An Exploration of Segmentation Strategies in Stream Decoding

Andrew Finch *Xiaolin Wang* *Eiichiro Sumita*

Multilingual Translation Group
National Institute of Information and
Communications Technology
Kyoto, Japan
{first.last}@nict.go.jp

Abstract

In this paper we explore segmentation strategies for the stream decoder - a method for decoding from a continuous stream of input tokens, rather than the traditional method of decoding from sentence segmented text. The behavior of the decoder is analyzed and modifications to the decoding algorithm are proposed to improve its performance. The experimental results show our proposed decoding strategies to be effective, and add support to the original findings that this approach is capable of approaching the performance of the underlying phrase-based machine translation decoder, at useful levels of latency. Our experiments evaluated the stream decoder on a broader set of language pairs than in previous work. We found most European language pairs were similar in character, and report results on English-Chinese and English-German pairs which are of interest due to the reordering required.

1. Introduction

Statistical machine translation (SMT) technology has advanced to the point where it is becoming capable enough to be useful for many applications. The process of automatic simultaneous interpretation however is another matter entirely. The interpretation process is difficult, even for skilled human interpreters, and presents a major challenge to a machine the since in addition to the translation process, decisions need to be made about when to commit to outputting a partial translation. Such decisions are critical since once such an output is made it can be difficult and highly undesirable to correct it later if it is in error.

In simultaneous interpretation, the input to the automatic interpretation system is often a continuous stream of tokens. Since the output from the system occurs periodically, the output of the system is segmented. In order to produce this output segmentation two strategies can be employed. In the first, the stream is segmented before the machine translation process begins, and the machine translation system is constrained to translate using the given segmentation. In order to distinguish the methods that segment the input prior to the decoding in a pre-processing step, we will refer to them as "pre-segmentation" in this paper. In the second, the segmentation process is performed during the decoding of the input stream. The work presented here is primarily concerned with the latter, but proposes and evaluates a method to integrate them.

2. Related Work

The work in this paper is based upon the *stream decoder* [1], an extension to a phrase-based statistical machine translation decoder that allows it to decode directly from continuous stream of tokens. We describe this methodology in more detail in Section 3.

In [2] the prosody information in the speech signal was used to segment a continuous stream of speech input for translation. In their experiments, a silence duration of approximately 100ms was found to be suitable for segmentation.

A number of diverse strategies for pre-segmentation were studied in [3]. They studied both non-linguistic techniques, that included fixed-length segments, and a "hold-output" method. The hold-output method method is relevant to the research in this paper because it relies the same principle used by the stream decoder. It identifies contiguous blocks of text that do not contain alignments to words outside them. An SVM was used to predict these blocks prior to the decoding process; the stream decoder operates by identifying similar structures during decoding. Their experimental results showed this method to be ineffective. Linguistically-motivated segmentation techniques were also considered. Conjunctions, sentence boundaries and commas were investigated, with commas being the most effective segmentation cue in their investigation.

In [4] a strategy for pre-segmentation based on searching for segmentation points while optimizing the BLEU score was presented. An attractive characteristic of this approach is that the granularity of the segmentation can be controlled by choosing the number of segmentation boundaries to be inserted, prior to the segmentation process.

The automatic interpretation from English into Japanese has been studied in [5]. Their approach used heuristics to identify predicates that are likely to be invertible from a dependency structure derived from a phrase-structure parse of the English. They exploit the somewhat free word order of Japanese to re-order the Japanese tokens into an order that is appropriate for interpretation. The resulting word order may be a little dis-fluent, but is nonetheless grammatically valid and is typical of the kind of compromise that needs to be made during interpretation.

There are also some related studies in translation process research (for example, [6, 7]) that study in detail the process of human simultaneous interpretation.

In [8] it was shown that the prediction and use of soft boundaries in the source language text, when used as reordering constraints can improve the quality of a speech translation system.

