
Dynamic Masking for Improved Stability in Online Spoken Language Translation

Yuekun Yao
Barry Haddow

ykyao.cs@gmail.com
bhaddow@staffmail.ed.ac.uk

School of Informatics, University of Edinburgh, 10 Crichton Street, Edinburgh, Scotland

Abstract

For spoken language translation (SLT) in live scenarios such as conferences, lectures and meetings, it is desirable to show the translation to the user as quickly as possible, avoiding an annoying lag between speaker and translated captions. In other words, we would like low-latency, online SLT. If we assume a pipeline of automatic speech recognition (ASR) and machine translation (MT) then a simple but effective approach to online SLT is to pair an online ASR system, with a *retranslation strategy*, where the MT system retranslates every update received from ASR. However this can result in annoying “flicker” as the MT system updates its translation. A possible solution is to add a fixed delay, or “mask” to the the output of the MT system, but a fixed global mask re-introduces undesirable latency to the output. We introduce a method for dynamically determining the mask length, which provides a better latency–flicker trade-off curve than a fixed mask, without affecting translation quality.

1 Introduction

A common approach to Spoken Language Translation (SLT) is to use a cascade (or pipeline) consisting of automatic speech recognition (ASR) and machine translation (MT). In a live translation setting, such as a lecture or conference, we would like the transcriptions or translations to appear as quickly as possible, so that they do not “lag” noticeably behind the speaker. In other words, we wish to minimise the latency of the system. Many popular ASR toolkits can operate in an *online* mode, where the transcription is produced incrementally, instead of waiting for the speaker to finish their utterance. However online MT is less well supported, and is complicated by the reordering which is often necessary in translation, and by the use of encoder-decoder models which assume sight of the whole source sentence.

Some systems for online SLT rely on the *streaming* approach to translation, perhaps inspired by human interpreters. In this approach, the MT system is modified to translate incrementally, and on each update from ASR it will decide whether to update its translation, or wait for further ASR output (Cho and Esipova, 2016; Ma et al., 2019; Zheng et al., 2019a,b; Arivazhagan et al., 2019).

The difficulty with the streaming approach is that the system has to choose between committing to a particular choice of translation output, or waiting for further updates from ASR, and does not have the option to revise an incorrect choice. Furthermore, all the streaming approaches referenced above require specialised training of the MT system, and modified inference algorithms.

To address the issues above, we construct our online SLT system using the *retranslation* approach (Niehues et al., 2018; Arivazhagan et al., 2020a), which is less studied but

more straightforward. It can be implemented using any standard MT toolkit (such as Marian (Junczys-Dowmunt et al., 2018) which is highly optimised for speed) and using the latest advances in text-to-text translation. The idea of retranslation is that we produce a new translation of the current sentence every time a partial sentence is received from the ASR system. Thus, the translation of each sentence prefix is independent and can be directly handled by a standard MT system. Arivazhagan et al. (2020b) directly compared the retranslation and streaming approaches and found the former to have a better latency–quality trade-off curve.

When using a completely unadapted MT system with the retranslation approach, however, there are at least two problems we need to consider:

1. MT training data generally consists of full sentences, and systems may perform poorly on partial sentences
2. When MT systems are asked to translate progressively longer segments of the conversation, they may introduce radical changes in the translation as the prefixes are extended. If these updates are displayed to the user, they will introduce an annoying “flicker” in the output, making it hard to read.

We illustrate these points using the small example in Figure 1. When the MT system receives the first prefix (“Several”) it attempts to make a longer translation, due to its bias towards producing sentences. When the prefix is extended (to “Several years ago”), the MT system completely revises its original translation hypothesis. This is caused by the differing word order between German and English.

Several	→	Mehrere Male <i>Several times</i>
Several years ago	→	Vor einigen Jahre <i>Several years ago</i>

Figure 1: Sample translation with standard en→de MT system. We show the translation output, and its back-translation into English.

The first problem above could be addressed by simply adding sentence prefixes to the training data of the MT system. In our experiments we found that using prefixes in training could improve translation of partial sentences, but required careful mixing of data, and even then performance of the model trained on truncated sentences was often worse on full sentences.

A way to address both problems above is with an appropriate retranslation strategy. In other words, when the MT system receives a new prefix, it should decide whether to transmit its translation in full, partially, or wait for further input, and the system can take into account translations it previously produced. A good retranslation strategy will address the second problem above (too much flickering as translations are revised) and in so doing so address the first (over-eagerness to produce full sentences).

In this paper, we focus on the retranslation methods introduced by Arivazhagan et al. (2020a) – mask- k and biased beam search. The former is a delayed output strategy which does not affect overall quality, reduces flicker, but can significantly increase the latency of the translation system. The latter alters the beam search to take into account the translation of the previous prefix, and is used to reduce flicker without influencing latency much, but can also damage translation quality.

