva Hasler, Aurelien Waite, and Bill Byrne. Syntactically guided neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 299–305, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://anthology.aclweb.org/P16-2049.

Felix Stahlberg, Adrià de Gispert, Eva Hasler, and Bill Byrne. Neural machine translation by minimising the bayes-risk with respect to syntactic translation lattices. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 362–368, Valencia, Spain, April 2017. Association for Computational Linguistics.

Miloš Stanojević and Khalil Sima'an. Fitting Sentence Level Translation Evaluation with Many Dense Features. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 202–206, Doha, Qatar, October 2014a. Association for Computational Linguistics.

Miloš Stanojević and Khalil Sima'an. Evaluating Word Order Recursively over Permutation-Forests. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 138–147, Doha, Qatar, October 2014b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W14-4017.

Miloš Stanojević and Khalil Sima'an. BEER: BEtter Evaluation as Ranking. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 414–419, Baltimore, Maryland, USA, June 2014c. Association for Computational Linguistics.

Miloš Stanojević and Khalil Sima'an. Reordering Grammar Induction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 44–54, Lisbon, Portugal, September 2015a. Association for Computational Linguistics. URL http://aclweb.org/anthology/D15-1005.

Miloš Stanojević and Khalil Sima'an. Evaluating MT systems with BEER. *The Prague Bulletin of Mathematical Linguistics*, 104:17–26, 2015b.

Miloš Stanojević and Khalil Sima'an. BEER 1.1: ILLC UvA submission to metrics and tuning task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 396–401, Lisbon, Portugal, September 2015c. Association for Computational Linguistics. URL http://aclweb.org/anthology/W15-3050.

Miloš Stanojević and Khalil Sima'an. Hierarchical permutation complexity for word order evaluation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, Osaka, Japan, December 2016. Association for Computational Linguistics.

Miloš Stanojević and Khalil Sima'an. Alternative objective functions for training mt evaluation metrics. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada, August 2017. Association for Computational Linguistics.

Miloš Stanojević, Amir Kamran, and Ondřej Bojar. Results of the wmt15 tuning shared task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 274–281, Lisbon, Portugal, September 2015a. Association for Computational Linguistics.

Miloš Stanojević, Amir Kamran, Philipp Koehn, and Ondřej Bojar. Results of the wmt15 metrics shared task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 256–273, Lisbon, Portugal, September 2015b. Association for Computational Linguistics.

Jinsong Su, Zhixing Tan, Deyi Xiong, Rongrong Ji, Xiaodong Shi, and Yang Liu. Lattice-based recurrent neural network encoders for neural machine translation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, pages 3302–3308, 2017.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proc. NIPS*, 2014.

David Talbot, Hideto Kazawa, Hiroshi Ichikawa, Jason Katz-Brown, Masakazu Seno, and Franz J. Och. A Lightweight Evaluation Framework for Machine Translation Reordering. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 12–21, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-937284-12-1. URL `http://dl.acm.org/citation.cfm?id=2132960.2132963`.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-Margin Markov Networks. In *NIPS 2014 - Advances in Neural Information Processing Systems 27*, 2003.

Christoph Tillmann. A Unigram Orientation Model for Statistical Machine Translation. In *Proceedings of HLT-NAACL 2004: Short Papers*, HLT-NAACL-Short '04, pages 101–104, 2004. ISBN 1-932432-24-8. URL `http://dl.acm.org/citation.cfm?id=1613984.1614010`.

Roy Tromble and Jason Eisner. Learning Linear Ordering Problems for Better Translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1007–1016, Singapore, August 2009. URL `http://cs.jhu.edu/~jason/papers/#tromble-eisner-2009`.

Takeaki Uno and Mutsunori Yagiura. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2):290–309, 2000. ISSN 1432-0541. doi: 10.1007/s004539910014.

Bernard Vauquois. A survey of formal grammars and algorithms for recognition and transformation in mechanical translation. In J.H. Morrell, editor, *Proceedings of the International Federation for Information Processing Congress (IFIP-68)*, volume 2, pages 1114–1122, Edinburgh, Scotland, August 1968.

Karthik Visweswariah, Rajakrishnan Rajkumar, Ankur Gandhe, Ananthakrishnan Ramanathan, and Jiri Navratil. A word reordering model for improved machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 486–496, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-937284-11-4.

