Dynamic Transcription for Low-latency Speech Translation

Jan Niehues, Thai Son Nguyen, Eunah Cho, Thanh-Le Ha,
Kevin Kilgour, Markus Müller, Matthias Sperber, Sebastian Stüker, Alex Waibel

Karlsruhe Institute of Technology
firstname.lastname@kit.edu

Abstract

Latency is one of the main challenges in the task of simultaneous spoken language translation. While significant improvements in recent years have led to high quality automatic translations, their usefulness in real-time settings is still severely limited due to the large delay between the input speech and the delivered translation.

In this paper, we present a novel scheme which reduces the latency of a large scale speech translation system drastically. Within this scheme, the transcribed text and its translation can be updated when more context is available, even after they are presented to the user. Thereby, this scheme allows us to display an initial transcript and its translation to the user with a very low latency. If necessary, both transcript and translation can later be updated to better, more accurate versions until eventually the final versions are displayed. Using this framework, we are able to reduce the latency of the source language transcript into half. For the translation, an average delay of 3.3s was achieved, which is more than twice as fast as our initial system.
Index Terms: speech translation, low-latency

1. Introduction

In our more and more globalized world we are witnessing both a growing opportunity and an increasing demand for communication among people speaking different languages. While in some cases this translation demand is met by human interpreters (e.g. in the European Parliament), this is not possible in many every day situations due to their costly nature and low availability.

Using modern machine translation (MT) and automatic speech recognition (ASR) technology, we can now build speech translation systems which can support people in these scenarios. Starting with domain dependent systems for very formalized dialogues in the 90s, recent systems are able to translate open-domain spontaneous speech. Such systems are currently employed to translate university lectures [1] or telephone conversations, for example the Skype translator [2]. Additionally to being affordable, these systems have the advantage of flexibly supporting multiple output modalities. For example, we can convert the translations back to audio using a text to speech component, display them as captions, or archive them for later use.

One of the most important challenges in speech translation is ensuring a low latency for both the transcription as well as the translation. Although the translation quality is often sufficient for communication, the user is required to wait until the translation is delivered. This greatly reduces the system's usefulness in practice [3]. In an interactive discussion it is important to be able to directly answer or comment on other participants, and in a lecture the speech should be in sync with the slides and gestures of the lecturer. Low latency systems can also be crucial when users have incomplete knowledge of the source language, since they can no longer combine their knowledge of the source language and delayed translation.

In this paper, we focus on live captions as the output modality. Since written captions can be dynamically changed, we propose updating initial sub-optimal outputs while the transcription and translation hypotheses stabilize over time during the search process, e.g., due to more context becoming available. As we apply updates from both the ASR and MT components, we can afford to directly display the current best translation at an early stage, with low latency. In general, later updates differ only slightly from the initial output. Using traditional approaches required balancing a trade off between latency and quality. With our new framework, however, we are able to display an initial output at a very low latency and a high quality output after a number of rewrites.

2. Related Work

Many previous works are dedicated to the relationship between latency and performance of speech translation. In [4], the authors investigated how utterance chunking influences the machine translation performance. In their work, they compared the machine translation performance for a given segment length. Therefore, they analyzed manual transcripts and on ASR output. They show that while chunking at the sentence boundaries is a reliable method, it is often not the best sentence boundary for the MT due to the segments' length.

An extensive study on the relationship between different segmentation strategies and their latency is made in [5]. In this work, the authors compared various segmentation schemes for ASR output, aiming for real-time translation applications. In their experiments, it was observed that a good performance can be achieved when a conjunction-based segmentation strategy is used in combination with a comma-based segmentation.

In [6], the authors developed and compared further algorithms for segmentation strategies. The authors showed that the greedy search and dynamic programming based methods, called Greedy-DP in their work, could successfully split source sentences into effective smaller units, while maintaining the translation performance. The authors in [7] extended this work further through a detailed study on segmentation optimization. In their work, the trade-off between latency and quality of spoken language translation was investigated. The potential drawback of the previously suggested Greedy-DP algorithm was that larger segments tend to be chosen, thereby increasing the latency in a real-time scenario. In order to address this issue,

This project has received funding from the European Unions Horizon 2020 Research and Innovation programme (call: H2020-ICT-2014-1, RIA) under grant agreement No 643950.

the authors of this work used an Pareto-optimal approach, in which latency is considered as an optimization parameter. By using this approach the authors achieved slightly better or similar translation quality while maintaining a relatively low latency.

