Syntax-based Simultaneous Translation through Prediction of Unseen Syntactic Constituents

Yusuke Oda **Graham Neubig** **Sakriani Sakti** **Tomoki Toda** **Satoshi Nakamura**
Graduate School of Information Science
Nara Institute of Science and Technology
Takayamacho, Ikoma, Nara 630-0192, Japan
{oda.yusuke.on9, neubig, ssakti, tomoki, s-nakamura}@is.naist.jp

Abstract

Simultaneous translation is a method to reduce the latency of communication through machine translation (MT) by dividing the input into short segments before performing translation. However, short segments pose problems for syntax-based translation methods, as it is difficult to generate accurate parse trees for sub-sentential segments. In this paper, we perform the first experiments applying syntax-based SMT to simultaneous translation, and propose two methods to prevent degradations in accuracy: a method to predict unseen syntactic constituents that help generate complete parse trees, and a method that waits for more input when the current utterance is not enough to generate a fluent translation. Experiments on English-Japanese translation show that the proposed methods allow for improvements in accuracy, particularly with regards to word order of the target sentences.

1 Introduction

Speech translation is an application of machine translation (MT) that converts utterances from the speaker's language into the listener's language. One of the most identifying features of speech translation is the fact that it must be performed in real time while the speaker is speaking, and thus it is necessary to split a constant stream of words into translatable segments before starting the translation process. Traditionally, speech translation assumes that each segment corresponds to a sentence, and thus performs sentence boundary detection before translation (Matusov et al., 2006). However, full sentences can be long, particularly in formal speech such as lectures, and if translation does not start until explicit ends of sentences, listeners may be forced to wait a considerable time until receiving the result of translation. For example, when the speaker continues to talk for 10 seconds, listeners must wait at least 10 seconds to obtain the result of translation. This is the major factor limiting simultaneity in traditional speech translation systems.

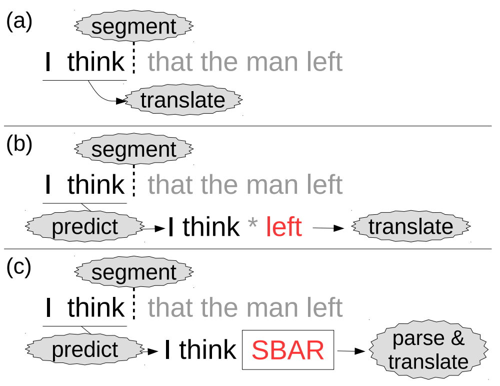


Figure 1: Simultaneous translation where the source sentence is segmented after "I think" and translated according to (a) the standard method, (b) Grissom II et al. (2014)'s method of final verb prediction, and (c) our method of predicting syntactic constituents.

Simultaneous translation (Section 2) avoids this problem by starting to translate before observing the whole sentence, as shown in Figure 1 (a). However, as translation starts before the whole sentence is observed, translation units are often not syntactically or semantically complete, and the performance may suffer accordingly. The deleterious effect of this missing information is less worrying in largely monotonic language pairs (e.g. English-French), but cannot be discounted in syntactically distant language pairs (e.g. English-Japanese) that often require long-distance reordering beyond translation units.

One way to avoid this problem of missing information is to explicitly predict information needed

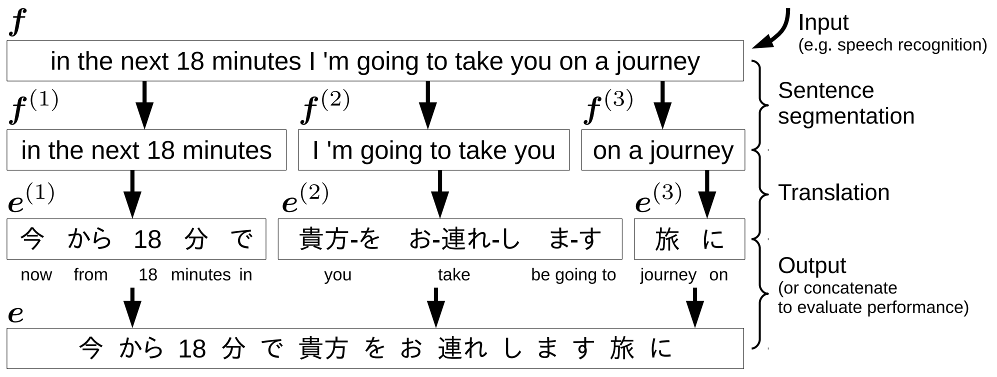


Figure 2: Process of English-Japanese simultaneous translation with sentence segmentation.

to translate the content accurately. An ambitious first step in this direction was recently proposed by Grissom II et al. (2014), who describe a method that predicts sentence-final verbs using reinforcement learning (e.g. Figure 1 (b)). This approach has the potential to greatly decrease the delay in translation from verb-final languages to verb-initial languages (such as German-English), but is also limited to only this particular case.

