Incremental Decoding and Training Methods for Simultaneous Translation in Neural Machine Translation

Fahim Dalvi* faimaduddin@qf.org.qa

Nadir Durrani* ndurrani@qf.org.qa

Hassan Sajjad hsajjad@qf.org.qa

Stephan Vogel svogel@qf.org.qa

Qatar Computing Research Institute – HBKU

Abstract

We address the problem of simultaneous translation by modifying the Neural MT decoder to operate with dynamically built encoder and attention. We propose a tunable agent which decides the best segmentation strategy for a user-defined BLEU loss and Average Proportion (AP) constraint. Our agent outperforms previously proposed Wait-if-diff and Wait-if-worse agents (Cho and Esipova, 2016) on BLEU with a lower latency. Secondly we proposed data-driven changes to Neural MT training to better match the incremental decoding framework.

1 Introduction

Simultaneous translation is a desirable attribute in Spoken Language Translation, where the translator is required to keep up with the speaker. In a lecture or meeting translation scenario where utterances are long, or the end of sentence is not clearly marked, the system must operate on a buffered sequence. Generating translations for such incomplete sequences presents a considerable challenge for machine translation, more so in the case of syntactically divergent language pairs (such as German-English), where the context required to correctly translate a sentence, appears much later in the sequence, and prematurely committing to a translation leads to significant loss in quality.

Various strategies to select appropriate segmentation points in a streaming input have been proposed (Fügen et al., 2007; Bangalore et al., 2012; Sridhar et al., 2013; Yarmohammadi et al., 2013; Oda et al., 2014). A downside of this approach is that the MT system translates sequences independent of each other, ignoring the context. Even if the segmenter decides perfect points to segment the input stream, an MT system requires lexical history to make the correct decision.

The end-to-end nature of the Neural MT architecture (Sutskever et al., 2014; Bahdanau et al., 2014) provides a natural mechanism[1] to integrate stream decoding. Specifically, the recurrent property of the encoder and decoder components provide an easy way to maintain historic context in a fixed size vector.

We modify the neural MT architecture to operate in an online fashion where i) the *encoder* and the *attention* are updated dynamically as new input words are added, through a `READ` operation, and ii) the *decoder* generates output from the available encoder states, through a `WRITE` operation. The decision of when to `WRITE` is learned through a tunable segmentation agent, based on user-defined thresholds. Our incremental decoder significantly outperforms the chunk-based decoder and restores the oracle performance with a deficit of ≤ 2 BLEU points across 4 language pairs with a moderate delay. We additionally explore whether modifying the Neural MT training to match the decoder can improve performance. While we observed significant restoration in the case of chunk decoding matched with chunk-based NMT training, the same was not found true with our proposed incremental training to match the incremental decoding framework.

The remaining paper is organized as follow: Section 2 describes modifications to the NMT decoder to enable stream decoding. Section 3 describes various agents to learn a `READ/WRITE` strategy. Section 4 presents evaluation and results. Section 5 describes modifications to the NMT training to mimic corresponding decoding strategy, and Section 6 concludes the paper.

*These authors contributed equally to this work

[1] as opposed to the traditional phrase-based decoder (Moses), which requires pre-computation of phrase-table, future-cost estimation (Durrani et al., 2013a) and separately maintaining each state-full feature (language model, OSM (Durrani et al., 2013b) etc.)

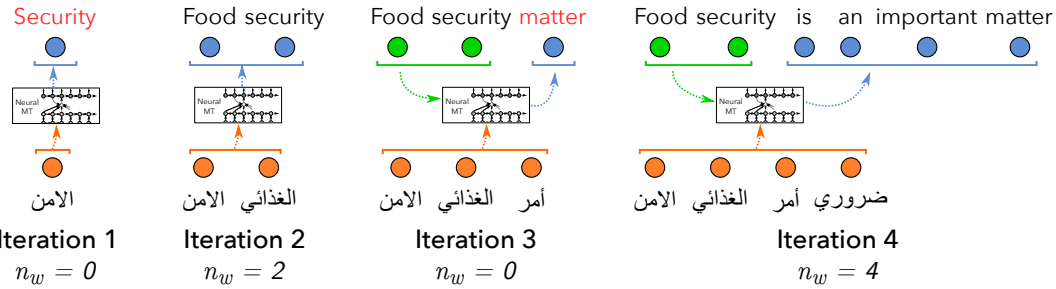

Figure 1: A decoding pass over a 4-word source sentence. n_w denotes the number of words the agent chose to commit. Green nodes = committed words, Blue nodes = newly generated words in the current iteration. Words marked in red are discarded, as the agent chooses to not commit them.

