Learning Segmentations that Balance Latency versus Quality in Spoken Language Translation

Hassan S. Shavarani Maryam Siahbani Ramtin M. Seraj Anoop Sarkar

School of Computing Science
Simon Fraser University, Burnaby BC. Canada
{sshavara,msiahban,rmehdiza,anoop}@cs.sfu.ca

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 (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, using Pareto-optimality, for finding 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) [1]. 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 latency and translation quality.

1. Introduction

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 is to segment the incoming speech stream into chunks that can capture reordering between source and target languages and translate these chunks in order to improve latency.

The technique of segmenting the input is often referred to as the "salami technique" in the field of conference interpreting (by humans) [2] 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 on monolingual features in the input such as pauses and other similar cues [3, 4, 5, 6] 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 [7, 8]. 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 [9]. The impact of the choice of segment length has been studied in some previous work on segmentation [10] and stream decoding [11]. 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 paper, 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. 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 [12] score.

In this work, we extend previous work [1] on finding optimal segments and provide a more appealing algorithm, using Pareto-optimality, for finding good segment boundaries that can balance the trade-off between latency and translation quality. Latency is measured in terms of segments translated per second and translation quality is measured using a translation evaluation measure such as BLEU score. Using data that was produced by simultaneous translation by human interpreters, the study in Mieno et al. [13] considers how humans view the tradeoff 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 paper we provide efficient algorithms to find segmentation decisions that explicitly rank these decisions based on the trade-off between latency and translation quality.

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 2013, 2012 and 2010. The results show that we can provide qualitatively better segments (compared to previous work) that improve latency without substantially hurting translation quality.

2. Segments that Maximize Translation Quality

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][1] has a parameter for the number of expected segments, K, which is given by Equation 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.

$$K := max(0, \left\lfloor \frac{\sum_{f \in F} |f|}{\mu} \right\rfloor - N) \quad (1)$$

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:

$$B(s) = \sum_{j=1}^{N} \beta(\mathcal{D}(f_j, s), e_j) \quad (2)$$

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. We use per-sentence smoothed BLEU score (BLEU+1) [12, 14] in this paper. $B(s)$ 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 $k = 1 \dots K$, 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.

$$B_\alpha(s) = B(s) - \alpha|\Phi| \quad (3)$$

In an English-German translation task, consider the three-sentence sample example of Figure 1 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 general 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 1: 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).

Table 1 shows the feature frequencies of the sample corpus. For $\mu = 1$ (setting each word as one segment) for the

[1] It might seem so, but we are not duplicating a lot of content from their paper, and what is included is necessary to understand our proposed algorithm. We provide an example that is used to explain our algorithm as well and which will help the reader understand the difference with our proposed algorithm. We also change their notation to match our own.

	$\Phi_{sent\ 2}$	#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 2: For the second sentence in Figure 1, 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 3), 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.

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 1: Frequencies of the bigram part of speech tags in the example from Figure 1.

example in Figure 1, 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 $s_{possible}$.

$s_{possible}$={{}, {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 2 shows the possible segmentations of the second sentence of the example in Figure 1 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 pick either sentence 8 or 10 from Table 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 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 in spite of the regularizer on the size of Φ. Equation 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. 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. To get to this point, we will first review the concept of Pareto-optimality as it shows how one could choose different equally important points in the two-dimensional space of latency and accuracy.

Considering translation latency-accuracy points depicted in Figure 2 as an example, a point will be *Pareto-Optimal if*