In this paper, we propose a more general method that focuses on a different variety of information: *unseen syntactic constituents*. This method is motivated by our desire to apply translation models that use source-side parsing, such as tree-to-string (T2S) translation (Huang et al., 2006) or syntactic pre-ordering (Xia and McCord, 2004), which have been shown to greatly improve translation accuracy over syntactically divergent language pairs. However, conventional methods for parsing are not directly applicable to the partial sentences that arise in simultaneous MT. The reason for this, as explained in detail in Section 3, is that parsing methods generally assume that they are given input that forms a complete syntactic phrase. Looking at the example in Figure 1, after the speaker has spoken the words "I think" we have a partial sentence that will only be complete once we observe the following SBAR. Our method attempts to predict exactly this information, as shown in Figure 1 (c), guessing the remaining syntactic constituents that will allow us to acquire a proper parse tree.

Specifically the method consists of two parts: First, we propose a method that trains a statistical model to predict future syntactic constituents based on features of the input segment (Section 4). Second, we demonstrate how to apply this syntactic prediction to MT, including the proposal of a heuristic method that examines whether a future constituent has the potential to cause a reordering problem during translation, and wait for more input in these cases (Section 5).

Based on the proposed method, we perform experiments in simultaneous translation of English-Japanese talks (Section 6). As this is the first work applying T2S translation to simultaneous MT, we first compare T2S to more traditional phrase-based techniques. We find that T2S translation is effective with longer segments, but drops off quickly with shorter segments, justifying the need for techniques to handle translation when full context is not available. We then compare the proposed method of predicting syntactic constituents, and find that it improves translation results, particularly with respect to word ordering in the output sentences.

2 Simultaneous Translation

In simultaneous translation, we assume that we are given an incoming stream of words f, which we are expected to translate. As the f is long, we would like to begin translating before we reach the end of the stream. Previous methods to do so can generally be categorized into *incremental decoding* methods, and *sentence segmentation* methods.

In incremental decoding, each incoming word is fed into the decoder one-by-one, and the decoder updates the search graph with the new words and decides whether it should begin translation. Incremental decoding methods have been proposed for phrase-based (Sankaran et al., 2010; Yarmohammadi et al., 2013; Finch et al., 2014) and hierarchical phrase-based (Siahbani et al., 2014) SMT models.[1] Incremental decoding has the advantage of using information about the decoding graph in the choice of translation timing, but also requires significant changes to the internal workings of the decoder, precluding the use of standard decoding tools or techniques.

Sentence segmentation methods (Figure 2) provide a simpler alternative by first dividing $\boldsymbol{f}$ into subsequences of 1 or more words $[\boldsymbol{f}^{(1)}, \ldots, \boldsymbol{f}^{(N)}]$. These segments are then translated with a traditional decoder into output sequences $[\boldsymbol{e}^{(1)}, \ldots, \boldsymbol{e}^{(N)}]$, which each are output as soon as translation finishes. Many methods have been proposed to perform segmentation, including the use of prosodic boundaries (Fügen et al., 2007; Bangalore et al., 2012), predicting punctuation marks (Rangarajan Sridhar et al., 2013), reordering probabilities of phrases (Fujita et al., 2013), or models to explicitly optimize translation accuracy (Oda et al., 2014). Previous work often assumes that $\boldsymbol{f}$ is a single sentence, and focus on sub-sentential segmentation, an approach we follow in this work.

Sentence segmentation methods have the obvious advantage of allowing for translation as soon as a segment is decided. However, the use of the shorter segments also makes it necessary to translate while part of the utterance is still unknown. As a result, segmenting sentences more aggressively often results in a decrease translation accuracy. This is a problem in phrase-based MT, the framework used in the majority of previous research on simultaneous translation. However, it is an even larger problem when performing translation that relies on parsing the input sentence. We describe the problems caused by parsing a segment $\boldsymbol{f}^{(n)}$, and solutions, in the following section.

3 Parsing Incomplete Sentences

3.1 Difficulties in Incomplete Parsing

In standard phrase structure parsing, the parser assumes that each input string is a complete sentence, or at least a complete phrase. For example, Figure 3 (a) shows the phrase structure of the complete sentence "this is a pen." However, in the case of simultaneous translation, each translation unit

[1]There is also one previous rule-based system that uses syntax in incremental translation, but it is language specific and limited domain (Ryu et al., 2006), and thus difficult to compare with our SMT-based system. It also does not predict unseen constituents, relying only on the observed segment.