2 Incremental Decoding

Problem: In a stream decoding scenario, the entire source sequence is not readily available. The translator must either wait for the sequence to finish in order to compute the encoder state, or commit partial translations at several intermediate steps, potentially losing contextual information.

Chunk-based Decoder: A straight forward way to enable simultaneous translation is to chop the incoming input after every N-tokens. A drawback of these approaches is that the translation and segmentation process operate independently of each other, and the previous contextual history is not considered when translating the current chunk. This information is important to generate grammatically correct and coherent translations.

Incremental Decoding: The RNN-based NMT framework provides a natural mechanism to preserve context and accommodate streaming. The decoder maintains the entire target history through the previous decoder state alone. But to enable incremental neural decoding, we have to address the following constraints: i) how to dynamically build the encoder and attention with the streaming input? ii) what is the best strategy to pre-commit translations at several intermediate points?

Inspired by Cho and Esipova (2016), we modify the NMT decoder to operate in a sequence of `READ` and `WRITE` operations. The former reads the next word from the buffered source sequence and translates it using the available context, and the latter is computed through an AGENT, which decides how many words should be committed from this generated translation. Note that, when a translation is generated in the `READ` operation, the already committed target words remain unchanged, i.e. the generation is continued from the last committed target word using the saved decoder state. See Algorithm 1 for details. The AGENT decides how many target words to `WRITE` after every `READ` operation, and has complete control over the context each target word gets to see before being committed, as well as the overall delay incurred. Figure 1 shows the incremental decoder in action, where the agent decides to not commit any target words in iterations 1 and 3. The example shows an instance where the incorrectly translated words are discarded when more context becomes available. Given this generic framework, we describe several AGENTS in Section 3, trained to optimize the BLEU loss and latency.

Algorithm 1 Algorithm for incremental decoder

s, Source sequence
s', Available source sequence
t_c, Committed target sequence
t, Current decoded sequence for s'
n_w, Number of tokens to commit

$s' \leftarrow$ empty
for $token$ in s **do** ▷ `READ` operation
  $s' \leftarrow s' + token$
  $t \leftarrow$NMTDECODER(s', t_c)
  if $s' \neq s$ **then**
    $n_w \leftarrow$AGENT(s', t_c, t)
  else
    $n_w \leftarrow$ length$(t) -$ length(t_c)
  end if ▷ commit all new words if we have seen the entire source
  $t'_c \leftarrow$GETNEWTOKENS(t_c, t, n_w)
  $t_c \leftarrow t_c + t'_c$ ▷ `WRITE` operation
end for

function GETNEWTOKENS(t_c, t, n_w)
  $start \leftarrow$length$(t_c) + 1$
  $end \leftarrow start + n_w$
  return $t[start : end]$
end function

Beam Search: Independent of the agent being used, the modified NMT architecture incurs some

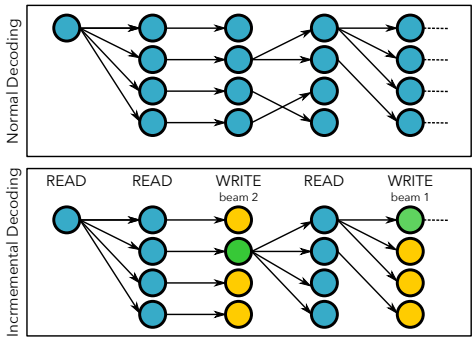

Figure 2: Beam Search in normal decoding vs incremental decoding. Green nodes indicate the hypothesis selected by the agent to WRITE. Since we cannot change what we have already committed, the other nodes (marked in yellow) are discarded and future hypotheses originate from the selected hypothesis alone. Normal beam search is executed for *consecutive* READ operations (blue nodes).

complexities for beam decoding. For example, if at some iteration the decoder generates 5 new words, but the agent decides to commit only 2 of these, the best hypothesis at the 2^{nd} word may not be the same as the one at the 5^{th} word. Hence, the agent has to re-rank the hypotheses at the last target word it decides to commit. Future hypotheses then continue from this selected hypothesis. See Figure 2 for a visual representation. The overall utility of beam decoding is reduced in the case of incremental decoding, because it is necessary to commit and retain only one beam at several points to start producing output with minimal delay.