3. Stream Decoding

The stream decoding strategy differs from approaches based on the pre-segmentation of the stream of input tokens in that the segmentation decisions are able to exploit information from the decoding process itself. In [9], it is stated that long segments of around 10-40 words are required in a pre-segmentation strategy in order to achieve performance close to the underlying machine translation system. These long segments give rise to long latencies, and the penalty for reducing the segment size in order to achieve acceptably latencies is typically severe. These issues have been addressed recently with more intelligent strategies for choosing the segmentation points [4], but nonetheless we believe the stream decoding approach deserves more attention in the literature, and merits further study for the following reasons:

- Stream decoding uses characteristics of the decoding process for segmentation, and requires no annotation of the input token stream.
- Stream decoding is able to enforce a maximum limit on the latency.
- The first results on English-Spanish translation ([1]) were very promising.

3.1. Overview of the Stream Decoding Process

The reader is referred to the original paper [1] for a complete description of the stream decoding process; in this section, for completeness, we provide a brief summary of the stream decoding methodology.

Figure 1 depicts a stream decoding process. The input to the stream decoder is a stream of tokens (it is also possible to configure the decoder to operate on tuples of confusable token sequences from a speech recognition decoder, but for the purposes of this paper we consider streams of tokens). A typical phrase-based machine translation system will decode token sequences, where a token (typically word) sequence usually represents a sentence in the source language. The decoder will construct a search graph from this sequence of tokens and output the n-best derivations of target token sequences from this graph.

The stream decoder, in contrast, operates on a potentially infinite sequence of tokens. As new tokens arrive, states in the search graph are extended with the new possible translation options arising from the new tokens. Periodically the stream decoder will commit to outputting a sequence of target tokens. At this point a state from the search graph is selected, the search graph leading from this state is kept, and the remainder discarded. The search then continues using the pruned search graph. In our implementation of the stream decoder the language model context is preserved at this state for use during the subsequence decoding. In this manner the stream decoder is able to operate on a stream of tokens that contains no segment boundary information. The segmentation occurs as a natural by-product of the decoding process.

3.2. Latency Parameters

The stream decoding process is governed by two parameters L_{max} and L_{min}. These parameters are illustrated in Figure 1. The L_{max} parameter controls the maximum latency of the system. That is, the maximum number of tokens the system is permitted to fall behind the current position. If interpreting from speech, the parameter represents the number of words the system is allowed to fall behind the speaker, before being required to provide an output translation. This parameter is a hard constraint that guarantees the system will always be within L_{max} tokens of the current last token in the stream of input tokens. The parameter L_{min} represents the minimum number of words the system will lag behind the last word spoken. It serves as a means of preventing the decoder from committing to a translation too early.

3.3. Determining the Segmentation Point

Algorithm 1 shows the algorithm used to select the segmentation point. The decoder maintains a sequence of tokens that represent the sequence of untranslated tokens from the input stream (see Figure 1). As new tokens arrive from the input stream, they are added to the end of the sequence. When the length of this sequence reaches L_{max}, the decoder is forced to provide an output. A search state is chosen from the sequence of states in the search graph representing the best hypothesis that covers the full sequence of untranslated words. In short, the best hypothesis is rolled back, state by state, until the remaining state sequence translates a contiguous sequence of source words starting from beginning of the sequence of untranslated words, and the number of words that would remain in the sequence of untranslated words after the translation is made, is at least L_{min}. It is possible that no

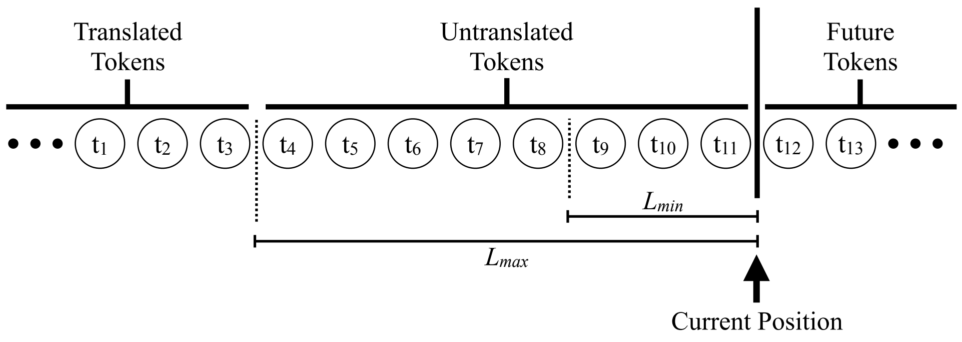


Figure 1: The stream decoding process.