Wilhelm von Humboldt. *ber die Verschiedenheit des menschlichen Sprachbaues und ihren Einflu auf die geistige Entwicklung des Menschengeschlechtes*. 1836. Reprint: Humboldt 1972:368-756.

Weiyue Wang, Jan-Thorsten Peter, Hendrik Rosendahl, and Hermann Ney. Character: Translation edit rate on character level. In *Proceedings of the First Conference on Machine Translation*, pages 505–510, Berlin, Germany, August 2016. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W/W16/W16-2342`.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. Online large-margin training for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 764–773, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Benjamin Wellington, Sonjia Waxmonsky, and I. Dan Melamed. Empirical lower bounds on the complexity of translational equivalence. In *In Proceedings of ACL 2006*, pages 977–984, 2006.

Dekai Wu. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Computational Linguistics*, 23(3):377–403, September 1997. ISSN 0891-2017.

Fei Xia and Michael Mccord. Improving a Statistical MT System with Automatically Learned Rewrite Patterns. In *Proceedings of Coling 2004*, pages 508–514, Geneva, Switzerland, August 2004. COLING. URL `http://www.mt-archive.info/Coling-2004-Xia.pdf`.

Bing Xiang, Xiaoqiang Luo, and Bowen Zhou. Enlisting the ghost: Modeling empty categories for machine translation. In *Proceedings of ACL*, pages 822–831, 2013.

Yang Ye, Ming Zhou, and Chin-Yew Lin. Sentence Level Machine Translation Evaluation As a Ranking Problem: One Step Aside from BLEU. In *Proceedings of*

the Second Workshop on Statistical Machine Translation, StatMT '07, pages 240–247, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1626355.1626391.

Daniel H. Younger. Recognition and parsing with context-free languages in time n3. *Information and Control*, 10(2):189–208, 1967.

Hui Yu, Xiaofeng Wu, Jun Xie, Wenbin Jiang, Qun Liu, and Shouxun Lin. Red: A reference dependency based mt evaluation metric. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2042–2051, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics.

Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 609–616, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-778-1. URL http://dl.acm.org/citation.cfm?id=645530.655658.

Richard Zens, Franz Josef Och, and Hermann Ney. *Phrase-Based Statistical Machine Translation*, pages 18–32. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.

Hao Zhang and Daniel Gildea. Factorization of Synchronous Context-Free Grammars in Linear Time. In *NAACL Workshop on Syntax and Structure in Statistical Translation (SSST)*, pages 25–32, 2007.

Hao Zhang, Daniel Gildea, and David Chiang. Extracting Synchronous Grammar Rules from Word-level Alignments in Linear Time. In *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 1081–1088, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. ISBN 978-1-905593-44-6. URL http://dl.acm.org/citation.cfm?id=1599081.1599217.

Andreas Zollmann and Ashish Venugopal. Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation*, StatMT '06, pages 138–141, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

Samenvatting

Natuurlijke taal kent een beperkte ruimte voor variatie in de woordvolgorde van talige uitingen. Taalwetenschappelijk gezien resulteert woordvolgorde uit de herhaalde toepassing van recursieve syntactische functies. Deze syntactische operaties brengen hierarchische syntactische structuren voort, alsmede een keten woorden die in een bepaalde volgorde verschijnen.

Verschillende talen worden echter door verschillende syntactische regels geregeerd. Een van de hoofdvraagstukken voor machinevertaling is dus het vinden van de relatie tussen de woordvolgorde in de brontaal en die in de doeltaal. Dit word vaak gedaan door middel van syntactische transfer, waar een syntactische boomstructuur uit de bronzin wordt herleid en vervolgens omgezet in een structuur dat overeen komt met de syntactische regels van de doeltaal.

In deze dissertatie stel ik een alternatief voor syntactische transfer dat de goede eigenschappen van deze methode behoudt—namelijk de compositionele en hierarchische structuur—maar dat, in tegenstelling tot syntactische transfer, rechtstreeks uit de data herleid wordt, zonder taalwetenschappelijke annotaties te behoeven. Deze benadering heeft twee hoofdvoordelen. Ten eerste maakt het mogelijk hierarchische herrangschikking op talen toe te passen waarvoor geen parsers bestaan. Ten tweede, in tegenstelling tot de in syntactische transfer gebruikte boomstructuren, die niet altijd met de herrangschikkingspatronen van de data overeenkomen, worden de boomstructuren in deze werk rechtstreeks uit herrangschikkingspatronen herleid, en komen zo per definitie daarmee overeen.

