Training Data Annotation for Segmentation Classification in Simultaneous Translation

by

Sayyedhassan Shavarani

B.Sc., Amirkabir University of Technology, 2014

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Spring 2016

Approval

Name:	**Sayyedhassan Shavarani**
Degree:	**Master of Science**
Title:	***Training Data Annotation for Segmentation Classification in Simultaneous Translation***
Examining Committee:	**Chair:** Arrvindh Shriraman Assistant Professor

Anoop Sarkar
Senior Supervisor
Professor _____

Fred Popowich
Supervisor
Professor _____

William D. Lewis
External Examiner
Microsoft Research
Principal Technical Project Manager
Affiliate Assistant Professor
Department of Linguistics
University of Washington _____

Date Defended: 9 May 2016

Abstract

Segmentation of the incoming speech stream and translating segments incrementally is a commonly used technique that improves latency in spoken language translation. Previous work of Oda et al. 2014 [1] has explored creating training data for segmentation by finding segments that maximize translation quality with a user-defined bound on segment length. In this work we provide a new algorithm that uses Pareto-optimality to find good segment boundaries that can balance the trade-off between latency versus translation quality. We compare against the state-of-the-art greedy algorithm from Oda et al. 2014. Our experimental results show that we can improve latency by up to 12% without harming the BLEU score for the same average segment length. Another benefit is that for any segment size, Pareto-optimal segments maximize both latency and translation quality.

Keywords: Simultaneous Translation; Machine Translation; Segmentation; Simon Fraser University

Dedication

It is my genuine gratefulness and warmest regard that I dedicate this work to my father, who taught me that the best kind of knowledge to have is that which is learned for its own sake. It is also dedicated to my mother, who taught me that even the largest task can be accomplished if it is done one step at a time.

Acknowledgements

It is with immense gratitude that I acknowledge the support and help of my supervisor Dr. Anoop Sarkar for his full support, expert guidance, understanding and encouragement throughout my study and research. Without his incredible patience and timely wisdom and counsel, my thesis work would have been a frustrating and overwhelming pursuit. In addition, I express my appreciation to Dr. Fred Popowich, Dr. William D. Lewis, and Dr. Arrvindh Shriraman for having served on my committee. I am gratefully indebted to their very valuable comments on this thesis. I thank my fellow labmates in Natlang Lab, as well. Specifically, I am indebted to my colleagues Maryam Siahbani and Ramtin Mehdizadeh Seraj for their support and help throughout this academic experience.

I am grateful to Yusuke Oda and Graham Neubig for the paper that inspired this thesis, as well as for the discussions carried on long-distance via e-mail. Yusuke also provided me with his original segmenter code. All this was only possible because of the Internet, so I would like to thank all the organizations worldwide responsible for its upkeep.

I wish to thank all my friends who helped me get through two years of graduate school. Some friends, like Sina Bahrasemani and Masoud Hashemian, were tolerant enough (or dumb enough) to live with me during this time. These fellows helped me learn music and accomplish one of the most interesting objectives of my life, understanding the language of music and playing an instrument. I lived in a place where I had the chance to neighbour Mohsen, Abdollah, Shahram, Akbar, Sasan, Kazem, Soheil, Saeed, and Sajjad. I am happy to thank these individuals who have always helped me to keep my life in context and assisted me to have a colorful and exciting life in Vancouver. I would also like to send my thanks out to my wonderful friends Mohammadreza and Mahshad for their support and friendship from miles away. Graduate school isn't the most important thing in life, but good friends, good times and happiness are.

Finally, I would like to thank my loved ones, who have supported me throughout the entire process, both by keeping me harmonious and helping me putting pieces together. I would not have been able to complete this thesis without their continued love and encouragement. I will be grateful forever for your love.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Simultaneous Translation (ST) is a real-time form of Machine Translation (MT) and focuses on the translation of the sentence before it comes to the end. ST systems assist people to communicate in languages they don't speak and understand (see Figure 1.1). Since the Paris Peace Conference of 1919, human interpreters have been used in situations where speakers and hearers have different native languages and where pre-translation or post-translation is not possible. Some estimate that the Nuremberg Trials in 1945 and 1956 would have taken ten years if all the material were translated in a batch mode compared to real-time conference interpretation by expert human translators. It is a wide-open research question if an MT system can replace humans in this capacity. This would create several additional opportunities for the use of translation technology. However, the task of taking machine translation systems into real-time conference interpretation is not trivial. There are many challenges to be overcome before an MT system can be competitive in terms of latency, quality and creativity compared to existing expert human interpreters. Thus the goal of simultaneous translation is to enable more natural spoken multilingual interactions through machine translation. This thesis targets one of the major current issues in simultaneous translation and the rest of this chapter tries to justify the definition of the domain of the problem. To this end, we first briefly overview the history of interpretation and simultaneous translation, and next, we will motivate the problem and specify our contributions to the problem.

1.1 The Birth of Conference Interpreting

Jesús Baigorri-Jalón [2] makes the case that the birth of simultaneous translation dates back to the Paris Peace Conference of 1919 when the representatives of the nations gathered together after World War I to decide on the strategies to spread peace around the world. This conference is important since it was the first European multi-nation conference after many years in which French was not the only official language. This conference had two

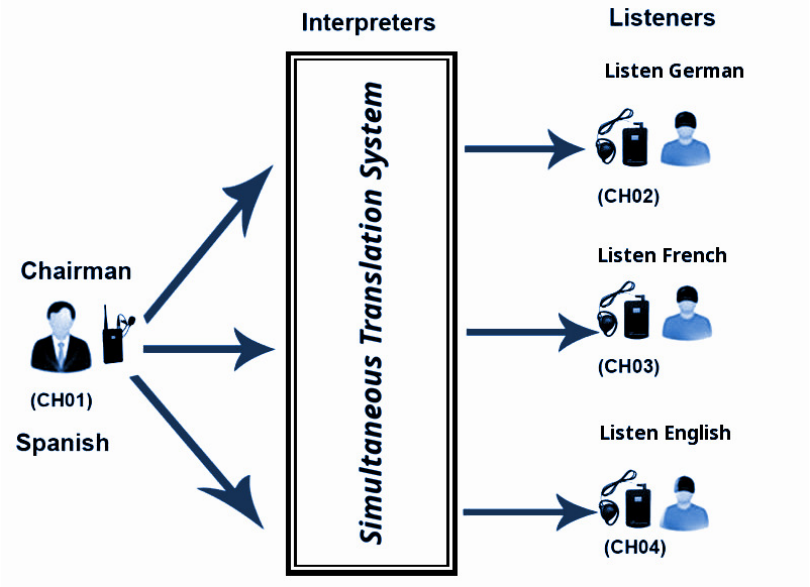

Figure 1.1: An example use of a Simultaneous Translation system

official languages English and French and people could use either of them to represent their ideas and communicate with the other diplomats. This was the reason for the urgent need to train interpreters to provide speech to speech simultaneous translation, on either of the languages not understood by diplomats.

Choosing interpreters was difficult in 1919 due to the absence of the definition of interpreter's professional activity. Since interpretation was practiced by chance, learned on the job and it was a temporary occupation with no necessity on dedication to work. Therefore no one could determine a good interpreter from a bad one using a specified test. That's why Jean Herbert says "conference interpreting only actually started during the first World War".

Early definitions of conference interpretation define four modes to it.

First *Long Consecutive* at which the interpreter had to listen to the whole speech, take notes and provide the translations after the speech. He also had to draft the minutes based on his notes.

Second *Short Consecutive* at which the interpreter had to listen to either of the diplomats says a few sentences and translate and tell them to the other diplomat. Taking notes have not been possible in this case due to the pace of communication.

Third *Sight Translation of Documents* at which the interpreter had to read the document in a language but say the translations instead while reading.

Fourth *Chuchotage* at which the interpreter was sitting next to the person whispering the interpretations to his ear as the speaker went along.

This occupation has been fairly difficult at the time since the interpretation had been not necessarily sentence by sentence (the worst case has been the long consecutive mode) and could be done in both directions, for example both French into English and English to French. This fact means that the interpreter should have had an exceptional memory as well as much amounts of attention and awareness regardless of fatigue and getting bored by the subject. In addition technical equipment has not been available at the time and the interpreter, sometimes, had to shout in the ears of the diplomat to help him hear the translations. Next problem for any interpreter had been dealing with different accents of the source language being presented while there could have been a very wide variety of different accents in it. Furthermore, since they did not know what the speakers are going to talk about they were not able to prepare themselves from before. Finally, the interpreters had reported that the diplomats were insisting in getting monotone (word-by-word) translations which might not have been good in quality. They reported that sometimes diplomats had spoken for a few seconds and then they said "correct it, it is not what I meant to say", so they believed it is good to buffer a few words and then start providing the translation.

All of these points together made the evaluation of interpretation a very hard and unclear task. Although, sometimes bilingual people were used to listen to both original and interpreted speech and evaluate the interpreters, but the quality of speeches and the noisy state of the auditorium (people who could not understand either of the languages started to talk to each other), made it very difficult to concentrate on the topic (while sometimes the interpreters themselves felt bored and lost their attention). In addition to the concentration problem, sometimes interpreters produced much more or much less information from the original and comparing the two could have been very difficult in such cases. The next point has been the relation between the quality and subjective criteria. The evaluator should have had sufficient knowledge about the topic to evaluate the translation quality. And the last point in evaluation has been the difference in the ideas about the interpretation. Some had believed that the interpretation can say something other than the words of the original speech while it is providing the content, and the interpretation can smooth the language or make it proper for the audience. The other believe that the translation should reserve fidelity to the speaker. Evaluation can be a very different problem from either of these perspectives.

Taking into account all of this information, we aim to define our problem in simulating the interpretation task in the area of a machine translation. We aim to use the long history of Machine Translation to attain a baseline for our simultaneous translation system. Given this brief history, the next section motivates the problem which this thesis is tackling in modern machine-based interpretation which is also called simultaneous translation.

1.2 Motivation

Conventional machine translation systems translate a sentence at a time. Such a strategy in translation guarantees a good translation quality but the delay in producing a translation is unacceptable in a simultaneous translation setting. Monotonic translation (word by word) is possible with current translation systems but the translations are lower quality in such a system due to the lack of reordering between the source language and the target language. Simultaneous translation (ST) systems try to solve the problem of long translation waits by finding the points between the two consecutive words in the stream which can split the stream into shorter chunks, called segments. Segmenting the stream on these points and translating each segment separately may decrease the translation accuracy since the rest of the sentence is not known while translating the earlier segments. However, despite this shortcoming the ST system will be able to provide faster translations and decrease translation delivery delays. Trading translation accuracy for achieving shorter delays (minimizing the translation latency) is a common strategy in ST systems. The segmentation points which can help achieve the best possible trade-off are called *optimal segmentation points*.

In addition to segmentation, predicting what words may come next, and paraphrasing some phrases may help the quality of a ST system. Furthermore, since the translation quality is not the only measure in the description of performance for a ST system, evaluating such a system can be one other major issue to be carefully taught about (see section 3.1). Graham Neubig in his presentation slides [3] summarizes previous literature in this area and makes the case that simultaneous translation has four challenges: segmentation, prediction, paraphrasing, evaluation. Although, we do not get a chance to perform experiments in all four major topics in ST in this thesis, we cover two of these important topics: segmentation and evaluation.

In this thesis, we will try to find optimal segmentation points between the two adjacent words in the stream. To perform this idea, the ST system needs a classifier which should classify each point as to be a segmentation point or not. This segmentation classifier, like any other supervised classifier, needs a segmented corpus to be trained on; however, as the date of writing this thesis, segmentation for ST is not a solved problem and there are no high quality annotated corpora for such a task. One of the reasons for this problem is that the research community working on ST is still searching for a unified strategy for creating such data and has not agreed on one yet. This is why this thesis will focus on the algorithms which annotate an unsegmented corpus with the segmentation points and will assess the quality of this annotated data.

1.3 Contributions

In this work, we extend previous work [1] on finding optimal segments and provide a more appealing algorithm, using Pareto-optimality (described in section 3.4), for finding good segment boundaries that can balance the trade-off between latency and translation quality. Using data that was produced by simultaneous translation by human interpreters, the study in Mieno et al. [4] considers how humans view the trade-off between latency and translation quality. What they found was that humans were very sensitive to translation quality, and this implies that we need algorithms that can make a careful choice between different segmentation decisions of the same latency to produce translations with the best translation quality possible (for that latency). In this thesis, we provide efficient algorithms to find segmentation decisions that explicitly rank these decisions based on the trade-off between latency and translation quality. In addition, we introduce a more accurate latency measure which will enhance the ability to deal with the concept of time in Simultaneous Translation.

We provide experimental results to evaluate our approach on the English-German TED talk translation task which uses data from the IWSLT shared task data from 2010 to 2013. We provide our experiments on one language pair because training on each language pair takes several months (as will become apparent when our method is described in detail). The English-German pair was chosen because we had access to more resources for these languages. In addition, English-German is a challenging direction for the translation task and good results in this language pair should be indicative of success in some other European language pairs as well.

The results of our experiments show that we can provide qualitatively better segments (compared to previous work) that improve latency without substantially hurting translation quality. In the second part of our experiments, we perform a set of experiments to analyze the impact of the feature space selection in increase/decrease of the segmentation quality in the results.

1.4 An Overview of This Thesis

The chapters in this thesis are organized as follows:

Chapter 2 provides relevant background from the Machine Learning and Machine Translation literature. This chapter will cover definitions and explanations of Statistical Machine Translation approaches from different perspectives. The chapter ends with an introductory section about different translation quality evaluation measures and provides and example to the one which is more important to this thesis.

Chapter 3 provides the background information about Simultaneous Translation and explains the concept of segmentation and relates it to Simultaneous Translation in detail. The state-of-the-art Greedy Segmentation approach is reviewed in this chapter, as the basis for our Pareto-Optimal method. The chapter ends with an introductory section about the concept of Pareto-Optimality.

Chapter 4 provides sufficient detail about the method of modeling latency and translation accuracy in our Pareto-Optimal segmentation approach. The chapter ends with the detailed description of the algorithm which trades off latency and translation accuracy in the unsupervised segmentation task.

Chapter 5 provides relevant information about the designed experiments to prove the superiority of our suggested algorithm to the other available segmentation algorithms and provides the experimental results and provides the analysis of the provided algorithms.

Chapter 6 concludes the thesis, reviews the most important points about it and mentions what has to be done in the future.

Chapter 2

Machine Translation

In this chapter, we review the most important background information about Machine Translation which is used in the rest of the thesis. Please note that our focus in this thesis will be on Statistical Machine Translation and we do not cover other Machine Translation types such as Neural Machine Translation. In this chapter, we first describe the generative solution process of machine translation and introduce the pipeline of the generative model in Machine Translation. Next, we will explain the discriminative machine translation process and we will highlight its difference with the generative process. Then, we will define different commonly used machine translation quality evaluation metrics. In the rest of the chapter, we will define the idea of Simultaneous Translation as a real-time form of Statistical Machine Translation. And, we will end the chapter with describing the idea of Pareto-Optimality which is going to be used in the next chapters.

2.1 Generative Machine Translation

Machine Translation (MT) is a computational approach to translate utterances of one natural language to another one using mathematical models and algorithms. *Statistical Machine Translation* (SMT) is an approach to Machine Translation which uses Machine Learning techniques (e.g. maximum likelihood estimation) to perform the task of translation. Approaches to SMT can be either *Generative* (described in this section) or *Discriminative* (described in section 2.2).