Algorithm 1: Selecting a segmentation point.

```
Input: A sequence of search states $s_0, \ldots, s_n$
       representing the best hypothesis. $s_0$ being the
       initial state, and $s_n$ being the final state.
Output: A state $s_i \neq s_0$ representing the end of the
        translation segment, or $s_0$ if the process
        failed to find a suitable state.
foreach $i = n$ to 1 do
    if the tokens translated by $s_0 \ldots s_i$ are a
    contiguous sequence starting immediately after
    the last translated source token then
        if the number tokens translated by the states
        following $s_i$ at least $L_{min}$ then
            return $s_i$
        end
    end
end
return $s_0$
```

Language Pair	Training	Dev	Test
English-Spanish	180853	887	1701
English-Chinese	179651	887	1397
English-German	171721	887	1700

Table 1: Statistics on corpora using in the stream decoding experiments. The numbers given are in segments, representing individual subtitles, corresponding approximately to sentences.

such state exists, in which case the algorithm returns s_0, and since the stream decoder is required to make an output, it must use an alternative strategy.

In this alternative strategy, the stream decoder will undertake a new decoding pass in which it is forced to make a monotonic step as the first step in the decoding process. Then, a state is selected from the best hypothesis using Algorthim 1. This process may also fail if the monotonic step would lead to the violation of L_{min}. In our implementation, we allow the decoder to violate L_{min} only in this case.

4. Experimental Methodology

4.1. Corpora

For the experiments that explore the operation of and enhancements to the stream decoder we use the TED[1] talks data sets from the IWSLT2014 campaign. We studied English to: Spanish, Italian, French, German and Chinese, and found the results on the set of European language pairs were mostly similar in character, and we therefore report results on only English-Spanish (a typical pair) and English-German (an exceptional pair) from this set. Statistics on the corpora are given in Table 1. The European language data was tokenized by the Stanford PTBTokenizer. The Chinese was segmented using the Stanford Chinese word segmenter [10] according to the Chinese Penn Treebank standard.

4.2. Decoder

Our stream decoder was implemented within the framework of the OCTAVIAN decoder, a phrase-based statistical machine translation decoder that operates in a similar manner to the MOSES decoder [11]. The training procedure was quite typical: 5-gram language models were used, trained with modified Kneser-Ney smoothing; MERT [12] was used to train the log-linear weights of the models; the decoding was performed with no limit on the distortion.

4.3. Evaluation

The BLEU score [13] was used to evaluate the machine translation quality in all our experiments. Where sentence segmentation was known we used both talk and sentence-level BLEU, and for the experiments where true stream decoding was performed on a stream of tokens with no segmentation information, talk-level BLEU was used. In talk level BLEU each talk is considered to be a single sentence in the BLEU computation. For consistency only the talk level BLEU results are reported in this paper, but the results from the sentence-level BLEU experiments were similar in character.

[1]http://www.ted.com

5. Alternative Stream Decoding Strategies

5.1. Increasing the Output Frequency

5.1.1. Methodology

As explained in the previous section, in the originally proposed stream decoder, the best hypothesis is unrolled backwards until a suitable point is found for output. The principle here is to find the longest subsequence of states in the best hypothesis that satisfies the constraints that determine whether the segmentation point is permissible. However, other strategies are possible. One plausible strategy is instead of committing to the longest permissible output, commit to the shortest. The algorithm is identical to that shown of Algorithm 1 except that the "foreach" loop that ranges from $i = n \dots 1$, ranges from $i = 1 \dots n$. The approach takes less of a risk, since it will commit to shorter translations. On the downside, it will lag behind the original strategy given the same values for its latency parameters.

For this reason, it is unfair to compare these approaches under the constraint that their L_{max} and L_{min} parameters are the same, since there may be a bias in favor of the strategy that commits to the shortest output, and this strategy will gain its advantage by increasing the latency of the tokens in the output stream. To remove this potential bias, we therefore compare these two methods in the experiment below by measuring the trade-off between machine translation quality (measured using the BLEU score) and average latency per token L_{avg}. That is the average number of tokens each token is behind the input stream, given by:

$$L_{avg} = \frac{\sum_{i=1,N} L(i)}{N} \tag{1}$$

where N is the total number of words in the input stream, and $L(i)$ is the latency after word i has been processed.