Ik behandel herrangschikking als het probleem van het voorspellen van de permutatie van bronwoorden dat de doelvolgorde het beste nadert. Deze permutatie can recursief in een hierarchische structuur opgebroken worden, ook genoemd *permutatie boom* (*permutation tree*, of PET) (Zhang and Gildea, 2007). In bepaalde gevallen kunnen meerdere permutatie bomen eenzelfde permutatie genereren. Deze set permutatie bomen heet een *permutatie forest*. Een permutatie forest geeft een rijkere representatie van een permutatie omdat het elke mogelijke segmentatie van de permutatie inhoudt. Hele forests zijn daarom meer aantrekkelijk voor het modelleren van permutaties dan

enkel bomen.

Ik pas permutatie bomen in twee subtaken van machinevertaling toe: voorspelling en evaluatie van woordvolgorde. In de woordvolgorde voorspellingstaak stel ik een probabilistische model voor dat non-terminals en de bracketing van een zin beide als latente variabelen behandelt. In het geval van evaluatie in machinevertaling, stel ik evaluatie metrieken voor die PET's gebruiken, en gebruik ik machine learning methoden om dichter by menselijke beoordeling van vertalingskwaliteit te komen.

De hier voorgestelde permutatieboom modellen zijn (i) compositioneel, (ii) hierarchisch en (iii) rechtstreeks van ongeannoteerde vertalingsdata herleid. Empirisch gezien hebben modellen met deze drie eigenschappen bewezen vertalingskwaliteit te verbeteren, en laten ook meer correlatie met menselijke beoordelingen zien wanneer ze voor machinevertalingsevaluatie ingezet worden.

Abstract

In natural language, there is only a limited space for variation in the word order of linguistic productions. From a linguistic perspective, word order is the result of multiple application of syntactic recursive functions. These syntactic operations produce hierarchical syntactic structures, as well as a string of words that appear in a certain order.

However, different languages are governed by different syntactic rules. Thus, one of the main problems in machine translation is to find the mapping between word order in the source language and word order in the target language. This is often done by a method of syntactic transfer, in which the syntactic tree is recovered from the source sentence, and then transduced so that its form is consistent with the syntactic rules of the target language.

In this dissertation, I propose an alternative to syntactic transfer that maintains its good properties—namely the compositional and hierarchical structure—but, unlike syntactic transfer, it is directly derived from data without requiring any linguistic annotation. This approach brings two main advantages. First, it allows for applying hierarchical reordering even on languages for which there are no syntactic parsers available. Second, unlike the trees used in syntactic transfer, which in some cases cannot cover the reordering patterns present in the data, the trees used in this work are built directly over the reordering patterns, so they can cover them by definition.

I treat reordering as a problem of predicting the permutation of the source words which permutes them into an order that is as close as possible to the target side order. This permutation can be recursively decomposed into a hierarchical structure called a *permutation tree* (PET) (Zhang and Gildea, 2007). In some cases there can be many permutation trees that can generate the same permutation. This set of permutation trees is called *permutation forest*. A permutation forest is a richer representation of a permutation because it covers all possible segmentations consistent with the permutation, so modeling permutations over the whole forest is a more promising approach than modeling a single tree.

I apply permutation trees in two sub-tasks of machine translation: word order pre-

diction and word order evaluation. In the word order prediction scenario I propose a probabilistic model that treats both the non-terminals and the bracketing of the sentence as latent variables. In the context of MT evaluation, I propose evaluation metrics that incorporate PETs and use machine learning methods to approximate human judgment of translation quality.

Overall, the permutation tree models proposed here are (i) compositional, (ii) hierarchical and (iii) directly derived from unannotated translation data. Empirically, the models satisfying these three properties have been shown to improve translation quality, and provide better correlation with human judgment when used for evaluation of machine translation output.

Titles in the ILLC Dissertation Series:

ILLC DS-2009-01: **Jakub Szymanik**
Quantifiers in TIME and SPACE. Computational Complexity of Generalized Quantifiers in Natural Language

ILLC DS-2009-02: **Hartmut Fitz**
Neural Syntax

ILLC DS-2009-03: **Brian Thomas Semmes**
A Game for the Borel Functions

ILLC DS-2009-04: **Sara L. Uckelman**
Modalities in Medieval Logic

ILLC DS-2009-05: **Andreas Witzel**
Knowledge and Games: Theory and Implementation

ILLC DS-2009-06: **Chantal Bax**
Subjectivity after Wittgenstein. Wittgenstein's embodied and embedded subject and the debate about the death of man.