Our contribution in this paper is to show that by using a straightforward method to predict how much to mask (i.e. the value of k in the mask- k strategy) we obtain a more optimal trade-off of flicker and latency than is possible with a fixed mask. We show that for many prefixes

the mask can be safely reduced. We achieve this by having the system make probes of possible extensions to the source prefix, and observing how stable the translation of these probes is – instability in the translation requires a larger mask. Our method requires no modifications to the underlying MT system, and has no effect on translation quality.

2 Related Work

Early work on incremental MT used prosody (Bangalore et al., 2012) or lexical cues (Rangarajan Sridhar et al., 2013) to make the translate-or-wait decision. The first work on incremental neural MT used confidence to decide whether to wait or translate (Cho and Esipova, 2016), whilst in (Gu et al., 2017) they learn the translation schedule with reinforcement learning. In Ma et al. (2019), they address simultaneous translation using a transformer (Vaswani et al., 2017) model with a modified attention mechanism, which is trained on prefixes. They introduce the idea of wait- k , where the translation does not consider the final k words of the input. This work was extended by Zheng et al. (2019b,a), where a “delay” token is added to the target vocabulary so the model can learn when to wait, through being trained by imitation learning. The MILk attention (Arivazhagan et al., 2019) also provides a way of learning the translation schedule along with the MT model, and is able to directly optimise the latency metric.

In contrast with these recent approaches, retranslation strategies (Niehues et al., 2018) allow the use of a standard MT toolkit, with little modification, and so are able to leverage all the performance and quality optimisations in that toolkit. Arivazhagan et al. (2020a) pioneered the retranslation system by combining a strong MT system with two simple yet effective strategies: biased beam search and mask- k . Their experiments show that the system can achieve low flicker and latency without losing much performance. In an even more recent paper, Arivazhagan et al. (2020b) further combine their retranslation system with prefix training and make comparison with current best streaming models (e.g. MILk and wait- k models), showing that such a retranslation system is a strong option for online SLT.

3 Retranslation Strategies

Before introducing our approach, we describe the two retranslation strategies introduced in Arivazhagan et al. (2020a): mask- k and biased beam search.

The idea of mask- k is simply that the MT system does not transmit the last k tokens of its output – in other words it masks them. Once the system receives a full sentence, it transmits the translation in full, without masking. The value of k is set globally and can be tuned to reduce the amount of flicker, at the cost of increasing latency. Arivazhagan et al. (2020a) showed good results for a mask of 10, but of course for short sentences a system with such a large mask would not produce any output until the end of the sentence.

In biased beam search, a small modification is made to the translation algorithm, changing the search objective. The technique aims to reduce flicker by ensuring that the translation produced by the MT system stays closer to the translation of the previous (shorter) prefix. Suppose that S is a source prefix, S' is the extension of that source prefix provided by the ASR, and T is the translation of S produced by the system (after masking). Then to create the translation T' of S' , biased beam search substitutes the model probability $p(t'_i|t'_{<i}, S')$ with the following expression:

$$p^B(t'_i|t'_{<i}, S') = (1 - \beta) \cdot p(t'_i|t'_{<i}, S') + \beta \cdot \delta(t'_i, t_i)$$

where t'_i is the i^{th} token of the translation hypothesis T' , and β is a weighting which we set to 0 when $t_{<i} \neq t'_{<i}$. In other words, we interpolate the translation model with a function that keeps it close to the previous translation, but stop applying the biasing once the new translation diverges from the previous one.

As we noted earlier, biased beam search can degrade the quality of the translation, and we show experiments to illustrate this in Section 6.1. We also note that biased beam search assumes that the ASR simply extends its output each time it updates, when in fact ASR systems may rewrite their output. Furthermore, biased beam search requires modification of the underlying inference algorithm (which in the case of Marian is written in highly optimised, hand-crafted GPU code), removing one of the advantages of the retranslation approach (that it can be easily applied to standard MT systems).

4 Dynamic Masking

In this section we introduce our improvement to the mask- k approach, which uses a variable mask, that is set at runtime. The problem with using a fixed mask, is that there are many time-steps where the system is unlikely to introduce radical changes to the translation as more source is revealed, and on these occasions we would like to use a small mask to reduce latency. However the one-size-fits-all mask- k strategy does not allow this variability.

The main idea of dynamic masking is to *predict* what the next source word will be, and check what effect this would have on the translation. If this changes the translation, then we mask, if not we output the full translation.