5.1.2. Experiment

Figure 2 shows the results on English-to-Spanish translation task. Experiments were run for values of L_{max} in the range 1 to 10, and the points are annotated with these values. The oracle values of L_{min}, that is the value of L_{min} that gave rise to the highest BLEU score, were used. The graph plots L_{avg} against BLEU score for each experiment. For high and low values of L_{max} the two strategies are similar in performance, but for $3 \leq L_{max} \leq 6$ the strategy that makes more frequent, but shorter output is the better strategy. Of course there may be human factors to consider, but in terms of the machine translation evaluation scores at least, the shorter output strategy would seem to be the more effective approach, especially when lower latencies are required. This approach varied in its effectiveness across language pairs however, with some European language pairs (for example English-to-French) showing almost no difference in performance. The proposed strategy was always at least as good as the baseline, and therefore it was adopted in the remainder of the experiments reported here.

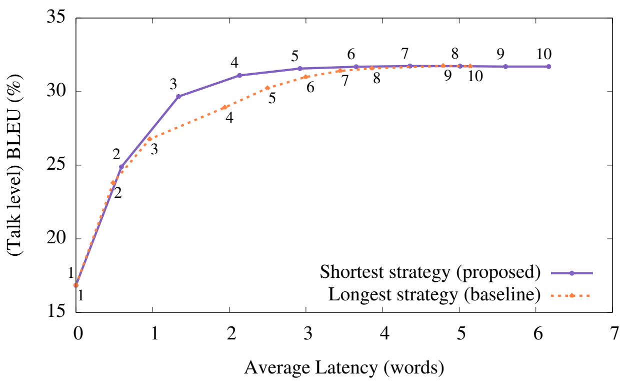


Figure 2: The trade-off between BLEU score and average latency for two different strategies for selecting the segmentation point.

5.2. Minimizing the Number of Forced Monotonic Steps

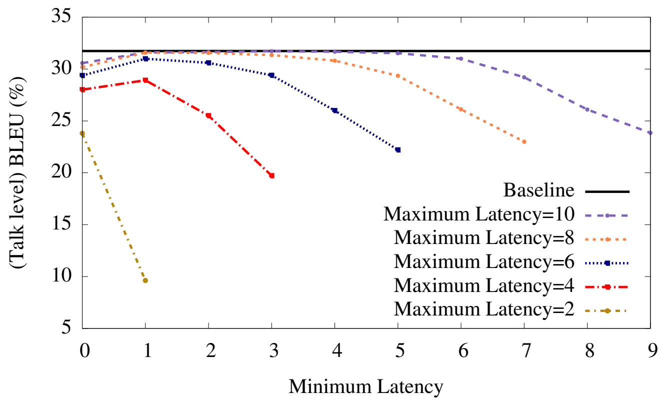
5.2.1. Motivation

In Section 3.3 it was explained that during stream decoding the best hypothesis is rolled back until a satisfactory segmentation point is found. In some cases, no such segmentation point exists and the decoder resorts to an alternative decoding strategy that forces the first step of the decoding process to be monotonic. This section is motivated by the concern that constraining the decoder in this manner will lead to translation hypotheses that diverge from the optimal path, impacting the overall translation performance. We therefore seek a method that can reduce the number of forced monotonic steps.

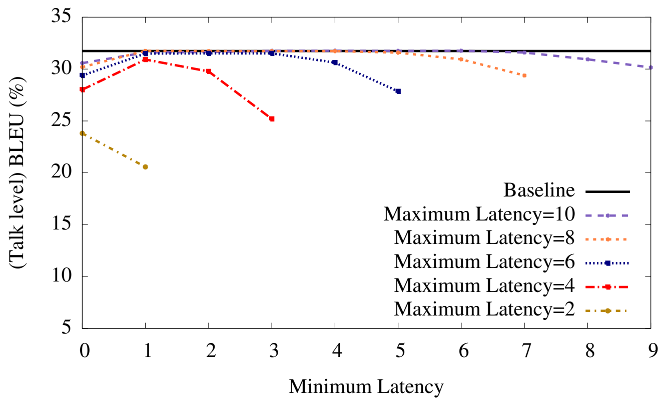
5.2.2. Methodology

One plausible method to alleviate the issue is to extend the stream decoding approach to allow it to select a segmentation point from the whole search graph, rather than from the 1-best hypothesis. We proposed a straightforward extension of the existing approach: to select a state from the n-best list. The proposed method applies Algorithm 1 iteratively over an n-best list of derivations, from rank 1 to n, terminating on the first rank in which a suitable segmentation point is found. Only if no segmentation point is found in the n-best list, does the decoder resort to a forced monotonic decoding step.