3 Segmentation Strategies

In this section, we discuss different AGENTS that we evaluated in our modified incremental decoder. To measure latency in these agents, we use *Average Proportion* (AP) metric as defined by Cho and Esipova (2016). AP is calculated as the total number of source words each target word required before being committed, normalized by the product of the source and target lengths. It varies between 0 and 1 with lesser being better. See supplementary material for details.

Wait-until-end: The WUE agent waits for the entire source sentence before decoding, and serves as an upper bound on the performance of our agents, albeit with the worst $AP = 1$.

Wait-if-worse/diff: We reimplemented the baseline agents described in Cho and Esipova (2016). The **Wait-if-Worse** (WIW) agent WRITES a target word if its probability does not decrease after a READ operation. The **Wait-if-Diff** (WID) agent instead WRITES a target word if the target word remains unchanged after a READ operation.

Static Read and Write: The STATIC-RW: agent is inspired from the chunk-based decoder and tries to resolve its shortcomings while maintaining its simplicity. The primary drawback of the chunk-based decoder is the loss of context across chunks. Our agent starts by performing $\mathcal{S}$ READ operations, followed by repeated $\mathcal{RW}$ WRITES and READS until the end of the source sequence. The number of WRITE and READ operations is the same to ensure that the gap between the source and target sequence does not increase with time. The initial $\mathcal{S}$ READ operations essentially create a buffer of $\mathcal{S}$ tokens, allowing some future context to be used by the decoder. Note that the latency induced by this agent in this case is only in the beginning, and remains constant for the rest of the sentence. This method actually introduces a class of AGENTS based on their $\mathcal{S}$,$\mathcal{RW}$ values. We tune $\mathcal{S}$ and $\mathcal{RW}$ to select the specific AGENT with the user-defined BLEU-loss and AP thresholds.

4 Evaluation

Data: We trained systems for 4 language pairs: German-, Arabic-, Czech- and Spanish-English pairs using the data made available for IWSLT (Cettolo et al., 2014). See supplementary material for data stats. These language pairs present a diverse set of challenges for this problem, with Arabic and Czech being morphologically rich, German being syntactically divergent, and Spanish introducing local reorderings with respect to English.

NMT System: We trained a 2-layered LSTM encoder-decoder models with attention using the `seq2seq-attn` implementation. Please see supplementary material for settings.

Results: Figure 3 shows the results of various streaming agents. Our proposed STATIC-RW agent outperforms other methods while maintaining an AP $<$ 0.75 with a loss of less than 0.5 BLEU points on Arabic, Czech and Spanish. This was found to be consistent for all test-sets 2011-2014 (See under "small" models in Figure 4). In the case of German the loss at AP $<$ 0.75 was around 1.5 BLEU points. The syntactical divergence and rich morphology of German posits a