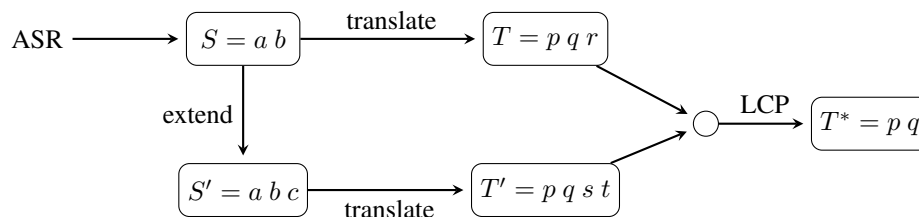


Figure 2: The source prediction process. The string $a b$ is provided by the ASR system. The MT system then produces translations of the string and its extension, compares them, and outputs the longest common prefix (LCP)

More formally, we suppose that we have a source prefix $S = s_1 \dots s_p$, a source-to-target translation system, and a function $pred_k$, which can predict the next k tokens following S . We translate S using the translation system to give a translation hypothesis $T = t_1 \dots t_q$. We then use $pred_k$ to predict the tokens following s_p in the source sentence to give an extended source prefix $S' = s_1 \dots s_p s_{p+1} \dots s_{p+k}$, and translate this to give another translation hypothesis T' . Comparing T and T' , we select the longest common prefix T^* , and output this as the translation, thus masking the final $|T| - |T^*|$ tokens of the translation. If S is a complete sentence, then we do not mask any of the output, as in the mask- k strategy. The overall procedure is illustrated in Figure 2.

In fact, after initial experiments, we found it was more effective to refine our strategy, and not mask at all if the translation after dynamic mask is a prefix of the previous translation. In this case we directly output the last translation. In other words, we do not mask if $is_prefix(T_i^*, T_{i-1}^*)$ but instead output T_{i-1}^* again, where T_i^* denotes the masked translation for the i th ASR input. We also notice that this refinement does not give any benefit to the mask- k strategy in our experiments. The reason that this refinement is effective is that the translation of the extended prefix can sometimes exhibit instabilities early in the sentence (which then disappear in a subsequent prefix). Applying a large mask in response to such instabilities increases latency, so we effectively “freeze” the output of the translation system until the instability is resolved. An example to illustrate this refinement can be found in Figure 10.

To predict the source extensions (i.e. to define the $pred_k$ function above), we first tried using a language model trained on the source text. This worked well, but we decided to add two simple strategies in order to see how important it was to have good prediction. The extension strategies we include are:

lm-sample We sample the next token from a language model (LSTM) trained on the source-side of the parallel training data. We can choose n possible extensions by choosing n distinct samples.

lm-greedy This also uses an LM, but chooses the most probable token at each step.

unknown We extend the source sentence using the UNK token from the vocabulary.

random We extend by sampling randomly from the vocabulary, under a uniform distribution. As with lm-sample, we can generalise this strategy by choosing n different samples.

5 Evaluation of Retranslation Strategies

Different approaches have been proposed to evaluate online SLT, so we explain and justify our approach here. We follow previous work on retranslation (Niehues et al., 2018; Arivazhagan et al., 2020a,b) and consider that the performance of online SLT should be assessed according to three different aspects – quality, latency and flicker. All of these aspects are important to users of online SLT, but improving on one can have an adverse effect on other aspects. For example outputting translations as early as possible will reduce latency, but if these early outputs are incorrect then either they can be corrected (increasing flicker) or retained as part of the later translation (reducing quality). In this section we will define precisely how we measure these system aspects. We assume that selecting the optimal trade-off between quality, latency and flicker is a question for the system deployer, that can only be settled by user testing.

Latency The latency of the MT system should provide a measure of the time between the MT system receiving input from the ASR, and it producing output that can be potentially be sent to the user. A standard (text-to-text) MT system would have to wait until it has received a full sentence before it produces any output, which exhibits high latency.

We follow Ma et al. (2019) by using a latency metric called average lag (AL), which measures the degree to which the output lags behind the input. This is done by averaging the difference between the number of words the system has output, and the number of words expected, given the length of the source prefix received, and the ratio between source and target length. Formally, AL for source and target sentences S and T is defined as:

$$AL(S, T) = \frac{1}{\tau} \sum_{t=1}^{\tau} g(t) - \frac{(t-1)|S|}{|T|}$$

where τ is the number of target words generated by the time the whole source sentence is received, $g(t)$ is the number of source words processed when the target hypothesis first reaches a length of t tokens. In our implementation, we calculate the AL at token (not subword) level with the standard tokenizer in sacreBLEU (Post, 2018), meaning that for Chinese output we calculate AL on characters.

This metric differs from the one used in Arivazhagan et al. (2020a)¹, where latency is defined as the mean time between a source word being received and the translation of that source word being finalised. We argue against this definition, because it conflates latency and

¹In the presentation of this paper at ICASSP, the authors used a latency metric similar to the one used here, and different to the one they used in the paper

flicker, since outputting a translation and then updating is penalised for both aspects. The update is penalised for flicker since the translation is updated (see below) and it is penalised for latency, since the timestamp of the initial output is ignored in the latency calculation.