Inspired by simultaneous interpretation, the authors in [8] proposed a new method of rewriting the reference translation, in order to reduce the latency caused by translation. The authors introduced a new reference translation where the word order is monotone to the order of source sentence. Using this reference translation, they aim to support good translations while producing them promptly. In this work, linguistically-driven rules are used to create a reference in which the word order is closer to the source language. In their Japanese to English translation experiments, where there is a substantially big difference in word order, the authors achieved better and faster translation.

A different approach to reduce the latency was introduced in [9]. Motivated by the run-on decoder for ASR systems, they introduce a stream-based decoder that does not wait for the next sentence boundary to translate the sentence. In contrast to our method, they need to modify the decoding algorithm heavily. Therefore, it is not straight forward to adapt this method to other translation systems.

3. Speech Translation Framework

Our speech translation framework employs a flexible service architecture with individual components like systems for ASR or MT [1]. In our terminology, these components are called "workers". Each worker registers on a central server, called the "mediator". The mediator keeps track of the individual workers and knows which type of service they offer (e.g., translating from German to English or performing speech recognition on an audio stream). Clients are able to connect to the mediator and query for certain services. The mediator then computes an appropriate path by connecting individual workers in the correct order. This architecture is independent of the services offered. It provides a generic communication framework where workers offer services. The different types of services are announced or queried by specifying certain "fingerprints" that identify the kind of service offered.

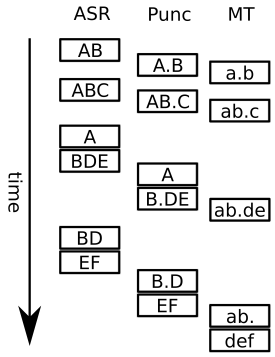
If a client asks for an English transcription of a German audio stream, the mediator then selects a German ASR worker and routes its output through an MT system for the translation from German to English and returns the English transcript. In addition to the workers for ASR and MT, our service architecture features another kind of worker for punctuation prediction.

4. Output Updates

Latency in a speech translation system is not mainly caused by long computation times, but is inevitably caused by models that require a certain amount of context. For instance, the statistical ASR framework is designed for decoding whole utterances. Only after all the data of one utterance has been processed the most probable hypothesis is decided on. The fact that the search algorithm uses a Viterbi beam-search as a heuristic allows it to output stable partial hypotheses after processing only parts of the data. The machine translation system is normally optimized to work on whole sentences. This means that the translation system can only generate a high-quality translation after the whole sentence is spoken and transcribed.

One possibility would be to develop a translation system that does not work on whole sentences. The disadvantage of this approach is that the translation of some words might depend on any word in the source sentence. Furthermore, most researchers work on the translation of whole sentences. New techniques in machine translation could therefore not be easily integrated into an MT system using an alternative scheme.

Figure 1: *Update Protocol*



4.1. Idea

We propose a framework that directly outputs the best current hypothesis at a current point. If a better translation is found later, we can replace the current translation to it.

Our strategy is inspired by human interpreters, who often produce the translation very quickly, but with the option of later altering it.

In our framework, the different components send messages $m(t_s, t_e, h)$ among each other. These contain the start time t_s, the end time t_e and the text hypothesis h. The only extension to the model is that we can replace the message with the latest start time t_s, by a new message $m'(t_s, t'_e, h')$. In this case the text hypothesis h will be replaced by the new text h'. All other segments before this segment are final and can no longer be replaced.

This small extension to the protocol allows us to correct errors made when directly outputting the current best hypothesis. Let us assume that we recognize the sequence *"A B C"* in the time span from t_1 to t_3. In this case we will generate the message $m(t_1, t_3, \text{“}ABC\text{”})$. Upon continuing recognition, we will be sure about A due to further contexts. Let us make another assumption that now C should be replaced by D with the following utterance E. In this case, the following two messages $m(t_1, t_2, \text{“}A\text{”})$ and $m(t_2, t_4, \text{“}BDE\text{”})$ will be generated. The messages between all three components for various examples are shown in Figure 1. The messages are shown according to the time they are sent.

For the ASR component, we added the possibility to send a non-stable, or unfixed, hypothesis. For the subsequent components, we used a mechanism in order to handle this flexible input as well as to output initial hypotheses.

4.2. Automatic Speech Recognition

In our framework, the ASR component was built to process an audio stream and output a stream of transcripts which is the input to the subsequent components. We use the Janus Recognition Toolkit (JRTk)[10] which features the IBIS decoder[11]. By using a dynamic decoding framework for speech recognition, we can avoid the detection of audio segments, and directly perform the decoding as soon as a small part of speech

Figure 2: *Algorithm to handle updates*

```
stable="";unstable="";
start=-1; end = -1
while(m = getMessage()):
  if (starttime(m) == start):
    unstable = text(m)
  else:
    stable.append(unstable)
    unstable = text(m)
    start = starttime(m)
  i = lastSentenceBoundary(stable)
  translate(stable[0:i])
  stable.remove(0,i)
  if(endtime(m) > end):
    t = join(stable,unstable)
    translate(t)
    end = endtime(m)
```

is recorded. However, as new speech is being processed, earlier parts of previously obtained results might change.

