Simultaneous Translation using Optimized Segmentation

Maryam Siahbani maryam.siahbani@ufv.ca
Department of Computer Information System,
University of Fraser Valley,
Abbotsford, V2S 7M7, Canada

Hassan S. Shavarani sshavara@sfu.ca
Ashkan Alinejad aalineja@sfu.ca
Anoop Sarkar anoop@sfu.ca
School of Computing Science,
Simon Fraser University,
Burnaby, V5A 1S6, Canada

Abstract

Previous simultaneous translation approaches either use a separate segmentation step followed by a machine translation decoder or rely on the decoder to segment and translate without training the segmenter to minimize delay or increase translation quality. We integrate a segmentation model and an incremental decoding algorithm to create an automatic simultaneous translation framework. Oda et al. (2014) propose a method to provide annotated data for sentence segmentation. This work uses this data to train a segmentation model that is integrated with a novel simultaneous translation decoding algorithm. We show that this approach is more accurate than previously proposed segmentation models when integrated with a translation decoder. Our results on the speech translation of TED talks from English to German show that our system can achieve translation quality close to the offline translation system while at the same time minimizing the delay in producing the translations incrementally. Our approach also outperforms other comparable simultaneous translation systems in terms of translation quality and latency.

1 Introduction

In simultaneous translation the incoming speech stream is segmented and translated incrementally to reduce the latency. There are two approaches for simultaneous translation task: *sentence segmentation* and *incremental decoding*, also called *stream decoding*. In incremental decoding, incoming words are fed into the decoder one-by-one, and the decoder updates its internal state. The decoder is responsible to decide when to begin the translation process and when to output the translation. Incremental decoding algorithms have been proposed for phrase-based (Kolss et al., 2008; Sankaran et al., 2010) translation, hierarchical phrase-based (Finch et al., 2015) and syntax-based (Oda et al., 2015) translation systems.

Real-world speech translation systems estimate the sentence boundaries using punctuation insertion methods (Rangarajan Sridhar et al., 2013). As a result, recent work in simultaneous machine translation assume the input is already segmented into sentences, and focus on splitting the sentences into shorter subsequences of words (*segments*). This approach is called sentence segmentation. As soon as a segment is recognized, it is given to a decoder to generate and output the translation for that segment.

Different methods have been proposed for sentence segmentation. Some use prosodic boundaries for segmentation (Fügen et al., 2007; Bangalore et al., 2012), while others use classification models. For example Rangarajan Sridhar et al. (2013) train a classifier to predict punctuation marks. The other approaches rely on the reordering probabilities of phrases to predict the segment boundaries (Fujita et al., 2013; Yarmohammadi et al., 2013; Siahbani et al., 2014). Oda et al. (2014) propose a method to provide annotated data for sentence segmentation which can be used in training a segmentation model. This method which later have been extended by (Shavarani et al., 2015) aims to find the best segmentation strategy for a given set of sentence which optimizes the translation accuracy. But the obtained annotated data has never been used in an end-to-end simultaneous translation system.

In this work, we focus on sentence segmentation approach for simultaneous translation. We model the segmentation task as a classification problem and investigate different methods to provide annotated data for training the segmentation model (Section 2). We modify Oda et al. (2014) approach by propose a new formula to compute the latency and use Pareto-optimality for finding good segment boundaries that can balance the trade-off between latency versus translation quality. We use the obtained annotated data to train a segmentation model. We conduct various experiments to evaluate the segmentation model and show that this model outperforms previous segmentation models in terms of accuracy.

Segmentation-based simultaneous translation approaches typically use a traditional phrase-based decoder to translate each input segment individually. Although hierarchical phrase-based (Hiero) translation system usually performs comparable to or better than conventional phrase-based systems, they use CKY based decoding algorithm which requires the entire input sentence to generate the translation. While phrase-based decoders generate translation in a left-to-right manner and it makes phrase-based systems more suitable for simultaneous translation than Hiero.

We use *LR-Hiero* for simultaneous translation which uses hierarchical phrase-based translation models while generates the translation in left-to-right manner (Watanabe et al., 2006; Siahbani et al., 2013). We modify LR-Hiero decoder and combine it with the segmentation model to incrementally translate the input sentence (stream of words).

We evaluate our simultaneous translation system on the speech translation of TED talks on English-German. The experimental results show that our system can achieve translation quality close to offline SMT system while generate the output translation words around twenty times faster. We also compare our simultaneous translation system to neural machine translation (NMT) simultaneous translation systems. Our system outperforms the state of the art NMT-based simultaneous translation system in both translation quality and latency.

2 Sentence Segmentation

The segmentation task is usually modeled as a binary classifier which is called for each input word to determine if it is a segment boundary or not. To train the segmentation classifier we need some training data in the form of sentences with labeled words showing if a word is a segment boundary or not. For each sentence $\mathbf{f} = \langle f_1 \dots f_J \rangle$ different possible segmentations exist which grows exponentially with the length of the sentence. Finding the best segmentation can be quite difficult, as it requires a brute-force search over all possible segmentations which is intractable.

Different heuristics have been proposed to efficiently solve this problem. The two main approaches are: *alignment-based* and *translation-based* heuristics. We modify the translation-based heuristic to use in our translation framework. We will briefly discuss both approaches in the following and compare their performance in Section 4.