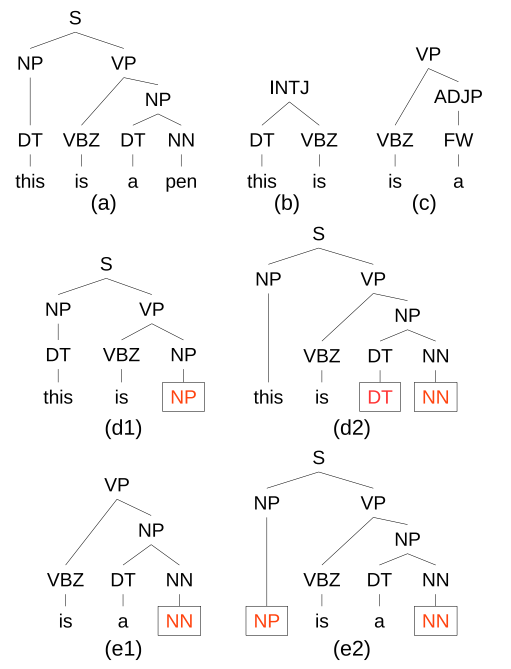


Figure 3: Phrase structures with surrounding syntactic constituents.

is not necessarily segmented in a way that guarantees that the translation unit is a complete sentence, so each translation unit should be treated not as a whole, but as a part of a spoken sentence. As a result, the parser input may be an incomplete sequence of words (e.g. "this is," "is a"), and a standard parser will generate an incorrect parse as shown in Figures 3(b) and 3(c).

The proposed method solves this problem by supplementing unseen syntactic constituents before and after the translation unit. For example, considering parse trees for the complete sentence in Figure 3(a), we see that a noun phrase (NP) can be placed after the translation unit "this is." If we append the syntactic constituent $\boxed{\text{NP}}$ as a "black box" before parsing, we can create a syntactically desirable parse tree as shown in Figure 3(d1) We also can construct another tree as shown in Figure 3(d2) by appending two constituents $\boxed{\text{DT}}$ and $\boxed{\text{NN}}$. For the other example "is a," we can create the parse tree in Figure 3(e1) by appending $\boxed{\text{NP}}$ before the unit and $\boxed{\text{NN}}$ after the unit, or can create the tree in Figure 3(e2) by appending only $\boxed{\text{NN}}$ after the unit.

3.2 Formulation of Incomplete Parsing

A typical model for phrase structure parsing is the probabilistic context-free grammar (PCFG). Parsing is performed by finding the parse tree T that maximizes the PCFG probability given a sequence of words $\boldsymbol{w} \equiv [w_1, w_2, \cdots, w_n]$ as shown by Eq. (2):

$$
\begin{aligned}
T^* &\equiv \arg\max_T \Pr(T|\boldsymbol{w}) \quad (1)\\
&\simeq \arg\max_T [\\
&\qquad \sum_{(X\to[Y,\cdots])\in T} \log \Pr(X \to [Y, \cdots]) + \\
&\qquad \sum_{(X\to w_i)\in T} \log \Pr(X \to w_i) \,], \quad (2)
\end{aligned}
$$

where $\Pr(X \to [Y, \cdots])$ represents the generative probabilities of the sequence of constituents $[Y, \cdots]$ given a parent constituent X, and $\Pr(X \to w_i)$ represents the generative probabilities of each word w_i $(1 \le i \le n)$ given a parent constituent X.

To consider parsing of incomplete sentences with appended syntactic constituents, We define $\boldsymbol{L} \equiv [L_{|\boldsymbol{L}|}, \cdots, L_2, L_1]$ as the sequence of preceding syntactic constituents of the translation unit and $\boldsymbol{R} \equiv [R_1, R_2, \cdots, R_{|\boldsymbol{R}|}]$ as the sequence of following syntactic constituents of the translation unit. For the example Figure 3(d1), we assume that $\boldsymbol{L} = [\,]$ and $\boldsymbol{R} = [\,\boxed{\text{NP}}\,]$.

We assume that both sequences of syntactic constituents $\boldsymbol{L}$ and $\boldsymbol{R}$ are predicted based on the sequence of words $\boldsymbol{w}$ before the main parsing step. Thus, the whole process of parsing incomplete sentences can be described as the combination of predicting both sequences of syntactic constituents represented by Eq. (3) and (4) and parsing with predicted syntactic constituents represented by Eq. (5):

$$
\begin{aligned}
\boldsymbol{L}^* &\equiv \arg\max_{\boldsymbol{L}} \Pr(\boldsymbol{L}|\boldsymbol{w}), \quad (3)\\
\boldsymbol{R}^* &\equiv \arg\max_{\boldsymbol{R}} \Pr(\boldsymbol{R}|\boldsymbol{w}), \quad (4)\\
T^* &\equiv \arg\max_T \Pr(T|\boldsymbol{L}^*, \boldsymbol{w}, \boldsymbol{R}^*). \quad (5)
\end{aligned}
$$

Algorithmically, parsing with predicted syntactic constituents can be achieved by simply treating each syntactic constituent as another word in the input sequence and using a standard parsing algorithm such as the CKY algorithm. In this process, the only difference between syntactic constituents and normal words is the probability, which we define as follows:

$$
\Pr(X \to \boxed{Y}) \equiv \begin{cases} 1, & \text{if } Y = X \\ 0, & \text{otherwise.} \end{cases} \quad (6)
$$

It should be noted that here $\boldsymbol{L}$ refers to syntactic constituents that have already been seen in the past. Thus, it is theoretically possible to store past parse trees as history and generate $\boldsymbol{L}$ based on this history, or condition Eq. 3 based on this information. However, deciding which part of trees to use as $\boldsymbol{L}$ is not trivial, and applying this approach requires that we predict $\boldsymbol{L}$ and $\boldsymbol{R}$ using different methods. Thus, in this study, we use the same method to predict both sequences of constituents for simplicity.

