Stream Decoding for Simultaneous Spoken Language Translation

Muntsin Kolss[1]*, Stephan Vogel*[2]*, Alex Waibel*[1,2]

[1]Fakultät für Informatik, Universität Karlsruhe (TH), Germany
[2]Language Technologies Institute, Carnegie Mellon University, USA
kolss@ira.uka.de, vogel@cs.cmu.edu, waibel@ira.uka.de

Abstract

In the typical speech translation system, the first-best speech recognizer hypothesis is segmented into sentence-like units which are then fed to the downstream machine translation component. The need for a sufficiently large context in this intermediate step and for the MT introduces delays which are undesirable in many application scenarios, such as real-time subtitling of foreign language broadcasts or simultaneous translation of speeches and lectures.

In this paper, we propose a statistical machine translation decoder which processes a continuous input stream, such as that produced by a run-on speech recognizer. By decoupling decisions about the timing of translation output generation from any fixed input segmentation, this design can guarantee a maximum output lag for each input word while allowing for full word reordering within this time window.

Experimental results show that this system achieves competitive translation performance with a minimum of translation-induced latency.

Index Terms: speech translation, machine translation, decoding, latency, real-time

1. Introduction

In speech translation real-time processing is required. This means, first of all, that both speech recognition and translation work in less than real-time. A second aspect is latency, i.e., the delay between words being uttered by the speaker and the translation of these words being delivered to the listener. In dialog systems, where utterances are typically short and speakers constantly take turns, this is not as important as in lecture translation, or translation of broadcast news. There, the speaker will not make pauses to allow the translation system to catch up. The problem is even more pronounced in situations where audio-visual material is used, like slides or video material. A long delay between the original utterance, during which the speaker may navigate with a light-pointer over a slide, and the generation of the translation will make it more difficult for the listener to follow the lecture, and could severely impact the understanding of the presented material.

Translating with low latency poses severe problems, as is well-known to human interpreters. For example, when translating from German to English the verb may come as the last word in the German sentence, while the English translation would require a sentence structure which puts the verb in the second position. More generally speaking, long-distance reordering and short latency do not go together. Trained interpreters have an arsenal of strategies to cope with these problems. While a long-term goal in speech translation should be to integrate such strategies, our current goal is much more modest: achieving low-latency translation without losing significantly in translation quality compared to sentence-level translation.

The standard solution to achieve low latencies, segmenting the ASR hypothesis into shorter segments before translating them, suffers from several drawbacks that significantly degrade translation quality: Choosing meaningful segment boundaries is difficult and error-prone, each boundary destroys the available context for language modeling and finding longer phrase translations, and no word reordering across segment boundaries is possible.

In this paper we describe a novel decoding strategy for continuous, low-latency speech translation and present experiments showing that low-latency translation is possible with only a small degradation in performance.

In section 3, we describe the baseline statistical machine translation system and decoder used for our experiments. Section 4 describes the design of the proposed stream decoder. Sections 5 and 6 present experimental results comparing the stream decoder with segmentation strategies.

2. Related Work

[1] introduces the concept of Anytime Translation, focusing on the interruptability and response time of speech translation systems.

[2], [3] and [4] most recently study segmentation as a means to improve translation performance, but do not consider latency in detail. They find that rather long segments of 10-40 words on average are needed to yield acceptable translation quality, which are unsuitable for real-time simultaneous translation.

[5] describe an end-to-end system for simultaneous translation of speeches and lectures. The system runs in real-time but uses the standard approach of an intermediary segmentation step, resulting in a combined latency of 1-2 sentences.

3. Baseline Machine Translation System

In our experiments, we use a phrase-based statistical machine translation system which uses a log-linear combination of several models: an n-gram target language model, phrase translation models for both directions $p(f|e)$ and $p(e|f)$, a distance-based distortion model, a word penalty and a phrase penalty. The model scaling factors for these features are optimized on the development set by Minimum Error Rate training [6].

The Pharaoh toolkit [7] was used to extract bilingual phrase pairs from the parallel training corpus.

3.1. Baseline Decoder

The baseline decoder operates on the sentence level, i.e., the best translation is output after the source sentence has been

completely translated. The decoding process can be described in the following way:

- Construction of a translation lattice, or word graph, from the source sentence, containing all available word and phrase translations.
- Finding the best combination of these partial translations, such that every word in the source sentence is covered exactly once. This amounts to doing a best path search through the translation lattice, which is extended to allow for word reordering.

To restrict the search space, only limited word reordering within a local window is performed.