These heuristics use parallel data on the source and target languages of the simultaneous translation task to create labeled training data for the segmentation model. We define $\mathcal{C} = \langle F, E \rangle$ as a parallel corpus

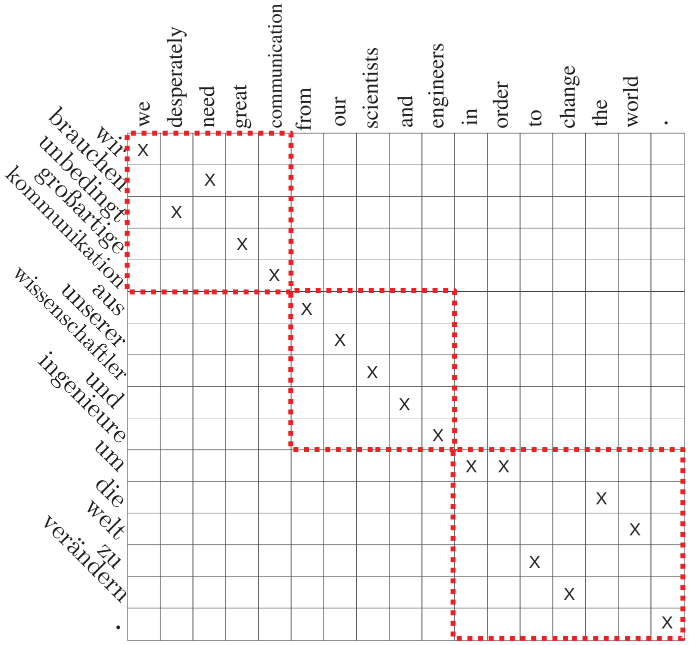


Figure 1: Word alignment matrix for an English-German sentence pair. Monotone phrases are shown in dashed line rectangles.

of source and target sentences used to extract training data.

2.1 Alignment-based Heuristic

The idea behind alignment-based heuristic is to split the input sentence into segments which can be translated to the target language monotonically (Yarmohammadi et al., 2013; Siahbani et al., 2014). To achieve this goal, the segmentation task is simplified to the problem of segmenting the source sentence in a way that reordering just occurs inside segments but not across segments. To find such segmentation we can leverage word alignment models. Given the word alignment $\mathbf{a} = \langle a_1 \dots a_J \rangle$ for a sentence pair $\langle \mathbf{f}, \mathbf{e} \rangle$, we can segment the source sentence $\mathbf{f} = \langle f_1 \dots f_J \rangle$, into a set of segments (phrases) $\mathbf{s} = \langle s_1 \dots s_K \rangle$:

$$s_k = \langle j_{k-1}, j_k \rangle \qquad \forall k = 1 \dots K, j_0 = 1 \tag{1}$$

To restrict the reorderings inside the segments, we should extract segments where $a_{j_{k-1}} < a_{j_k}$ for $k = 1 \dots K$. This segmentation results in a phrase alignment for the sentence pair $\langle \mathbf{f}, \mathbf{e} \rangle$ called *monotonic phrase alignment*. Figure 1 shows word alignment matrix and monotonic phrase alignment for an English-German sentence pair. Monotonic phrase alignment for a sentence pair can be found in linear time, given the word alignment. Experimentally, it has been shown that translation quality improves significantly with longer phrases (Koehn et al., 2003). Therefore to avoid too many short segments which could lead to word-to-word translation, the segmentation algorithm is given a constraint based on a constant μ and segments of length less than μ are disallowed[1].

Usually a word alignment model is trained over a parallel corpus containing the parallel data of the

[1] μ is usually set to 4 (Yarmohammadi et al., 2013; Siahbani et al., 2014).

translation task and parallel corpus $\mathcal{C}$. It provides us the oracle word alignment for $\mathcal{C}$ which can be used to extract labeled data for the segmentation classifier.

2.2 Translation-based Heuristic

The translation-based segmentation heuristic (Oda et al., 2014) focuses on obtaining segmentation points that are the least harmful to translation accuracy. This heuristic performs the segmentation using an iterative greedy approach. It starts with an empty set of segmentation points and each time tries to find a segmentation point in the given corpus which is the least harming to the translation accuracy. Segmentation points are described using a set of features. Different kinds of features can be used such as bigram POS tags, lexical terms, parsing related features and etc. Each feature is used as a metric to recognize segmentation points in a given input sentence. For instance, suppose we find a feature which is a bigram POS tag: *NNS-IN*. Using this feature we can find one segmentation point at index 10 in the English-German sentence shown in Figure 1 (the segmentation point is surrounded by two words *engineers* and *in* corresponding to POS tags NNS and IN). This approach finds an optimal set of features, based on a parallel corpus[2].

Given a parallel corpus $\mathcal{C} = \langle F, E \rangle$ and an expected number of segments, K, the translation-based segmentation heuristic first extracts all features over the corpus along with their frequencies, $\langle c_1 \ldots c_m \rangle$. The translation-based segmentation heuristic tries to find a feature set s containing $l(\leq m)$ features according to the least harmful segmentation criterion for the translation accuracy where $\sum_{i=1}^{l} c_{s_i} = K$ (features of set s appear in K points of source corpus which result in K segments).

Oda et al. (2014) define translation accuracy as the summation of sentence-level BLEU score (Lin and Och, 2004) of the translations of segmented sentences. The feature set is initialized as empty ($s = \{\}$), then the best feature (adding it to the corpus causes the least translation loss) is greedily chosen and added to the feature set. Once a feature has been chosen, all the points exhibiting that feature are segmented at the same time. This approach requires running the translation system for each possible feature in each iteration which takes a long time. To overcome this issue, they propose dynamic programming (DP) and call their approach Greedy-DP (GDP)[3].