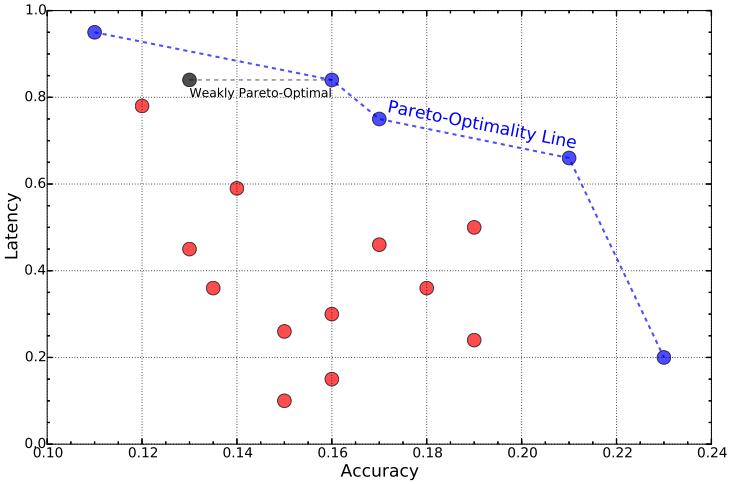

Figure 2: Pareto-Optimal and Weakly Pareto-Optimal points as well as the dominated points scored on the two metrics of interest in this paper: latency and translation accuracy scores (e.g. BLEU*).*

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

$$\Lambda\{p_2\} < \Lambda\{p_1\} \;\&\; B\{p_2\} < B\{p_1\} \tag{4}$$

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

Based on these concepts and paying attention to the Pareto-Optimality Line in Figure 2, 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. 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 segments.

In our approach, we use the same notation of K and μ introduced in the Greedy approach of Section 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 quality. 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 latency (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. We consider different values of K in our algorithm to balance the latency-accuracy trade-off.

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).

$\mathcal{S}^* \in \mathcal{S}_{all}$ is defined as a set of best segmentation strategies which maximizes an evaluation function over latency and accuracy (Equation 7). We propose two scoring functions for latency and accuracy (Equations 5 and 6 respectively) which are used in Equation 7.

We modify the accuracy function of Equation 3 to address the problem of spreading the segmentation positions (Equation 5).

$$B_\alpha(s) = \sum_{j=1}^{N} \frac{\beta(\mathcal{D}(f_j, s_j), e_j)}{|s_j|} - \alpha|\Phi| \tag{5}$$

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.

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 the total number of segments by the total translation time as follows:

$$\Lambda_\alpha(s) = \frac{|s|}{\sum_{j=1}^{N} \gamma(\mathcal{D}(f_j, s))} - \alpha|\Phi| \tag{6}$$

where γ function measures the time taken for computing $\mathcal{D}(f_j, s)$. Note that, here we use the same regularization strategy used in Equation 3 (see Section 2).

$$\mathcal{S}^* = \underset{s \in \mathcal{S}_{all}}{\arg \text{pareto frontier}} \{B_\alpha(s), \Lambda_\alpha(s)\} \tag{7}$$

Note that in Equation 7, 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 1 Pareto-Optimal Segmentation

1: $\mathcal{S}_0^* \leftarrow \emptyset$
2: **for** $k = 1$ to K **do**
3:

$$\mathcal{S}_k^* \leftarrow \underset{p \in FSS \wedge p \notin \mathcal{S}_{k-1}^*}{\arg \text{pareto frontier}} \left\{ \begin{array}{c} B_\alpha(\mathcal{S}_{k-1}^* \cup \{p\}), \\ \Lambda_\alpha(\mathcal{S}_{k-1}^* \cup \{p\}) \end{array} \right\}$$

4: **end for**
5: **return** $\mathcal{S}_K^*$

Algorithm 2 depicts our *Computationally Efficient Pareto-Optimal Segmentation Method* to find $\mathcal{S}^*$. The main

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: $\Phi_{k,j} \leftarrow \Phi_j \cup \left\{ \arg \text{pareto frontier}_{\phi \in \Phi'} \{B_\alpha(s(\mathcal{F}, \Phi_j \cup \{\phi\})), \Lambda_\alpha(s(\mathcal{F}, \Phi_j \cup \{\phi\}))\} \right\}$
6: **end for**
7: **if** $k < K$ **then** ▷ *To reduce the computational complexity*
8: $\Phi_{k,j} \leftarrow \text{argmax}_{\phi \in \{\Phi_{k,j} : 0 \le j \le k\}} B_\alpha(s(\mathcal{F}, \phi))$
9: **end if**
10: $\Phi_k \leftarrow \arg \text{pareto frontier}_{\Phi \in \{\Phi_{k,j} : 0 \le j \le k\}} \{B_\alpha(s(\mathcal{F}, \Phi)), \Lambda_\alpha(s(\mathcal{F}, \Phi))\}$
11: **end for**
12: **return** $s(\mathcal{F}, \Phi_K)$

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 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 2 reports PO segmenter accuracy, total translation time and latency measurement values besides the reported accuracy of GDP segmenter over different segmented versions of second sample example (in Figure 1) as well as the actual feature set (Φ) for each specific segmentation and translation.