Flicker The idea of flicker is to obtain a measure of the potentially distracting changes that are made to the MT output, as its ASR-supplied source sentence is extended. We assume that straightforward extensions of the MT output are fine, but changes which require re-writing of part of the MT output should result in a higher (i.e. worse) flicker score. Following Arivazhagan et al. (2020a), we measure flicker using the normalised erasure (NE), which is defined as the minimum number of tokens that must be erased from each translation hypothesis when outputting the subsequent hypothesis, normalised across the sentence. As with AL, we also calculate the NE at token level for German, and at character level for Chinese.

Quality As in previous work Arivazhagan et al. (2020a), quality is assessed by comparing the full sentence output of the system against a reference, using a sentence similarity measure such as BLEU. We do not evaluate quality on prefixes, mainly because of the need for a heuristic to determine partial references. Further, quality evaluation on prefixes will conflate with evaluation of latency and flicker and thus we simply assume that if the partial sentences are of poor quality, that this will be reflected in the other two metrics (flicker and latency). Note that our proposed dynamic mask strategy is only concerned with improving the flicker–latency trade-off curve and has no effect on full-sentence quality, so MT quality measurement is not the focus of this paper. Where we do require a measure of quality (in the assessment of biased beam search, which does change the full-sentence translation) we use BLEU as implemented by sacreBLEU (Post, 2018).

6 Experiments

6.1 Biased Beam Search and Mask- k

We first assess the effectiveness of biased beam search and mask- k (with a fixed k), providing a more complete experimental picture than in Arivazhagan et al. (2020a), and demonstrating the adverse effect of biased beam search on quality. For these experiments we use data released for the IWSLT MT task (Cettolo et al., 2017), in both English→German and English→Chinese. We consider a simulated ASR system, which supplies the gold transcripts to the MT system one token at a time².

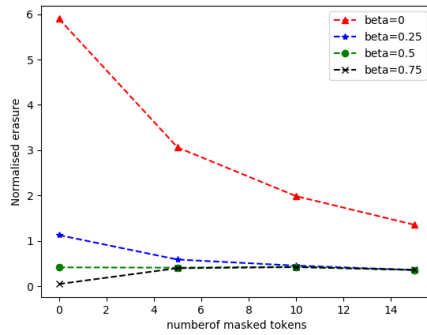
For training we use the TED talk data, with dev2010 as heldout and tst2010 as test set. The raw data set sizes are 206112 sentences (en-de) and 231266 sentences (en-zh). We preprocess using the Moses (Koehn et al., 2007) tokenizer and truecaser (for English and German) and jieba³ for Chinese. We apply BPE (Sennrich et al., 2016) jointly with 90k merge operations. For our MT system, we use the transformer-base architecture (Vaswani et al., 2017) as implemented by Nematus (Sennrich et al., 2017). We use 256 sentence mini-batches, and a 4000 iteration warm-up in training.

As we mentioned in the introduction, we did experiment with prefix training (using both alignment-based and length-based truncation) and found that this improved the translation of prefixes, but generally degraded translation for full sentences. Since prefix translation can also be improved using the masking and biasing techniques, and the former does not degrade full sentence translation, we only include experimental results when training on full sentences.

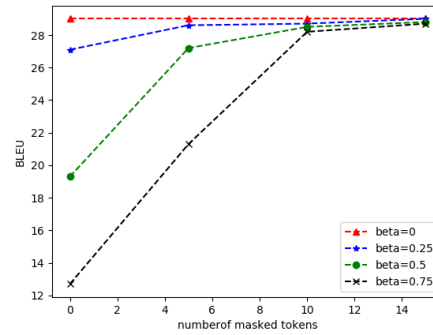
²A real online ASR system typically increments its hypothesis by adding a variable number of tokens in each increment, and may revise its hypothesis. Also, ASR does not normally supply sentence boundaries, or punctuation, and these must be added by an intermediate component. Sentence boundaries may change as the ASR hypothesis changes. In this work we make simplifying assumptions about the nature of the ASR, in order to focus on retranslation strategies, leaving the question of dealing with real online ASR to future work.

³<https://github.com/fxsjy/jieba>

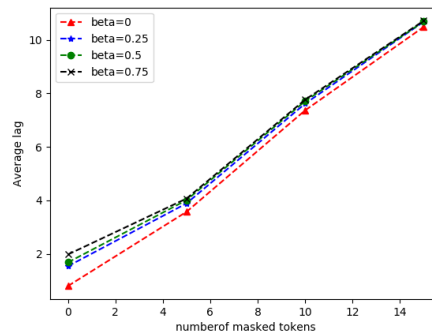
In Figure 3 we show the effect of varying β and k on our three evaluation measures, for English→German.