Generative methods in SMT try to translate a given sentence using estimations over probability distributions (based on counts) of words (or phrases) in the given training data. This training data (input to the generative process, also called *parallel corpus or bitext*) usually contains a number of sentences in the source language (F) and the corresponding human translations in the target language (E). A parallel corpus (containing N parallel sentences) will be of the form Equation (2.1) in which each f_i represents a source sentence and each e_i is supposed to be the corresponding translation of it.

$$Parallel\_Corpus = \{(f_i, e_i) \mid i = 1 \dots N\} \tag{2.1}$$

The assumption in the process of generative SMT is that sentences are independent of each other and, therefore, each sentence contains enough information to be used for translating it. This assumption results in the definition of the *translation input unit* of the generative process which is a *sentence*. So, to start the translation, we need to know the whole sentence from the beginning to the end.

Given a sentence (f) in the source language, our generative SMT system tries to find the most probable translation (e) in the target language among all possible translations (Equation 2.2).

$$e = \underset{e' \in E}{\operatorname{argmax}} P(e'|f) \tag{2.2}$$

This step is called *'Decoding'* due to a memorandum published in 1949, from Warren Weaver, one of the pioneering minds in machine translation [5]. One of the pages of the memo reads:

> *When I look at an article in Russian, I say "This is really written in English, but it has been coded in some strange symbols." I will now proceed to decode.*

Clearly, it is impossible to explore the whole space of possible sentences in the target language to find the most probable one based on Equation (2.2). Generative SMT uses the input parallel corpora, introduced in this chapter, to compute the result of this *maximum likelihood estimation* formula in an indirect way. Using *Bayes rule*, we can write the probability distribution function of Equation (2.2) as Equation (2.3).

$$P(e'|f) = \frac{P(e') \cdot P(f|e')}{P(f)} \tag{2.3}$$

So, taking the argmax of Equation (2.3) will result in the indirect procedure of computing the best translation for f in Equation (2.4). In this equation, note that taking argmax over different values of e' will treat the denominator ($P(f)$) as a constant. So, we can refrain from considering it in the computations while still getting correct results.

$$e = \underset{e' \in E}{\operatorname{argmax}} P(e'|f) = \underset{e' \in E}{\operatorname{argmax}} \frac{P(e') \cdot P(f|e')}{P(f)} = \underset{e' \in E}{\operatorname{argmax}} P(e') \cdot P(f|e') \tag{2.4}$$

$$e = \underset{e' \in E}{\operatorname{argmax}} \{ \underbrace{P(e')}_{Language\ Model} \cdot \underbrace{P(f|e')}_{Translation\ Model} \} \tag{2.5}$$

Equation (2.4) states that finding the best translation in the decoding step consists of two sub-steps; finding the probability distributions of words, phrases and sentences in the target language which is called *Language Modeling*, and finding the conditional probabilities

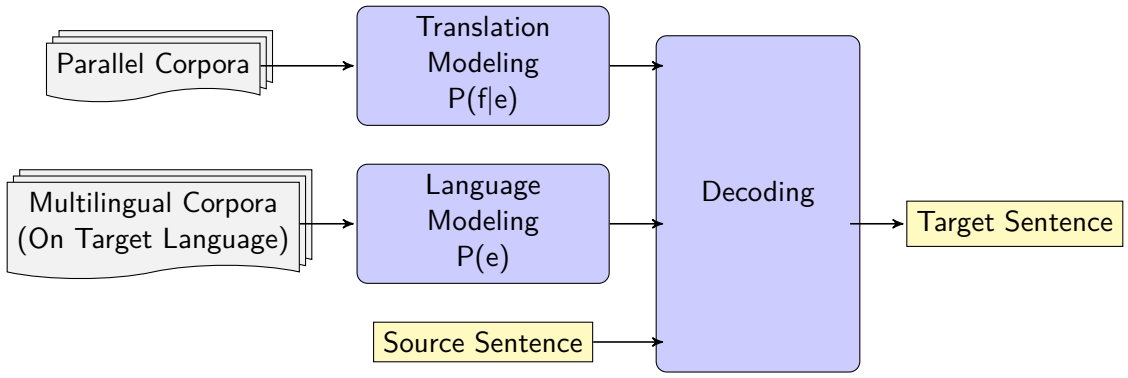

Figure 2.1: The general architecture of a generative Statistical Machine Translation system

of each sentence in the source language given a specific sentence in the target language which is called *Translation Modeling*. Figure 2.1 depicts the general architecture of a generative SMT system. Sections 2.1.1 and 2.1.2 describe the two mentioned sub-steps.

2.1.1 Language Modeling

Language Modeling aims to provide a probability distribution over sentences in a language. Language models are built using *monolingual* training data. They can help us attain a scoring function which can tell a fluent translation from a less acceptable one to the native speakers. The most commonly used language models in natural language processing are *n-gram* language models where the probability of a sentence is built up from probabilities of overlapping groups of n words. Suppose we have a sentence $e = w_1 w_2 \dots w_N$ in the language A. The language model probability of this sentence in the given language is computed based on Equation (2.6).

$$P(e) = p(w_1) \cdot p(w_2|w_1) \cdot \ \dots \ \cdot p(w_N|w_1, w_2, \dots, w_{N-1}) \tag{2.6}$$

We are not interested in the frequency distribution of large phrases (e.g. $n > 10$) because increasing the number of words in n-grams will overfit the distributions to the training sentences. The Markov independence assumption truncates the conditional probability to consider a limited fixed history. The 5-gram language model that uses (at most) 4 words of context to predict each word in the sentence is shown in Equation (2.7).

$$P(e) = p(w_1) \cdot p(w_2|w_1) \cdot \ \dots \ \cdot p(w_N|w_{N-4}, w_{N-3}, w_{N-2}, w_{N-1}) \tag{2.7}$$

Although this formula is good enough to assist in discovering more accurate sentences (based on the language rules), out of vocabulary (OOV) words or phrases (the ones which have not been observed) may result in the probability of a sentence that is zero. *Smoothing* is an approach to address this issue. Smoothing tends to decrease a small portion of the

seen probabilities and assign that portion to the unseen words. We do not cover the different smoothing methods because they are out of the scope of this thesis (see [6] for more information about different smoothing methods).

2.1.2 Translation Modeling

The *translation model* is the second ingredient of Equation (2.5) in addition to the language model which can help finding the best translation given an unknown source sentence. In translation modeling, we aim to model $P(f|e')$ for all source *translation units* (f) given the most probable target *translation units* (e') based on an available $Parallel\_Corpus$ (Equation 2.1). The so-called translation units can be either a word or a phrase. The difference in the translation units has led to different approaches which will be covered in this section.

2.1.2.1 Word-Based Translation Models

The simplest model for translation can be considered as a *lexical translation model* (word-to-word translation model) [7, 8, 9]. During the translation, this model considers the words solely and does not take the previous and coming words into account. In other words, the model will look like a dictionary trying to align the words from one language to another (one word may be mapped to more than one word with a different mapping confidence for each). The model can be constructed by collecting the frequency statistics of words and estimating their lexical probability distributions based on a maximum likelihood estimation method. Using these estimated probabilities words can get aligned in a parallel corpus (words of a source sentence to the words of the target sentence), so this process is called *alignment*. Alignments can be illustrated using alignment diagrams (e.g. Figure 2.2).

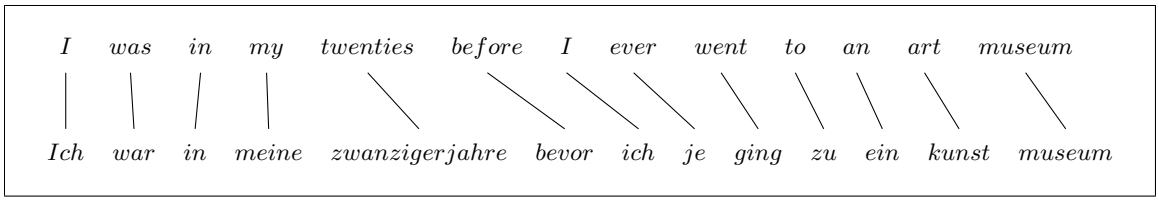

Figure 2.2: An Example Alignment Diagram

Given a word-alignment model, the generative MT system can use the table to compute the $P(f|e')$ term (Equation 2.5) using Equation (2.8) where the source sentence contains W words and $T(w)$ contains the list of valid translations for the word f_w and $p(f_w|w')$ demonstrates the word-alignment probability of words f_w and w' in the source and target language, respectively.

$$P(f|e') = \prod_{w=1}^{W} \underset{w' \in T(w)}{\operatorname{argmax}}\, p(f_w|w') \tag{2.8}$$

Brown et. al [8] propose five unsupervised machine learning approaches (IBM Models 1 through 5) which focus on the problem of word alignment. The Expectation Maximization (EM) algorithm can be one of the iterative algorithms which can be used to implement these approaches. The EM algorithm contains two phases;

1. compute expectations for the lexical probability distributions (it usually starts by uniformly aligning all source sentence words to all target sentence words)

2. maximize the log-likelihood of the lexical probability distributions (in this case updating the probabilities based on the counts of words happening together in source and target)

These two steps are repeated till the convergence of probability distributions. The higher the number of IBM Model is the more complicated the log-likelihood becomes and the more details it considers. For example, IBM Model 1 just considers word translations and reordering of the translations in the target side while limiting the length of the source and target sentence to be the same. IBM Model 2 removes the limit on the length of the alignments but does not let a source word to be translated into many target words. IBM Model 3 removes the limit on the number of target words a source word is allowed to be translated into and so on. Since these concepts are well-known to the NLP community and there has been quite a lot of work done in this area, a wide variety of tools to perform the word alignment exists. GIZA++ [10] is one of such tools which has been used in our experiments.

2.1.2.2 Phrase-Based Translation Models

Word-based translation models omit the context of the training sentences by removing the adjacency information of the words in the sentence. This information can help producing more fluent translations since there may be many translations for one word in the target language and capturing the best fitting one for this specific case is not possible without considering the context in which the word appears. Phrase-based translation models try to address this issue [11, 12, 13]. In the phrase-based approach, the translation unit is considered to be a *phrase*, an ordered collection of adjacent words. However, the approach does not guarantee that phrases are syntactically or semantically aligned (e.g. "engineers in order" in Figure 2.3 can be a phrase).

A phrase can contain just one word; therefore, the word-based approach is a specific case of the phrase-based one. The easiest algorithm to create a phrase-table is obtainable through the modification of the available word-alignment algorithms, but this is not

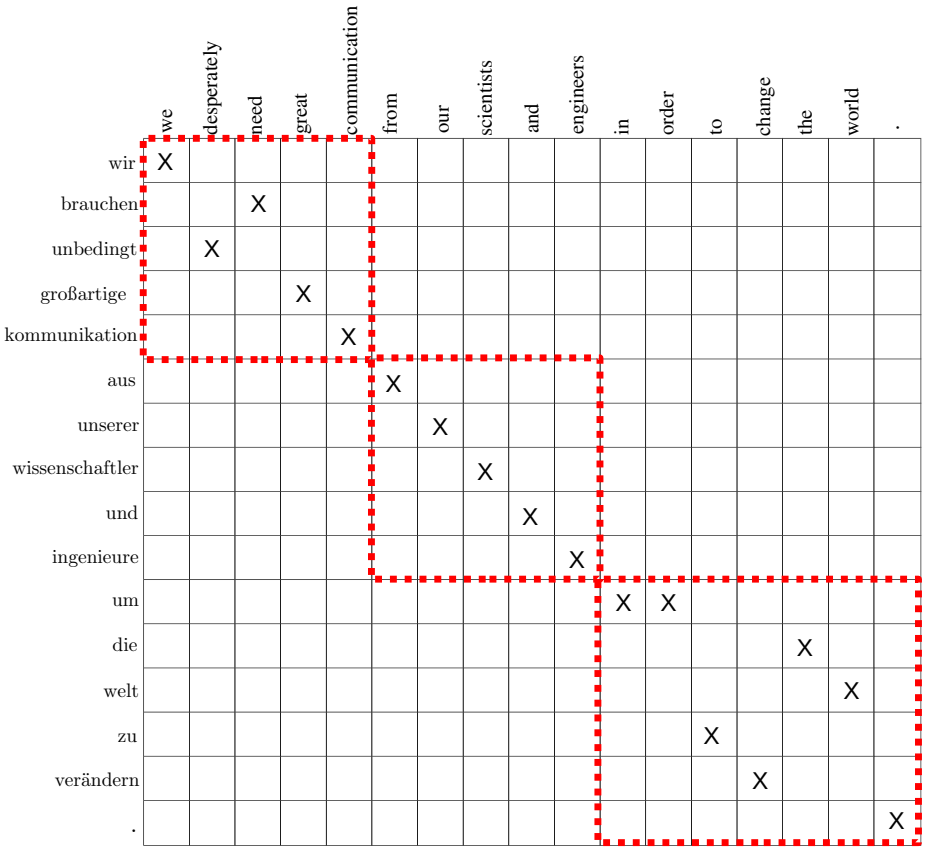

Figure 2.3: An example phrase alignment diagram

usually done in the state-of-the-art approaches. Instead, phrase-tables are normally extracted from the word-alignments. Figure 2.3 depicts a phrase alignment diagram extracted from the word-alignment table. A parallel phrase (e.g. "*in order to change the world* ||| *um die welt zu verandern*") can be consistent with the word-alignment table and extracted as a valid phrase if both source and target side word collections are covered in the parallel extracted phrase (e.g. "*in order to* ||| *um die welt zu verandern*" is not valid).

In addition, a phrase-based translation may *reorder* the phrases, while translation, and put the translation of the i^{th} phrase in the source sentence in the j^{th} phrase position of the translated sentence where $(i \neq j)$. This idea is believed to capture the local relativity of phrases in relation to each other, e.g. put the 'object phrase' before the 'verb phrase' in the target language while the correct form is the other way around in the source language. However, the *distortion model*, responsible for performing the reorderings, tends to prevent long-distance phrase relocations; assuming that the translation of the i^{th} phrase in the source sentence might be somewhere near the i^{th} phrase position in the translated sentence.

Given an extracted phrase-table, the generative MT system can use the phrase-table to compute the $P(f|e')$ term (Equation 2.5) using Equation (2.9) where I is the number of phrases to be translated, $\phi(.)$ is a function of translation probability of the phrases (it

may not demonstrate a probability) and $d(.)$ is a function which penalizes the score if the translated phrase is reordered.

$$P(f|e') = \prod_{i=1}^{I} \phi(f_i|e'_i) d(start_i - end_{i-1} - 1) \tag{2.9}$$

2.1.2.3 Hierarchical Phrase-Based Translation Models

Due to translation quality improvements by phrase-based machine translation (PBMT) systems, Kohen et al. [13] performed a series of analytical experiments on the performance of PBMT systems given different corpus sizes, different maximum valid phrase lengths, and different language pairs. The experiments over the maximum phrase length of translation models revealed that phrases over 4 words long do not contribute much to improving the translation quality since longer phrases are much less frequent resulting in more sparse distribution vectors. Due to this fact, some implementations suggested that the phrase length should be limited to a threshold (e.g. 3 or 4) to get faster translations without substantially hurting the quality.

Chiang [14] introduced the idea of *hierarchical phrases* to benefit from longer phrases instead of simply ignoring them. A hierarchical phrase is defined as a translation unit with discontiguous phrases that can be modeled by a synchronous context-free grammar. As of section 2.1.2.2, the main objective of introducing phrases was to learn local reorderings among words to capture the local context while translating. The hierarchical model (*Hiero* model) tries to exploit this idea to localize the phrases themselves. Given phrases can capture the word localizations, the longer phrases also can capture the subphrase localizations.

He uses the idea of *synchronous context-free grammars* (SCFG)s to describe the structure of the hierarchical phrases. This type of grammar breaks the construction of a large phrase into the construction of smaller subphrases and combining (*gluing*) the subphrases together. A synchronous CFG can map a source sentence (f) to a combination of terminal terms and non-terminal terms for each of which there will be in the target language. Terminals are atomic subphrases which will be directly translated while non-terminals are subphrases which will break down to other subphrases. For example the sentence 'f = *we desperately need great communication*' containing 5 terminals can break down into the hierarchy of Figure 2.4.