To explain the algorithm we have used the running example in Figure 1 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 3. 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

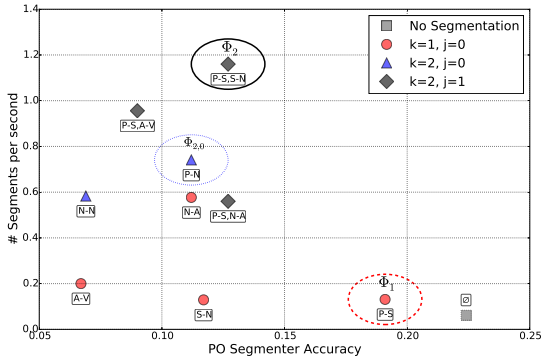

Figure 3: Evaluation results of different segmentation strategies in one loop of the algorithm 2

segmentation candidates (depicted as filled circles in Figure 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.

4. Experiments

4.1. Experimental Setup

We evaluate our approach on the English-German TED speech translation data [15]. We used Moses [16] 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 [17] to tokenize and produce the POS tags over the train and test data. We used *IWSLT 2013 Train data* plus half of the *Europarl data* [18] to train our MT system on English-German and *IWSLT Test 2012* to tune it using MERT [19]. Our German language model was trained using

the *monolingual data from WMT 2013 Shared Task* [2]. 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*. Table 3 shows the statistics of data used in our experiments.

	Sentences	Types	Tokens
MT Train	1033491	105267	27948041
MT Tune	1730	3937	31568
Seg Train	3669	6773	74883
Seg Test	1025	3181	22026

Table 3: Size of datasets used in our experiments.

For the evaluation metrics used to evaluate segment translation quality and latency, we use BLEU+1 [12, 14] and the number of translated segments per time unit (S/T), respectively. We set the α regularizer coefficient to 0.5, for both GDP and Pareto-Optimal (PO) segmenters. This value for α avoids selecting features with extremely high or low frequency.

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 7).

We compute the time of translations over each segment in order to evaluate the latency of translations. 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 many runs so we do not consider such issues to be dominant in our experimental results.

4.2. Accuracy vs. Latency-Accuracy Evaluation

In this experiment, we would like to assess the effect of adding latency to accuracy metric in the segmentation task. In our experiments, we use two baselines: the state-of-the-art speech segmenters (Rangarajan et al. 2013) [5] and GDP (Oda et al. 2014) [1]. We implemented a heuristic segmenter based on (Rangarajan et al. 2013) [5] which segments 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 [4]. 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 4.1. Due to the large number of generated points and outputs, we summarize the results in Figure 4 and Figure 5.

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

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 depiceted. 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 *X*s, 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 work 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 4 shows the latency (average number of segments translated per second) and translation quality (BLEU) on the training data. Figure 5 shows the latency and accuracy on the unseen test set. These figures show that Pareto-optimality is a useful methodology 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 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 4. 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 4: 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.