Since waiting until the end of the utterance for to output the ASR result leads to high latency, we output as soon as a part of the hypothesis becomes stable and will not change anymore, because competing search paths were pruned away in the beam-search.

With the update protocol in our framework, the ASR component can send and resend a hypothesis to the subsequent components if the hypothesis changes. When a stable part is found the ASR component sends it out with the corresponding time spans and then moves the start time span to the end of the stable hypothesis.

4.3. Machine Translation

In most speech translation systems the MT component waits until it receives the whole sentence and then translates the whole sentence. While this enables an easy integration of a standard statistical machine translation system, it adds a considerable latency to the system. The final translation will be generated only when the whole sentence is defined.

In order to reduce the latency of the MT component, a translation is generated directly if new words are received by the ASR component, not waiting for the whole sentence. When new words are added to the segment, the existing translation is replaced by the translation of this updated segment. In addition, we need to handle updates of the ASR and punctuation prediction module. A first step is to add a management component to the MT system. This component organizes which text will be translated and sends the complete sentences as well as the current unfinished segment to the MT component. This approach, however, significantly increases the number of sentences required to be translated.

An overview on the algorithm to handle the translation with later updates from the ASR component is shown in Figure 2. The main idea is to store the input in two stacks: one that is already fixed (stable) and another one which can still be updated (unstable). Depending on the incoming message, the unstable stack is updated or marked as stable stack. In the latter case, the incoming message is added into a new unstable stack. First, all sentences for which no more updates are expected are translated. Finally, the unfinished segment of the fixed stack and the temporary input are translated. The translation of all parts is then sent out. The punctuation and segmentation is applied before the MT component, in the similar method. It will be described in Section 4.4.

In cases like the previously discussed example, we would first delete the translation of *B D E* when receiving the message "A", and then we would regenerate this translation. Therefore, a new translation is generated when receiving a message with a newer end time t_e.

This technique has an advantage that it is possible to generate a translation instantly when new words are recognized by the ASR component. The re-translation of the last sentence, however, also dramatically increases the number of words that need to be re-translated. Since this leads to an issue in real-time scenarios, we devise a scheme to save intermediate results of translation to speed up the general translation process.

For the translation of the unfinished segment, we store the last translation. Also, some of the last words are removed, which are less reliably estimated due to missing future context. The last two source words and their translation are considered for this task. If the beginning of the input segment is the same for the updated translation, the existing translation of this part is reused. A new translation is performed only on the remaining parts. This scheme is implemented by adding dynamic phrase pair entries to the MT system.

4.4. Punctuation

The punctuation prediction component needs to handle unstable input, generating the output hypotheses consisting of stable and unstable stacks correctly. Since the computation for this part is not as demanding as for MT, it was not required to modify the actual component. In our experiments we inserted punctuation marks based on language model probabilities and pause information [1].

5. Evaluation

The proposed framework does not change the final hypothesis, but the time in which it is delivered to the user. Therefore, we evaluated the latency by measuring the elapsed time between the spoken words and when they are displayed. In our evaluation we have for every message the time stamps t_s, t_e, t_r, where t_s and t_e are the start and end times of the message and t_r is the time the message is received by the display client. Their order is therefore always $t_s < t_e < t_r$. First, we calculated the average delay $d(t_s^i, t_e^i, t_r^i)$ before the first output of m^i.

$$d(t_s^i, t_e^i, t_r^i) = t_r^i - \frac{t_s^i + t_e^i}{2} \tag{1}$$

If the m^{i+1} overwrites the text of message m^i, this only adds a delay for the new transcript between time t_e^i and t_e^{i+1}. This delay is then calculated as

$$\begin{aligned} &d(t_s^i, t_e^i, t_r^i) = (t_r^i - \frac{max(t_s^i, t_e^{i-1}) + t_e^i}{2}) \\ &\text{s.t. } max(t_s^i, t_e^{i-1}) < t_e^i \end{aligned} \tag{2}$$

Since the messages have different length, we calculate the average delay as the weighted sum over all delays

$$D = \frac{\sum_{i=1}^{m} d(t_s^i, t_e^i, t_r^i) * (max(t_s^i, t_e^{i-1}) - t_e^i)}{\sum_{\substack{i=1 \\ \text{s.t. } max(t_s^i, t_e^{i-1}) < t_e^i}}^{m} (max(t_s^i, t_e^{i-1}) - t_e^i)} \tag{3}$$