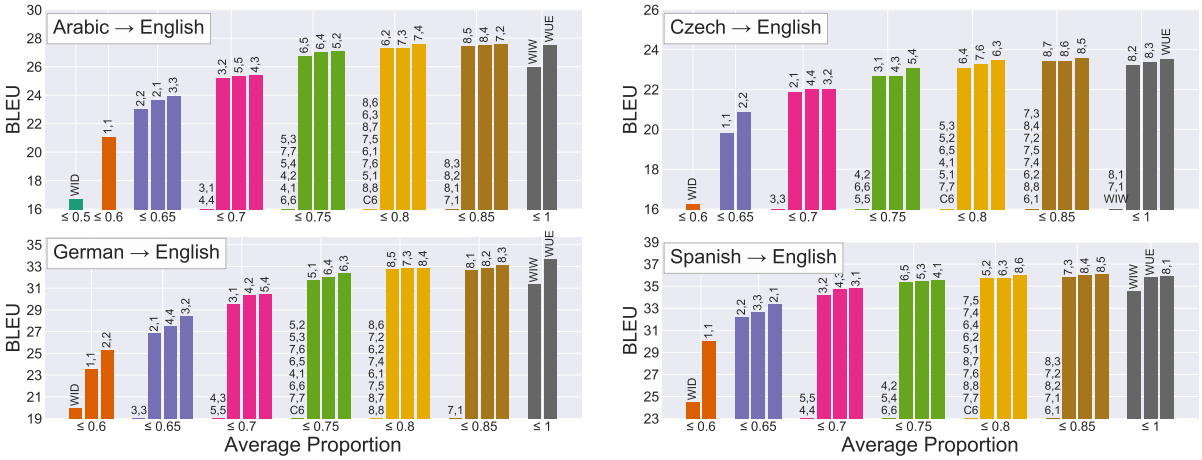

Figure 3: Results for various streaming AGENTS (WID, WIW, WUE, C6 (Chunk decoding with a N=6) and $\mathcal{S}$,$\mathcal{RW}$ for STATIC-RW) on the tune-set. For each AP bucket, we only show the Agents with the top 3 BLEU scores in that bucket, with remaining listed in descending order of their BLEU scores.

bigger challenge and requires larger context than other language pairs. For example the conjugated verb in a German verb complex appears in the second position, while the main verb almost always occurs at the end of the sentence/phrase (Durrani et al., 2011). Our methods are also comparable to the more sophisticated techniques involving Reinforcement Learning to learn an agent introduced by Gu et al. (2017) and Satija and Pineau (2016), but without the overhead of expensive training for the agent.

Scalability: The preliminary results were obtained using models trained on the TED corpus only. We conducted further experiments by training models on larger data-sets (See the supplementary section again for data sizes) to see if our findings are scalable. We fine-tuned (Luong and Manning, 2015; Sajjad et al., 2017b) our models with the in-domain data to avoid domain disparity. We then re-ran our agents with the best $\mathcal{S}$,$\mathcal{RW}$ values (with an AP under 0.75) for each language pair. Figure 4 ("large" models) shows that the BLEU loss from the respective oracle increased when the models were trained with bigger data sizes. This could be attributed to the increased lexical ambiguity from the large amount of out-domain data, which can only be resolved with additional contextual information. However our results were still better than the WIW agent, which also has an AP value above 0.8. Allowing similar AP, our STATIC-RW agents were able to restore the BLEU loss to be ≤ 1.5 for all language pairs except German-English. Detailed test results are available in the suplementary material.

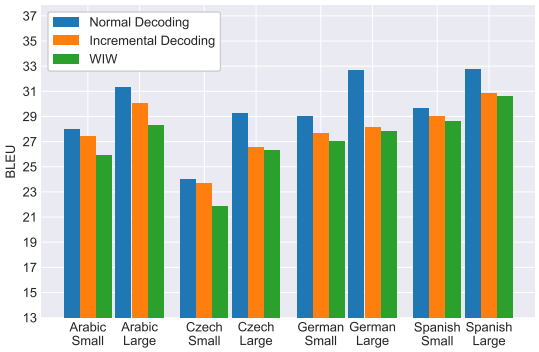

Figure 4: Averaged results on test-sets (2011-2014) using the models trained on small and large datasets using $AP \leq 0.75$. Detailed test-wise results are available in the supplementary material.

5 Incremental Training

The limitation of previously described decoding approaches (chunk-based and incremental) is the mismatch between training and decoding. The training is carried on full sentences, however, at the test time, the decoder generates hypothesis based on incomplete source information. This discrepancy between training and decoding can be potentially harmful. In Section 2, we presented two methods to address the partial input sentence decoding problem, the *Chunk Decoder* and the *Incremental Decoder*. We now train models to match the corresponding decoding scenario.