This figure shows that the hierarchical phrase-based models are trying to find a set of *rules* for decomposing the sentence into phrases while some of them may not contain any terminal phrases (to be translated) at all. The hierarchical phrase-based model will assign a probability to each of the rules which may lead the sentence one step closer to the full decomposition. The translation system will try to find the best set (R) of such rules which lead to the decomposition with the highest probability. The formula to compute the $P(f|e')$ term (Equation 2.5) in hierarchical phrase-based models will be as shown in Equation (2.10).

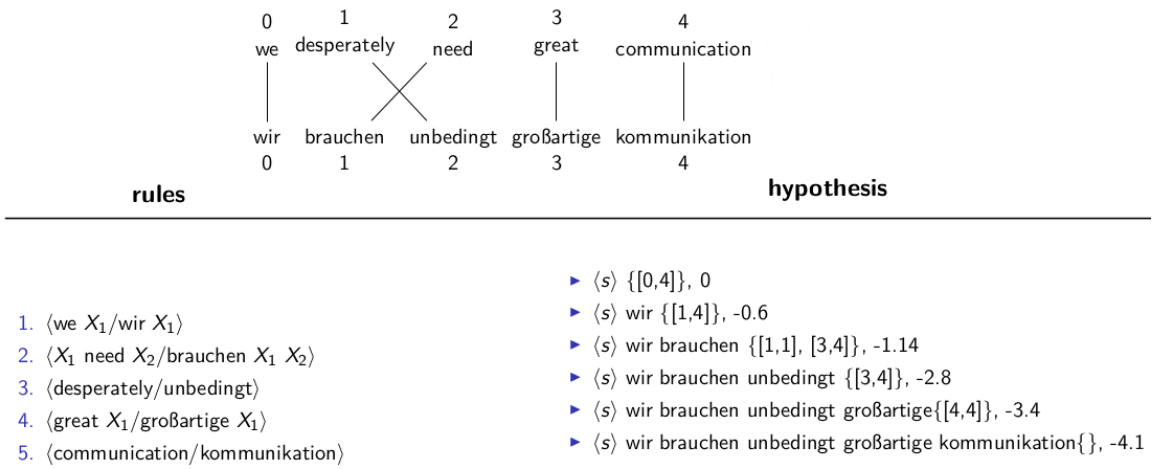

Figure 2.4: An example hierarchical phrase decomposition in Hierarchical Phrase-Based Translation Models

$$P(f|e') = \prod_{r \in R(f,e')} p(r) \tag{2.10}$$

2.1.2.4 Left-to-Right Hierarchical Phrase-Based Translation Models

The Hiero models were able to catch the longer-distance reorderings between source and target languages. In addition, they were able to decrease the need for expensive language model calls (due to storing more content in a hierarchical phrase). These properties helped to introduce translation modeling approaches which were more focused on the concept of real-time translation (more details in section 3). The first step towards this idea was to think about *left-to-right (incremental) translation* models [15, 16].

In real-time translation, the output should be generated incrementally as the input is being provided. The preference, in this case, will not be to hold the audience for a long time to show them the translation result of a long sentence. *Left-to-Right Hierarchical Phrase-Based* (Incremental) translation models try to address this issue by justifying the Hiero models. As a review, Hiero models were using synchronous context-free grammars (containing terminal and non-terminal terms) to produce the hierarchical translations. Incremental translation models try to filter the SCFG rules into the ones which are in Greibach Normal Form (GNF). A rule will be in GNF if the right-hand side of it starts with a terminal symbol, optionally followed by some non-terminals. Such rules will guarantee that no terminal is appearing after a non-terminal (which is assuring the left-to-right translation). In this case, any terminal which appears in a rule can be directly sent to the audience in a real-time manner. As an example, in Figure 2.4, all the rules are in GNF since terminals are appearing before non-terminals in all of them.

The formula to compute the $P(f|e')$ term (Equation 2.5) in the left-to-right phrase-based models will be the same as Equation (2.10) with a difference in the set of rules R. In the left-to-right models, the hierarchical phrase alignment rules are filtered to the left-to-right ones where terminals appear before non-terminals.

2.2 Discriminative Machine Translation

Generative machine translation methods show acceptable results as well as reasonable running times. But, they consider the noisy channel model components (language and translation models) as the only effecting components to the superiority of a translation over another. Therefore, there is no straightforward way to extend a baseline generative statistical MT model by including additional dependencies. Och and Ney [17] mention that this assumption results in extendability problems when extra prior knowledge exists. In addition, they believe that the current formulae to approximate language and translation model distributions are only poor approximations of the true probability distributions. Due to the mentioned problems, they provide an alternative formula to Equation (2.3) where they directly compute the probability distribution of $P(e'|f)$ using a discriminative log-linear model (also called a *Maximum Entropy* model).

The model, called *Discriminative Translation Model*, considers a set of M feature functions $h_m(e, f)$, $m = 1 \dots M$, for each of which there exists a model parameter λ_m. The maximum entropy objective of the training phase will be to find the best model parameter set (λ) based on the collected feature values from the parallel sentences in the training set. Once best model parameter set is discovered, $P(e'|f)$ can be calculated using the Equation (2.11).

$$P(e'|f) = P_{\lambda_1^M}(e'|f) = \frac{exp[\sum_{m=1}^{M} \lambda_m h_m(e', f)]}{\sum_{e' \in E} exp[\sum_{m=1}^{M} \lambda_m h_m(e', f)]} \tag{2.11}$$

Note that the denominator performs as the normalization term in the model; however, it can be neglected since our search problem is from the argmax type and we can exclude fixed values. This simplification will omit a very time-consuming part and will help to achieve faster search results.

Based on the discriminative translation model, the most probable translation (e) in Equation (2.2) can be the output of Equation (2.12).

$$e = \underset{e' \in E}{\operatorname{argmax}} P(e'|f) = \underset{e' \in E}{\operatorname{argmax}} \sum_{m=1}^{M} \lambda_m h_m(e', f) \tag{2.12}$$

This approach can describe the generative noisy-channel model as a special case where the model contains only two feature functions $h_1(e', f) = \log P_{LM}(e', f)$ and $h_2(e', f) = \log P_{TM}(e', f)$, given $\lambda_1 = \lambda_2 = 1$.

We use the standard feature set used in [16] to build our log-linear discriminative translation model for the experiments in this thesis. The features in this set are as follows.

- relative-frequency translation probabilities $p(f|e)$ and $p(e|f)$
- lexical translation probabilities $p_l(f|e)$ and $p_l(e|f)$
- language model probabilities, word counts, and phrase counts
- word and phrase reordering penalties
- source side distortion features (total length of jumps between source phrases to produce the translation, ...)

2.3 Evaluation of Machine Translation Systems

Given different translation models, a source sentence may have many valid translations in the target language. Therefore, defining an evaluation measure which can rank the translations and find the higher quality ones seems necessary. Though, this task is not as trivial as it appears to be since there may be many equally correct translations for one sentence (consider them as the paraphrases of each other) and it is fairly impossible to formulate all possible translations for any possible source sentence. With this in mind, different translation evaluation measures have been suggested with different intuitions about what matters in the translation evaluation.

Evaluation can be performed at the word level, the sentence level, or the corpus level. A sentence level evaluation will assume that there exists, at least, one human translation of the source sentence (called *reference sentence* or *gold standard*) against which the machine translated target sentence is evaluated. In the meantime, such an approach can perform in the corpus level if the concatenation of all translated sentences (a huge sentence) be fed to the evaluation method in addition to the concatenation of all reference sentences. Here, we mention the most promising evaluation methods which are currently being used during the recent years of research in Machine Translation.

TER The intuition behind *Translation Edit Rate* is to measure the edit distance between the translation and human reference sentences. Equation (2.13) depicts the method of deriving this measure [18].

$$TER = \frac{count(edits)}{average(reference\ length\ in\ corpus)} \tag{2.13}$$

METEOR The idea of flexible matching has been resulted in the *Metric for Evaluation of Translation with Explicit ORdering*. This evaluation metric is based on a word-to-word (unigram) matching between the translation and reference sentence(s).

Candidate: we desperately require great communication

Reference: we desperately need great communication

$p_1 = \frac{4}{5},\ p_2 = \frac{2}{4},\ score = \min(1, \frac{5}{5})(\prod_{n=1}^{2} p_i)^{\frac{1}{2}} = (\frac{4}{5} * \frac{2}{4})^{\frac{1}{2}} = 0.632455532$

Figure 2.5: An example BLEU score computation for $N = 2$

In addition, the measure is not only matching the identical words but is also detecting the morphological variants of them. It considers partial credits for matching stems, matching synonyms and the use of paraphrases. The formula to compute METEOR is a variation of F-Score with different descriptions of precision and recall as explained in the beginning of this paragraph [19].

BLEU Another explored idea as an automated translation evaluation measure has been to count the number of n-gram overlaps between the reference and the translated sentence, since such an algorithm may tend to assign higher scores to shorter target sentence translations, a brevity penalty is considered in the scoring formula. Equation (2.14) depicts the BLEU scoring function[20].

$$\text{BLEU} = \min(1, \frac{\textit{output-length}}{\textit{reference-length}})(\prod_{n=1}^{N} \textit{precision}_i)^{\frac{1}{N}} \tag{2.14}$$

Please note that this measure is typically computed over the entire corpus (entire corpus is assumed to be a huge sentence). The main vulnerability of the mentioned scoring function is related to the time when one precision value (especially for larger values of N) is zero (turning the total score to be zero). N is normally limited to 4 and precision values may get smoothed.

BLEUp1 The *Oracle Ranking for Gisting Evaluation* (ORANGE) is a variation of BLEU (also called BLEU+1 or BLEUp1) which focuses on the problem of ranking different translations of one source sentence. The main assumption about this variation is that each sentence in the corpus is independent of the other sentences and the scoring function should not consider credits for the translation of this sentence if the n-gram reference translation of this sentence has happened to be available in another corpus sentence translation. BLEUp1 scores the corpus by computing the local BLEU scores for each sentence and taking the average of the computed scores for all the sentences [21].

Due to the importance of BLEU evaluation measure in this thesis, we provide the BLEU computation example of Figure 2.5 for a candidate-reference pair with just one word difference and $N = 2$. As shown in the example, this measure is sensitive to the reference

sentence to the extent that even one word difference may substantially decrease the score. We will use this feature to increase our accuracy in finding the optimal segmentations (see section 3.2).

2.4 Summary

In this chapter, we reviewed the most important background knowledge about Machine Translation and we introduced different modeling techniques in it. We explained the concepts of language modeling and translation modeling and how they come to importance to get acceptable results in Machine Translation. In the last section, we introduced the most important translation quality evaluation metrics and suggested further information for the measure which is going to be used in this thesis. In the next chapter, we will introduce the concept of Simultaneous Translation and explain the way it is different from Machine Translation.

Chapter 3

Simultaneous Translation

Sections 2.1 and 2.2 briefly explained the problem of machine translation and provided the best practices for modeling the solutions. But, the solutions were assuming that the whole sentence from the beginning to the end is available when the translation is starting. This assumption is not valid for *Real-Time Machine Translation* (also called *Simultaneous Translation and Machine Interpretation*).

In Simultaneous Translation, the process starts before receiving the end of the sentence, and the evaluation objective is not only sensitive to the translation quality, but it is also caring about the translation *latency*; the difference between the receiving time of the utterance in the source language and the delivery time of its translation in the target language. Figure 3.1 provides an example stream of words in Japanese as well as the outputs of both Machine Translation and Simultaneous Translation strategies (borrowed from Graham Neubig - NAIST 2015 [3]). We will use this figure to define and explain four major issues in Simultaneous Translation; *Segmentation*, *Prediction*, *Paraphrasing*, and *Evaluation*.

3.1 Major Issues in Simultaneous Translation

The Simultaneous Translation output (*Interpretation*) is fairly different from the Machine Translation output in terms of length, wording and content, in some cases (one example is shown in Figure 3.1). Clearly this difference comes from the major differences in producing the output.

The first difference is in segmentation task done in ST where we have to segment the stream of receiving utterances into a number of chunks and produce the translation of each segment separately, to achieve lower latency translations. Each of these segments are translated separately regardless of what comes next and may or may not consider what utterances have come before. This strategy enhances the *speed* of translation (consider it as $\frac{1}{latency}$) one the one hand, but it also diminishes the quality of translation, on the other hand. This fact defines a trade-off between the translation quality and latency in

Source	今ご覧いただいた|この映像は|今から五年前、|日本で|世間を賑わせていた裁判員制度が|始まる一年前、大学四年生だった私が模擬裁判用の|資料として作った物です
Translation	Five years ago, as a college senior, I created the video that you just saw as a reference material for a mock trial, one year before the much-talked-about jury system commenced in Japan.
Interpretation	You just saw | this video clip. | Five years ago, at that time | in Japan, | the ordinary people's justice system, jury system, was very much talked about in Japan, | and I created this video as a reference material for that.

Figure 3.1: The translation and interpretation outputs of a Japanese stream of words (from Graham Neubig in a NAIST 2015 presentation). The 'Translation' row has been produced as the whole input stream has been accessible to the translator, while the 'Interpretation' row has been produced by a human interpreter, segment by segment.

the Segmentation task. The segment separators are shown in the example of Figure 3.1 as vertical bars in the Source and Interpreted output rows.

The second difference is in the prediction task which is a crucial task when the linguistic structure of the source and target languages are not the same and monotone translation is not an option, e.g. the source language is subject-object-verb (like Japanese) while the target language is subject-verb-object (like English). In this case, when the system receives the object phrase in the source language, has to predict the verb phrase to produce the correct translation of the object phrase in the target language. Prediction will help in choosing the correct word and phrase translations of the phrase we have not still received. For example, in Figure 3.1, when we receive the phrases 'you' and 'the video', we have to predict the correct form of the verb 'to visit' as being 'to see' and use the correct form of it.

The third difference is in paraphrasing (or rewording) which is a necessary task which helps improving latency measures by producing less translation holding the same content. In addition to reducing the delivery delay, paraphrasing can help reduce disfluencies by converting passive to active sentence voice or using clearer words to covey the actual meaning of the source sentence. This task can even remove irrelevant parts of the source sentence (to the pre-specified topic) to improve the extent of understanding about the topic. In Figure 3.1, the fourth segment actually means "in the country of the rising sun", but considering the context, the actual paraphrase "in Japan" seems more reasonable to convey the meaning in this case.

The last difference is in evaluation task. The most important fact about human interpretation evaluation is that people do evaluate how interpreting is done not just based on the translation accuracy but with the understanding that there is some time pressure. Therefore quality scores (such as BLEU) cannot evenly evaluate the ST outputs. Another measure is necessary in this case which may consider the translation creation in a timely way, and that measure is going to score based on the latency.

Considering all of these issues and the solutions suggested for them, the ST output may still need some improvements given the fact that currently available interpretation corpora may lack the translation of some segments exactly such as the translation of the fifth segment in the example of Figure 3.1 which is available in the Translation output as the underlined parts but is not existing in the Interpretation output since the interpreter has simply not been able to catch up with the speaker and decided to translate the next segment with a better quality instead of focusing on what he has not heard correctly.

Among these four different issue types, the segmentation and evaluation types are the issues targeted by this thesis. Sections 3.2 and 3.3 provide more explanations and details in these areas.

3.2 Segmentation

Segmentation is the task of finding reasonable chunks, regarding an evaluation function, in a given text or speech. This task has shown to be useful in various applications one of which is Simultaneous Translation. To illustrate this application, we demonstrate different segmentation strategies over a source sentence in Figure (3.2) for the German language. Figures (3.2a) and (3.2b) depict two extreme cases of segmentation where Figure (3.2a) is preferring to make certain segmentations (with a high accuracy and a bad latency) and Figure (3.2b) has decided to segment any received word at the arrival time without even waiting for one more word (fairly low latency but a bad translation quality).