3.2. Input Segmentation for Low-Latency Decoding

One possibility to achieve low-latency translation with such a decoder is to segment the input into shorter segments and run the standard decoding algorithm. Such an approach has been used in [2], [3] and [4]. Segmentation can be done based on pauses in the speech signal, by using a hidden-event language model to detect meaningful boundaries, or simply by cutting the input word stream into fixed-length segments. However, all these segmentation approaches suffer from the following problems:

- Phrases which would match across the segment boundaries can no longer be used.
- Language model context is lost across the segment boundaries.
- If the language model is trained on sentence segmented data there will often be a mismatch for the begin-of-sentence and end-of-sentence LM events.
- Finally, making the decision, which of the many alternative translations is the best one, at intermediate positions rather than at the sentence end only may result in making wrong decisions more often.

All this will lead to a drop in performance, and experiments reported by different authors have shown that the drop in translation quality is quite severe as soon as a short latency is enforced.

4. Stream Decoder Design

4.1. Continuous Translation Lattice

In the baseline stack decoder, one translation lattice is created from each input utterance or sentence with translation alternatives attached as edges. After the utterance has been decoded, the translation lattice is discarded for the next input utterance.

In contrast, the stream decoder maintains a continuous translation lattice to be able to process an "infinite" input stream from the speech recognizer in real-time. New incoming source words are added to the end of the translation lattice, and the lattice is truncated at the start to remove the part for which translation output has been committed. When a new source word is added to the end, the lattice is immediately expanded with all newly matching word and phrase translation alternatives.

Because the translation lattice has the property of a confusion network, the decoder can also handle real-time lattice input from the speech recognizer instead of a first-best ASR hypothesis if it has the form of a confusion network as well. In that case, the decoder expects each confusion network tuple, representing a time window of alternative "confusable" source words, to arrive at the same time, and adds them to the translation lattice at the same newly created node.

4.2. Asynchronous Input and Output

Each new incoming source word or confusion network tuple triggers a search for the best path through the current translation lattice. However, this best translation hypothesis is not necessarily output immediately as translation; the output can be partially or completely delayed, until either a certain time span has passed or new input has arrived, which leads to lattice expansion and a new search.

Thus a sliding time window is created during which the translation output lags the corresponding incoming source stream, with the current translation lattice spanning the still untranslated input.

Once the decision has been made to output a part of the best current translation hypothesis, the translation lattice is truncated at the start to reflect this and the initial decoder hypothesis for the next search is set to the state at the end of the committed output. The remaining part of the translation hypothesis can optionally be displayed as "unfixed" live output, in the style that computer-assisted translation systems use to suggest possible continuations.

4.3. Output segmentation

To decide which part of the current best translation hypothesis to output, if any at all, the decoder uses two parameters:

- Minimum Latency L_{min}. The translation covering the last L_{min} untranslated source words received from the speech recognizer at any point is never output, except at the very end. This allows the decoder to postpone translating the current end of the input stream until more context becomes available.
- Maximum Latency L_{max}. When the latency reaches L_{max} source words, translation output covering the source words exceeding this value is forced.

To find the output boundary, the currently best translation hypothesis is unrolled backwards until the last L_{min} source words have been passed. If the hypothesis reached has no reordering gap the translation up to this point is generated. Committing to a partial translation means that the lattice up to this point will be deleted. Therefore, if the hypothesis has some open reordering gaps some source words would remain untranslated.

If the hypothesis contains open gaps we follow further back until a state is reached where all word reorderings are closed. If no such state is found, a new restricted search through the lattice is performed that only expands hypotheses which have no open reorderings at the node where the maximum latency would be exceeded, i.e. that have translated all source words up to that point.

It is instructive to take a closer look at the effect of the minimum and maximum latency parameters. At each point we have a translation lattice which has at most L_{max} source words. L_{min} words are kept as context information. Using a n-gram LM means that we need a context of about n words to decide which alternative translation is the best one. Therefore, we can expect that making the minimum latency too small will impact translation quality.

$L_{max} - L_{min}$ is the number of source words for which the partial translation can be committed. If this difference is very

small then many of the hypotheses will still have open reordering gaps. In other words, decoding degenerates more and more to monotone decoding the smaller the difference between maximum and minimum latency is.

The phrases which can be used by the decoder are also directly affected by these parameters. Essentially, no phrases will be used in any translation for which the number of source words is larger than $L_{max} - L_{min}$. As has been shown, translation quality improves significantly with longer phrases up to 3 source words. So we can expect to see a drop in translation quality if $L_{max} - L_{min}$ is less than 3.

5. Experimental Setup

5.1. Task and Corpus Statistics

The experiments were performed on the English-to-Spanish European Parliament Plenary Sessions (EPPS) task, i.e., parliamentary speech data from native speakers in the European Parliament. Development and testset correspond to the ASR condition of the 2007 TC-STAR evalation [8]. The word error rate of the English ASR input used was 6.9%.