(a) Masking versus flicker (measured by erasure)



(b) Masking versus quality (measured by BLEU).



(c) Masking versus latency (measured by average lagging).

Figure 3: Effect of varying mask- k , at different values of the biased beam search interpolation parameter, on the three measures proposed in Section 5.

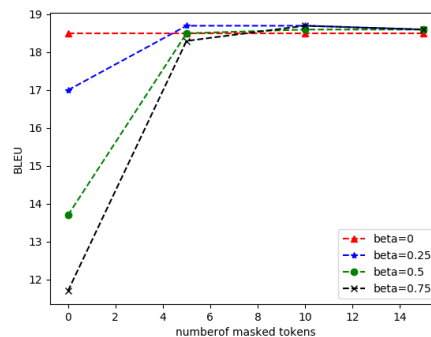
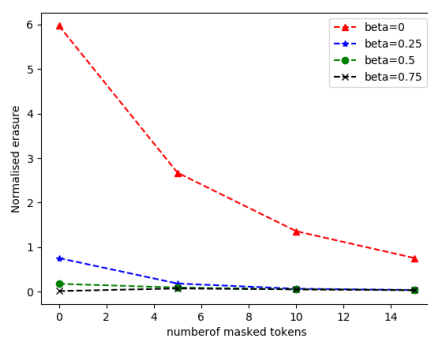
Looking at Figure 3(a) we notice that biased beam search has a strong impact in reducing flicker (erasure) at all values of β . However the problem with this approach is clear in Figure 3(b), where we can see the reduction in BLEU caused by this biasing. This can be offset by increasing masking, also noted by Arivazhagan et al. (2020a), but as we show in Figure 3(c) this comes at the cost of an increase in latency.

Our experiments with en→zh show a roughly similar pattern, as shown in Figure 4. We find that lower levels of masking are required to reduce the detrimental effect on BLEU of the biasing, but latency increases more rapidly with masking.

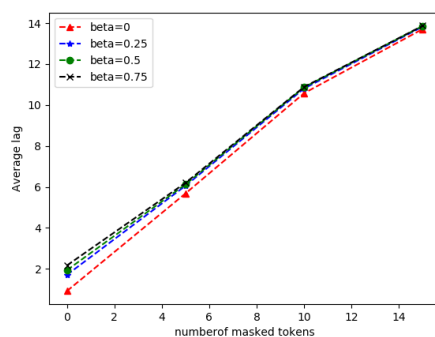
6.2 Dynamic Masking

We now turn our attention to the dynamic masking technique introduced in Section 4. We use the same data sets and MT systems as in the previous section. To train the LM, we use the source side of the parallel training data, and train an LSTM-based LM.

To assess the performance of dynamic masking, we measure latency and flicker as we vary the length of the source extension (k) and the number of source extensions (n). We consider the 4 different extension strategies described at the end of Section 4. We do not show translation



(a) Masking versus flicker (measured by erasure) (b) Masking versus quality (measured by BLEU).

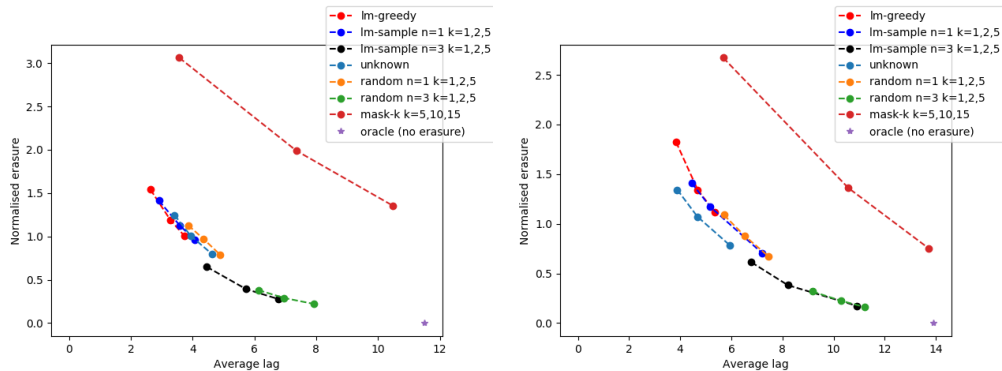


(c) Masking versus latency (measured by average lagging).

Figure 4: Effect of varying mask- k , at different values of the biased beam search interpolation parameter, on English-to-Chinese corpus

quality since the translation of the complete sentence is unaffected by the dynamic masking strategy. The results for both $en \rightarrow de$ and $en \rightarrow zh$ are shown in Figure 5, where we compare to the strategy of using a fixed mask- k . The oracle data point is where we use the full-sentence translations to set the mask so as to completely avoid flicker.