A good segmentation strategy, tries to find a tradeoff between these two extreme choices to decrease the number of segmentation points from all possible to some of the good ones (improving accuracy in Figure (3.2b)) while it does not wait untill the end of sentence (making shorter latency gaps in Figure (3.2a)). This segmentation strategy is implemented as a *segmentation classifier* in the Simultaneous Translation system which is receiving the stream of utterances while they are being produced and makes a decision upon receiving each word to segment the stream or not. Figure (3.2c) is depicting two segmentation decisions made by a modest segmentation classifier.

The technique of segmenting the input is often referred to as the "salami technique" in the field of conference interpreting (by humans) [22] referring to the slicing up of the input into small, predictably sized units for translation. In spoken language translation, the "salami technique" has been mostly focused on fixed length segments or segments based

Figure 3.2: Different segmentation startegies on a source sentence

on monolingual features in the input such as pauses and other similar cues [23, 24, 25, 26] to break the input into segments for incremental translation. In order to train a segmentation classifier, one can go beyond simple cues such as pauses and annotate training data with good segmentation boundaries [27, 28]. These techniques require either heuristic or human annotation of segment boundaries for some data in the source language. The segmentation classifier can be tightly integrated into a stream decoding process for incremental translation [16]. The impact of the choice of segment length has been studied in some previous work on segmentation [29] and stream decoding [30]. However, none of these approaches explicitly consider the impact of selecting between different segments (perhaps of the same size) on the translation quality in the target language. In the context of this thesis, we want to choose segments that are optimal in some way with respect to latency and/or translation quality and we wish to train a segmentation strategy that provides such an optimality guarantee (on the training data).

Oda et al. (2014) [1] have explored finding segments that maximize translation quality with a user-defined bound on segment length (more details in section 3.2.2). The training data set required for this is much more complex because, in order to optimize for segments with good translation quality, we need a training set translated with all possible segment choices and sizes and the eventual translation quality for each possible segmentation choice. Once such a training data set is built, one can apply the algorithms in [1] to find segmentation decisions that are optimal with respect to some evaluation measure of translation quality such as BLEU [20] score.

3.2.1 Review on Segmentation Approaches

In speech translation, the segmentation task can be performed on speech or the transcribed text. Early work on speech translation uses prosodic pauses detected in speech as segmentation boundaries [23, 24]. Segmentation methods applied on the transcribed text can be divided to two categories: heuristic methods which use linguistic cues, like conjunctions, commas, etc. [25]; and statistical methods which train a classifier to predict the segmentation boundaries, like what Xiong et al. [31] have done in detecting the translation zones (which can be interpreted as the segment). Some early methods use prosodic and lexical cues as features to predict soft boundaries [32]; while most recent methods rely on word alignment information to identify contiguous blocks of text that do not contain alignments to words outside them [27, 28]. In addition to these segmentation approaches which are applied before calling the translation decoder, there is another strategy which performs the segmentation during decoding which is usually called stream or incremental decoding. Different incremental decoding approaches have been proposed for phrase-based [30, 33] and hierarchical phrase- based translation [28, 34]. He et al. [35] focus on language pairs with divergent word order by designing syntactic transformations and rewriting batch translations into more monotonic translations. Some research has been conducted on human simultaneous interpretation to determine the effect of the latency and accuracy metrics on the human evaluation of the output of simultaneous translation. The results indicate that latency is not as important as accuracy [4]. This implies that we need algorithms that can make a careful choice between different segmentation decisions of the same latency to produce translations with the best translation quality possible (for that latency) which we have done in this thesis.

3.2.2 Greedy Segmentation Approach

Greedy segmentation (Oda et al. 2014) [1] is the state-of-the-art method for creating segmentation training data. In this approach, the best possible segmentation points are found over an unsegmented corpus which maximize the translation accuracy of the segmented sentences in a greedy way.

The algorithm in [1] has a parameter for the number of expected segments, K, which is given by Equation 3.1. Using this equation, the segmentation model is trained on a parallel corpus $\mathcal{F} = \langle F, E \rangle$ which has N source/target sentence pairs. $|f|$ provides the length of sentence f in words and μ is the average segment length.

$$K := max(0, \left\lfloor \frac{\sum_{f \in F} |f|}{\mu} \right\rfloor - N) \tag{3.1}$$

Finding each of these K segmentation points in the algorithm involves searching through all the N sentences in the corpus and examining each segment boundary in the whole corpus.

For $K = 1$, one sentence in the corpus is segmented into two chunks. This way, they will produce all possible hypothesized segmentations of the entire corpus, one of which is going to be the optimal one.

Given an MT system, $\mathcal{D}$, which is already tuned on a given development set, $\mathcal{D}(f, s)$ is the translation output of the MT system $\mathcal{D}$ for a given source sentence f obtained by concatenating the translations of the individual segments defined by the set of segmentation decisions s. This set s is created by adding a segmentation point at each place where a segmentation classifier fires. In [1] the segmentation classifier is determined by checking a single feature firing. This single feature is a bigram part of speech (POS) tag. Each segmentation of the corpus is a collection of such features called Φ. Thus, s, the set of segmentation points is proportional to the number of sentences in $\mathcal{F}$ and the features Φ that determine the segments: $s \propto \{\mathcal{F}, \Phi\}$.

The accuracy score of each possible segmentation choice for a given number of segments s is computed for the whole corpus as follows:

$$B(s) = \sum_{j=1}^{N} \beta(\mathcal{D}(f_j, s), e_j) \tag{3.2}$$

where $\mathcal{D}(f_j, s)$ produces target translations for each source sentence f_j based on the segments in s. Each output sentence is scored by β which can be any automatic evaluation measure for translation quality. [1] uses per-sentence smoothed BLEU score (BLEU+1) [20, 21] as the translation quality evaluation measure. $B(s)$ is the sum of the translation quality scores for each segmented sentence. The *argmax* of $B(s)$ finds the optimal segmentation for the entire corpus, searching over all possible s segment boundary points. This *argmax* of $B(s)$ is repeatedly computed for every segmentation set of size $k = 1 \ldots K$, and the set of size K is returned.

Because such an approach is computationally complex, Oda et al. (2014) [1] introduce the idea of feature grouping. Using feature grouping, once a feature has been greedily chosen, all the points exhibiting that feature are segmented at the same time and added to the set of selected features. Moreover, they take advantage of dynamic programming (DP) implementation of the greedy approach to reflect optimal feature grouping. DP is used to build larger sets of segmentation points from smaller sets. This method is called Greedy-DP or the GDP Segmentation approach in their paper.

Finally, they introduce a regularizer coefficient α to their accuracy scoring function which is aimed to control the number of selected features out of the set Φ; as a higher α will choose a smaller set of features in Φ which occur frequently to produce the necessary number of segments while a lower α tends to prefer a larger set of features in Φ, each of which occur less frequently.

$$B_\alpha(s) = B(s) - \alpha|\Phi| \tag{3.3}$$

In an English-German translation task, consider the three-sentence sample example of Figure 3.3 (a corpus with 43 tokens) and the features used for choosing the segmentation points to be the bigram part of speech (POS) tags (like [1]). In this example, each point has been labeled with a coarse POS tag out of the set $\mathcal{P}$={N[noun], V[verb], D[determiner], J[adjective], P[preposition], S[possessive pronoun], A[adverb], R[particle], .[dot]}.

(1)	I	am	a	contemporary	artist	with	a	bit	of	an	unexpected	background	.
	N	V	D	J	N	P	D	N	P	D	J	N	.

(2)	I	was	in	my	twenties	before	I	ever	went	to	an	art	museum	.
	N	V	P	S	N	P	N	A	V	P	D	N	N	.

(3)	I	grew	up	in	the	middle	of	nowhere	on	a	dirt	road	in	rural	Arkansas	.
	N	V	R	P	D	N	P	N	P	D	N	N	P	J	N	.

Figure 3.3: Example training set for segmentation choices containing the source sentences and part of speech tags (target German sentences are not shown in this figure but appear later).

Feat	Freq	Feat	Freq	Feat	Freq
N-P	6	J-N	3	V-R	1
P-D	5	N-N	2	P-S	1
D-N	4	P-N	2	P-J	1
N-.	3	D-J	2	S-N	1
N-V	3	R-P	1	A-V	1
V-D	3	N-A	1		
FSS Size				40	

Table 3.1: Frequencies of the bigram part of speech tags in the example from Figure 3.3.

Table 3.1 shows the feature frequencies of the sample corpus. For $\mu = 1$ (setting each word as one segment) for the example in Figure 3.3, the GDP segmentation algorithm will set $K = 40 = max(0, \left\lfloor \frac{[\sum_{f \in F} |f|=43]}{[\mu=1]} \right\rfloor - [N = 3])$. Likewise, if we set $\mu = 8$, we will have $K = 2$, and our possible segmentation sets will be in {{N-N}, {P-N}, {D-J}, {[R-P],[N-A]}, {[V-R],[P-S]}, ...} for our running example. Therefore, the segmentation set will contain all the different ways to segment the segmentation training data to obtain the average segment length of 8. If we want to consider different possible segmentations of the second sentence in our sample corpus with $\mu = 8$, the possible segmentations will be one of the sets inside $s_{possible}$= {{}, {N-N}, {P-N}, {{N-A}, {P-S}}, {{N-A}, {S-N}}, {{A-V}, {P-S}}, {{A-V}, {S-N}}, {{A-V}, {N-A}}, {{P-S}, {S-N}}} .

Table 3.2 shows the possible segmentations of the second sentence of the example in Figure 3.3 for $K = 2$. We show Φ only for the second sentence, so when $\Phi_{sent\ 2}$ is $\emptyset$ the two segments were chosen in other sentences not shown in this table. The GDP algorithm will choose the segmentation that maximizes accuracy, so for $K = 2$, the GDP algorithm will

	$\Phi_{sent\ 2}$	#segments	Segmented Sentence & Translation	GDP Accuracy	PO Accuracy	Time	Segs/Sec
1	$\emptyset$	1	[I was in my twenties before I ever went to an art museum .] Ich war in meinen zwanzig vor Ich in ein kunstmuseum ging .	0.224	0.224	16.097	0.062
2	P-S	2	[I was in][my twenties before I ever went to an art museum .] Ich war in meine zwanziger vor Ich in ein kunstmuseum ging .	0.382	0.191	15.206	0.131
3	S-N	2	[I was in my][twenties before I ever went to an art museum .] Ich war in meinem zwanziger vor Ich in ein kunstmuseum ging .	0.235	0.117	15.487	0.129
4	A-V	2	[I was in my twenties before I ever][went to an art museum .] Ich war in meinen zwanzig Ich je vor ging zu einer kunst museum .	0.134	0.067	9.983	0.200
5	N-A	2	[I was in my twenties before I][ever went to an art museum .] Ich war in meinen zwanzig Ich vor in ein kunstmuseum ging .	0.224	0.112	3.462	0.577
6	N-N	2	[I was in my twenties before I ever went to an art][museum .] Ich war in meinen zwanzig vor Ich jemals zu einer kunst museum .	0.138	0.069	3.426	0.583
7	P-N	2	[I was in my twenties before][I ever went to an art museum .] Ich war in meinen zwanzig vor Ich in ein kunstmuseum ging .	0.224	0.112	2.697	0.741
8	P-S,S-N	3	[I was in][my][twenties before I ever went to an art museum .] Ich war in meine zwanziger vor Ich in ein kunstmuseum ging .	0.382	0.127	2.586	1.160
9	P-S,A-V	3	[I was in][my twenties before I ever][went to an art museum .] Ich war in meine zwanziger vor Ich je ging zu einer kunst museum .	0.272	0.090	3.137	0.956
10	P-S,N-A	3	[I was in][my twenties before I][ever went to an art museum .] Ich war in meine zwanziger vor Ich in ein kunstmuseum ging .	0.382	0.127	5.350	0.560
11	S-N,A-V	3	[I was in my][twenties before I ever][went to an art museum .] Ich war in meinem zwanziger vor Ich je ging zu einer kunst museum .	0.141	0.047	2.762	1.086
12	S-N,N-A	3	[I was in my][twenties before I][ever went to an art museum .] Ich war in meinem zwanziger vor Ich in ein kunstmuseum ging .	0.235	0.078	2.586	1.160
13	N-A,A-V	3	[I was in my twenties before I][ever][went to an art museum .] Ich war in meinen zwanzig Ich vor je ging zu einer kunst museum .	0.134	0.044	2.632	1.139

Table 3.2: For the second sentence in Figure 3.3, we show the bigram part of speech features that pick the segment boundaries, the number of segments in this sentence, the accuracy for both the Greedy-DP (GDP) algorithm of [1] and our Pareto-Optimal (PO) algorithm (see Section 4), the translation times and latency measurements (with parameter $\mu = 8$). GDP accuracy is different from PO accuracy because accuracy is measured differently in the two approaches.

pick either sentence 8 or 10 from Table 3.2 (the algorithm has to break ties arbitrarily in the sorted order for segmentations with equal accuracy).

The GDP algorithm thus picks the segmentation decisions that result in the best accuracy on the training set. However, the GDP algorithm considers only accuracy to find the optimal segmentations, so it tends to prefer larger segments that can result in worsening the latency. Furthermore, the trade-off between accuracy and latency is not modelled in the search for good segmentations. This trade-off is crucial in the design of simultaneous translation systems. Another issue can be observed in Table 3.2, in choosing to spread the segmentation points to more sentences or concentrating them in fewer sentences, the GDP algorithm tends to choose the latter (oversegmentation) in spite of the regularizer on the size of Φ. Equation 3.3 does not consider the number of segments which are placed in each individual sentence. We try to address both of these issues in our Pareto-optimal segmentation approach.

3.2.3 Integrating Segmentation Into Simultaneous Translation Systems

Considering the need for segmentation (see section 3.2) for Simultaneous Translation, integration of a segmentation strategy with the ST system can be done in different manners.

The most obvious idea of integration can be to perform the translation in a segment by segment manner and paste the outputs together. Taking one step further, you can keep track of what you have translated so far (keep all the hypotheses) and by receiving the new segment you update the whole translation but you just output the translation of the last segment (the previous segments have been already delivered), even if it is inconsistent. And finally, the stream decoding technique which is producing the output by looking at the stream word by word (not just considering fixed rules to segment), and consults with a heuristic function to decide on generating the output. This heuristic mostly takes care of the length of the current phrase (stream of untranslated words) in comparison to the maximum valid phrase in the phrase table.

Karlsruhe (KIT) Lecture Translator [36] contains a reinforcement learning classifier which is deciding on the length of translation creation delay in addition to the stream decoding conditions. If the translation generation delay goes higher than a threshold, the system decides to produce the translation even though the results are not of a superior quality.

NICT Speech Translator [37] and Skype Translator [38] are two other Speech-to-Speech translation systems which are trying to use the Simultaneous Translation concepts in a product. The issue of training data selection and preparation has been one of the problems that Skype Translator project has been concentrating on [39, 40, 41, 42].

Domain Adaptation [43] is also one of the concepts which makes a difference when such Simultaneous Translation systems are commercialized, since what has been done in the area of stream decoding. Since the training phase is mainly done in the *news domain* while a

Speech-to-Speech system needs to perform the translation in other domains, as well. With this point in mind, we still focus on the news domain and let the reader to adjust the information using the domain adaptation techniques to the other domains.