5.1 Chunk Training

In chunk-based training, we simply split each training sentence into chunks of N tokens.[2] The corresponding target sentence for each chunk is generated by having a span of target words that are word-aligned[3] with the words in the source span. Chunking the data into smaller segments increases the training time significantly. To overcome this problem, we train a model on the full sentences using all the data and then fine-tune it with the in-domain chunked data.

5.2 Add-M Training

Next we formulate a training mechanism to match the incremental decoding described in Section 2. A way to achieve this is to force the attention on a local span of encoder states and block it from giving weight to the non-local (rightward) encoder states. The hope is that in the case of long-range dependencies, the model learns to predict these dependencies without the entire source context. Such a training procedure is non-trivial, as it requires dynamic inputs to the attention mechanism while training, including backpropagation where some encoder states which have been seen by the attention mechanism a greater number of times dynamically receiving more gradient inputs. We leave this idea as future work, while focusing on a data-driven technique to mimic this kind of training as described below.

We start with the first N words in a source sentence and generate target words that are aligned to these words. We then generate the next training instances with $N+M$, $N+2M$, $N+3M$ ... source words until the end of sentence has been reached.[4] The resulting training roughly mimics the decoding scenario where the source-side context is gradually built. The down-side of this method is that the data size increases quadratically, making the training infeasible. To overcome this, we fine-tune a model trained on full sentences with the in-domain corpus generated using this method.

[2] Although randomly segmenting the source sentence based on number of tokens is a naïve approach that does not take into account the linguistic properties, our goal here was to exactly match the training with the chunk-based decoding scenario.

[3] We used fast-align (Dyer et al., 2013) for alignments.

[4] We trained with $N = 6$ and $M = 1$ for our experiments.

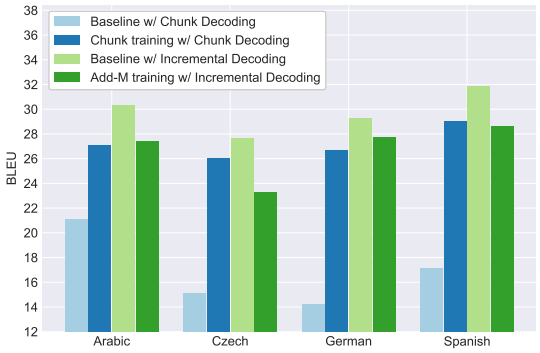

Figure 5: Averaged test set results on various training modifications

5.3 Results

The results in Figure 5 show that matching the chunk-decoding with corresponding chunk-based training significantly improves performance, with a gain of up to 12 BLEU points. However, we were not able to improve upon our incremental decoder, with the results deteriorating notably. One reason for this degradation is that the training/decoding scenarios are still not perfectly matched. The training pipeline in this case also sees the beginning of sentences much more often, which could lead to unnatural distributions being inferred within the model.

6 Conclusion

We addressed the problem of simultaneous translation by modifying the architecture in Neural MT decoder. We presented a tunable agent which decides the best segmentation strategy based on user-defined BLEU loss and AP constraints. Our results showed improvements over previously established WIW and WID methods. We additionally modified the Neural MT training to match the incremental decoding, which significantly improved the chunk-based decoding, but we did not observe any improvement using *Add-M Training*. The code for our incremental decoder and agents has been made available.[5] While were able to significantly improve the the chunk-based decoder, we did not observe any improvement using the *Add-M Training*. In the future we would like to change the training model to dynamically build the encoder and the attention model in order to match our incremental decoder.

[5] https://github.com/fdalvi/seq2seq-attn-stream

References

Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. 2016. Farasa: A fast and furious segmenter for arabic. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*. Association for Computational Linguistics, San Diego, California, pages 11–16.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473.

Srinivas Bangalore, Vivek Kumar Rangarajan Sridhar, Prakash Kolan, Ladan Golipour, and Aura Jimenez. 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*. Association for Computational Linguistics, Montréal, Canada, pages 437–445.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT Evaluation Campaign. *Proceedings of the International Workshop on Spoken Language Translation, Lake Tahoe, US* .

Kyunghyun Cho and Masha Esipova. 2016. Can neural machine translation do simultaneous translation? *CoRR* abs/1606.02012.