However, this approach does not consider the latency to choose the features and therefore does not model the trade-off between accuracy and latency. This trade-off is crucial in designing a simultaneous translation systems. Shavarani et al. (2015) used *Pareto Optimality* to model the trade-off between accuracy and latency. In this approach, the algorithm iteratively goes over the corpus and examines the available features by computing the difference of translation accuracy (Δ) before and after applying each available feature to the source corpus. The features which cause least translation loss (the smallest Δ) are selected as candidate points. Among them, the feature which causes the least latency or the highest throughput is selected to be added to the feature set s. In the previous works latency was simply defined as the number of segments divided by the total translation time (Oda et al., 2014; Shavarani et al., 2015).

We extend the Pareto optimality approach by modifying the definition of both objective functions: translation accuracy and latency.

2.2.1 Translation Accuracy

Our primarily experiments show that using the sentence-level BLEU to measure the translation accuracy in GDP (and Pareto Optimality approach) tends to oversegment some sentences in the corpus and leave the other sentences untouched. To overcome this issue, we propose to use corpus-level BLEU (Papineni et al., 2002) to measure translation accuracy. The corpus-level BLEU gives a general view over the corpus

[2] The feature set which results in the best segmentation strategy (a set of segmentation points which gives us the best translation for the given parallel corpus).

[3] Please refer to (Oda et al., 2014) for more details

therefore it alleviate the tendency to localize the segmentation.

2.2.2 Latency

In simultaneous translation, the translation process starts before receiving the end of 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.

Based on this idea we define the translation delay measure as a function of two different types of delay factors; the transmission delay and the translation delay. The transmission delay measures the amount of time the system is waiting to reach the end of current segment after producing the translation of the previous segment, completely. The translation delay on the other side is the amount of time it takes for the system to perform the mapping of source side utterances into the target side equivalents.

Equation 2 formulates the idea about the latency where we assume latency measure Λ to assess the latency of a sentence containing N segments and each segment being translated as soon as it is ready. We assume the target side will have a buffer which will keep the already translated segments in a queue and pops the translated segments from the queue while any is available and will wait to receive one if the buffer is empty. In Equation 2, t_i represents the time moment that the i^{th} segment is ready and t'_{i-1} points to the time moment that the translated segment $i-1$ has been completely delivered to the audience. Both of the measures start from 0 for each new sentence and $t'_0 = 0$ always holds. The phrase $\max(t_i - t'_{i-1}, 0)$ means that we will not have any transmission delay if the time moment that we receive the segment i is before the time moment that we finish delivering the translated segment t'_{i-1} otherwise the transmission delay will be equal to $t_i - t'_{i-1}$. δ_i represents the duration of the time it takes to translate the i^{th} source side segment into the target side language.

$$\Lambda = \sum_{i=1}^{N} \max(t_i - t'_{i-1}, 0) + \delta_i \tag{2}$$

2.3 Segmentation Model

Given a set of sentences along with the gold segmentations, we can prepare the training data for the segmentation model. For each segment in the gold segmentation we create a positive training example corresponding to the whole segment and a set of negative examples corresponding to each smaller segment. For example for a segment $\langle i, j\rangle$, the positive example is (i, j), and negative examples are $[(i, i+1), (i, i+2), \ldots (i, j-1)]$.

Using this training data, we train a binary classifier (using a log-linear model) based on different feature sets. Basic features, used in (Yarmohammadi et al., 2013), are: the last word of the segment (candidate segment boundary), the position of the boundary in the sentence, and the candidate segment length (set1). Siahbani et al. (2014) proposed different sets of features for segmentation task including Part Of Speech (POS) tags and feedback from the decoder (given from the partial hypotheses of decoder during translation). POS tags showed promising results and fast to be computed. In addition to POS tags we also propose to use two features created based on reordering. We compare four different sets of features including the basic features (set1) to train the segmentation model:

- **Part of Speech tags**: The first group uses POS tags of the candidate segment as features. We considered the last three POS tags in a segment and also bigrams and trigrams of the POS tags for each segment (set2). In addition to these features we consider POS bigram surrounding the segment boundary (set3).

- **Reordering Features**: The lexicalized reordering model (Koehn et al., 2007) of phrase-based translation system determines the orientation of phrases with respect to the previous phrase, monotone (M),

Set1: Word, Position, Length	“engineers”, 9, 5
Set2: + POS tags	[NNS],[CC-NNS],[NN-CC-NNS]
Set3: + Cross POS tag	[NNS-IN]
Set4: + Reordering	0.8904, 0.6

Table 1: Feature sets and an example (for segment “from our scientist and engineers” in the English sentence in Figure 1).

swap (S) and discontinuous (D). We expect the segments to be monotonically ordered. For each segment, we define two reordering features corresponding to the monotone feature orientation of the first and last phrases of the segment[4]. To compute the feature values we use lexicalized reordering model of Moses (Koehn et al., 2007) for monotone orientation of both left-to-right and right-to-left (corresponding to the first and last phrases of the segment). Adding reordering features to the previous features creates the last set of features (set4).

Table 1 shows an example for POS-based and reordering-based features defined on the second segment (“from our scientist and engineers”) of the stream “we desperately need great communication from ...” (see Figure 1). To simplify the comparison, we consider each set-i contains the features of the previous sets. For example set2 includes the POS tags and features in set1.