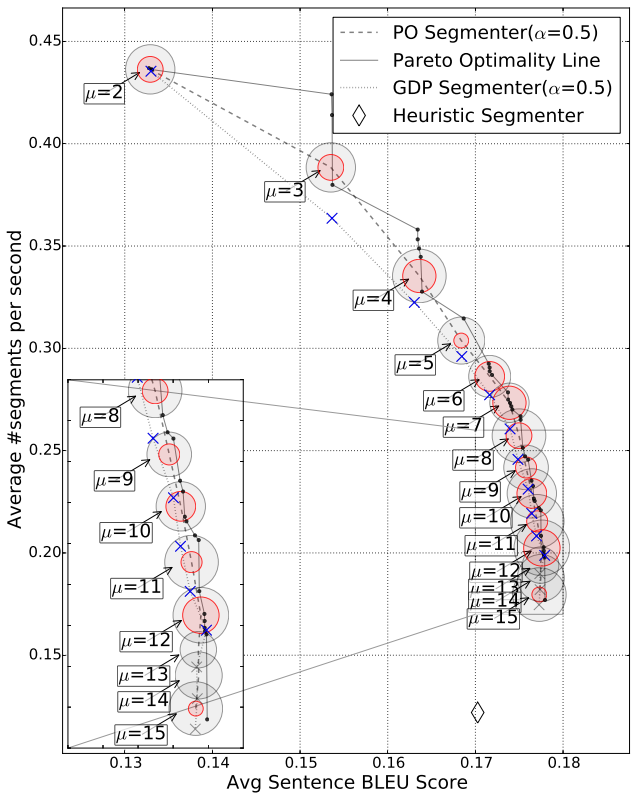

Figure 4: Comparison on the segmentation training data.

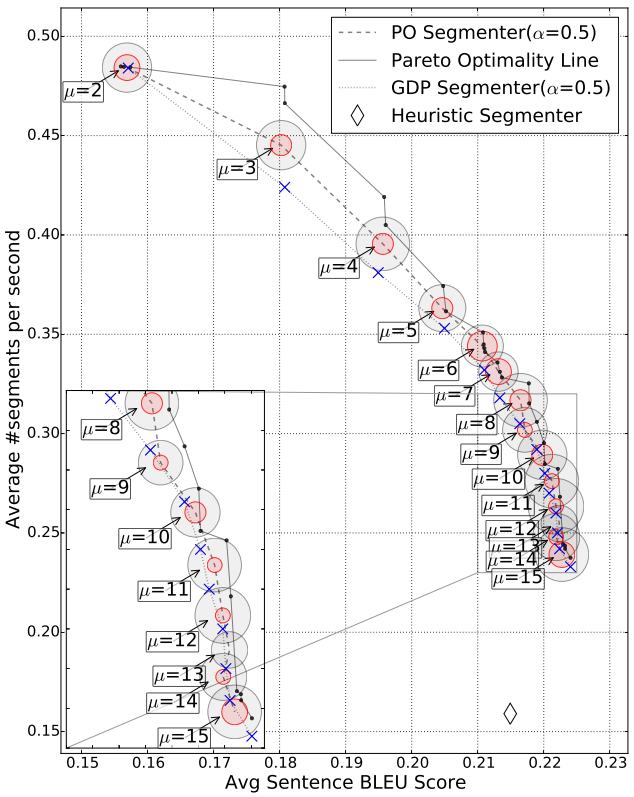

Figure 5: Comparison on the segmentation test data.

5. Related Work

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 [3, 4]. Segmentation methods applied on the transcribed text can be divided to two categories: heuristic methods which use linguistic cues, like conjunctions, commas, etc. [5]; and statistical methods which train a classifier to predict the segmentation boundaries. Some early methods use prosodic and lexical cues as features to predict soft boundaries [20]; while most recent methods rely on word alignment information to identifies contiguous blocks of text that do not contain alignments to words outside them [7, 8]. In addition to these segmentation approaches which are applied before calling the translation decoder, there is another strategy which perform the segmentation during decoding which is usually called stream or incremental decoding. Different incremental decoding approaches have been proposed for phrase-based [11, 21] and hierarchical phrase- based translation [8, 22]. He et al. [23] 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 [13]. 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 paper.

6. Conclusion

This paper explores multi-metric optimization in simultaneous translation that learns segmentations that optimize both latency and translation quality. We provide 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.

In future work, we plan to iteratively use a weighted segmentation model that is trained using the Pareto frontier in order to iteratively find new weights for the segmentation model that will extend the "knee" of the Pareto frontier. Such an approach was explored in [24] for multi-metric tuning of SMT models, but has not been explored for training a segmentation model.

7. References

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

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

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

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

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

[15] 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.

[16] 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.

[17] 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.

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

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

[20] 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.

[21] 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.

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

[23] 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.

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