Nadir Durrani, Alexander Fraser, and Helmut Schmid. 2013a. Model With Minimal Translation Units, But Decode With Phrases. In *Proceedings of the NAACL-HLT'13*. Atlanta, Georgia, USA.

Nadir Durrani, Alexander Fraser, Helmut Schmid, Hieu Hoang, and Philipp Koehn. 2013b. Can Markov Models Over Minimal Translation Units Help Phrase-Based SMT? In *Proceedings of ACL 2013*. Sofia, Bulgaria, pages 399–405.

Nadir Durrani, Helmut Schmid, and Alexander Fraser. 2011. A Joint Sequence Translation Model with Integrated Reordering. In *Proceedings of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT'11)*. Portland, OR, USA.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of NAACL'13*.

Andreas Eisele and Yu Chen. 2010. MultiUN: A Multilingual Corpus from United Nation Documents. In *Proceedings of the Seventh conference on International Language Resources and Evaluation*. Valleta, Malta.

Christian Fügen, Alex Waibel, and Muntsin Kolss. 2007. Simultaneous translation of lectures and speeches. *Machine Translation* 21(4):209–252.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O.K. Li. 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*. Association for Computational Linguistics, Valencia, Spain, pages 1053–1062.

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

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford Neural Machine Translation Systems for Spoken Language Domains. In *Proceedings of the International Workshop on Spoken Language Translation*. Da Nang, Vietnam.

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2014. Optimizing segmentation strategies for simultaneous speech translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 551–556.

Hassan Sajjad, Fahim Dalvi, Nadir Durrani, Ahmed Abdelali, Yonatan Belinkov, and Stephan Vogel. 2017a. Challenging Language-Dependent Segmentation for Arabic: An Application to Machine Translation and Part-of-Speech Tagging. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Vancouver, Canada.

Hassan Sajjad, Nadir Durrani, Fahim Dalvi, Yonatan Belinkov, and Stephan Vogel. 2017b. Neural Machine Translation Training in a Multi-Domain Scenario. In *Proceedings of the 14th International Workshop on Spoken Language Technology (IWSLT-14)*.

Harsh Satija and Joelle Pineau. 2016. Simultaneous machine translation using deep reinforcement learning. In *Abstraction in Reinforcement Learning Workshop*. International Conference on Machine Learning, New York, USA.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 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)*. Association for Computational Linguistics, Berlin, Germany, pages 1715–1725.

Rangarajan Sridhar, Vivek Kumar, John Chen, Srinivas Bangalore, Andrej Ljolje, and Rathinavelu Chengalvarayan. 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. Association for Computational Linguistics, Atlanta, Georgia, pages 230–238.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in neural information processing systems*. pages 3104–3112.

Mahsa Yarmohammadi, Vivek Kumar Rangarajan Sridhar, Srinivas Bangalore, and Baskaran Sankaran. 2013. Incremental segmentation and decoding strategies for simultaneous translation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. Asian Federation of Natural Language Processing, Nagoya, Japan, pages 1032–1036.

A Supplementary Material

A.1 Data and Preprocessing

We trained systems for four language pairs namely German-English, Arabic-English, Czech-English and Spanish-English using the data for the translation task of the International Workshop on Spoken Language Translation (Cettolo et al., 2014). Apart from using the in-domain TED corpus (≈ 200K sentences), we additionally used Europarl and News Corpus made available for the recent WMT campaign. For Arabic-to-English, we also news Corpus and a subset of UN corpus (1 Million sentences) (Eisele and Chen, 2010). We used a concatenation of dev- and test-2010 for tuning Neural MT models, test-2011 for development (tuning the Static Read and Write agent) and tests 2012-14 for testing. We used Moses (Koehn et al., 2007) preprocessing pipeline including tokenization and truecasing. For Arabic we used Farasa segmentation (Abdelali et al., 2016) with BPE (Sennrich et al., 2016) as suggested in (Sajjad et al., 2017a). We trained the BPE models separately for both the source and target datasets instead of jointly training limiting the number of operations to 49,500, as suggested in (Sennrich et al., 2016).