3 Integrating Segmentation and Decoding

In sentence segmentation approaches, the input stream is segmented and for each recognized segment the machine translation decoder is called to translate the segment individually. In this approach the decoder treat each segment as an independent input, while we are translating the input stream. We integrate the segmentation model and decoder. This approach can be also considered as a stream decoding method which the decoder exploit other resources beyond just decoding cues.

Hiero models encode the translation correspondences in *hierarchical* phrases, unlike the phrase-based models that use contiguous translation phrases. The notion of hierarchy allows the Hiero models to capture long-distance reordering between source and target languages unlike phrase-based models. Additionally they also model discontiguous translations, e.g. translating the English word *not* as *ne* ___ *pas* in French (with an appropriate verb form inserted between *ne* and *pas*). These properties make Hiero models more appropriate for some language pairs than phrase-based models (Marcu and Wong, 2002; Och and Ney, 2002, 2004).

Hiero uses a lexicalized synchronous context-free grammar (SCFG) extracted from word and phrase alignments of a bitext. Typically, Hiero uses a CKY-style decoding algorithm with time complexity $O(n^3)$ where the source input has n words.

Previous translation services proposed for real-time translation environments, are mainly phrase-based (Fügen et al., 2007; Sankaran et al., 2010; Bangalore et al., 2012; Yarmohammadi et al., 2013; Oda et al., 2014). Since a phrase-based decoder generates translations in a left-to-right manner, it is more suited than the CKY based decoding which requires the entire input sentence before generating the translation.

We propose to use left-to-right hierarchical phrase-based translation in our simultaneous translation framework. It has been shown that left-to-right hierarchical (LR-Hiero) decoder can translate using Hiero translation model much faster than CKY Hiero decoder (Siahbani et al., 2013). In addition, it generates the translation in left-to-right manner. These properties make it a suitable decoder for simultaneous translation (Siahbani et al., 2014). We augment LR-Hiero decoder to incrementally translate the input and integrate it with our segmentation model. We briefly review the LR-Hiero decoder and then explain our incremental

[4] We consider the longest phrase which is available in the phrase-table.

Algorithm 1: Simultaneous Translation

```
Input stream: $\mathbf{f} = f_0 f_1 \ldots$
$buffer = []$
$h_0 = (\langle s \rangle, null, null, 0)$                                        (Initial history is $\langle s \rangle$)
$history = \{h_0\}$
while $f_i \neq \langle /s \rangle$ do
    if $Segmenter(buffer, f_i) == True$ then
        $trans = Decoder(buffer, history)$
        print $trans$
        $buffer = [f_i]$
        Update $history$
    else
        Add $f_i$ to $buffer$                          (Add the current word to the end of buffer)
$trans = Decoder(buffer, history)$                                  (Translate the last segment)
print $trans$
```

version of the decoder.

3.1 LR-Hiero Decoder

LR-Hiero uses a constrained lexicalized SCFG usually called GNF grammar: $X \rightarrow \langle \gamma, \bar{b}\,\beta \rangle$, where X is a non-terminal, γ is a string of non-terminal and terminal symbols, $\bar{b}$ is a string of terminal symbols and β is a possibly empty sequence of non-terminals. Using GNF rules ensures that in derivations the target side is always generated from left to right. The rules are obtained from a word and phrase aligned bitext by replacing the smaller source-target phrase pair within a larger phrase pair with some non-terminal.

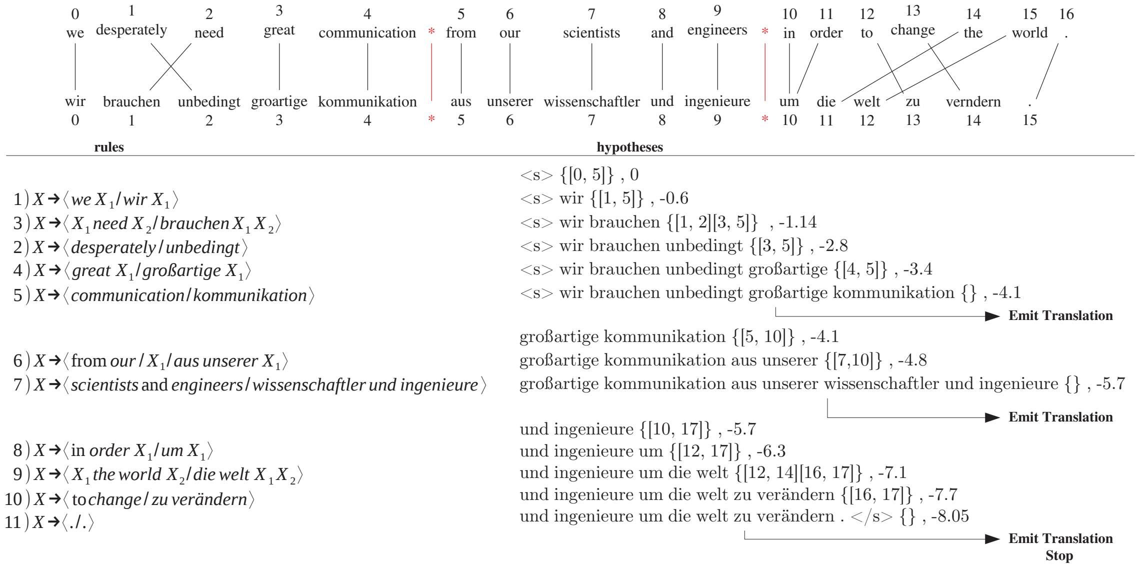
The decoding algorithm in LR-Hiero follows an Earley-style search (Earley, 1970) on the source side. The dot jumps around on the source side of the rules based on the order of nonterminals on the target side. Thus the target side derivation is strictly developed in left to right order. The search algorithm is integrated with beam search or cube pruning to find the k-best translations.