In the next section, we describe the actual method used to create a predictive model for these strings of syntactic constituents.

4 Predicting Syntactic Constituents

In order to define which syntactic constituents should be predicted by our model, we assume that each final parse tree generated by $\boldsymbol{w}$, $\boldsymbol{L}$ and $\boldsymbol{R}$ must satisfy the following conditions:

1. The parse tree generated by $\boldsymbol{w}$, $\boldsymbol{L}$ and $\boldsymbol{R}$ must be "complete." Defining this formally, this means that the root node of the parse tree for the segment must correspond to a node in the parse tree for the original complete sentence.
2. Each parse tree contains only $\boldsymbol{L}$, $\boldsymbol{w}$ and $\boldsymbol{R}$ as terminal symbols.
3. The number of nodes is the minimum necessary to satisfy these conditions.

As shown in the Figure 3, there is ambiguity regarding syntactic constituents to be predicted (e.g. we can choose either $[\,\boxed{\text{NP}}\,]$ or $[\,\boxed{\text{DT}}, \boxed{\text{NN}}\,]$ as $\boldsymbol{R}$ for $\boldsymbol{w} = [$ "this", "is" $]$). These conditions avoid ambiguity of which syntactic constituents should predicted for partial sentences in the training data. Looking at the example, Figures 3(d1) and 3(e1) satisfy these conditions, but 3(d2) and 3(e2) do not.

Figure 4 shows the statistics of the lengths of $\boldsymbol{L}$ and $\boldsymbol{R}$ sequences extracted according to these criteria for all substrings of the WSJ datasets 2 to 23 of the Penn Treebank (Marcus et al., 1993), a standard training set for English syntactic parsers. From the figure we can see that lengths of up to 2 constituents cover the majority of cases for both $\boldsymbol{L}$ and $\boldsymbol{R}$, but a significant number of cases require longer strings. Thus methods that predict a fixed number of constituents are not appropriate here. In Algorithm 1, we show the method we propose to

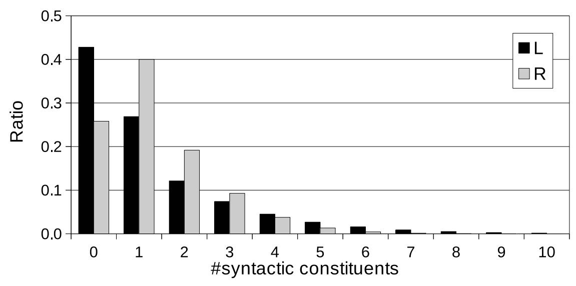


Figure 4: Statistics of numbers of syntactic constituents to be predicted.

Algorithm 1 Prediction algorithm for following constituents $\boldsymbol{R}^*$

```
$T' \leftarrow \arg\max_T \Pr(T|\boldsymbol{w})$
$\boldsymbol{R}^* \leftarrow [\,]$
**loop**
    $R^+ \leftarrow \arg\max_R \Pr(R|T', \boldsymbol{R}^*)$
    **if** $R^+ = \text{nil}$ **then**
        **return** $\boldsymbol{R}^*$
    **end if**
    $\boldsymbol{R}^* \leftarrow \boldsymbol{R}^* +\!\!+ [R^+]$
**end loop**
```

predict $\boldsymbol{R}$ for constituent sequences of an arbitrary length. Here $+\!\!+$ represents the concatenation of two sequences.

First, our method forcibly parses the input sequence $\boldsymbol{w}$ and retrieves a potentially incorrect parse tree T', which is used to calculate features for the prediction model. The next syntactic constituent R^+ is then predicted using features extracted from $\boldsymbol{w}$, T', and the predicted sequence history $\boldsymbol{R}^*$. This prediction is repeated recurrently until the end-of-sentence symbol ("nil" in Algorithm 1) is predicted as the next symbol.

In this study, we use a multi-label classifier based on linear SVMs (Fan et al., 2008) to predict new syntactic constituents with features shown in Table 1. We treat the input sequence $\boldsymbol{w}$ and predicted syntactic constituents $\boldsymbol{R}^*$ as a concatenated sequence $\boldsymbol{w} +\!\!+ \boldsymbol{R}^*$. For example, if we have $\boldsymbol{w} = [\text{ this, is, a }]$ and $\boldsymbol{R}^* = [\ \boxed{\text{NN}}\ ]$, then the word features "3 rightmost 1-grams" will take the values "is," "a," and $\boxed{\text{NN}}$. Tags of semi-terminal nodes in T' are used as part-of-speech (POS) tags for corresponding words and the POS of each predicted syntactic constituent is simply its tag. "nil" is used when some information is not available. For example, if we have $\boldsymbol{w} = [\text{ this, is }]$ and $\boldsymbol{R}^* = [\,]$ then "3 rightmost 1-grams" will take the values "nil," "this," and "is." Algorithm 1 and Table 1 shows the method used to predict $\boldsymbol{R}^*$ but $\boldsymbol{L}^*$ can be predicted by performing the prediction process in the reverse order.