Pair	ID	Cat	test11	test12	test13	test14
ar-en	229K	1.26M	1199	1702	1169	1107
	4.4M	28.7M	22K	28K	24K	20K
	4.7M	30.2M	26K	32K	28K	24K
de-en	209K	2.4M	1433	1700	993	1305
	4.0M	61.9M	26K	29K	20K	24K
	4.3M	64.7M	27K	31K	21K	25K
cs-en	122K	900K	1013	1385	1327	–
	2.0M	20.3M	15K	21K	24K	–
	2.5M	23.6M	18K	25K	28K	–
es-en	188K	2.3M	1435	1385	–	–
	3.6M	66.9M	25K	27.5K	–	–
	3.8M	64.4M	27K	31K	–	–

Table 1: Data Statistics: First Row = Number of Sentences, Second Row: Number of Tokens in Source Language, Third Row: Number of Tokens in Target Language. First Column = statistics for the in-domain TED corpus, Second Column = Statistics for the Concatenated Data

A.2 Neural MT system

We train a 2-layer LSTM encoder-decoder with attention using the `seq2seq-attn` implementation (**?**) with the following settings: word vectors and LSTM states with 500 dimensions, SGD with an initial learning rate of 1.0, a decay rate of 0.5, and dropout rate of 0.3. The MT systems are trained for 13 epochs. We used uni-directional encoder because it is not possible to compute the encoder in right-to-left direction in the streaming scenario, due to unavailability of the full input sentence. Computing right-to-left encoder states with whatever input sequence is available is also not viable as it requires expensive re-computation after each input word is added.[6] We also trained the models by initializing the first decoder state with zeros, rather than using the final encoder state, which will not be available during stream decoding.

A.3 Average Proportion

In normal decoding, the BLEU metric is commonly used to calculate the quality of translations from a system. In stream decoding, we have to also consider the delay induced by the system along with its BLEU. In our work, we use *Average Proportion* (AP) as defined by Gu et al. (2017). AP is calculated as the total number of source words each target word required before being committed, normalized by the product of the source and target lengths. Formally, if $s(t_i)$ is the number of source words required for target word i before being committed, X is the source sequence and Y is the generated target sequence:

$$AP = \frac{1}{|X| \cdot |Y|} \sum_{t_i}^{Y} s(t_i) \qquad (1)$$

A.4 Incremental Decoder

Figure 4 shows the average results on the test-sets for the models trained on in-domain TED corpus. Here, we present the test-wise results for the interested reader. Missing table values correspond to unavailable test-sets on the IWSLT webpage. See Table 2.

A.5 Scalability

In section 4 we note that even though the `WIW` agent's performance is not significantly below our selected `STATIC-RW` agent, its AP is much higher. When we allow our `STATIC-RW` agent an AP similar to that of the `WIW` agent, we are able to restore the BLEU loss to be less than 1.5 for all

[6] Unlike left-to-right encoder which only requires single computation after each input word is added.

Pair	Agent	test12	test13	test14	Agent	test12	test13	test14
ar-en	WUE	30.16	28.16	25.53	WUE	32.84	32.23	28.95
	5, 2	29.31	27.72	25.21	7, 2	31.71	31.46	28.29
	WIW	28.06	25.86	23.75	WIW	29.48	28.82	26.52
	WID	19.89	17.24	15.64				
cs-en	WUE	22.95	25.03	–	WUE	27.97	30.50	–
	5, 4	22.97	24.46	–	8, 3	26.68	29.37	–
	WIW	21.78	21.99	–	WIW	25.20	27.43	–
	WID	16.37	17.07	–				
de-en	WUE	29.20	31.31	26.61	WUE	35.52	35.01	30.44
	6, 3	27.94	29.90	25.07	8, 3	28.62	31.71	27.09
	WIW	27.77	29.55	23.88	WIW	27.94	30.05	25.56
	WID	19.15	20.73	16.46				
es-en	WUE	29.65	–	–	WUE	32.78	–	–
	4, 1	29.04	–	–	8, 1	32.05	–	–
	WIW	28.65	–	–	WIW	30.59	–	–
	WID	21.90	–	–				

Table 2: Left Side: Test-wise results for "Small" models in Figure 4, Right Side: Test-wise results for "Large" models in Figure 4

.

language pairs except German-English. Here are the results in detail. See Table 2.