We slightly modify LR-Hiero decoder proposed by (Siahbani et al., 2013) and explain it over an example [5] (Figure 2). Each partial hypothesis h contains (h_t, h_s, h_c): a translation prefix h_t, a (LIFO-ordered) *list* h_s of uncovered spans and the hypothesis cost h_c which includes future cost and a model score computed based on feature values (using a log-linear model).

In the standard LR-Hiero decoder, translation prefix for the initial hypothesis is $\langle s \rangle$ and the initial hypothesis would be $h_0 = (\langle s \rangle, \{[0, n]\}, 0)$. The hypotheses are stored in stacks $S_0, \ldots, S_n$, where S_p contains hypotheses covering p source words, just like in stack decoding for phrase-based SMT (Koehn et al., 2003). Decoding process finishes when stack S_n has been filled.

To Expand each hypothesis we find a rule that matched the first uncovered span ($h_s[0]$). For example to expand the initial hypothesis in Figure 2, we apply rule #1 which is matched to the first uncovered span ([0,5]). The new hypothesis will be generated by appending the lexical part of target side of the rule to the translation prefix of the previous hypothesis ("wir" is appended to "$\langle s \rangle$"). The list of uncovered span, h_s, is created by removing the first uncovered spand and pushing the new uncovered spans after applying the rule. In Figure 2, after applying rule #1 we translate the first word so a new uncovered span is generated (matched to non-terminal X_1 in rule #1) and is pushed to the h_s after popping the first uncovered span from the initial hypothesis.

[5] Please refer to (Siahbani et al., 2013) for more details.



rules

1) $X \rightarrow \langle \mathit{we}\ X_1 / \mathit{wir}\ X_1 \rangle$
3) $X \rightarrow \langle X_1\ \mathit{need}\ X_2 / \mathit{brauchen}\ X_1\ X_2 \rangle$
2) $X \rightarrow \langle \mathit{desperately} / \mathit{unbedingt} \rangle$
4) $X \rightarrow \langle \mathit{great}\ X_1 / \mathit{großartige}\ X_1 \rangle$
5) $X \rightarrow \langle \mathit{communication} / \mathit{kommunikation} \rangle$
6) $X \rightarrow \langle \text{from}\ \mathit{our} / X_1 / \mathit{aus\ unserer}\ X_1 \rangle$
7) $X \rightarrow \langle \mathit{scientists}\ \text{and}\ \mathit{engineers} / \mathit{wissenschaftler\ und\ ingenieure} \rangle$
8) $X \rightarrow \langle \text{in}\ \mathit{order}\ X_1 / \mathit{um}\ X_1 \rangle$
9) $X \rightarrow \langle X_1\ \mathit{the\ world}\ X_2 / \mathit{die\ welt}\ X_1 X_2 \rangle$
10) $X \rightarrow \langle \text{to}\mathit{change} / \mathit{zu\ verändern} \rangle$
11) $X \rightarrow \langle . / . \rangle$

hypotheses

<s> {[0, 5]} , 0
<s> wir {[1, 5]} , -0.6
<s> wir brauchen {[1, 2][3, 5]} , -1.14
<s> wir brauchen unbedingt {[3, 5]} , -2.8
<s> wir brauchen unbedingt großartige {[4, 5]} , -3.4
<s> wir brauchen unbedingt großartige kommunikation {} , -4.1
großartige kommunikation {[5, 10]} , -4.1
großartige kommunikation aus unserer {[7,10]} , -4.8
großartige kommunikation aus unserer wissenschaftler und ingenieure {} , -5.7
und ingenieure {[10, 17]} , -5.7
und ingenieure um {[12, 17]} , -6.3
und ingenieure um die welt {[12, 14][16, 17]} , -7.1
und ingenieure um die welt zu verändern {[16, 17]} , -7.7
und ingenieure um die welt zu verändern . </s> {} , -8.05

Figure 2: Simultaneous translation for an English-German sentence using LR-Hiero. The word alignment is shown on the top. The segmentation points are shown by red stars. On the bottom, different steps of the decoder are shown. The left side shows the rules used in the derivation.The hypotheses column shows partial hypotheses containing the translation prefix, h_t, the ordered list of yet-to-be-covered spans, h_s and cost h_c.

3.2 Incremental Translation

In our simultaneous translation framework, we integrate LR-Hiero with the segmentation model. This framework is shown in Algorithm 1. The input is a stream of words ($\mathbf{f} = f_0 f_1 \ldots$) which is fed to the translation system word by word. In this algorithm, *buffer* always contains the sequence of yet-to-be-translated words (initially empty), and *history* keeps the previous state of the decoder. We define *history* as the set of best hypotheses generated by the decoder while translating the previous segment. *history* is initialized by a null hypothesis (containing the sentence initial marker).

For each new input word, f_i, the translation system queries the segmentation model. The content of *buffer* and f_i are passed to the *Segmenter* to determine whether the sequence of words in *buffer* is a valid segment to be translated or not. Once a segment is recognized, the segment (content of *buffer*) and *history* are passed to the decoder. The decoder translates the given source segment, and produces the translation output for that segment. After emitting the translation to the output, *buffer* is initialized with the last input word, f_i which has not been translated yet and *history* is updated with the set of best hypotheses generated by the decoder. In other words, we freeze the the state of the decoder and continue the translation when the new segment arrives.

