Prediction Improves Simultaneous Neural Machine Translation

Ashkan Alinejad [1], Maryam Siahbani[2], Anoop Sarkar[1]
[1]Simon Fraser University, Burnaby, BC, Canada
{aalineja,anoop}@sfu.ca
[2]University of the Fraser Valley, Abbotsford, BC, Canada
maryam.siahbani@ufv.ca

Abstract

Simultaneous speech translation aims to maintain translation quality while minimizing the delay between reading input and incrementally producing the output. We propose a new general-purpose prediction action which predicts future words in the input to improve quality and minimize delay in simultaneous translation. We train this agent using reinforcement learning with a novel reward function. Our agent with prediction has better translation quality and less delay compared to an agent-based simultaneous translation system without prediction.

1 Introduction

One of the next significant challenges in machine translation research is to make translation ubiquitous using real-time translation. Simultaneous machine translation aims to address this issue by interleaving reading the input with writing the output translation. Current Simultaneous Neural Machine Translation (SNMT) systems (Satija and Pineau, 2016; Cho and Esipova, 2016; Gu et al., 2017) use an AGENT to control an incremental encoder-decoder (or sequence to sequence) NMT model. Each READ adds more information to the encoder RNN, and each WRITE produces more output using the decoder RNN. In this paper, we propose adding a new action to the AGENT: a PREDICT action that predicts what words might appear in the input stream. Prediction was previously proposed in simultaneous statistical machine translation (Grissom II et al., 2014) but has not been studied in the context of Neural Machine Translation (NMT). In SNMT systems, prediction of future words augments the encoder-decoder model with possible future contexts to produce output translations earlier (minimize delay) and/or produce better output translations (improve translation quality). Our experiments show that prediction improves SNMT in both these measures.

2 Simultaneous Translation Framework

An agent-based framework whose actions decide whether to translate or wait for more input is a natural way to extend neural MT to simultaneous neural MT and has been explored in (Satija and Pineau, 2016; Gu et al., 2017) which contains two main components: The ENVIRONMENT which receives the input words $X = \{x_1, \ldots, x_N\}$ from the source language and incrementally generates translated words $W = \{w_1, \ldots, w_M\}$ in the target language; And the AGENT which decides an action for each time step, a_t. The AGENT generates an action sequence $A = \{a_1, \ldots, a_T\}$ to control the ENVIRONMENT.

Previous models only include two actions: READ and WRITE. We extend the model by adding the third action called PREDICT. Action READ is simply sending a new word to the ENVIRONMENT and generating a candidate word in the target language. In action WRITE, the AGENT takes current candidate word and sends it to the output. For PREDICT, the AGENT predicts the next word in the input and treats it like a READ action. The following section explains how the ENVIRONMENT deals with different actions.

2.1 ENVIRONMENT

The ENVIRONMENT is an attention-based Encoder-Decoder MT system (Bahdanau et al., 2014) which is adopted to simultaneous translation task. The Encoder receives the embedded representation of the input words (including predicted ones) and converts them into context vectors $H_n^\rho = \{h_1, \ldots, h_{n+\rho}\}$ using a gated RNN (GRU) where n is the number of input words so far and ρ is the number of predicted words since the last READ. Whenever the AGENT decides to READ, ρ will be set to 0, and $h_n = f_{\text{ENC}}(h_{n-1}, x_n)$, where x_n is the next input words ($n \leq N$). But if the action is PREDICT, $\rho > 0$, the AGENT predicts a new word x'_ρ and the

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3022–3027
Brussels, Belgium, October 31 - November 4, 2018.

context vector $h_{n+\rho} = f_{\text{ENC}}(h_{n+\rho-1}, x'_\rho)$ will be added to $H_n^\rho = \{h_1, \ldots, h_{n+\rho}\}$.

At each time step t, the decoder uses the current context vectors (H_n^ρ) to generate the next candidate output (y_t):

$$c_t = a_{\text{ATT}}(z_{m-1}, H_n^\rho)$$
$$s_t = f_{\text{DEC}}(z_{m-1}, w_{m-1}, c_t)$$
$$p(y_t|y_{<t}, H_n^\rho) = g(w_{m-1}, z_m, c_t)$$
$$y_t = \arg\max_y p(y|y_{<t}, H_n^\rho)$$

where w_{m-1} is the previous output word, and a_{ATT} is an attention model (Bahdanau et al., 2014), f and g are nonlinear functions, and c_t is the current context vector.

If the action a_t is either READ or PREDICT the current candidate y_t will be ignored (wait for better predictions). But in the case of WRITE, the candidate y_t is produced as the next output word w_m and then the decoder state will be updated ($w_m \leftarrow y_t, z_m \leftarrow s_t$).

Note that as soon as the AGENT decides to READ, all the hidden vectors generated by PREDICT actions will be discarded ($H_n^\rho = H_n^0 = \{h_1, \ldots, h_n\}$).

Figure 1 shows an example of how a sentence can be translated using our modified translation framework [1].