5 Tree-to-string SMT with Syntactic Constituents

Once we have created a tree from the sequence $\boldsymbol{L}^* +\!\!+ \boldsymbol{w} +\!\!+ \boldsymbol{R}^*$ by performing PCFG parsing with predicted syntactic constituents according to Eqs. (2), (5), and (6), the next step is to use this tree in translation. In this section, we focus specifically on T2S translation, which we use in our experiments, but it is likely that similar methods are applicable to other uses of source-side syntax such as pre-ordering as well.

Table 1: Features used in predicting syntactic constituents.

Type	Feature
Words	3 leftmost 1,2-grams in $\boldsymbol{w} +\!\!+ \boldsymbol{R}^*$
	3 rightmost 1,2-grams in $\boldsymbol{w} +\!\!+ \boldsymbol{R}^*$
	Left/rightmost pair in $\boldsymbol{w} +\!\!+ \boldsymbol{R}^*$
POS	Same as "Words"
Parse	Tag of the root node
	Tags of children of the root node
	Pairs of root and children nodes
Length	$\lvert\boldsymbol{w}\rvert$
	$\lvert\boldsymbol{R}^*\rvert$

It should be noted that using these trees in T2S translation models is not trivial because each estimated syntactic constituent should be treated as an aggregated entity representing all possibilities of subtrees rooted in such a constituent. Specifically, there are two problems: the possibility of *reordering* an as-of-yet unseen syntactic constituent into the middle of the translated sentence, and the calculation of *language model probabilities* considering syntactic constituent tags.

With regards to the first problem of reordering, consider the example of English-Japanese translation in Figure 5(b), where a syntactic constituent $\boxed{\text{PP}}$ is placed at the end of the English sequence ($\boldsymbol{R}^*$), but the corresponding entity in the Japanese translation result should be placed in the middle of the sentence. In this case, if we attempt to translate immediately, we will have to omit the as-of-yet unknown $\boxed{\text{PP}}$ from our translation and translate it later, resulting in an unnatural word ordering in the

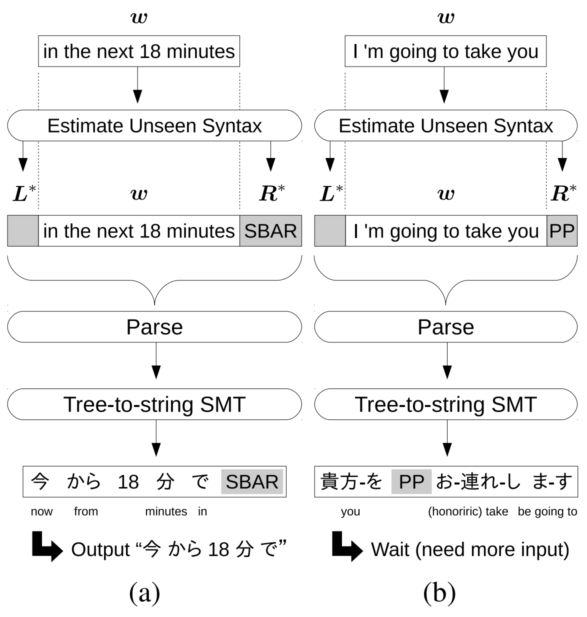


Figure 5: Waiting for the next translation unit.

target sentence.[2]

Thus, if any of the syntactic constituents in $\boldsymbol{R}$ are placed anywhere other than the end of the translation result, we can assume that this is a hint that the current segmentation boundary is not appropriate. Based on this intuition, we propose a heuristic method that ignores segmentation boundaries that result in a translation of this type, and instead wait for the next translation unit, helping to avoid problems due to inappropriate segmentation boundaries. Algorithm 2 formally describes this *waiting* method.

The second problem of language model probabilities arises because we are attempting to generate a string of words, some of which are not actual words but tags representing syntactic constituents. Creating a language model that contains probabilities for these tags in the appropriate places is not trivial, so for simplicity, we simply assume that every syntactic constituent tag is an unknown word, and that the output of translation consists of both translated normal words and non-translated tags as shown in Figure 5. We relegate a more complete handling of these tags to future work.

[2]It is also potentially possible to create a predictive model for the actual content of the $\boxed{\text{PP}}$ as done for sentence-final verbs by Grissom II et al. (2014), but the space of potential prepositional phrases is huge, and we leave this non-trivial task for future work.