The models were trained on the parallel *Final Text Edition* data from the European Parliament's website, with a verbalization of number and date expressions applied as preprocessing. The statistics of the training and test corpora are shown in Table 1.

		English	Spanish
Train	Sentences	1.24M	
	Words	34.8M	36.4M
Test	Sentences	849	
	Words	26,812	-

Table 1: *Statistics for training and test corpus.*

We used 4-gram language models in all experiments, estimated with modified Kneser-Ney smoothing as implemented in the SRILM toolkit [9].

5.2. Evaluation Criteria

We report translation results using the well-established objective error measure BLEU [10]. To calculate scores on automatically produced segments, the translation output must be resegmented to match the number of reference segments. This is done using the method described in [11], which determines the best alignment with the multiple reference translations based on the word error rate. Scores were calculated on two reference translations, including casing and punctuation marks.

To measure latency in a platform- and implementation-independent manner, we use the delay in number of source words until the translation of a particular source word appears in the output. The rationale is that the actual real-time performance of an integrated system is usually limited by the ASR component, while we are interested in algorithmically induced latencies here which cannot be eliminated by faster hardware.

6. Results and Discussion

6.1. Baseline

A manual sentence segmentation produced by humans serves as a baseline to evaluate a segmentation optimized towards translation quality. For our testset, the BLEU score is 37.28, and the segments had an *average* length of 24.4 words. Some segments were considerably longer, however, and a system using only manual sentence boundaries would only start translating after the speaker has finished a sentence, leading to long delays.

In an integrated system, reliable sentence boundaries cannot be assumed to be available, and for the following experiments, the manual sentence segmentation was discarded.

6.2. Fixed Segment Lengths

Getting a lower translation latency is easy: simply use shorter segments. Figure 1 shows the results of using a fixed segment length, for segments of 1-10 words and a selection of longer segment lengths.

The case of segment length 1 has the shortest possible latency (zero words delay) but degenerates into single word to single word translation. At the other end, the segment length of 10000 lets the decoder translate the entire document as a single utterance.

Not surprisingly, translation performance suffers heavily if the input segments are very short, as at each segment boundary the context is destroyed completely and no word reordering across boundaries is possible.

It can also be seen that in this simple approach, rather long segments are needed to get near optimal performance. Translation quality still improves when going to higher segment lengths, indicating that introducing a wrong segment boundary at an arbitrary place hurts more than translating across boundaries.

Note that a constant segment length of N words corresponds to an average latency of roughly N/2 words.

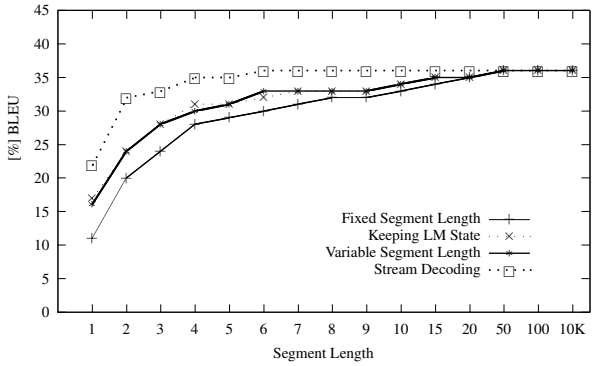

Figure 1: Translation performance of the different approaches on the English-Spanish EPPS task.

6.3. Keeping Target Language Model State

So far, each input segment was translated in isolation with no information crossing the segment boundaries, even though the output segments are concatenated afterwards to form the translation output presented to the user. Since the translated segments will be viewed as a longer contiguous unit, we can as well keep the language model state across segment boundaries rather than discard it - i.e., the decoding of each new segment is started with a language model history of the words of the preceding segment(s). The improvement from this approach is shown in Figure 1.

6.4. Variable Length Segments

Various approaches using variable-length segmentation have been found to improve translation quality at longer segment lengths. In the 2007 TC-STAR evaluation [8], an automatic segmentation was produced for our testset based on pause information from the original audio signal and a source language model, yielding an average segment length of 32 words.

Since we require relatively short segments for real-time translation and would like to be able to specify a guaranteed maximum delay, we further segment the provided fragments into shorter fixed-length segments if the maximum latency is exceeded. As can be seen in Figure 1, this gave no significant improvement over keeping the language model state across segments.

6.5. Stream Decoding

The stream decoder organization described in section 4 allows translating without an explicit segmentation.