3.3 Latency

Minimizing latency is a challenge for any spoken language translation system that does simultaneous translation. Ideally the system should produce the translation of an utterance soon after it has been produced. However, translation often involves reordering and this means that a monotone translation which immediately translates as soon as possible can be quite poor in translation quality. Waiting until the end of the input can typically improve the quality of translation but has very bad latency, while translating short segments improves latency but typically makes the quality of translation much worse. A common technique in the literature [25, 44, 45, 46] is to segment the incoming speech stream into chunks that can capture re-ordering between source and target languages and translate these chunks in order to improve latency.

The state-of-the-art [28, 47] define latency as the average time spent on segment translation in a sentence. In these works, the underlying translation system greatly affects the general performance of the system. In this thesis, we define another latency measure which is more careful about the elements which will be effective to the performance of the ST system. In the designed experiments, we will compare the two described measures in terms of performance and quality of the results.

3.4 Pareto-Optimality

Pareto-optimality shows how one could choose different equally important n-dimensional points ($n > 1$) in the space while one point may be superior to the others in one dimension but not in the other dimensions. Here, we consider the two-dimensional space of latency and accuracy.

Considering translation latency-accuracy points depicted in Figure 3.4 as an example, a point will be *Pareto-Optimal if and only if there is no other point which is both faster and more accurate than this point (or even equal in one aspect).* In other words, a point p_1 is Pareto-optimal if and only if for each point p_2 in the region we have

$$\Lambda\{p_2\} < \Lambda\{p_1\} \ \& \ B\{p_2\} < B\{p_1\} \tag{3.4}$$

where Λ and B are representing functions measuring latency and accuracy. Therefore, point p_1 dominates any such point p_2, shown as $p_1 \rhd p_2$. If the dominated point p_2 has an equal

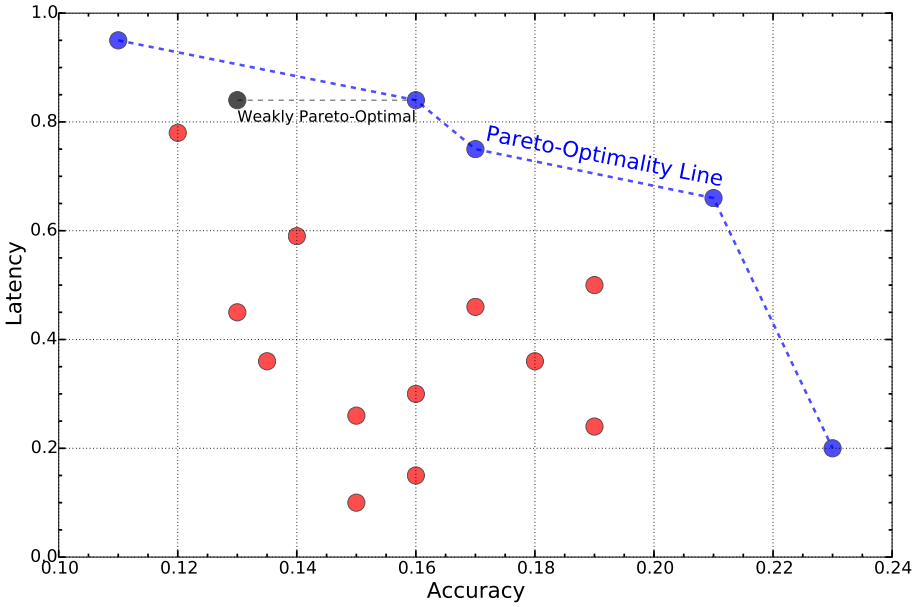

Figure 3.4: Pareto-Optimal and Weakly Pareto-Optimal points as well as the dominated points scored on the two metrics of interest in this thesis: latency and translation accuracy scores (e.g. BLEU).

latency or accuracy measure to the dominating point p_1, we call p_2 a Weakly Pareto-Optimal point [48].

Based on these concepts and paying attention to the Pareto-Optimality Line in Figure 3.4, we see that there may be more than one optimal point on which one could tune the MT system to enhance the performance of stream decoding in Simultaneous Translation. Each of these points is one *Pareto Frontier Point*. The Pareto frontiers provide a range of equally optimal points rather than one most accurate point and we use this fact in our search for optimal segmentation points.

3.5 Summary

In this chapter, we defined the two main concepts of Simultaneous Translation, latency and translation accuracy, and we related these two using the idea of segmentation. We mentioned that the segmentation classifier is responsible for segmentation decisions and training such a classifier needs high quality annotated data which is not currently available, and we reviewed the Greedy Segmentation Approach as the latest method for creating the training data for the segmentation classifier. In the next chapters, we will introduce our Pareto-Optimal segmentation approach which will consider both latency and accuracy in training data creation for segmentation classifier and we will explain the experiments done to prove the superiority of our approach.

Chapter 4

Pareto-Optimal Segmentation Approach

In this section, we will show how Pareto-optimality can help producing a better segmentation with respect to both latency and accuracy. In our approach, we use the same notation of K and μ introduced in the Greedy approach of chapter 3.2.2 to explore the space of possible segmentations in the training corpus. However, in our algorithm, the parameter μ (the average segment length) can be seen as a way to explore the trade-off between latency and accuracy. Longer segments (with a higher μ value) tend to be associated with higher translation qualities. But, the cost of this higher accuracy is that our translation system will have a worse latency. Shorter segments (with a smaller μ value) tend to be associated with better latencies (on average there will be more segments translated per second). In this case the translation fluency scores tend to become worse. We compare our approach to the Greedy approach by (Oda et al., 2014) [1] which takes the value of K as an input, in addition to the naïve heuristic on prosodic cues (comma, end of sentence, ...). We consider different values of K in our algorithm to balance the latency-accuracy trade-off. To describe our algorithm, we first describe the way we model the translation accuracy in section 4.1 and the way we model the translation delivery latency in section 4.2. We finally describe the model and provide a running example of it in section 4.3.

4.1 Modeling the Translation Accuracy

We demonstrate our translation accuracy model by virtue of Equation 4.1 which is a modification of Equation 3.3 where $K = |s| = \sum_{j=1}^{N} |s_j|$ holds and $|s_j|$ is the number of segments

(i.e. the number of segmentation points plus one) for each sentence f_j.

$$B_\alpha(s) = \underbrace{\sum_{j=1}^{N} \frac{\beta(\mathcal{D}(f_j, s_j), e_j)}{|s_j|}}_{Accuracy\ Scoring\ Function} - \underbrace{\alpha|\Phi|}_{Weighted\ Regularizer} \tag{4.1}$$

In the Equation (4.1), the first term is the actual accuracy scoring function (ASF) which is aimed to score the translation quality and like any other machine learning model, the second term is going to regulate the modeling error. As mentioned before, the ASFs in Equations (3.3) and (4.1) are sentence level functions. Although, there is no necessity to stick with the sentence-level measures, and one can consider using the corpus level BLEU scores as an ASF.

One main reason of adding the $|s_j|$ term denominator to the Equation 4.1 or considering another level of calculating accuracy in the mentioned ASF is to distinguish the effect of the number of segments or the accuracy measuring level to the *oversegmentation* phenomenon which was pointed to be a drawback of the Greedy segmentation approach. We think manipulating the ASF, to some extent, can solve the problem of corpus oversegmentation and we aim to provide experiments to test this idea out in this thesis.

Furthermore, in both Equations (3.3) and (4.1), the regularizer term is supposed to be fixed and equal to the size of selected features set ($|\Phi|$). While there can be other useful regularizer terms (a Γ term) with useful features. Equations (4.2),(4.3), and (4.4) can be three example regularization terms which can be used in the related experiments.

$$\Gamma_0 = |\Phi| \tag{4.2}$$

$$\Gamma_1 = \frac{|\Phi|}{K} \tag{4.3}$$

$$\Gamma_2 = \sqrt{\sum_{i=1}^{l} (\frac{1}{\sum_{i=1}^{l} c_i})^2} \quad \& \ c_i = count(i^{th}\ selected\ feature) \tag{4.4}$$

Based on the provided explanations, we describe our general translation quality model as Equation 4.5 where $Accuracy(.)$ can be either a sentence-level or a corpus level translation accuracy measuring function and Γ function can be any regularizer term defined for the model given the specific segmentation set of s.

$$B(s) = Accuracy(s) - \alpha\Gamma \tag{4.5}$$

4.2 Modeling Translation Latency

As mentioned in section 3.3, in the state-of-the-art, the latency scoring function is defined as the average number of segments translated in the unit of time, which can be simply computed by dividing the total number of segments by the total translation time as Equation (4.6) where γ function measures the time taken for computing $\mathcal{D}(f_j, s)$.

$$\Lambda(s) = \frac{|s|}{\sum_{j=1}^{N} \gamma(\mathcal{D}(f_j, s))} \tag{4.6}$$

However, latency is defined as the time gap between the presentation of the word in the source language and delivering the translation in the target language. The more the latency of the ST system, the more the audience should wait to receive the translation. Although Equation (4.6) has provided good results in the prior experiments, this equation does not accurately reflect all the factors that affect latency, especially in the case of speech translation, one of the important use cases of simultaneous translation, where the translation time is not the only factor causing the time gap, and the speaking time of the speaker and the text-to-speech time of the translated sentences are not negligible. As the study by Bendazzoli et al. [49] shows that the probability of simultaneous speaking of both the speaker and interpreter is less than twenty percent for English-Spanish. We assume that this time gaps are frequent, to some extent, for other language pairs as well.

On the other hand, translation times are normally computed using hardware queries (e.g. taking the run time or subtracting the end translation time from the start time) and these computations are not accurate enough to differentiate between close alternatives. We need to consider these facts in modeling the latency. Figure 4.1 tries to illustrate this idea by presenting latency gaps as hatched rectangles.

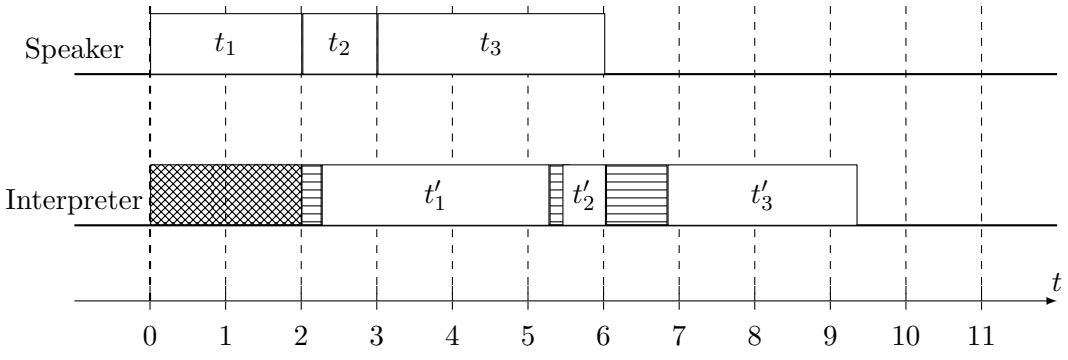

Figure 4.1: The figure to simulate the concept of latency

In this figure, when the speaker starts to talk, the ST system holds to receive the first segment. During this time, no translation is performed and the audience should wait. So, the length of the first segment is shown as a cross-hatched segment as the first term

calculated in measuring latency. After receiving the first segment, the interpreter needs some time to perform the translation of the segment; a time during which the audience should wait again. The latency caused by this translation process is shown as a horizontally hatched rectangle in Figure 4.1. Unlike the latency caused by an empty buffer (the cross-hatched rectangle), the latency caused by the translation process happens exactly before translating each segment.

Figure 4.1 can be the best case of providing the input stream to the ST system since each segment has been completely received before the system needs that segment to start the translation; however, this is not always the case. As an example of this case, we present Figure 4.2 in which the second segment is longer than the translation of the first segment. In this example, because the translation of the first segment (t'_1) has been presented before receiving the end of the second segment, the ST system has to hold for some more time (presented as the dotted rectangle in Figure 4.2) to get the end of the segment before starting the translation. This example implies that the segment reception delay can be the third factor which may impose the latency to our ST system (in addition to the empty buffer delay and segment translation delay). Note that the empty buffer delay is a special case of segment reception delay in which the system faces delay since it has not received any segments.

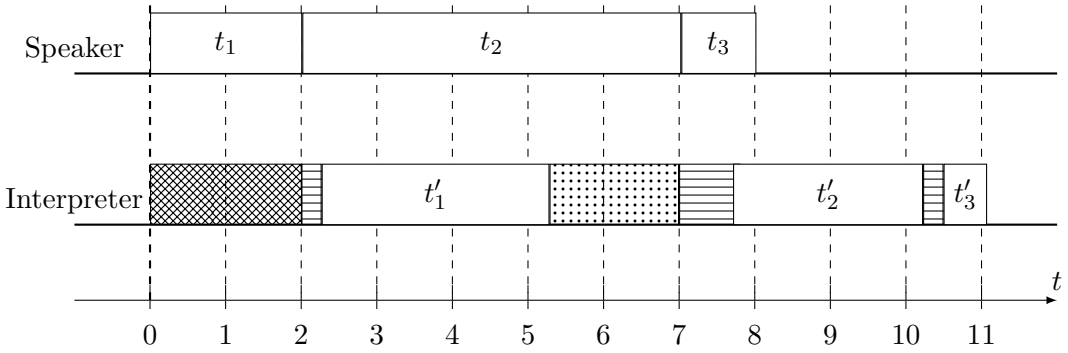

Figure 4.2: The figure to simulate the concept of latency - second case

Considering the presented examples, the latency scoring function can be described based on the two major factors resulting in it; the *segment reception delay* (SRD) and the *segment translation delay* (STD). STD can be calculated by measuring the time taken by the ST system to translate the segment. SRD can be calculated by subtracting the time duration of each segment in source side (t_i) from the time duration of the previous presented segment on the target side (t'_{i-1}) [considering $t'_0 = 0$ for the empty buffer delay]. Note that SRD can be zero if the end of current segment has been received before the previous translated segment has been completely presented ($SRD = 0$ if $t_i < t'_{i-1}$). Equation (4.7) represents the latency scoring function of a sentence (a unit of translation containing only one empty buffer delay case and an arbitrary number of segments). Note that in this equation, Λ is

the total latency (measured in time units) of a sentence with N segments, $\max(t_i - t'_{i-1}, 0)$ represents the SRD, γ_i represents the STD, and $t'_0 = 0$.

$$\Lambda = \sum_{i=1}^{N} \max(t_i - t'_{i-1}, 0) + \gamma_i \tag{4.7}$$

In Equation (4.7) the weight of SRD and STD terms are assumed to be equal while in real-time application this may not be the case. SRD may not be as important to the users as the translation delay.

4.3 Latency-Accuracy Trade-off Model

We search for the best set s, containing K segments (total number of expected chunks) over the stream of an expected known size. The cardinality of this segmentation set may vary from 0 (no segmentation at all), to $W = \sum_{i=1}^{N}\{|f_i|-1\}$ (take each word as a segment). A *'full segmentation set'* (FSS) will contain all possible W segments. $\mathcal{S}_{all}$ represents a superset containing all possible segmentation sets over F (source sentences in parallel corpora).

Algorithm 1 Pareto-Optimal Segmentation

1: $\mathcal{S}^*_0 \leftarrow \emptyset$
2: **for** $k = 1$ to K **do**
3:
$$\mathcal{S}^*_k \leftarrow \underset{p \in FSS \wedge p \notin \mathcal{S}^*_{k-1}}{\arg \text{pareto frontier}} \left\{ \; B(\mathcal{S}^*_{k-1} \cup \{p\}), \Lambda(\mathcal{S}^*_{k-1} \cup \{p\}) \; \right\}$$
4: **end for**
5: **return** $\mathcal{S}^*_K$

$\mathcal{S}^* \in \mathcal{S}_{all}$ is defined as a set of best segmentation strategies which maximizes an evaluation function over latency and accuracy (Equation 4.8). Using the proposed scoring models for latency and accuracy in sections 4.1 and 4.2, We define our search method for $\mathcal{S}^*$ as Equation (4.8).

$$\mathcal{S}^* = \underset{s \in \mathcal{S}_{all}}{\arg \text{pareto frontier}} \{B(s), \Lambda(s)\} \tag{4.8}$$

Note that in Equation (4.8), the output of *"arg pareto frontier"* is the Pareto-optimality line in the accuracy-latency plot. Therefore, $\mathcal{S}^*$ might contain more than one best set of segmentations.