Figure 2 illustrates the process of generating just one translation. After reading the sixth word in the input stream ("from") the segmentation model recognize a segment ("we desperately need great communication"). This segment is passed to the decoder. The decoder generates the translation and the best translation will be emitted. Then *history* will be updated by the set of best hypotheses generated by the decoder. The translation system keeps reading the input stream and after recognizing the next segment (after reading "in") the new segment along with the *history* is passed to the decoder. The decoder translated the given

Task	Sentences	Tokens
MT Train	1033491	27948039
Tune	3669	74883
Seg. model Train	3669	74883
Test	1025	22026

Table 2: Corpus statistics in number of sentences and tokens (source side).

segment while using partial hypotheses in *history* as initial hypotheses (the figure just shows one of them $\langle$groBartige kommunikation, $\{[5, 10]\}, -4.1\rangle$). The process is repeated until the end of sentence is detected.

In this approach, the translation output is updated over time by adding the translation of the next input segments and the decoder does not change the output which is already produced and emitted.

4 Experimental Results

Following the *International Workshop on Spoken Language Translation* (IWSLT) shared task, we evaluate our approach on the speech translation of TED talks for English-German.Section 4.2 describe the experimental setting. We conduct many experiments to evaluate our approach. We first evaluate different approaches to create the segmentation model and experiment on various feature sets to obtain the best segmentation model (Section 4.2). Then we use the trained segmentation model in an end-to-end simultaneous translation system (Section 4.3). We evaluate the performance of end-to-end simultaneous translation system in terms of translation quality and latency and compare it with different baselines.

4.1 System Setup

We use the parallel text provided as training data of IWSLT 2013 and about one million sentence pairs of Europarl (v7), to train the translation system. We use development set 2010 and 2012 and test set 2010 of IWSLT shared task as development set to tune the translation system (LR-Hiero) and test set of IWSLT 2013 is used as the test set to evaluate the simultaneous translation system. We use a 5-gram LM trained on the monolingual German data provided by WMT 2013 shared task using KenLM (Heafield, 2011).

In LR-Hiero, we set pop limit 500, maximum source rule length 7 and at most 2 non-terminals. The standard feature set of LR-Hiero (Siahbani et al., 2013) is used in a discriminative log-linear model. The weights in the log-linear model are tuned by minimizing BLEU loss through MERT (Och, 2003) on the dev set for each language pair. In these experiments, we use the reference transcript of the utterance for dev and test sets. LR-Hiero is trained once and used in all experiments.

We use Stanford POS-Tagger (Toutanova et al., 2003) to obtain the POS tags to extract features for the segmentation model [6].

4.2 Evaluating the Segmentation Model

In Section 2 we discussed two heuristics: translation-based and alignment-based, to provide training data for segmentation model. We conduct some experiments to compare different feature sets for these heuristics. We use Dev 2010 and 2012 and Test 2010 from IWSLT to provide the training date for the segmentation model. Table 2 shows the statistics of data used in our experiments.

To evaluate segment translation quality, we use corpus level BLEU (Papineni et al., 2002). To compute the latency model, we use *sayit*[7] script by Hal Daumê III which receives the content of the segment (in text format) and estimates the time it takes from a human to say the segment in some languages: English (US),

[6] In our experiments we use the standard POS-Tagger.

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

Labeled data Heuristic	Features	P	R	F1
Translation-based	Set1	81.38	52.56	63.87
	Set2	82.03	53.90	65.06
	Set3	**97.18**	**69.89**	**81.31**
	Set4	93.41	64.14	76.06
Alignment-based	Set1	71.78	62.88	67.04
	Set2	74.58	56.46	64.27
	Set3	**79.78**	58.39	**67.43**
	Set4	79.09	**59.62**	**67.97**

Table 3: Results of segmentation model trained on different labeled data using various feature sets.

Segmentation model	Features	BLEU	Latency	Number of segments
Translation-based	Set3	**20.86**	**0.311**	3313
Alignment-based	Set3	20.60	0.540	2648
	Set4	20.62	0.524	2654
Prosodic heuristic	-	**20.88**	0.514	2709
Fixed Segmentation	-	19.81	0.283	3580
Random Segmentation	-	19.63	0.218	3980
No Segmentation	-	21.04	6.353	1025

Table 4: Results of our simultaneous translation using different segmentation models on English-German translation task. The last row shows the offline translation (regular SMT without segmentation). Segment length is set to 6 in *Fixed Segmentation* and *Random Segmentation*.

German, French, Italian, Spanish, and Japanese. These estimates are then used to evaluate the terms t_i and t'_{i-1} in Equation 2.

For the alignment-based heuristic, we concatenate the training data of segmentation model (3669 sentence pairs) to the training data of the translation system and run GIZA++ to get the word alignment. Then the heuristic discussed in Section 2 is used to extract segments. To have fair comparison, we choose $\mu = 5$ in translation-based heuristic which provides comparable number of segments on the training data in both alignment-based and translation-based heuristics.

We train separate segmentation models using the training data created by translation-based and alignment-based heuristics and different feature sets. To compare the feature sets, we test the models on a heldout set[8] (5000 sentences randomly selected from training data of IWSLT 2013). Table 3 shows the results in terms of precision, recall and F1 measure. Feature set3 outperforms the other feature sets for the segmentation models trained on the data obtained by translation-based heuristic. Therefore we use this model in the further experiments (Section 4.3). Hence feature set3 and set4 show comparable results, for the alignment-based heuristic, in these experiments, we will use both trained models in the end-to-end simultaneous translation system (Section 4.3).