We analyzed the effect of the approach on the number of forced monotonic steps for English-Spanish. The results are shown in Figure 3, the oracle value of L_{min} is used. The figure shows the percentage of translated segments that were the result of a decoding hypothesis that contained a forced monotonic step. The results clearly show that the proposed method can have a substantial impact on the number of forced monotonic steps. We investigate whether or not this leads to an improvement in machine translation performance in the next sections.

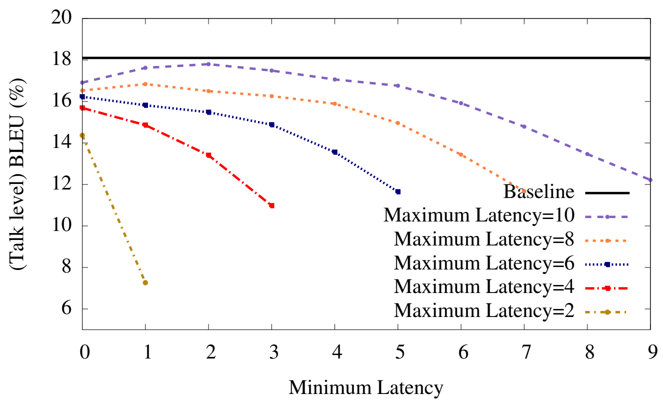


(a) Selecting from the best hypothesis.

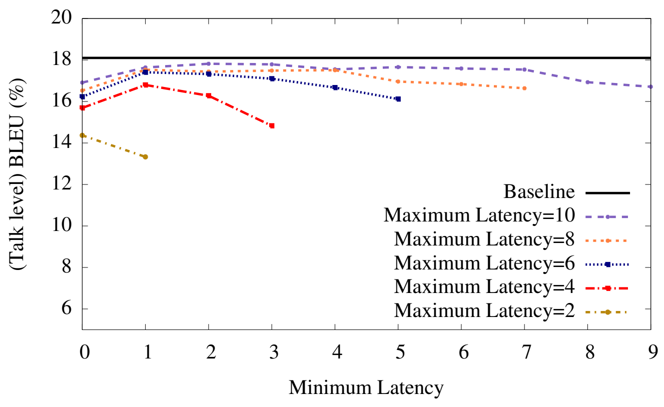


(b) Selecting from the 20-best hypotheses.

Figure 4: Using the n-best hypotheses to select the segmentation point for English-Spanish.



(a) Selecting from the best hypothesis.



(b) Selecting from the 20-best hypotheses.

Figure 5: Using the n-best hypotheses to select the segmentation point for English-Chinese.

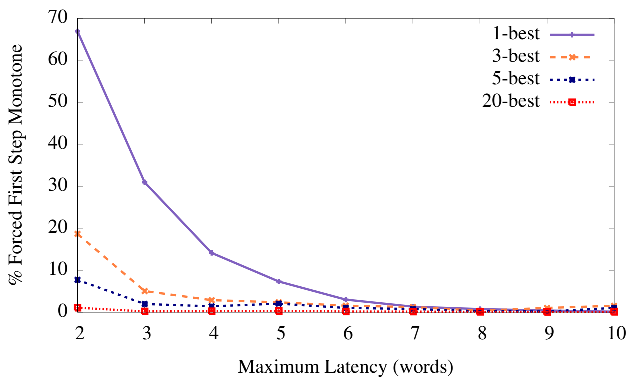


Figure 3: The proportion of output segments containing forced monotonic decoding steps for different length n-best lists.

5.2.3. English-Spanish Translation

Even though the number of forced monotonic decoding steps can be reduced by using an n-best list, it does not guarantee an improvement in performance. Selecting a state from a hypothesis other than the 1-best comes with a price as hypotheses further down the n-best list are likely to represent translations of lower quality.