$\mathcal{S}^*$ can be found using a naïve algorithm as described in Algorithm 1. However, this algorithm is computationally expensive and its time complexity is exponentially increased by increasing the size of K.

Algorithm 2 Computationally Efficient Pareto-Optimal Segmentation

1: $\Phi_0 \leftarrow \emptyset$
2: **for** $k = 1$ to K **do**
3: **for** $j = 0$ to $k - 1$ **do**
4: $\Phi' \leftarrow \{\phi : (\phi \notin \Phi_j) \wedge (count(\phi; \mathcal{F}) = k - j)\}$
5: $\Phi_{k,j} \leftarrow \Phi_j \cup \Big\{ \arg \text{pareto frontier}_{\phi \in \Phi'} \{B_\alpha(s(\mathcal{F}, \Phi_j \cup \{\phi\})), \Lambda_\alpha(s(\mathcal{F}, \Phi_j \cup \{\phi\}))\} \Big\}$
6: **end for**
7: **if** $k < K$ **then** ▷ *To reduce the computational complexity*
8: $\Phi_{k,j} \leftarrow \text{argmax}_{\phi \in \{\Phi_{k,j} : 0 \leq j \leq k\}} B_\alpha(s(\mathcal{F}, \phi))$
9: **end if**
10: $\Phi_k \leftarrow \arg \text{pareto frontier}_{\Phi \in \{\Phi_{k,j} : 0 \leq j \leq k\}} \{B_\alpha(s(\mathcal{F}, \Phi)), \Lambda_\alpha(s(\mathcal{F}, \Phi))\}$
11: **end for**
12: **return** $s(\mathcal{F}, \Phi_K)$

Algorithm 2 depicts our *Computationally Efficient Pareto-Optimal Segmentation Method* to find $\mathcal{S}^*$. The main loop (lines 2-11) each time finds the next best segmentation feature (ϕ) and adds it to the set of best segmentation points which are already found (creating a set of k points). Each feature is a bigram part of speech tag. The inner loop (line 3) implements the dynamic programming (DP) condition as in [1]. For instance, for Φ_3 this inner loop would combine the features in the set Φ_0, Φ_1 and Φ_2 (for $j = 0, 1, 2$) with the features that occur with a count of $3, 2$ and 1 respectively. So take $\Phi_{3,1}$ which is the set that is updated in line 5, the points satisfying the Pareto frontier criteria are selected out of Φ' and combined with the segmentation points of the chunked sub-segments. $\Phi_{3,1}$ contains the union of all features in Φ_1 computed previously in the DP table with new features of count 2 collected in line 4. Eventually $\Phi_{3,1}$ is used to search over Pareto frontier candidates to produce Φ_3 in line 10. Line 7 limits the computational complexity of producing Pareto frontiers out of a set containing previously computed Pareto frontiers. This line sets the most accurate point out of currently discovered Pareto frontiers, to be the only Φ of the next step (to build Φ_{j+1}). In Line 10, all possible segmentation points are analyzed (for $k < K$ there is just one point) and the Pareto frontiers out of them are stored as Φ_k. Finally, in line 12, the result of segmentation with the discovered segmentation points of Φ_K is produced and returned.

Performing the same segmentation task from Section 3.2.2, over our running example using this Pareto-optimal segmentation approach, will initially result in the same segmented sentences but our algorithm has a different intuition about choosing the best translations. Table 3.2 reports PO segmenter accuracy (using Equation (4.1)), total translation time and latency measurement values besides the reported accuracy of GDP segmenter over different segmented versions of second sample example (in Figure 3.3) as well as the actual feature set (Φ) for each specific segmentation and translation.

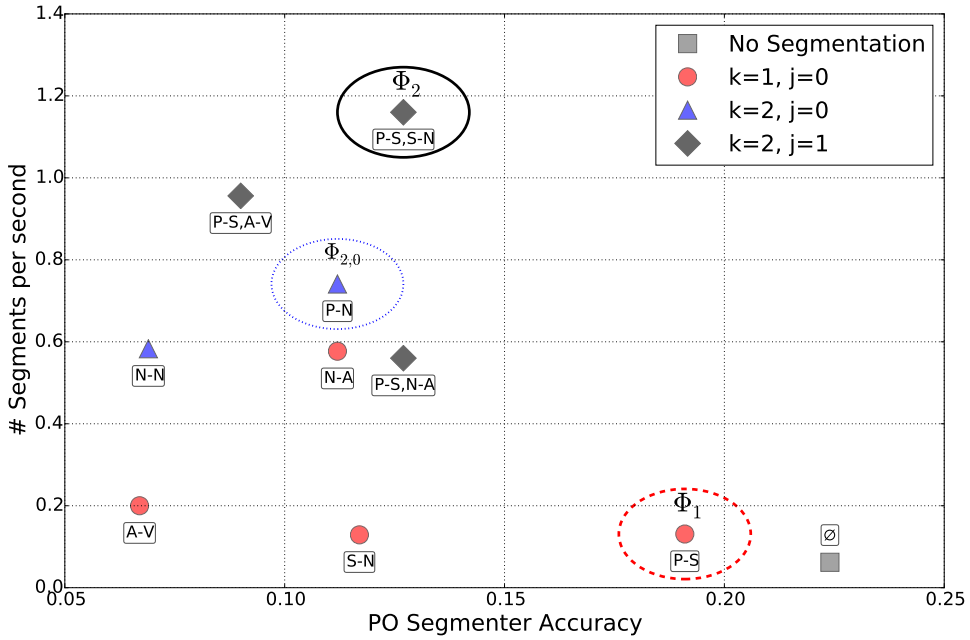

Figure 4.3: Evaluation results of different segmentation strategies in one loop of the algorithm 2

To explain the algorithm we have used the running example in Figure 3.3 and traced the output of our Algorithm 2 for $K = 2$ and provided the plot of accuracy-latency values during one execution of this algorithm in Figure 4.3. We have used Equations (4.1) and (4.6) to model the translation accuracy and latency.

By running the algorithm, we get the highest accuracy with the worst latency in the beginning. Then the algorithm starts to find the first best segmentation point ($K = 1; j = 0$) and it finds four possible segmentation candidates (depicted as filled circles in Figure 4.3). It chooses the best accuracy as Φ_1 and moves to the next round to find the second (last) segmentation point. It first considers features happening twice ($K = 2; j = 0$), then it again chooses the best accuracy as $\Phi_{2,0}$. Next, it examines the strategy of adding a single repeated feature to Φ_1 which ends up to the points depicted as diamonds. When it finds the second strategy which dominates the first strategy, it chooses the Pareto-optimal points out of the new strategy and reports it as Φ_2. Although in this example, the final segmentation set (Φ_2) contains just one point, this is not always the case.

Chapter 5

Experiments

5.1 Experimental Setup

We evaluate our approach on the English-German TED speech translation data [50]. We used Moses [51] which is a conventional phrase-based SMT system using the standard set of features in the discriminative log-linear model for SMT to produce the translations for each possible segmentation decision in our segmentation training data. We used the Stanford POS-Tagger [52] to tokenize and produce the POS tags over the train and test data (the default tag set has been the *Penn Treebank POS tag set* [53]). We used *IWSLT 2013 Train data* plus half of the *Europarl data* [54] to train our MT system on English-German and *IWSLT Test 2012* to tune it using MERT [55]. Our German language model was trained using the *monolingual data from WMT 2013 Shared Task* [1]. The segmentation training data was taken from IWSLT Shared Task *Dev 2010 and 2012 and Test 2010* and it has been tested on IWSLT Shared Task *Test 2013*. Our model parameter estimation experiments have been over IWSLT Shared Task *Test 2011* as a held-out set. Table 5.1 shows the statistics of data used in our experiments.

	Sentences	Types	Tokens
MT Train	1033491	105267	27948041
MT Tune	1730	3937	31568
Seg Train	3651	6628	72964
Seg Held-out	1461	3561	27567
Seg Test	1025	3179	22000

Table 5.1: Size of cleaned datasets used in our experiments.

The corpus translation evaluation results will be reported in BLEU+1 [20, 21] and the number of translated segments per time unit (S/T) as segment translation quality and latency, respectively, although different translation quality and latency measures have been

[1] http://statmt.org/wmt13/translation-task.html

explored in the model training phase (which will result in a set of fixed features to be segmented on). These two measures have been reported to unify the testing results reporting mechanism and make it easier to compare different modeling methods. We have tried different α regularizer coefficients for parameter tuning which will be reported in each experiment, separately.

We train the MT system and use it in all experiments. Using the trained MT system, we translate all possible segments and store them in a lattice (like [1]). In this way, we can access to a translation instantly while computing the evaluation metrics (Equation (4.8)).

We compute the translation time of each segment (γ in Equations (4.6) and (4.7)) using the Moses [51] log reported time. However, this computed translation time is the result of many factors and different seek and search algorithms and may depend on low-level issues such as cache misses on the hardware where the MT system is running. Our results were consistent across more than 50 runs in corroborating the conclusions mentioned at the end of this section. So, we do not consider such issues to be dominant in our experimental results.

In case of using Equation (4.7) as the latency model, the segment reception and delivery times (t_i and t'_i) have been computed using the *sayit*[2] script by Hal Daumé III (University of Maryland) which receives the content of the segment (in text format) and estimates the time it takes from a human to say the segment in any of the languages English (US), German, French, Italian, Spanish, and Japanese.

Based on Hal Daumé III, the script is using a pre-trained model on a speech synthesis program to extract the saying time of each word. The speech synthesis program has produced the equivalent speech form of 50K-100K most frequent words for each of the mentioned languages to extract the time lengths. Next step in preparing the 'sayit' model has been to extract features out of the words; features like number of characters, number of vowels, number of consonant groups, number of non-letters, number of vowel-consonant switches, number of digits of various lengths, and whether the word starts or ends with a vowel. The last step has been to relate the features and extracted times to estimate the regression weight vector of the feature space over the training data. The actual script estimates the saying time of a segment based on a regression mixture model over the segment tokens.

5.2 Accuracy vs. Latency-Accuracy Evaluation

In this experiment, we would like to assess the effect of adding latency to the accuracy metric in the segmentation task. In our experiments, we use two baselines: the state-of-the-art speech segmenter (Rangarajan et al. 2013) [25] and GDP (Oda et al. 2014) [1]. We implemented a heuristic segmenter based on (Rangarajan et al. 2013) [25] which segments

[2]http://www.umiacs.umd.edu/ hal/sayit.py

on surface clues such as punctuation marks. These segments reflect the idea of segmentation on silence frames of around 100ms in the ASR output used in [24]. This type of heuristic segmenter is a special case of a PO segmenter which inserts segment boundaries only for POS bigrams that end with a punctuation POS tag.

We ran our PO segmenter and the GDP segmenter with different values of parameter μ (average segment length) between 2 and 15 as well as the heuristic segmenter over the same data explained in Section 5.1. Considering the main objective of this experiment which was to compare our segmenter with the other state-of-the-art ones, we chose a single value $\alpha = 0.5$ for both GDP and PO, since this value for α avoids selecting features with extremely high or low frequency. Furthermore, in this experiment, we have used Equations (4.1) and (4.6) to model the translation accuracy and latency in our PO segmenter. The reason for this decision has been to keep our PO model as close as possible to the GDP segmenter and the heuristic segmenter to get a better sense of comparison between the results.

Due to the large number of generated points and outputs, we summarize the results in Figure 5.1 and Figure 5.2. Our experiments show that different possible values of μ will divide the accuracy-latency area into districts and each experiment is expected to exhibit a number of samples of each district for each μ. We show each district with a circle in the figures as the representative of the group of obtained points relating to one specific μ. This circle is put in the centroid of the points in the group. To show the size ratio of districts to each other, the more points found in one district, the bigger the circle is depicted. But not all the points in the group are Pareto-optimal, so we add another circle inside the outer one showing the ratio of Pareto-optimal points to the whole group of points. If all of the points for one μ were Pareto-optimal, both circles would have the same radius and the inner circle would not have been visible. In addition, we show the results of the baseline heuristic and GDP segmenter using $\Diamond$s and Xs, respectively. Moreover, we plot the real Pareto-optimality line with the actual points on it to give the reader the chance to compare the actual results of the experiments to the baseline results.

Our choice of the axis is different from previous works in this area. Commonly, segmentation results are reported with accuracy on the y-axis, but we use the x-axis instead in order to easily get a better visual understanding of pushing the Pareto-optimality line towards the trade-off area we care about (the "knee" of our plots).

Figure 5.1 shows the latency (average number of segments translated per second) and translation quality (Bleu) on the segmented translated training data. Figure 5.2 shows the latency and accuracy on the segmented translated unseen test set. These figures show that Pareto-optimality is a useful idea to explore the various options for segmentation boundary selection. Optimizing for Pareto-optimality leads to segmentations that provide latency and accuracy improvements simultaneously and provide choices for the trade-off between latency and accuracy.

While Figure 5.2 shows the overall trend for various segment sizes on the test data, we chose some specific segment lengths and show a head to head comparison between the two segmenters in Table 5.2. This comparison shows that our PO segmenter can provide faster latency compared to the GDP segmenter while retaining translation accuracy.

	$\mu = 3$		$\mu = 8$	
	Latency	Accuracy	Latency	Accuracy
GDP	0.424	0.18	0.305	0.21
PO	0.474	0.18	0.315	0.21

Table 5.2: Result comparison for $\mu = 3$ and $\mu = 8$.

Our approach of optimizing over the latency in addition to the translation quality always results in better latencies compared to the baseline while keeping the same translation quality or even improving it in some cases.

At the end of this section, we would like to bring up an important tip about the latency measure. Although, it is possible to increase the accuracy at the expense of latency, users may not tolerate the latency more than a threshold and may give up on using the ST system. There are different works on analyzing the relationship between the *maximum tolerable latency* and accuracy [4, 56] but none have stated a specific boundary on the maximum tolerable latency, to the best of our knowledge. In data selection if one increases μ one should attempt to evaluate the maximum tolerable latency for your source and target language pair for your chosen latency measure.

5.3 Granularity of Feature Space Evaluation

Comparison between different segmentation approaches showed that we can use the PO segmenter to generate segmentation models which consider both latency and accuracy. One important factor in the process of modeling the segmentation distributions over the corpus can be the feature space by which we describe the segmentation points. As mentioned in section 5.1, the default POS tag set in the experiments done in section 5.2 has been the *Penn Treebank* tag set which contains 36 different POS tags.

The first question to be answered about the feature space is how sensitive is the segmentation strategy to it. We needed to know what would happen if we choose a coarser or finer feature space or what would happen if we change the tag set. To find out the result, we designed the experiment to take the *Universal POS Tag Set* [57] with twelve coarse grained POS tags of the Table 5.3 and two finer grained sets of a hundred and four-hundred tags created using Brown Clustering [58] ID (BCID) generation. Brown Clustering is a hierarchical clustering approach that tries to maximize the mutual information between the words. Using brown clustering idea, we can cluster the whole vocabulary of words in a

language (or the top 100K most frequent words) into q Clusters and assign each cluster a unique label which can be used instead of the POS tag in our application.