In the experiment included in Figure 1, the maximum segment length that the other approaches respect was enforced by setting the maximum latency parameter L_{max} to the corresponding length value. The minimum latency parameter L_{min} was chosen between 0 and L_{max} such that the translation quality was optimized.

It can be seen that the proposed stream decoder significantly outperforms all other approaches, and across all segment lengths. It is able to produce nearly optimal translation quality with a maximum latency of just six words, and still has good translation performance when using a maximum latency of three or even only two words delay.

Only at the very shortest latency values does the streaming approach suffer from adverse effects occurring at hard segment boundaries: loss of the immediate context, reduced possibilities for word reordering, and a loss of longer matching phrase-to-phrase translations.

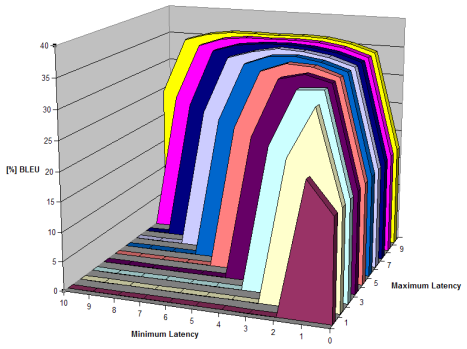

Figure 2: Translation performance of the stream decoder for different minimum and maximum latencies.

What is the optimal value of L_{min} for a given maximum latency L_{max}? Figure 2 shows the BLEU scores for all combinations of L_{min} and L_{max} up to a maximum of 10 words. Across all segment lengths, a minimum latency of about one half of the maximum latency is optimal. As can be seen from the description in section 4.3, the point where the minimum latency is passed is also where most decisions to commit translation output are taken. Placing this spot in the middle of the sliding window keeps the output decisions as far away as possible from the fringes where the decoder's choices are either more restricted by the already committed translation output, or where it may have to commit prematurely to a translation which restricts the search space for the following downstream input.

7. Conclusion

In this paper, we have studied the issue of reducing latencies in speech translation systems for simultaneous translation in real-time.

We have presented a novel decoder structure and algorithm capable of processing continuous input streams in real-time which not only outperforms alternative strategies based on segmenting the ASR input prior to translation, but is also able to produce nearly optimal translation quality at very low latencies. Moreover, by decoupling the decisions of when to generate translation output from any fixed input segmentation, the proposed stream decoder can be configured to guarantee a maximum delay between input and translation, while allowing for full word reordering within a sliding window. Thus, this approach is especially suited for building integrated real-time simultaneous translation systems.

8. References

[1] J. W. Amtrup, *Incremental Speech Translation.* Springer Verlag, 1999.

[2] E. Matusov, D. Hillard, M. Magimai-Doss, D. Hakkani-Tür, M. Ostendorf, and H. Ney, "Improving Speech Translation by Automatic Boundary Prediction." in *Interspeech*, Antwerp, Belgium, 2007.

[3] C. Fügen and M. Kolss, "The Influence of Utterance Chunking on Machine Translation Performance." in *Interspeech*, Antwerp, Belgium, 2007.

[4] S. Rao, I. Lane, and T. Schultz, "Optimizing Sentence Segmentation for Spoken Language Translation," in *Interspeech*, Antwerp, Belgium, 2007.

[5] C. Fügen, M. Kolss, D. Bernreuther, M. Paulik, S. Stüker, S. Vogel, and A. Waibel, "Open Domain Speech Recognition & Translation: Lectures and Speeches." in *Proc. ICASSP*, Toulouse, France, 2006.

[6] F. Och, "Minimum Error Rate Training in Statistical Machine Translation," in *41st Annual Meeting of the Association for Computational Linguistics (ACL)*, Sapporo, Japan, 2003.

[7] P. Koehn and C. Monz, "Manual and Automatic Evaluation of Machine Translation between European Languages." in *NAACL 2006 Workshop on Statistical Machine Translation*, New York, USA, 2006.

[8] O. Hamon, D. Mostefa, and K. Choukri, "End-to-end Evaluation of a Speech-to-Speech Translation System in TC-STAR," in *Machine Translation Summit XI*, Copenhagen, Denmark, 2007.

[9] A. Stolcke, "SRILM – An Extensible Language Modeling Toolkit." in *Proc. ICSLP*, Denver, Colorado, USA, 2002.

[10] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a Method for Automatic Evaluation of Machine Translation." IBM Research Division, T. J. Watson Research Center, Tech. Rep. RC22176 (W0109-022), 2002.

[11] E. Matusov, G. Leusch, O. Bender, and H. Ney, "Evaluating Machine Translation Output with Automatic Sentence Segmentation." in *Internat. Workshop on Spoken Language Translation*, Pittsburgh, USA, 2005.