Figure 4 shows the results of an experiment using the proposed method in the previous section on the English-to-Spanish task. The experiments used identical settings apart from the length of the n-best list used to select the segmentation point. The baseline on both graphs represents the performance of the underlying phrase-based SMT decoder when decoding the data according to the segmentation provided in the corpus.

The results show that the stream decoder, which must provide its own segmentation is able to achieve evaluation performance comparable to the baseline SMT system. The stream decoder may have been helped by the fact that the baseline system was decoding without a distortion limit. Typically languages such as English and Spanish, having similar word orders benefit from a constraint on the reordering, which the stream decoder may be providing as a consequence of more monotonic decoding process. Nonetheless we feel its performance is impressive.

The results of this experiment show our proposed method is very effective in improving the stream decoder. There are two important differences in the graphs, firstly the curves do not drop as sharply as L_{min} is increased, making the approach less sensitive to the selection of this parameter. Secondly, and more importantly, the performance on the experiments with lower latencies (where L_{max} is less than 6), is improved overall. We ran a set of experiments on the English-Spanish task to determine the effect of varying the size of the n-best list. We found that the approach was not very sensitive to the size of the n-best list for small values of n. The best results were obtained with $5 \leq n \leq 20$.

5.2.4. *Other Language Pairs*

The original stream decoder was evaluated on an English-Spanish task, and for consistency with the original work, so far we have shown results on the same language pair (but a different corpus). We ran experiments on all of the languages for which data was provided for the IWSLT2014 machine translation shared tasks. The stream decoder proved robust to differences in the language pair chosen. The results were generally similar in character to those presented for English-Spanish. We have omitted these results for brevity, and instead present results on the English-Chinese and English-German pairs which are interesting because their word orders are not similar, and as a consequence a substantial amount of reordering is necessary in the decoding process. These language pairs were expected to present more of a challenge to the stream decoder.

We conducted the same experiment presented in the previous section on an English-to-Chinese task, and the results are shown in Figure 5. As expected, it can be seen in the Figure 5a that the cost in terms of BLEU score is greater when lower latencies are required than for English-to-Spanish. The results have the same general character as before; the use of the n-best list has improved the performance of the lower latency curves, and also made the decoder far less sensitive to variations in the L_{min} parameter.

Among the European languages, German has some significant structural differences that can be expected to create difficulties for simultaneous interpretation. We show the results on the English-German pair in Figure 6. The results appear similar to the English-Chinese results, with a larger penalty in BLEU for shorter latencies. Moreover, the curves on the graph fall more sharply than the other languages tested with increasing L_{min}, indicating that the stream decoder is more sensitive to the value chosen for this parameter.

5.3. Selecting the Most Productive State

Instead of selecting a state in the set of 1-best or n-best hypotheses according to the algorithms described in the previous section, it is also possible to use other criteria to select the search state from the full search graph. One plausible

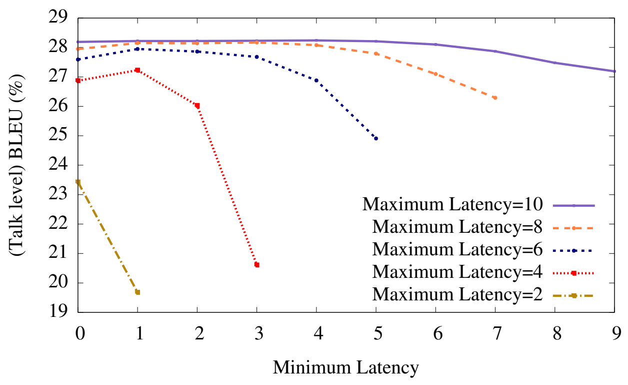


Figure 6: Performance on the English-German task (Using a 20-best list).

heuristic is to select the state that is in the greatest number of search paths leading to the final stack; the "most productive" state. The intuition behind this idea was that this state might provide the greatest number of good alternative search paths for decoding the future tokens. In the event of a tie in which several states gave rise the same number of hypotheses on the final stack, the state on the highest probability path was given precedence. If this failed to break the tie, the state closest to the initial state on the path was selected.

Unfortunately this strategy proved to be less effective than the simpler strategies described previously. We believe the reason may have caused by this strategy selecting states on sets of paths where the best path in the set had too low a rank. We would like to pursue similar ideas in the future, with the overall goal of removing the parameter L_{min} entirely from the decoding process, allowing the decoder more freedom to decode.