4.3 Evaluation of Simultaneous Translation

We evaluate our simultaneous translation framework on a English-German translation task. We calculate latency as the total time taken to translate the whole sentence divided by the number of segments. Latency

[8] We use the set of POS tags obtained by translation-based heuristic to create the gold reference for this experiment.

in Table 4 shows the result of taking the average over 5 different runs for 50 sentences randomly selected from the test set.

The first three rows of Table 4 compare the results of the end-to-end simultaneous translation using segmentation models trained by translation-based and alignment-based heuristics. We can see that segmentation model trained on translation-based heuristic outperforms the other segmentation models both in translation accuracy and latency.

To evaluate our simultaneous translation framework we use four baselines. We implemented a heuristic segmenter based on (Rangarajan Sridhar et al., 2013) 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 (Bangalore et al., 2012). The results of this heuristic (*prosodic*) has been shown in the forth row of Table 4. The last row in Table 4 shows the results of the regular translation strategy (with no segmentation employed). For a relatively small loss in the BLEU score we obtain a much faster incremental translation system. To evaluate the impact of segmentation model we add two more baselines in which decoder segments the input stream without using the segmentation model: (i) *Fixed Segmentation*: a segmentation with equally sized fragments; (ii) *Random Segmentation*: decoder randomly segments the input. These two baselines show comparable performance. The reduction in the BLEU score for these segmentation models shows that we need a more informative segmentation model.

We also compare our output against a state-of-the-art simultaneous neural MT approach (Gu et al., 2017), which uses a reinforcement learning style agent which is trained using a policy gradient algorithm to find segments that minimize delay and maximize the BLEU score. The agent uses a softmax policy over the segmentation outcomes (either read or write, aka segment) and trains its parameters by learning segmentations decisions based on a fully-trained non-simultaneous NMT encoder-decoder. Gu et al. (2017) use a new metric to measure the latency called *average proportion* proposed by(Cho and Esipova, 2016). Average proportion is defined as the average number of source words being used, when translating each word. The average proportion $d(X, Y)$ for a source sentence X and translation output Y is defined as $\frac{\sum_{t=1}^{|Y|} s(t)}{|X||Y|}$ where $|X|$ and $|Y|$ are the length of source and translation sentences respectively, and $s(t)$ is the number of already seen words from source sentence, when translating each word. We ran the Gu et al. (2017) approach on our English-German task (Figure 3). In order to compare the latency, we compute the average proportion for the output of our translation system which are shown in Figure 1. In this figure we have shown the results of our translation framework using different segmentation models trained for different segment lengths (μ values 3 to 8). We trained the NMT system with $\mu = 8$. Figure 3 also shows the results of offline translation for our approach and the NMT system (which results in average proportion of 1). There is a substantial loss in translation quality for simultaneous NMT (consistent with the results in Gu et al. (2017)).

5 Related Work

Early work on speech translation uses prosodic pauses detected in speech as segmentation boundaries (Fügen et al., 2007; Bangalore et al., 2012). Segmentation methods applied on the transcribed text can be divided to two categories: heuristic methods which use linguistic cues, like conjunctions, commas, etc. (Rangarajan Sridhar et al., 2013); 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 (Matusov et al., 2007); while some other methods rely on word alignment information to identifies contiguous blocks of text that do not contain alignments to words outside them (Yarmohammadi et al., 2013; Siahbani et al., 2014). 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. Various incremental decoding approaches have been proposed for phrase-based (Kolss et al., 2008; Sankaran et al., 2010), hierarchical phrase-based (Siahbani et al., 2014; Finch et al.,

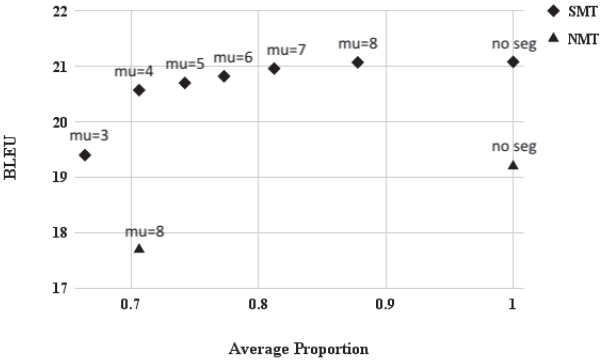


Figure 3: Comparing translation quality versus average proportion (latency) for our approach (SMT) and the simultaneous neural MT approach (NMT). We show the comparison of different μ values from 3 to 8 for the SMT system. The NMT system was trained with a μ value of 8. The offline translation systems with no segmentation (*no seg*) (for both SMT and NMT) have average proportion 1.

2015), and syntax-based (Oda et al., 2015) translation systems. In most incremental decoding algorithms, the decoder waits for more input and commit the translation when the current utterance is enough to generate a fluent translation. Oda et al. (2015) propose a method to predict the future syntactic constituents and use it in generating complete parse trees which helps to find a good point to commit the translation. Some researches have been focused on language pairs with divergent word order. Grissom II et al. (2014) predict sentence-final verbs using reinforcement learning which greatly affects the delay. He et al. (2015) design syntactic transformations to rewrite 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 (Mieno et al., 2015). 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.

Neural machine translation has also been extended to perform simultaneous translation. Cho and Esipova (2016) proposed a manually defined heuristic waiting criteria to define an optimal segmentation point. A trainable agent which considers both quality and delay during segmentation first introduced by Satija and Pineau (2016). This work was extended by Gu et al. (2017) who designed a segmentation agent trained to incrementally translate using a policy gradient over a linear combination of translation quality (based on a sentence level BLEU score) and latency (calculated as minimizing delay). They showed that such an approach can learn a trade-off between quality and delay. However, in our comparison with their results our system provides a higher BLEU score while providing a comparable latency (see Figure 3).