We observe from Figure 5 that our dynamic mask mechanism improves over the fixed mask in all cases, Pareto-dominating it on latency and flicker. Varying the source prediction strategy and parameters appears to preserve the same inverse relation between latency and flicker, although offering a different trade-off. Using several random source predictions (the green curve in both plots) offers the lowest flicker, at the expense of high latency, whilst the shorter LM-based predictions show the lowest latency, but higher flicker. We hypothesise that this happens because extending the source randomly along several directions, then comparing the translations is most likely to expose any instabilities in the translation and so lead to a more conservative masking strategy (i.e. longer masks). In contrast, using a single LM-based predictions gives more plausible extensions to the source, making the MT system more likely to agree on its translations of the prefix and its extension, and so less likely to mask. Using several LM-based samples (the black curve) adds diversity and gives similar results to random (except that random is quicker to execute). The pattern across the two language pairs is similar, although we observe a more dispersed picture for the $en \rightarrow zh$ results.



(a) Dynamic Masking for en→de MT system trained on full sentences (measured by AL and NE) (b) Dynamic Masking for en→zh MT system trained on full sentences (measured by AL and NE)

Figure 5: Effect of dynamic mask with different source prediction strategies. The strategies are explained in Section 3, and the oracle means that we use the full-sentence translation to set the mask.

We note that, whilst Figure 5 shows that our dynamic mask approach Pareto-dominates the mask- k , it does not tell the full story since it averages both the AL and NE across the whole corpus. To give more detail on the distribution of predicted masks, we show histograms of all masks on all prefixes for two of the source prediction strategies in Figure 6. We select the source extension strategies at opposite ends of the curve, i.e. the highest AL / lowest NE strategy and the lowest AL / highest NE strategy, and show results from en→de. We can see from the graphs in Figure 6 that the distribution across masks is highly skewed, with majority of prefixes having low masks, below the mean. This is in contrast with the mask- k strategy where the mask has a fixed value, except at the end of a sentence where there is no mask. Our strategy is able to predict, with a reasonable degree of success, when it is safe to reduce the mask.

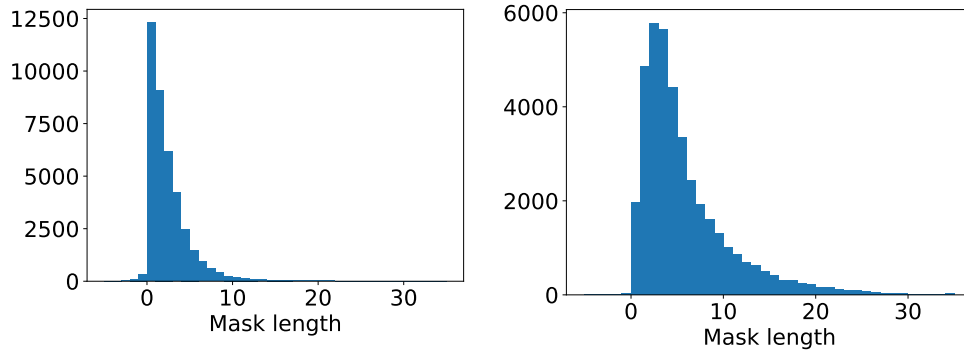
In order to provide further experimental verification of our retranslation model, we apply the strategy to a larger-scale English→Chinese system. Specifically, we use a model that was trained on the entire parallel training data for the WMT20 en-zh task⁴, in addition to the TED corpus used above. For the larger model we prepared as before, except with 30k BPE merges separately on each language, and then we train a transformer-regular using Marian. By using this, we show that our strategy is able to be easily slotted in a different model from a different translation toolkit. The results are shown in Figure 7. We can verify that the pattern is unchanged in this setting.

We illustrate the improved stability provided by our masking strategy, by drawing three examples from the English→German system’s translation of our test set, in Figures 8–10. In each example we show the source prefix (as provided by the ASR), its translation, its extension (in these examples, provided by the *lm-greedy* strategy with $n = 1, k = 1$), the translation of its extension, and the output after applying the dynamic mask.

In the first example (Figure 8) the MT system produces quite unstable translations of short prefixes, but the dynamic masking algorithm detects this instability and applies short masks to cover it up. Since the translation of the prefix and the translation of its extension differ, a short mask is applied and the resulting translation avoids erasure.

In the second example in Figure 9, there is genuine uncertainty in the translation because

⁴www.statmt.org/wmt20/translation-task.html



(a) Masks for *lm-greedy* strategy, $n = 1, k = 1$. (b) Masks for *random* strategy, $n = 3, k = 5$.

Figure 6: Histograms of mask lengths over all prefixes for *en – de* test set. The strategy on the left has low latency but higher erasure ($AL=1.54, NE=3.83$) whereas the one on the right has low erasure but higher latency ($AL=11.2, NE=0.22$). In both cases we see that the vast majority of prefixes have low masks; in other words the skewed distribution means most are below the mean.