Algorithm 2 Waiting algorithm for T2S SMT

```
$\boldsymbol{w} \leftarrow [\,]$
loop
    $\boldsymbol{w} \leftarrow \boldsymbol{w} +\!\!+ \mathrm{NextSegment}()$
    $\boldsymbol{L}^* \leftarrow \arg\max_{\boldsymbol{L}} \Pr(\boldsymbol{L}|\boldsymbol{w})$
    $\boldsymbol{R}^* \leftarrow \arg\max_{\boldsymbol{R}} \Pr(\boldsymbol{R}|\boldsymbol{w})$
    $T^* \leftarrow \arg\max_{T} \Pr(T|\boldsymbol{L}^*, \boldsymbol{w}, \boldsymbol{R}^*)$
    $\boldsymbol{e}^* \leftarrow \arg\max_{\boldsymbol{e}} \Pr(\boldsymbol{e}|T^*)$
    if elements of $\boldsymbol{R}^*$ are rightmost in $\boldsymbol{e}^*$ then
        $\mathrm{Output}(\boldsymbol{e}^*)$
        $\boldsymbol{w} \leftarrow [\,]$
    end if
end loop
```

6 Experiments

6.1 Experiment Settings

We perform 2 types of experiments to evaluate the effectiveness of the proposed methods.

6.1.1 Predicting Syntactic Constituents

In the first experiment, we evaluate prediction accuracies of unseen syntactic constituents $\boldsymbol{L}$ and $\boldsymbol{R}$. To do so, we train a predictive model as described in Section 4 using an English treebank and evaluate its performance. To create training and testing data, we extract all substrings $\boldsymbol{w}$ s.t. $|\boldsymbol{w}| \geq 2$ in the Penn Treebank and calculate the corresponding syntactic constituents $\boldsymbol{L}$ and $\boldsymbol{R}$ by according to the original trees and substring $\boldsymbol{w}$. We use the 90% of the extracted data for training a classifier and the remaining 10% for testing estimation recall, precision and F-measure. We use the Ckylark parser(Oda et al., 2015) to generate T' from $\boldsymbol{w}$.

6.1.2 Simultaneous Translation

Next, we evaluate the performance of T2S simultaneous translation adopting the two proposed methods. We use data of TED talks from the English-Japanese section of WIT3 (Cettolo et al., 2012), and also append dictionary entries and examples in Eijiro[3] to the training data to increase the vocabulary of the translation model. The total number of sentences/entries is 2.49M (WIT3, Eijiro), 998 (WIT3), and 468 (WIT3) sentences for training, development, and testing respectively.

We use the Stanford Tokenizer[4] for English tokenization, KyTea (Neubig et al., 2011) for

[3]http://eijiro.jp/

[4]http://nlp.stanford.edu/software/tokenizer.shtml

Japanese tokenization, GIZA++ (Och and Ney, 2003) to construct word alignment, and KenLM (Heafield et al., 2013) to generate a 5-gram target language model. We use the Ckylark parser, which we modified to implement the parsing method of Section 3.2, to generate T^* from $\boldsymbol{L}^*$, $\boldsymbol{w}$ and $\boldsymbol{R}^*$.

We use Travatar (Neubig, 2013) to train the T2S translation model used in the proposed method, and also Moses (Koehn et al., 2007) to train phrase-based translation models that serve as a baseline. Each translation model is tuned using MERT (Och, 2003) to maximize BLEU (Papineni et al., 2002). We evaluate translation accuracies by BLEU and also RIBES (Isozaki et al., 2010), a reordering-focused metric which has achieved high correlation with human evaluation on English-Japanese translation tasks.

We perform tests using two different sentence segmentation methods. The first is n-words segmentation (Rangarajan Sridhar et al., 2013), a simple heuristic that simply segments the input every n words. This method disregards syntactic and semantic units in the original sentence, allowing us to evaluate the robustness of translation against poor segmentation boundaries. The second method is the state-of-the-art segmentation strategy proposed by Oda et al. (2014), which finds segmentation boundaries that optimize the accuracy of the translation output. We use BLEU+1 (Lin and Och, 2004) as the objective of this segmentation strategy.

We evaluate the following baseline and proposed methods:

PBMT is a baseline using phrase-based SMT.

T2S uses T2S SMT with parse trees generated from only $\boldsymbol{w}$.

T2S-Tag further predicts unseen syntactic constituents according to Section 4. Before evaluation, all constituent tags are simply deleted from the output.

T2S-Wait uses *T2S-Tag* and adds the waiting strategy described in Section 5.

We also show *PBMT-Sent* and *T2S-Sent* which are full sentence-based *PBMT* and *T2S* systems.

6.2 Results

6.2.1 Predicting Syntactic Constituents

Table 2 shows the recall, precision, and F-measure of the estimated $\boldsymbol{L}$ and $\boldsymbol{R}$ sequences. The table shows results of two evaluation settings, where the order of generated constituents is considered or not.

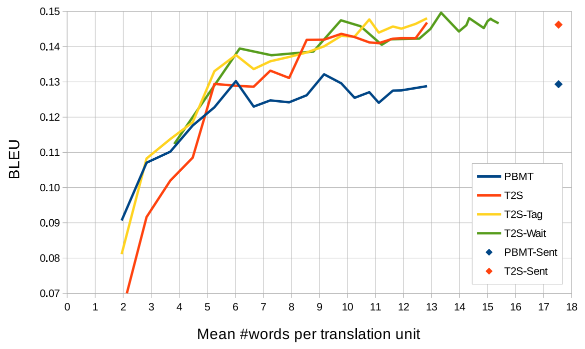
Table 2: Performance of syntactic constituent prediction.