6 Summary and Conclusion

This work combines segmentation with incremental decoding. The segmentation model is trained to minimize latency of producing translations as it reads from the input stream and maximize translation quality as measured by the BLEU score. Our framework is able to produce fast yet accurate translation in a simultaneous translation setting. Our experiments show that we obtain higher quality translations with near similar latency compared to a simultaneous neural machine translation system.

References

Bangalore, S., Rangarajan Sridhar, V. K., Kolan, P., Golipour, L., and Jimenez, A. (2012). Real-time incremental speech-to-speech translation of dialogs. In *Proc. of NAACL HLT 2012*, pages 437–445.

Cho, K. and Esipova, M. (2016). Can neural machine translation do simultaneous translation? *CoRR*, abs/1606.02012.

Earley, J. (1970). An efficient context-free parsing algorithm. *Commun. ACM*, 13(2):94–102.

Finch, A., Wang, X., Utiyama, M., and Sumita, E. (2015). Hierarchical phrase-based stream decoding. In *Proc. of EMNLP*, Lisbon, Portugal.

Fügen, C., Waibel, A., and Kolss, M. (2007). Simultaneous translation of lectures and speeches. *Machine Translation*, 21(4):209–252.

Fujita, T., Neubig, G., Sakti, S., Toda, T., and Nakamura, S. (2013). Simple, lexicalized choice of translation timing for simultaneous speech translation. In *INTERSPEECH*, pages 3487–3491.

Grissom II, A. C., Boyd-Graber, J., He, H., Morgan, J., and Daume III, H. (2014). Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *EMNLP*, pages 1342–1352.

Gu, J., Neubig, G., Cho, K., and Li, V. O. (2017). Learning to translate in real-time with neural machine translation. In *15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Valencia, Spain.

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

Heafield, K. (2011). KenLM: Faster and smaller language model queries. In *In Proc. of the Sixth Workshop on Statistical Machine Translation*.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Koehn, P., Och, F. J., and Marcu, D. (2003). Statistical phrase-based translation. In *Proc. of NAACL*.

Kolss, M., Vogel, S., and Waibel, A. (2008). Stream decoding for simultaneous spoken language translation. In *INTERSPEECH*, pages 2735–2738.

Lin, D. and Och, F. (2004). Orange: a method for evaluating automatic evaluation metrics for machine translation. In *COLING 2004*, pages 501–507.

Marcu, D. and Wong, W. (2002). A phrase-based, joint probability model for statistical machine translation. In *Proc. of EMNLP-2002*, pages 133–139.

Matusov, E., Hillard, D., Magimai-doss, M., Hakkani-tur, D., Ostendorf, M., and Ney, H. (2007). Improving speech translation with automatic boundary prediction. In *In Proc. Interspeech*, pages 2449–2452.

Mieno, T., Neubig, G., Sakti, S., Toda, T., and Nakamura, S. (2015). Speed or accuracy? a study in evaluation of simultaneous speech translation. In *INTERSPEECH*.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proc. of ACL*.

Och, F. J. and Ney, H. (2002). Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of ACL*.

Och, F. J. and Ney, H. (2004). The alignment template approach to statistical machine translation. *Computational Linguistics*, 30.

Oda, Y., Neubig, G., Sakti, S., Toda, T., and Nakamura, S. (2014). Optimizing segmentation strategies for simultaneous speech translation. In *The 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, Baltimore, USA.

Oda, Y., Neubig, G., Sakti, S., Toda, T., and Nakamura, S. (2015). Syntax-based simultaneous translation through prediction of unseen syntactic constituents. In *ACL*.

Papineni, K., Roukos, S., Ward, T., and jing Zhu, W. (2002). Bleu: a method for automatic evaluation of machine translation. pages 311–318.

Rangarajan Sridhar, V. K., Chen, J., Bangalore, S., Ljolje, A., and Chengalvarayan, R. (2013). Segmentation strategies for streaming speech translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 230–238, Atlanta, Georgia. Association for Computational Linguistics.

Sankaran, B., Grewal, A., and Sarkar, A. (2010). Incremental decoding for phrase-based statistical machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and Metrics-MATR*, WMT.

Satija, H. and Pineau, J. (2016). Simultaneous machine translation using deep reinforcement learning. *Abstraction in Reinforcement Learning Workshop, ICML 2016*, (33).

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

Siahbani, M., Mehdizadeh Seraj, R., Sankaran, B., and Sarkar, A. (2014). Incremental translation using a hierarchical phrase-based translation system. In *In Proceedings of the 2014 IEEE Spoken Language Technology Workshop (SLT 2014)*, Nevada, USA.

Siahbani, M., Sankaran, B., and Sarkar, A. (2013). Efficient left-to-right hierarchical phrase-based translation with improved reordering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Seattle, USA.

Toutanova, K., Klein, D., Manning, C., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL 2003*, pages 252–259.

Watanabe, T., Tsukada, H., and Isozaki, H. (2006). Left-to-right target generation for hierarchical phrase-based translation. In *Proc. of ACL*.

Yarmohammadi, M., Sridhar, V. K. R., Bangalore, S., and Sankaran, B. (2013). Incremental segmentation and decoding strategies for simultaneous translation. In *Proc. of IJCNLP-2013*.