ILLC DS-2009-07: **Kata Balogh**
Theme with Variations. A Context-based Analysis of Focus

ILLC DS-2009-08: **Tomohiro Hoshi**
Epistemic Dynamics and Protocol Information

ILLC DS-2009-09: **Olivia Ladinig**
Temporal expectations and their violations

ILLC DS-2009-10: **Tikitu de Jager**
"Now that you mention it, I wonder…": Awareness, Attention, Assumption

ILLC DS-2009-11: **Michael Franke**
Signal to Act: Game Theory in Pragmatics

ILLC DS-2009-12: **Joel Uckelman**
More Than the Sum of Its Parts: Compact Preference Representation Over Combinatorial Domains

ILLC DS-2009-13: **Stefan Bold**
Cardinals as Ultrapowers. A Canonical Measure Analysis under the Axiom of Determinacy.

ILLC DS-2010-01: **Reut Tsarfaty**
Relational-Realizational Parsing

ILLC DS-2010-02: **Jonathan Zvesper**
Playing with Information

ILLC DS-2010-03: **Cédric Dégremont**
The Temporal Mind. Observations on the logic of belief change in interactive systems

ILLC DS-2010-04: **Daisuke Ikegami**
Games in Set Theory and Logic

ILLC DS-2010-05: **Jarmo Kontinen**
Coherence and Complexity in Fragments of Dependence Logic

ILLC DS-2010-06: **Yanjing Wang**
Epistemic Modelling and Protocol Dynamics

ILLC DS-2010-07: **Marc Staudacher**
Use theories of meaning between conventions and social norms

ILLC DS-2010-08: **Amélie Gheerbrant**
Fixed-Point Logics on Trees

ILLC DS-2010-09: **Gaëlle Fontaine**
Modal Fixpoint Logic: Some Model Theoretic Questions

ILLC DS-2010-10: **Jacob Vosmaer**
Logic, Algebra and Topology. Investigations into canonical extensions, duality theory and point-free topology.

ILLC DS-2010-11: **Nina Gierasimczuk**
Knowing One's Limits. Logical Analysis of Inductive Inference

ILLC DS-2010-12: **Martin Mose Bentzen**
Stit, Iit, and Deontic Logic for Action Types

ILLC DS-2011-01: **Wouter M. Koolen**
Combining Strategies Efficiently: High-Quality Decisions from Conflicting Advice

ILLC DS-2011-02: **Fernando Raymundo Velazquez-Quesada**
Small steps in dynamics of information

ILLC DS-2011-03: **Marijn Koolen**
The Meaning of Structure: the Value of Link Evidence for Information Retrieval

ILLC DS-2011-04: **Junte Zhang**
System Evaluation of Archival Description and Access

ILLC DS-2011-05: **Lauri Keskinen**
Characterizing All Models in Infinite Cardinalities

ILLC DS-2011-06: **Rianne Kaptein**
Effective Focused Retrieval by Exploiting Query Context and Document Structure

ILLC DS-2011-07: **Jop Briët**
Grothendieck Inequalities, Nonlocal Games and Optimization

ILLC DS-2011-08: **Stefan Minica**
Dynamic Logic of Questions

ILLC DS-2011-09: **Raul Andres Leal**
Modalities Through the Looking Glass: A study on coalgebraic modal logic and their applications

ILLC DS-2011-10: **Lena Kurzen**
Complexity in Interaction

ILLC DS-2011-11: **Gideon Borensztajn**
The neural basis of structure in language

ILLC DS-2012-01: **Federico Sangati**
Decomposing and Regenerating Syntactic Trees

ILLC DS-2012-02: **Markos Mylonakis**
Learning the Latent Structure of Translation

ILLC DS-2012-03: **Edgar José Andrade Lotero**
Models of Language: Towards a practice-based account of information in natural language

ILLC DS-2012-04: **Yurii Khomskii**
Regularity Properties and Definability in the Real Number Continuum: idealized forcing, polarized partitions, Hausdorff gaps and mad families in the projective hierarchy.