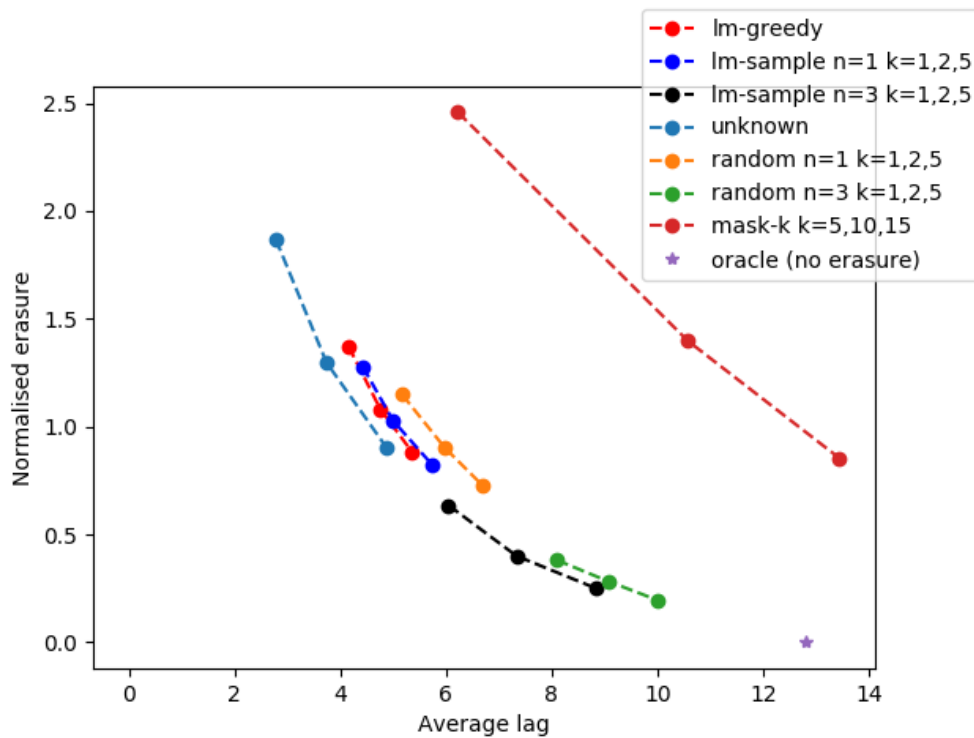


Figure 7: Effect of dynamic mask with different strategies, applied to a larger-scale *en*→*zh* system.

	Source	Translation	Output
prefix	Here	Hier sehen sie es	
extension	Here's	Hier ist es	Hier
prefix	Here are	Hier sind sie	
extension	Here are some	Hier sind einige davon	Hier sind
prefix	Here are two	Hier sind zwei davon	
extension	Here are two things	Hier sind zwei Dinge	Hier sind zwei
prefix	Here are two patients	Hier sind zwei Patienten	
extension	Here are two patients.	Hier sind zwei Patienten.	Hier sind zwei Patienten

Figure 8: In this example, the dynamic mask covers up the fact that the MT system is unstable on short prefixes. Using the translation of the LM's predicted extension of the source, the system can see that the translation is uncertain, and so applies a mask.

	Source	Translation	Output
prefix	But you know	Aber Sie wissen es	
extension	But you know,	Aber wissen Sie, sie wissen schon	Aber
prefix	But you know what?	Aber wissen Sie was?	Aber wissen Sie was?

Figure 9: Here the dynamic masking is able to address the initial uncertainty between a question and statement, which can have different word orders in German. The translation of the initial prefix and its extension produce different word orders, so the system conservatively outputs "Aber", until it receives further input.

	Source	Translation	Output
prefix	to paraphrase : it's not the strongest of	Um zu Paraphrasen: Es ist nicht die Stärke der Dinge	
extension	to paraphrase : it's not the strongest of the	Um zu paraphrasen: Es ist nicht die Stärke der Dinge	Um zu paraphrasen: Es ist nicht die Stärke der Dinge
prefix	to paraphrase : it's not the strongest of the	Um zu Paraphrasen: Es ist nicht die Stärke der Dinge	
extension	to paraphrase : it's not the strongest of the world	Zum paraphrasen: Es ist nicht die stärksten der Welt	Um zu paraphrasen: Es ist nicht die Stärke der Dinge

Figure 10: In this example we illustrate the refinement to the dynamic masking method, that is used typically when there are small uncertainties early in the sentence. For the second prefix, its translation and the translation of its extension differ at the beginning ("um zu" vs "zum"). Applying the LCP rule would result in a large mask, but instead we output the translation of the previous prefix

the fragment "But you know" can be translated both as subject-verb and verb-subject in German, depending on whether its continuation is a statement or a question. The extension predicted by the LM is just a comma, but it is enough to reveal the uncertainty in the translation and mean that the algorithm applies a mask.