Target		P %	R %	F %
$\boldsymbol{L}$	(ordered)	31.93	7.27	11.85
	(unordered)	51.21	11.66	19.00
$\boldsymbol{R}$	(ordered)	51.12	33.78	40.68
	(unordered)	52.77	34.87	42.00

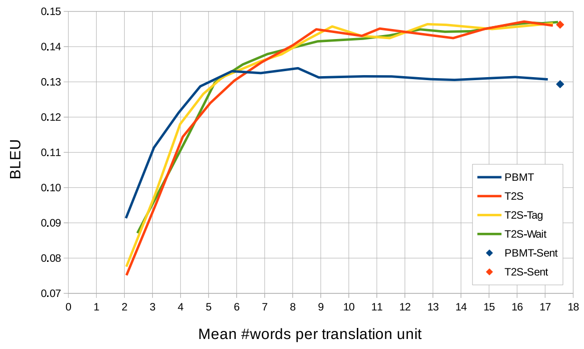
We can see that in each case recall is lower than the corresponding precision and the performance of $\boldsymbol{L}$ differs between ordered and unordered results. These trends result from the fact that the model generates fewer constituents than exist in the test data. However, this trend is not entirely unexpected because it is not possible to completely accurately guess syntactic constituents from every substring $\boldsymbol{w}$. For example, parts of the sentence “in the next 18 minutes” can generate the sequence “in the next $\boxed{\text{CD}}$ $\boxed{\text{NN}}$” and “$\boxed{\text{IN}}$ $\boxed{\text{DT}}$ $\boxed{\text{JJ}}$ 18 minutes,” but the constituents $\boxed{\text{CD}}$ in the former case and $\boxed{\text{DT}}$ and $\boxed{\text{JJ}}$ in the latter case are not necessary in all situations. In contrast, $\boxed{\text{NN}}$ and $\boxed{\text{IN}}$ will probably be inserted most cases. As a result, the appearance of such ambiguous constituents in the training data is less consistent than that of necessary syntactic constituents, and thus the prediction model avoids generating such ambiguous constituents.

6.2.2 Simultaneous Translation

Next, we evaluate the translation results achieved by the proposed method. Figures 6 and 7 show the relationship between the mean number of words in the translation segments and translation accuracy of BLEU and RIBES respectively. Each horizontal axis of these graphs indicates the mean number of words in translation units that are used to generate the actual translation output, and these can be assumed to be proportional to the mean waiting time for listeners. In cases except *T2S-Wait*, these values are equal to the mean length of translation unit generated by the segmentation strategies, and in the case of *T2S-Wait*, this value shows the length of the translation units concatenated by the waiting strategy. First looking at the full sentence results (rightmost points in each graph), we can see that *T2S* greatly outperforms *PBMT* on full sentences,

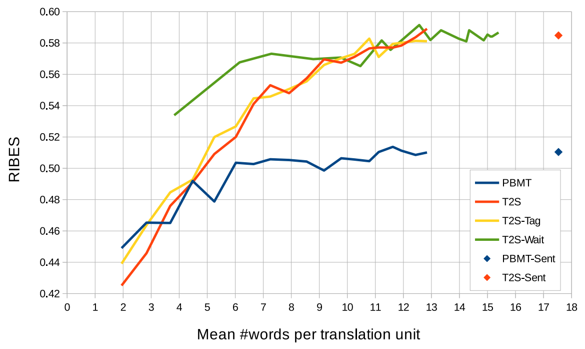


(a) n-words segmentation

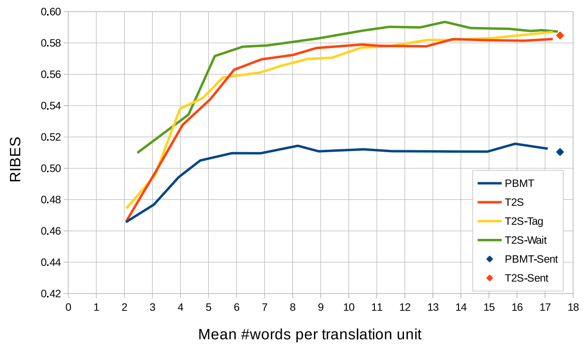


(b) optimized segmentation

Figure 6: Mean #words and BLEU scores of each method.



(a) n-words segmentation



(b) optimized segmentation

Figure 7: Mean #words and RIBES scores of each method.

underlining the importance of considering syntax for this language pair.

Turning to simultaneous translation, we first consider the case of n-words segmentation, which will demonstrate robustness of each method to poorly formed translation segments. When we compare *PBMT* and *T2S*, we can see that *T2S* is superior for longer segments, but on shorter segments performance is greatly reduced, dropping below that of *PBMT* in BLEU at an average of 6 words, and RIBES at an average of 4 words. This trend is reasonable, considering that shorter translation units will result in syntactically inconsistent units and thus incorrect parse trees. Next looking at the results for *T2S-Tag*, we can see that in the case of the n-words segmentation, it is able to maintain the same translation performance of *PBMT*, even at the shorter settings. Furthermore, *T2S-Wait* also maintains the same performance of *T2S-Tag* in BLEU and achieves much higher performance than any of the other methods in RIBES, particularly with regards to shorter translation units. This result shows that the method of waiting for more input in the face of potential reordering problems is highly effective in maintaining the correct ordering of the output.