5.4. Introducing Segmentation Points into the Stream

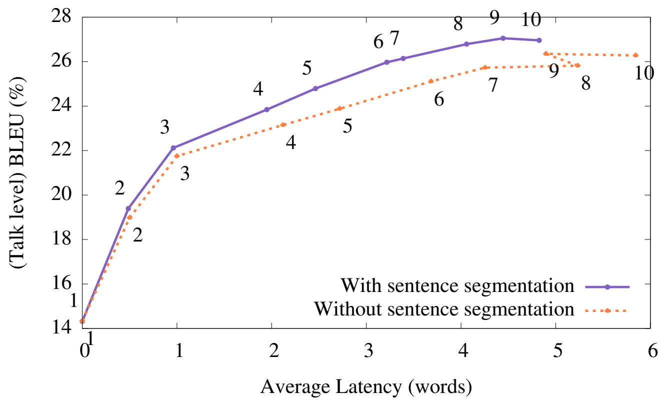
5.4.1. *Motivation*

As mentioned in Section 2, it has been shown that an input stream can be segmented effectively prior to the decoding process, using information derived from the input word sequence itself (punctuation, part-of-speech tags etc.) and also information from the speech recognition system (for example prosody). In this section we explore the idea of introducing segmentation information into the input stream, to support the segmentation process during stream decoding.

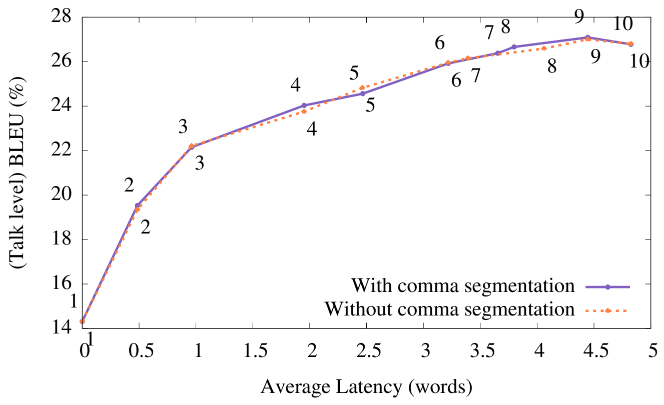
5.4.2. *Methodology*

In [3] the most effective segmentation strategy was to place segmentation boundaries at commas in the input. In addition segmenting at sentence boundaries also proved to be effective. Using predicted rather than reference commas did not seem to have a negative impact on machine translation performance.

We study the effect of introducing special tokens into the



(a) Using sentence segmentation information.



(b) Using sentence and comma segmentation information.

Figure 7: The effect of introducing segmentation information into the stream for English-to-Spanish.

stream to mark the ends of both sentence internal and sentence final segments. In our experiments we use the positions of commas in the corpus as the position at which to introduce sentence internal segment termination tokens (denoted ⟨p⟩), and the sentence segmentation in the corpus to delimit sentences (using the token ⟨s⟩).

There are a number of plausible strategies for using these tokens during decoding, and we wish to explore more of these in future research. In these experiments we study the case where priority is given to the segmentation indicated by the tokens in the input stream in the following manner: when an ⟨s⟩ or a ⟨p⟩ token arrives on the input stream, the stream decoder translates all untranslated words, and creates an initial search state from which to continue the decoding process. In the case of the ⟨p⟩ token, the language model context is preserved; in the case of ⟨s⟩ it is discarded. In both cases the decoder can violate the L_{min} constraint.

5.4.3. Experiments

The experiments were carried out on data with the punctuation removed from both source and target sides to eliminate the ambiguity of where to place the segmentation tokens in the stream. The punctuation was not used in training the machine translation systems' models, nor was it used in evaluation, but it was used to place the ⟨s⟩ and ⟨p⟩ tokens. The results are shown in Figure 7. It is clear from Figure 7a that sentence boundaries were useful to the stream decoder. The experiment in Figure 7b shows that adding ⟨p⟩ information surprisingly did not give any additional benefit.