The last example (Figure 10) shows the effect of our refinement to the dynamic masking method, in reducing spurious masking. For the second prefix, the translations of the prefix and its extension differ at the start, between "Um zu" and "Zum". Instead of masking to the LCP (which would mask the whole sentence in this case) the dynamic mask algorithm just sticks with the output from the previous prefix.

7 Conclusion

We propose a dynamic mask strategy to improve the stability for the retranslation method in online SLT. We have shown that combining biased beam search with mask- k works well in retranslation systems, but biased beam search requires a modified inference algorithm and hurts the quality and additional mask- k used to reduce this effect gives a high latency. Instead, the dynamic mask strategy maintains the translation quality but gives a much better latency–flicker trade-off curve than mask- k . Our experiments also show that the effect of this strategy depends on both the length and number of predicted extensions, but the quality of predicted extensions is less important.

For future research, we would like to combine biased beam search with dynamic mask to see if it can give a better trade-off between quality and latency than (Arivazhagan et al., 2020a) when the flicker is small enough. We would also like to experiment our dynamic masking strategy on prefixes provided by a real ASR system.

Acknowledgements



This work has received funding from the European Union’s Horizon 2020 Research and Innovation Programme under Grant Agreement No 825460 (Elitr).

References

- Arivazhagan, N., Cherry, C., I, T., Macherey, W., Baljekar, P., and Foster, G. (2020a). Re-Translation Strategies For Long Form, Simultaneous, Spoken Language Translation. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.
- Arivazhagan, N., Cherry, C., Macherey, W., Chiu, C.-C., Yavuz, S., Pang, R., Li, W., and Raffel, C. (2019). Monotonic Infinite Lookback Attention for Simultaneous Machine Translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1313–1323, Florence, Italy. Association for Computational Linguistics.
- Arivazhagan, N., Cherry, C., Macherey, W., and Foster, G. (2020b). Re-translation versus Streaming for Simultaneous Translation. *arXiv e-prints*, page arXiv:2004.03643.
- Bangalore, S., Rangarajan Sridhar, V. K., Kolan, P., Golipour, L., and Jimenez, A. (2012). Real-time incremental speech-to-speech translation of dialogs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 437–445, Montréal, Canada. Association for Computational Linguistics.
- Cettolo, M., Federico, M., Bentivogli, L., Niehues, J., Stüker, S., Sudoh, K., Yoshino, K., and Federmann, C. (2017). Overview of the IWSLT 2017 Evaluation Campaign. In *Proceedings of IWSLT*.
- Cho, K. and Esipova, M. (2016). Can neural machine translation do simultaneous translation? *CoRR*, abs/1606.02012.
- Gu, J., Neubig, G., Cho, K., and Li, V. O. (2017). Learning to translate in real-time with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1053–1062, Valencia, Spain. Association for Computational Linguistics.

- Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H., Heafield, K., Necker mann, T., Seide, F., Germann, U., Fikri Aji, A., Bogoychev, N., Martins, A. F. T., and Birch, A. (2018). Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.
- Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.
- Ma, M., Huang, L., Xiong, H., Zheng, R., Liu, K., Zheng, B., Zhang, C., He, Z., Liu, H., Li, X., Wu, H., and Wang, H. (2019). STACL: Simultaneous translation with implicit anticipation and controllable latency using prefix-to-prefix framework. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3025–3036, Florence, Italy. Association for Computational Linguistics.
- Niehues, J., Pham, N.-Q., Ha, T.-L., Sperber, M., and Waibel, A. (2018). Low-latency neural speech translation. In *Proceedings of Interspeech*.
- Post, M. (2018). A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.
- Rangarajan Sridhar, V. K., Chen, J., Bangalore, S., Ljolje, A., and Chengalvarayan, R. (2013). Segmentation strategies for streaming speech translation. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 230–238, Atlanta, Georgia. Association for Computational Linguistics.
- Sennrich, R., Firat, O., Cho, K., Birch, A., Haddow, B., HITSCHLER, J., Junczys-Dowmunt, M., Läubli, S., Miceli Barone, A. V., Mokry, J., and Nădejde, M. (2017). Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68, Valencia, Spain. Association for Computational Linguistics.
- Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.
- Zheng, B., Zheng, R., Ma, M., and Huang, L. (2019a). Simpler and faster learning of adaptive policies for simultaneous translation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1349–1354, Hong Kong, China. Association for Computational Linguistics.

Zheng, B., Zheng, R., Ma, M., and Huang, L. (2019b). Simultaneous translation with flexible policy via restricted imitation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5816–5822, Florence, Italy. Association for Computational Linguistics.