While the first output is a very important clue to the system's performance, it should not be the only measure. Otherwise, a system that always directly generates mediocre output

can be considered as a perfect system. Therefore, we measure the delay of the final hypothesis as well. A word is defined as *final* if the word and all its preceding words are not changed by any of the following messages. We calculated for every message m the words $w_j, \ldots, w_{j'}$ that are finalized by this message. This does not mean that the system has marked them already as stable. Afterwards, we calculated the start and end times t^i_{s*} and t^i_{e*} of the word sequence $w_j, \ldots, w_{j'}$. Since we only have the start and end times of the whole message, we approximated those times assuming a linear correlation between the elapsed time and the number of characters. Then we can calculate the average delay for the final hypothesis similar to the first output as follows:

$$D = \frac{\sum_{i=1}^{m} d(t^i_{s*}, t^i_{e*}, t^i_r) * (t^i_{s*} - t^i_{e*})}{\sum_{i=1}^{m} t^i_{s*} - t^i_{e*}} \quad (4)$$

6. Experiments

We evaluated the delay of the proposed framework for the language directions English→French and German→English.

6.1. System Description

We tested the framework using a lecture translation system consisting of an ASR, punctuation prediction and MT component. Speech recognition is performed in an online setup as described above. The MT components are trained on 1.8 million sentences of parallel data, including EPPS, NC and in-domain data. Before the training, preprocessing is applied to the parallel data. A phrase-based decoder [13] using several advanced models including domain adaptation, bilingual and cluster language models as well as Discriminative Word Lexica is used to generate the translation. The different word order between the languages is modeled using POS-based reordering [14, 15]. A detailed description can be found in [1]. The overall system is similar to the best performing system in the IWSLT SLT evalutation [**?**].

6.2. Results

The results of applying the suggested scheme on translating English to French and German to English IWSLT 2013 test sets are shown in Table 1. Each system setting is compared in two conditions: the first transcript (left column) and the final transcript (right column). Table 1 show the latency for the transcript in the source language. The baseline system (static) without any modifications has a delay of 4.9 s and 5.7 s (the latency for the initial and the final hypothesis of course being the same). Outputting preliminary sentence segmentations reduces the delay to 3.3 s for the initial and 3.5 s for the final output in English to French and 3.8 s and 3.9 s for the other language pair, reducing the latency by up to 1.8 s. By using an ASR system that sends frequent partial hypotheses we are able to reduce the latency by a further 1 s and 1.7 s respectively.

Table 1: *Latency for speech translation in seconds*

	System	EN-FR		DE-EN	
ASR	static	4.9	4.9	5.7	5.7
	+ dynamic Seg	3.3	3.5	3.8	3.9
	+ dynamic ASR	1.7	2.3	1.6	2.2
SMT	static	7.5	7.5	8.6	8.6
	+ dynamic MT	5.0	5.4	5.9	5.7
	+ dynamic Seg	3.4	3.3	4.0	5.3

In summary, the new method is able to reduce the latency of the initial output by 4 s and of the final output by 3.5 s. This means that we are able to reduce the waiting time for the transcript by more than a factor two. The improvements in German are higher than the ones in English.

Looking at the latency for the translation, the unmodified baseline system delivers the translation with an average delay of 7.5 s and 8.6 s. An MT system that directly outputs the current best translation and later corrects wrongly translated words improves the latency by 2 s. Integrating our modified segmentation and ASR systems each improve the latency further by roughly one second for English to French. For German to English the latency could be improved by 1.4 s.

In total, the latency of the first system could be reduced from 7.5 s to 1.8 s for the initial output and to 3.3 s for the final output. This means that the initial translation is delivered more than 4 times faster than in the baseline system and the final one is presented more than twice as fast. As can be seen in the table, these improvements are consistent over all talks.

Compared to the English to French system, a similar latency for the initial output is achieved, while the latency for the final output is 2 s larger than that of the English to French system. We believe that this bigger latency is caused by longer reorderings in this language pair. Since the German verb is often at the end of the sentence, we can only generate the correct beginning of the English sentence, when we have recognized the whole German sentence.

6.3. Analysis

In order to compare our approach to traditional ideas of reducing the latency, we performed an additional experiment on a subset of five German to English TEDx talks. In this experiment, we measure the latency and the performance of the MT component. While not allowing updates, partial sentences are translated if they exceed n words. Some segments might still be longer, since we do not receive the output of the ASR system word by word. The results for $n \in \{1, 2, 5, 10\}$ are shown in Table 2. Same as before, the latency is reported in seconds.