ILLC DS-2012-05: **David García Soriano**
Query-Efficient Computation in Property Testing and Learning Theory

ILLC DS-2012-06: **Dimitris Gakis**
Contextual Metaphilosophy - The Case of Wittgenstein

ILLC DS-2012-07: **Pietro Galliani**
The Dynamics of Imperfect Information

ILLC DS-2012-08: **Umberto Grandi**
Binary Aggregation with Integrity Constraints

ILLC DS-2012-09: **Wesley Halcrow Holliday**
Knowing What Follows: Epistemic Closure and Epistemic Logic

ILLC DS-2012-10: **Jeremy Meyers**
Locations, Bodies, and Sets: A model theoretic investigation into nominalistic mereologies

ILLC DS-2012-11: **Floor Sietsma**
Logics of Communication and Knowledge

ILLC DS-2012-12: **Joris Dormans**
Engineering emergence: applied theory for game design

ILLC DS-2013-01: **Simon Pauw**
Size Matters: Grounding Quantifiers in Spatial Perception

ILLC DS-2013-02: **Virginie Fiutek**
Playing with Knowledge and Belief

ILLC DS-2013-03: **Giannicola Scarpa**
Quantum entanglement in non-local games, graph parameters and zero-error information theory

ILLC DS-2014-01: **Machiel Keestra**
Sculpting the Space of Actions. Explaining Human Action by Integrating Intentions and Mechanisms

ILLC DS-2014-02: **Thomas Icard**
The Algorithmic Mind: A Study of Inference in Action

ILLC DS-2014-03: **Harald A. Bastiaanse**
Very, Many, Small, Penguins

ILLC DS-2014-04: **Ben Rodenhäuser**
A Matter of Trust: Dynamic Attitudes in Epistemic Logic

ILLC DS-2015-01: **María Inés Crespo**
Affecting Meaning. Subjectivity and evaluativity in gradable adjectives.

ILLC DS-2015-02: **Mathias Winther Madsen**
The Kid, the Clerk, and the Gambler - Critical Studies in Statistics and Cognitive Science

ILLC DS-2015-03: **Shengyang Zhong**
Orthogonality and Quantum Geometry: Towards a Relational Reconstruction of Quantum Theory

ILLC DS-2015-04: **Sumit Sourabh**
Correspondence and Canonicity in Non-Classical Logic

ILLC DS-2015-05: **Facundo Carreiro**
Fragments of Fixpoint Logics: Automata and Expressiveness

ILLC DS-2016-01: **Ivano A. Ciardelli**
Questions in Logic

ILLC DS-2016-02: **Zoé Christoff**
Dynamic Logics of Networks: Information Flow and the Spread of Opinion

ILLC DS-2016-03: **Fleur Leonie Bouwer**
What do we need to hear a beat? The influence of attention, musical abilities, and accents on the perception of metrical rhythm

ILLC DS-2016-04: **Johannes Marti**
Interpreting Linguistic Behavior with Possible World Models

ILLC DS-2016-05: **Phong Lê**
Learning Vector Representations for Sentences - The Recursive Deep Learning Approach

ILLC DS-2016-06: **Gideon Maillette de Buy Wenniger**
Aligning the Foundations of Hierarchical Statistical Machine Translation

ILLC DS-2016-07: **Andreas van Cranenburgh**
Rich Statistical Parsing and Literary Language

ILLC DS-2016-08: **Florian Speelman**
Position-based Quantum Cryptography and Catalytic Computation

ILLC DS-2016-09: **Teresa Piovesan**
Quantum entanglement: insights via graph parameters and conic optimization

ILLC DS-2016-10: **Paula Henk**
Nonstandard Provability for Peano Arithmetic. A Modal Perspective

ILLC DS-2017-01: **Paolo Galeazzi**
Play Without Regret

ILLC DS-2017-02: **Riccardo Pinosio**
The Logic of Kant's Temporal Continuum

ILLC DS-2017-03: **Matthijs Westera**
Exhaustivity and intonation: a unified theory

ILLC DS-2017-04: **Giovanni Cinà**
Categories for the working modal logician

ILLC DS-2017-05: **Shane Noah Steinert-Threlkeld**
Communication and Computation: New Questions About Compositionality

ILLC DS-2017-06: **Peter Hawke**
The Problem of Epistemic Relevance

ILLC DS-2017-07: **Aybüke Özgün**
Evidence in Epistemic Logic: A Topological Perspective

ILLC DS-2017-08: **Raquel Garrido Alhama**
Computational Modelling of Artificial Language Learning: Retention, Recognition & Recurrence

ILLC DS-2017-09: **Miloš Stanojević**
Permutation Forests for Modeling Word Order in Machine Translation