We trained our monolingual English language most frequent vocabulary over *English Gigaword* Corpus [59] and used the mkcls Brown clustering tool [60] to assign the Brown clustering ids to the words. We considered two strategies for OOVs one was to assign all OOVs to one BCID equal to NULL class. The other was to assign each new OOV to a new randomly generated BCID. Both of these strategies have been tried out in our experiments. We used the mentioned held-out set (details in Table 5.1) for different parameters to find the best settings showing the differences between different feature space descriptors. Using this held-out set, we changed the settings of the experiments in section 5.2 since we were not worried about the comparison with other works anymore. Here, we explain the settings used for this set of experiments to fulfill the building blocks of Equation 4.8.

VERB	verbs (all tenses and modes)	CONJ	conjunctions
NOUN	nouns (common and proper)	DET	determiners
PRON	pronouns	NUM	cardinal numbers
ADJ	adjectives	PRT	particles or other function words
ADV	adverbs	X	other
ADP	prepositions and postpositions	.	punctuation

Table 5.3: The course Universal POS tags used in the second experiment

In the translation accuracy modeling (Equation (4.5)), we selected the corpus-level Bleu score as our ASF since we discovered that the locality of the sentence-level scoring functions will most of the times result in the oversegmentation (the evaluator does not see the translations together, so it cannot tell whether a sentence has been segmented too much, or another is remaining unsegmented). We tried all three mentioned regularization functions of Equations (4.2), (4.3), and (4.4) during our parameter estimation phase, but we just report the results of the experiments with the Equation (4.3) as the best regularizer due to the reporting space limit. Our best reporting α parameter has had the value 0.5.

In the translation latency modeling, we used the more detailed model of Equation (4.7) to consider the segment receive and delivery times in addition to the segment translation times which were not exact due to the system errors in data collection. Using the proper estimated receive and delivery times could not only help us achieve a more detailed description of one aspect of our trade-off algorithm (latency), but also it could help us decrease the effect of translation time calculation error to our modeling approach (using Equation (4.7), the segment translation time is not the only factor which can control the latency aspect of the trade-off).

Given these explanations, we ran our PO segmenter using the four mentioned POS tag sets and evaluated the results in terms of what has been reported in section 5.2. Table 5.4 shows a head-to-head comparison for $\mu = 5$. The plotted results of Figure 5.3 show that

the larger feature sets decrease the segmented stream translation speed. Another point out of the results is that the smaller sets provide the system with a larger number of Pareto-Optimal points which can be used in further steps of training and tuning the system. The last point we can grasp out of the results can be the range of parameter μ in the discovered Pareto-Optimal points which decreases as the size of feature space goes up. All in all, the results of this experiment showed that the size of feature space is very important to the performance of the unsupervised segmenter and large sized feature spaces will not perform as good as smaller ones.

After performing this experiment, we concluded that there is another adjustable parameter in the problem of training data annotation for segmenter classifier and that is the *size of feature set*. Figure 5.4 tries to depict the 3 dimensional space of (latency,accuracy,feature set size). In this figure we have compared the Pareto-optimal points of different μs for each individual plot in Figure 5.3 and we have chosen the overal Pareto-optimal points. Indeed, we have selected the best that each feature space can perform and we have put these best point sets besides each other in Figure 5.4. As it is shown in figure, except the plot which belongs to Penn Treebank tag set, all the other graphs contain at least one best point which is not dominated by any other plot. This shows that no feature space can be obviously preferred to another and we have to consider the best points belonging to each space if we need better annotations, although, computational costs and running time limits may force us to decide to choose one of the spaces and perform the training over that space, solely. In this case we recommend picking the feature space which puts more emphasis on the measure of interest (accuracy or latency).

	$\mu = 5$		
	SetSize	Segs/Sec	BLEU
Universal Tagset	12	1.06	19.63
Penn Treebank Tagset	36	0.853	20.38
Brown Uniq 100C	677	0.873	21.69
Brown Null 100C	99	0.864	21.71
Brown Uniq 400C	976	0.647	22.11
Brown Null 400C	389	0.782	22.29

Table 5.4: Result comparison for $\mu = 5$

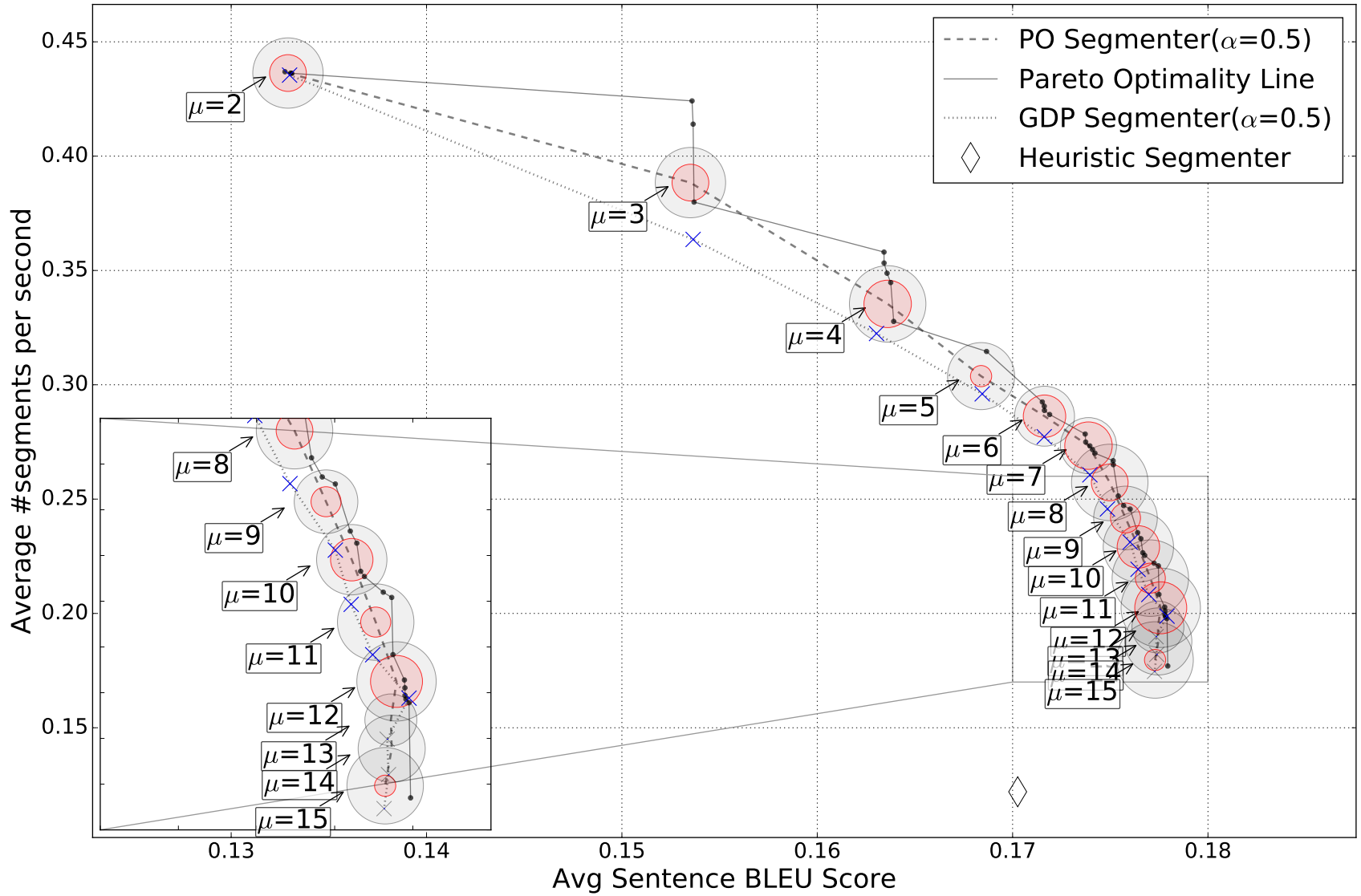

Figure 5.1: Comparison on the segmentation training data.

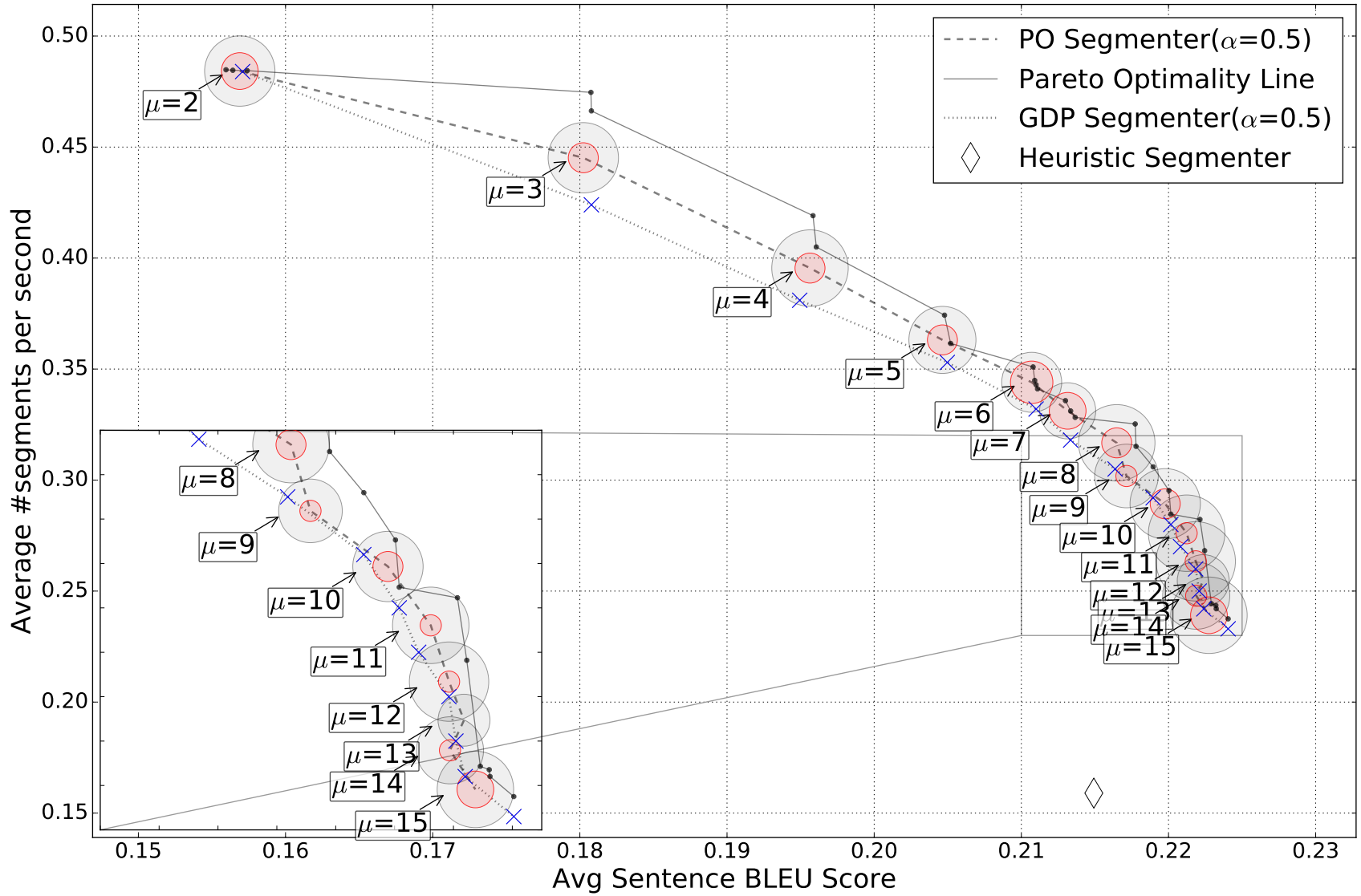

Figure 5.2: Comparison on the segmentation test data.