6. Conclusions

In this paper we have presented a study of several variations of the stream decoder. The stream decoder is able to decode from a continuous stream of tokens, and is capable of performing segmentation as it decodes. Previous studies have shown this technique can achieve respectable levels of performance whilst maintaining a usefully low level of latency. The experiments in this paper support the original findings and also broaden the study of this decoder by evaluating it on new datasets and new language pairs. Of particular interest were English-Chinese and English-German tasks, which are challenging due to the differences in word order. Our results show that the although BLEU score was impacted at shorter latencies, the behavior of the stream decoder was quite similar in character to that of the language pairs. We believe the original claims that stream decoding can achieve low latency translation with only a small degradation in performance are valid, and can be extended to a broad range of language pairs.

During the course of the research for this paper, we studied a number of alternative strategies for increasing the performance of the decoder. We found a simple but highly effective variant of the stream decoder was one that selected the segmentation point using the n-best list of hypotheses rather than the 1-best. In our experiments this technique substantially improved the performance of the decoder at shorter latencies and also made the decoder less sensitive to the value of the minimum latency constraint.

This paper also proposed a technique for integrating segmentation information from an external source into the stream decoding process. Our experiments show that reliable sentence segmentation information may be used effectively in stream decoding to guide the segmentation process.

In future research we would like to study the behavior of the stream decoder on language pairs with longer distance reordering such as Japanese or Korean to the European languages.

7. References

[1] M. Kolss, S. Vogel, and A. Waibel, "Stream decoding for simultaneous spoken language translation," in *Proceedings of Interspeech*, Brisbane, Australia, 2008.

[2] S. Bangalore, V. K. R. Sridhar, P. Kolan, L. Golipour,

and A. Jimenez, “Real-time incremental speech-to-speech translation of dialogs,” in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics – Human Language Technologies (HLT-NAACL)*, Montreal, Canada, 2012, pp. 437–445.

[3] V. K. R. Sridhar, J. Chen, S. Bangalore, A. Ljolje, and R. Chengalvarayan, “Segmentation strategies for streaming speech translation.” in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics – Human Language Technologies (HLT-NAACL)*, Atlanta, USA, 2013, pp. 230–238.

[4] Y. Oda, G. Neubig, S. S. T. Toda, and S. Nakamura, “Optimizing segmentation strategies for simultaneous speech translation,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Baltimore, USA: The Association for Computer Linguistics, June 2014.

[5] K. Ryu, S. Matsubara, and Y. Inagaki, “Simultaneous English-Japanese spoken language translation based on incremental dependency parsing and transfer,” in *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. Sydney, Australia: The Association for Computer Linguistics, 2006.

[6] P. Padilla, M. T. Bajo, and F. Padilla, “Proposal for a cognitive theory of translation and interpreting,” *The Interpreters Newsletter*, 1999.

[7] S. Tirkkonen-Condit, *Tapping and mapping the processes of translation and interpreting: outlooks on empirical research*. John Benjamins Publishing Company, 2000, vol. 37.

[8] E. Matusov, D. Hillard, M. Magimai-Doss, D. Hakkani-Tur, M. Ostendorf, and H. Ney, “Improving speech translation with automatic boundary prediction,” in *Proceedings of Interspeech*, Antwerp, 2007, pp. 2449–2452.

[9] M. Kolss, M. Wölfel, F. Kraft, J. Niehues, M. Paulik, and A. Waibel, “Simultaneous German-English lecture translation,” in *Proceedings of the International Workshop on Spoken Language Translation (IWSLT)*, Waikiki , Hawai’i, USA, 2008, pp. 174–181.

[10] H. Tseng, P. Chang, G. Andrew, D. Jurafsky, and C. Manning, “A conditional random field word segmenter for sighan bakeoff 2005,” in *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, vol. 171. Jeju Island, Korea, 2005.

[11] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowa, 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 Association for Computational Linguistics (ACL 2007): demo and poster sessions*, Prague, Czeck Republic, June 2007, pp. 177–180.

[12] F. J. Och, “Minimum error rate training for statistical machine translation,” in *Proceedings of the 41st Meeting of the Association for Computational Linguistics (ACL 2003)*, Sapporo, Japan, 2003.

[13] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “BLEU: a method for automatic evaluation of machine translation,” in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Morristown, NJ, USA: Association for Computational Linguistics, 2001, pp. 311–318.