Table 2: *Latency and performance comparison to a sentence length based model*

n	1	2	5	10	∞	Dynamic
Latency	5.3	5.4	6.0	7.3	7.9	**6.0**
BLEU	8.5	9.3	10.2	11.2	11.4	**11.4**

From this result we can see that we can achieve the same latency as the system that translates every hypothesis that extends five words, while obtaining the same translation performance as the system translating always whole sentences. Our system using the proposed update scheme outperforms the system with the same latency by 1.2 BLEU points.

7. Conclusion

In this paper we presented a framework that substantially reduces the latency of a speech translation system maintaining the performance. By delivering unstable hypothesis and allowing for later updates, we reduce its latency of the final output by half. At the same time, it is possible to present an initial translation to the user with a latency of less than two seconds, while only minimal changes are required to both the ASR and MT components. While the current system mainly focuses on reducing the latency, we plan to analyze the trade off between the number of updates and the latency of the system in the future.

8. References

[1] E. Cho, C. Fügen, T. Herrmann, K. Kilgour, M. Mediani, C. Mohr, J. Niehues, K. Rottmann, C. Saam, S. Stüker *et al.*, "A real-world System for Simultaneous Translation of german Lectures." in *INTERSPEECH*, 2013, pp. 3473–3477.

[2] W. D. Lewis, "Skype translator: Breaking down language and hearing barriers," 2015.

[3] M. Müller, S. Fünfer, S. Stüker, and A. Waibel, "Evaluation of the KIT Lecture Translation System," in *LREC*, 2016.

[4] C. Fügen and M. Kolss, "The Influence of Utterance Chunking on Machine Translation Performance," in *Proceedings of the eighth Annual Conference of the International Speech Communication Association (INTERSPEECH 2007)*, Antwerp, Belgium, 2007, pp. 2837–2840.

[5] 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 (NAACL-HLT 2013)*, Atlanta, Georgia, USA, 2013, pp. 230–238.

[6] Y. Oda, G. Neubig, S. Sakti, 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 (ACL 2014)*, Baltimore, Maryland, USA, 2014.

[7] H. S. Shavarani, M. Siahbani, R. M. Seraj, and A. Sarkar, "Learning Segmentations that Balance Latency versus Quality in Spoken Language Translation," in *Proceedings of the Eleventh International Workshop on Spoken Language Translation (IWSLT 2015)*, Da Nang, Vietnam, 2015.

[8] H. He, I. Alvin Grissom, J. Boyd-Graber, J. B. Graber, and H. Daumé III, "Syntax-based Rewriting for Simultaneous Machine Translation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, Lisbon, Portugal, 2015.

[9] M. Kolss, S. Vogel, and A. Waibel, "Stream Decoding for Simultaneous Spoken Language Translation," in *INTERSPEECH*. ISCA, 2008, pp. 2735–2738.

[10] M. Woszczyna, N. Aoki-Waibel, F. D. Bu, N. Coccaro, K. Horiguchi, T. Kemp, A. Lavie, A. McNair, T. Polzin, I. Rogina, C. Rose, T. Schultz, B. Suhm, M. Tomita, and A. Waibel, "JANUS 93: Towards Spontaneous Speech Translation," in *International Conference on Acoustics, Speech, and Signal Processing 1994*, Adelaide, Australia, 1994.

[11] H. Soltau, F. Metze, C. Fugen, and A. Waibel, "A one-pass decoder based on polymorphic linguistic context assignment," in *Automatic Speech Recognition and Understanding, 2001. ASRU'01. IEEE Workshop on*. IEEE, 2001, pp. 214–217.

[12] S. Vogel, "SMT Decoder Dissected: Word Reordering." in *Proceedings of the International Conference on Natural Language Processing and Knowledge Engineering*, Beijing, China, 2003.

[13] K. Rottmann and S. Vogel, "Word Reordering in Statistical Machine Translation with a POS-Based Distortion Model," in *Proceedings of the International Conference on Theoretical and Methodological Issues in Machine Translation (TMI 2007)*, Skövde, Sweden, 2007.

[14] J. Niehues and M. Kolss, "A POS-Based Model for Long-Range Reorderings in SMT," in *Proceedings of the Workshop on Statistical Machine Translation*, ser. WMT 2009, Athens, Greece, 2009.

[15] M. Cettolo, J. Niehues, S. Stker, L. Bentivogli, R. Cattoni, and M. Federico, "The IWSLT 2015 Evaluation Campaign," in *Proceedings of the 12th Workshop on Spoken Language Translation(IWST*, 2015.