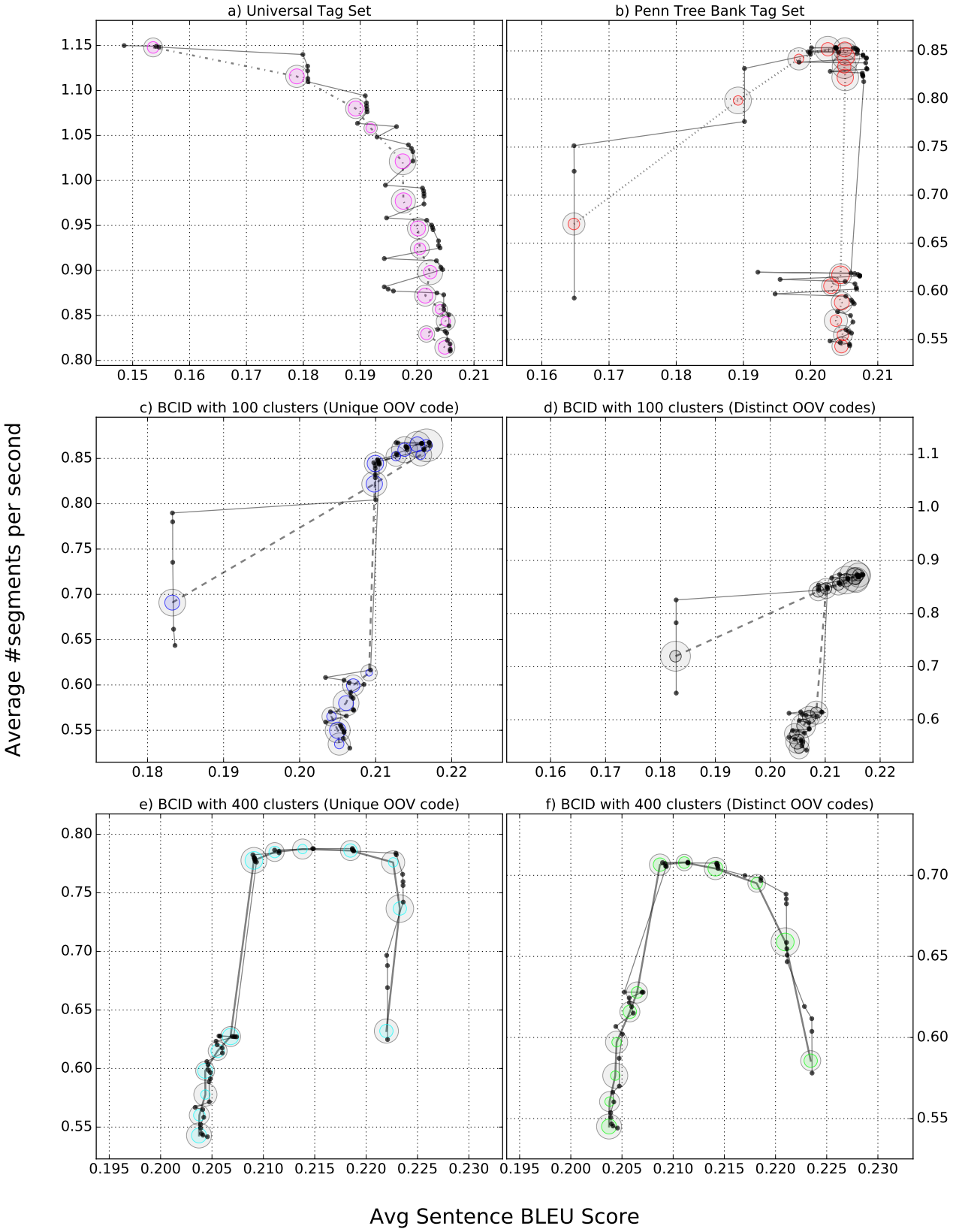

Figure 5.3: Experimental results of experiment over different feature spaces

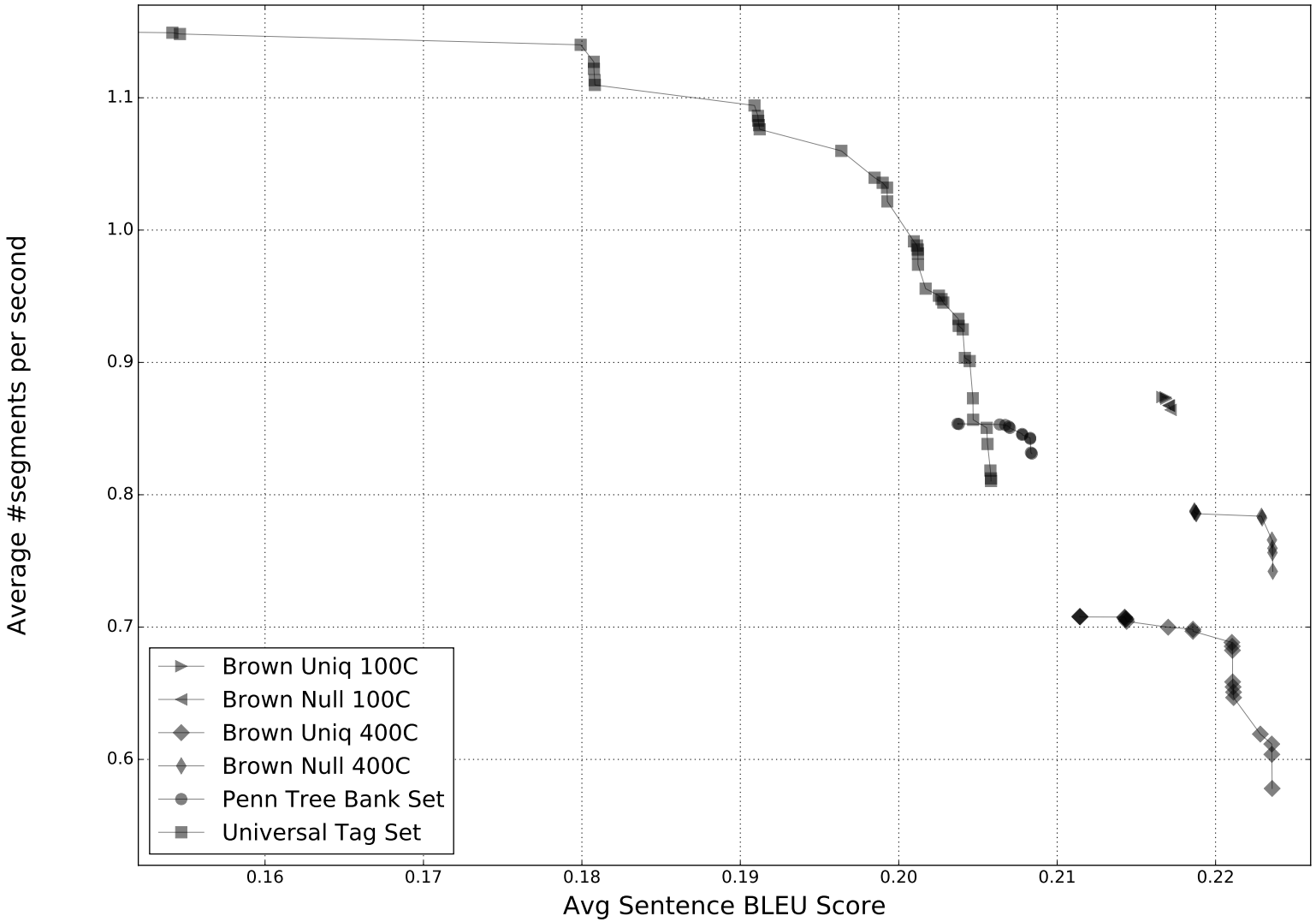

Figure 5.4: Comparison on the best each feature space can do

Chapter 6

Conclusion

This thesis explores multi-metric optimization in simultaneous translation that learns segmentations that optimize both latency and translation quality. We provided an efficient algorithm for Pareto-optimal segmentation and conducted a series of experiments that compared our approach to Oda et al. [1] which used translation quality as the only criteria to select segmentation choices. We showed that Pareto-optimality provides a better trade-off between latency and translation quality. For any segment size, Pareto-optimal segments maximize latency and translation quality. We also performed a set of experiments regarding the feature set selection and we showed that there is a negative relation between the number of selected features and the number of Pareto-Optimal points achieved.

In future work, we would like to perform experiments which can capture more context information (e.g. tri-grams and four-grams) and compare the results with the current ones. We plan to iteratively use a weighted segmentation model that is trained using the Pareto frontier points in order to iteratively find new weights for the segmentation model that will extend the "knee" of the Pareto frontier curve. Such an approach was explored in [61] for multi-metric tuning of SMT models, but has not been explored for training a segmentation model.

In addition, since the European Parliament Interpretation Corpus (EPIC) [49], an actual interpreted speech set, is available, we aim to try segmenting the original transcripted speech and translate the segmented data and compare the translation quality of our translated segments with the quality of human interpreted translated chunks.

As another next step, we would like to train a segmentation classifier using the actual annotated data to be used in a LR-Hiero stream decoder [16] and compare the translation results with the translations by other available ST systems.

Bibliography

[1] Y. Oda, G. Neubig, S. Sakti, T. Toda, and S. Nakamura, "Optimizing segmentation strategies for simultaneous speech translation," in *ACL*, 2014.

[2] J. Baigorri-Jalón, *From Paris to Nuremberg: The birth of conference interpreting*. John Benjamins Publishing Company, 2014, vol. 111.

[3] G. Neubig, "Simultaneous speech translation," 2015. [Online]. Available: http://www.phontron.com/slides/neubig15simultrans.pdf

[4] T. Mieno, G. Neubig, S. Sakti, T. Toda, and S. Nakamura, "Speed or accuracy? a study in evaluation of simultaneous speech translation," in *INTERSPEECH*, 2015.

[5] W. Weaver, "Warren weaver memorandum," 1949. [Online]. Available: http://www.mt-archive.info/Weaver-1949.pdf

[6] S. F. Chen and J. Goodman, "An empirical study of smoothing techniques for language modeling," in *Proceedings of the 34th annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 1996, pp. 310–318.

[7] P. F. Brown, J. Cocke, S. A. D. Pietra, V. J. D. Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin, "A statistical approach to machine translation," *Computational linguistics*, vol. 16, no. 2, pp. 79–85, 1990.

[8] P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, and R. L. Mercer, "The mathematics of statistical machine translation: Parameter estimation," *Computational linguistics*, vol. 19, no. 2, pp. 263–311, 1993.

[9] A. L. Berger, P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, A. S. Kehler, and R. L. Mercer, "Language translation apparatus and method using context-based translation models," Apr. 23 1996, uS Patent 5,510,981.

[10] F. J. Och and H. Ney, "A systematic comparison of various statistical alignment models," *Computational Linguistics*, vol. 29, no. 1, pp. 19–51, 2003.

[11] F. J. Och, "An efficient method for determining bilingual word classes," in *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 1999, pp. 71–76.

[12] D. Marcu and W. 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*. Association for Computational Linguistics, 2002, pp. 133–139.

[13] P. Koehn, F. J. Och, and D. 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*. Association for Computational Linguistics, 2003, pp. 48–54.

[14] D. Chiang, "A hierarchical phrase-based model for statistical machine translation," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2005, pp. 263–270.

[15] T. Watanabe, H. Tsukada, and H. Isozaki, "Left-to-right target generation for hierarchical phrase-based translation," in *Proc. of ACL*, 2006.

[16] M. Siahbani, B. Sankaran, and A. Sarkar, "Efficient left-to-right hierarchical phrase-based translation with improved reordering," in *Proc. of EMNLP*, Seattle, USA, October 2013.

[17] F. J. Och and H. Ney, "Discriminative training and maximum entropy models for statistical machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2002, pp. 295–302.

[18] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul, "A study of translation edit rate with targeted human annotation," in *Proceedings of association for machine translation in the Americas*, 2006, pp. 223–231.

[19] S. Banerjee and A. Lavie, "Meteor: An automatic metric for mt evaluation with improved correlation with human judgments," in *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, vol. 29, 2005, pp. 65–72.

[20] K. Papineni, S. Roukos, T. Ward, and W. jing Zhu, "Bleu: a method for automatic evaluation of machine translation," in *ACL*, 2002, pp. 311–318.

[21] D. Lin and F. Och, "Orange: a method for evaluating automatic evaluation metrics for machine translation," in *COLING 2004*, 2004, pp. 501–507.

[22] R. Jones, *Conference Interpreting Explained*, ser. Translation Practices Explained. Taylor & Francis, 2014.

[23] C. Fügen, A. Waibel, and M. Kolss, "Simultaneous translation of lectures and speeches," *Machine Translation*, vol. 21, no. 4, pp. 209–252, 2007.

[24] S. Bangalore, V. K. Rangarajan Sridhar, P. Kolan, L. Golipour, and A. Jimenez, "Real-time incremental speech-to-speech translation of dialogs," in *Proc. of NAACL HLT 2012*, 2012, pp. 437–445.

[25] V. K. Rangarajan Sridhar, J. Chen, S. Bangalore, A. Ljolje, and R. Chengalvarayan, "Segmentation strategies for streaming speech translation," in *NAACL*, 2013.

[26] T. Fujita, G. Neubig, S. Sakti, T. Toda, and S. Nakamura, "Simple, lexicalized choice of translation timing for simultaneous speech translation." in *INTERSPEECH*, 2013, pp. 3487–3491.

[27] M. Yarmohammadi, V. K. R. Sridhar, S. Bangalore, and B. Sankaran, "Incremental segmentation and decoding strategies for simultaneous translation," in *Proc. of IJCNLP-2013*, 2013.

[28] M. Siahbani, R. Mehdizadeh Seraj, B. Sankaran, and A. Sarkar, "Incremental translation using a hierarchical phrase-based translation system," in *Proceedings of IEEE Spoken Language Technology Workshop (SLT 2014)*, 2014.

[29] M. Wolfel, M. Kolss, F. Kraft, J. Niehues, M. Paulik, and A. Waibel, "Simultaneous machine translation of german lectures into english: Investigating research challenges for the future," in *Spoken Language Technology Workshop, 2008. SLT 2008. IEEE.* IEEE, 2008, pp. 233–236.

[30] M. Kolss, S. Vogel, and A. Waibel, "Stream decoding for simultaneous spoken language translation." in *INTERSPEECH*, 2008, pp. 2735–2738.

[31] D. Xiong, M. Zhang, and H. Li, "Learning translation boundaries for phrase-based decoding," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010, pp. 136–144.

[32] E. Matusov, D. Hillard, M. Magimai-doss, D. Hakkani-tur, M. Ostendorf, and H. Ney, "Improving speech translation with automatic boundary prediction," in *In Proc. Interspeech*, 2007, pp. 2449–2452.

[33] B. Sankaran, A. Grewal, and A. Sarkar, "Incremental decoding for phrase-based statistical machine translation," in *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, ser. WMT, 2010.

[34] A. Finch, X. Wang, M. Utiyama, and E. Sumita, "Hierarchical phrase-based stream decoding," in *Proc. of EMNLP*, Lisbon, Portugal, September 2015.

[35] H. He, A. Grissom II, J. Morgan, J. Boyd-Graber, and H. Daumé III, "Syntax-based rewriting for simultaneous machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, 2015.

[36] M. Kolss, M. Wölfel, F. Kraft, J. Niehues, M. Paulik, and A. Waibel, "Simultaneous german-english lecture translation." in *IWSLT*, 2008, pp. 174–181.

[37] S. Matsuda, X. Hu, Y. Shiga, H. Kashioka, C. Hori, K. Yasuda, H. Okuma, M. Uchiyama, E. Sumita, H. Kawai *et al.*, "Multilingual speech-to-speech translation system: Voicetra," in *Mobile Data Management (MDM), 2013 IEEE 14th International Conference on*, vol. 2. IEEE, 2013, pp. 229–233.

[38] W. D. Lewis, "Skype translator: Breaking down language and hearing barriers," in *Proceedings of Translating and the Computer (TC37)*, November 2015.

[39] A. Axelrod, Q. Li, and W. Lewis, "Applications of data selection via cross-entropy difference for real-world statistical machine translation," in *Proceedings of the International Workshop on Spoken Language Translation (IWSLT 2012)*. Internaltional Workshop on Spoken Language Translation (IWSLT), December 2012.

[40] W. Lewis and S. Eetemadi, "Dramatically reducing training data size through vocabulary saturation," in *Proceedings of the Eighth Workshop on Statistical Machine Translation, ACL 2013*. ACL, August 2013.

[41] S. Eetemadi, W. Lewis, K. Toutanova, and H. Radha, "Survey of data-selection methods in statistical machine translation," *Machine Translation*, December 2015.

[42] W. D. Lewis, C. Federmann, and Y. Xin, "Applying cross-entropy difference for selecting parallel training data from publicly available sources for conversational machine translation," in *Proceedings of IWSLT 2015*, December 2015.

[43] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine learning*, vol. 79, no. 1-2, pp. 151–175, 2010.

[44] S. Rao, I. R. Lane, and T. Schultz, "Optimizing sentence segmentation for spoken language translation." in *INTERSPEECH*, 2007, pp. 2845–2848.

[45] E. Matusov, D. Hillard, M. Magimai-Doss, D. Z. Hakkani-Tür, M. Ostendorf, and H. Ney, "Improving speech translation with automatic boundary prediction." in *INTERSPEECH*, vol. 7, 2007, pp. 2449–2452.

[46] M. Cettolo and M. Federico, "Text segmentation criteria for statistical machine translation," in *Advances in Natural Language Processing*. Springer, 2006, pp. 664–673.

[47] H. S. Shavarani, M. Siahbani, R. M. Seraj, and A. Sarkar, "Learning segmentations that balance latency versus quality in spoken language translation," in *The 12th International Workshop on Spoken Language Translation (IWSLT 2015)*, 2015, pp. 217–224.

[48] K. Duh, K. Sudoh, X. Wu, H. Tsukada, and M. Nagata, "Learning to translate with multiple objectives," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2012.

[49] C. Bendazzoli and A. Sandrelli, "An approach to corpus-based interpreting studies: developing epic (european parliament interpreting corpus)," *2005-2007), Proceedings of the Marie Curie Euroconferences MuTra: Challenges of Multidimensional Translation-Saarbrücken*, pp. 2–6, 2005.

[50] M. Cettolo, C. Girardi, and M. Federico, "Wit^3: Web inventory of transcribed and translated talks," in *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, Trento, Italy, May 2012, pp. 261–268.

[51] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. 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*, ser. ACL '07. Stroudsburg, PA, USA: Association for Computational Linguistics, 2007, pp. 177–180.

[52] K. Toutanova, D. Klein, C. Manning, and Y. Singer, "Feature-rich part-of-speech tagging with a cyclic dependency network," in *Proceedings of HLT-NAACL 2003*, 2003, pp. 252–259.

[53] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of english: The penn treebank," *Computational linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[54] P. Koehn, "Europarl: A parallel corpus for statistical machine translation," in *MT Summit*, 2005.

[55] F. J. Och, "Minimum error rate training in statistical machine translation," in *Proc. of ACL*, 2003.

[56] V. K. R. Sridhar, J. Chen, and S. Bangalore, "Corpus analysis of simultaneous interpretation data for improving real time speech translation." in *INTERSPEECH*, 2013, pp. 3468–3472.

[57] S. Petrov, D. Das, and R. McDonald, "A universal part-of-speech tagset," *arXiv preprint arXiv:1104.2086*, 2011.

[58] P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai, "Class-based n-gram models of natural language," *Computational linguistics*, vol. 18, no. 4, pp. 467–479, 1992.

[59] D. Graff, J. Kong, K. Chen, and K. Maeda, "English gigaword," *Linguistic Data Consortium, Philadelphia*, 2003.

[60] F. J. Och, "An efficient method for determining bilingual word classes," in *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 1999, pp. 71–76.

[61] B. Sankaran, A. Sarkar, and K. Duh, "Multi-metric optimization using ensemble tuning," in *NAACL*, 2013.