In the case of the optimized segmentation, all three T2S methods maintain approximately the same performance, consistently outperforming *PBMT* in RIBES, and crossing in BLEU around 5-6 words. From this, we can hypothesize that the optimized segmentation strategy learns features that maintain some syntactic consistency, which plays a similar role to the proposed method. However, RIBES scores for *T2S-Wait* is still generally higher than the other methods, demonstrating that waiting maintains its reordering advantage even in the optimized segmentation case.

7 Conclusion and Future Work

In this paper, we proposed the first method to apply SMT using source syntax to simultaneous translation. Especially, we proposed methods to maintain the syntactic consistency of translation units by predicting unseen syntactic constituents, and waiting until more input is available when it is necessary to achieve good translation results. Experiments on an English-Japanese TED talk translation task demonstrate that our methods are more robust to short, inconsistent translation segments.

As future work, we are planning to devise more sophisticated methods for language modeling using constituent tags, and ways to incorporate previously translated segments into the estimation process for left-hand constituents. Next, our method to predict additional constituents does not target the grammatically correct translation units for which $\boldsymbol{L} = [\,]$ and $\boldsymbol{R} = [\,]$, although there is still room for improvement in this assumption. In addition, we hope to expand the methods proposed here to a more incremental setting, where both parsing and decoding are performed incrementally, and the information from these processes can be reflected in the decision of segmentation boundaries.

Acknowledgement

Part of this work was supported by JSPS KAKENHI Grant Number 24240032, and Grant-in-Aid for JSPS Fellows Grant Number 15J10649.

References

Srinivas Bangalore, Vivek Kumar Rangarajan Sridhar, Prakash Kolan, Ladan Golipour, and Aura Jimenez. 2012. Real-time incremental speech-to-speech translation of dialogs. In *Proc. NAACL*.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. WIT^3: Web inventory of transcribed and translated talks. In *Proc. EAMT*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*.

Andrew Finch, Xiaolin Wang, and Eiichiro Sumita. 2014. An exploration of segmentation strategies in stream decoding. In *Proc. IWSLT*.

Christian Fügen, Alex Waibel, and Muntsin Kolss. 2007. Simultaneous translation of lectures and speeches. *Machine Translation*, 21.

Tomoki Fujita, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2013. Simple, lexicalized choice of translation timing for simultaneous speech translation. In *Proc. Interspeech*.

Alvin Grissom II, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III. 2014. Dont until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Proc. EMNLP*.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proc. ACL*.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proc. AMTA*.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic evaluation of translation quality for distant language pairs. In *Proc. EMNLP*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. ACL*.

Chin-Yew Lin and Franz Josef Och. 2004. ORANGE: a method for evaluating automatic evaluation metrics for machine translation. In *Proc. COLING*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The Penn treebank. *Computational linguistics*, 19(2).

Evgeny Matusov, Arne Mauser, and Hermann Ney. 2006. Automatic sentence segmentation and punctuation prediction for spoken language translation. In *Proc. IWSLT*.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable japanese morphological analysis. In *Proc. ACL-HLT*.

Graham Neubig. 2013. Travatar: A forest-to-string machine translation engine based on tree transducers. In *Proc. ACL*.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics*.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. ACL*.

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Optimizing segmentation strategies for simultaneous speech translation. In *Proc. ACL*.

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Ckylark: A more robust PCFG-LA parser. In *Proc. NAACL-HLT*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proc. ACL*.

Vivek Kumar Rangarajan Sridhar, John Chen, Srinivas Bangalore, Andrej Ljolje, and Rathinavelu Chengalvarayan. 2013. Segmentation strategies for streaming speech translation. In *Proc. NAACL-HLT*.

Koichiro Ryu, Shigeki Matsubara, and Yasuyoshi Inagaki. 2006. Simultaneous english-japanese spoken language translation based on incremental dependency parsing and transfer. In *Proc. COLING*.

Baskaran Sankaran, Ajeet Grewal, and Anoop Sarkar. 2010. Incremental decoding for phrase-based statistical machine translation. In *Proc. WMT*.

Maryam Siahbani, Ramtin Mehdizadeh Seraj, Baskaran Sankaran, and Anoop Sarkar. 2014. Incremental translation using hierarchical phrase-based translation system. In *Proc. SLT*.

Fei Xia and Michael McCord. 2004. Improving a statistical MT system with automatically learned rewrite patterns. In *Proc. COLING*.

Mahsa Yarmohammadi, Vivek Kumar Rangarajan Sridhar, Srinivas Bangalore, and Baskaran Sankaran. 2013. Incremental segmentation and decoding strategies for simultaneous translation. In *Proc. IJCNLP*.