2.2 AGENT

The AGENT is a separate component which examines the ENVIRONMENT at each time step and decides on the actions that lead to better translation quality and lower delay. The agent in the *greedy decoding* framework (Gu et al., 2017) was trained using reinforcement learning with the policy gradient algorithm (Williams, 1992), which observes the current state of the ENVIRONMENT at time step t as o_t where $o_t = [c_t;\ s_t;\ w_m]$. A RNN with one hidden layer passed through a softmax function generates the probability distribution over the actions a_t at each step. Therefore, policy π_θ will be computed as:

$$u_t = f_\theta(u_{t-1}, o_t)$$
$$\pi_\theta(a_t|a < t, o \leq t) \propto g_\theta(u_t)$$

Where u_t is the hidden state of the AGENT's RNN.

3 Training the AGENT with Prediction

In order to speed-up the training process, we have restricted AGENT's options by removing redundant operations. As illustrated in Figure 2, after a series of WRITE, the AGENT cannot choose to PREDICT, and after a sequence of PREDICTs, READ is not an option.

[1]The pseudo-code is available in supplementary material (Algorithm 1).

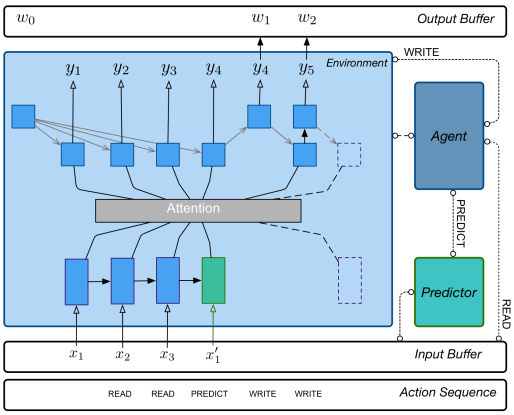


Figure 1: A schematic of our model. The ENVIRONMENT starts with reading the 'Start of Sentence' symbol as x_1 and generating y_1 from the decoder. Based on the output of the network at each time step, the AGENT decides whether to READ, WRITE or PREDICT for the following time steps.

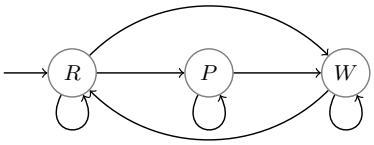


Figure 2: Action transition graph. R, P, and W stands for READ, PREDICT and WRITE actions respectively.

Reward Function: The total reward at any time step is calculated as the cumulative sum of rewards for actions at each preceding step. All the evaluation metrics have been modified to be computed for every time step.

Quality: We use a modified smoothed version of BLEU score (Chen and Cherry, 2014) multiplied by Brevity Penalty (Lin and Och, 2004) for evaluating the impact of each action on translation quality. At each point in time, the reward for translation quality is:

$$r_t^Q = \begin{cases} \Delta\text{BLEU}(t) & t < T \\ \text{BLEU}(W, W^*) & t = T \end{cases}$$

The ΔBLEU(t) is the difference between BLEU score of the translated sentence at the previous time step and the current time step; $\Delta\text{BLEU}(t) = \text{BLEU}(W^t, W^*) - \text{BLEU}(W^{t-1}, W^*)$; where W^t is the prefix of the translated sentence at time t.

Delay: The Delay reward is used to motivate the AGENT to minimize delay. We use Average Proportion (AP) (Cho and Esipova, 2016) for this purpose, which is the average number of source words

needed when translating each word. Given the source words X and translated words W, AP can be computed as:

$$d(X,W) = \frac{1}{|X||W|}\sum_t s(t)$$

$$r_t^D = \begin{cases} 0 & t < T \\ d(X,W) & t = T \end{cases}$$

Where $s(t)$ denotes the number of source words the WRITE action uses at time step t (for any other actions, $s(t)$ would be zero). The delay reward is smoothed using a *Target Delay* which is a scalar constant denoted by d^* (Gu et al., 2017):

$$r_t^D = \lfloor d_t - d^* \rfloor$$

Prediction Rewards for Quality and Delay alone do not motivate the AGENT to choose prediction and in preliminary experiments, after a number of steps, the number of prediction actions became zero. We address this problem by defining Prediction Quality (PQ) which rewards the AGENT for changes in BLEU score after each prediction action. By initializing $r_0^p = 0$, the prediction reward can be written as:

$$r_t^p = \begin{cases} \Delta\text{BLEU}(t) & a_t = \text{W}, a_{t-1} = \text{P} \\ r_{t-1}^p & a_t = \text{W}, a_{t-1} \neq \text{P} \\ 0 & \text{otherwise} \end{cases}$$

Final Reward The final reward function is calculated as the combination of quality, delay, and prediction rewards:

$$r_t = \alpha\, r_t^Q + \beta\, r_t^D + \lambda\, r_t^p \qquad (1)$$

The trade-off between better translation quality and minimal delay is achieved by modifying the parameters α, β, and λ.

Reinforcement Learning is used to train the AGENT using a policy gradient algorithm (Gu et al., 2017; Williams, 1992) which searches for the maximum in $J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=1}^T r_t\right]$ using the gradient: $\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a)\right]$ where $Q = \sum_{t=1}^T r_t$. The gradient for a sentence is the cumulative sum of gradients at each time step. We pre-train the ENVIRONMENT on full sentences using log-loss $\log p(\mathbf{y}|\mathbf{x})$.

4 Experiments

We train and evaluate our model on English-German (EN-DE) in both directions. We use WMT 2015 for training and Newstest 2013 for validation and testing. All sentences have been tokenized and the words are segmented using byte pair encoding (BPE) (Sennrich et al., 2016).

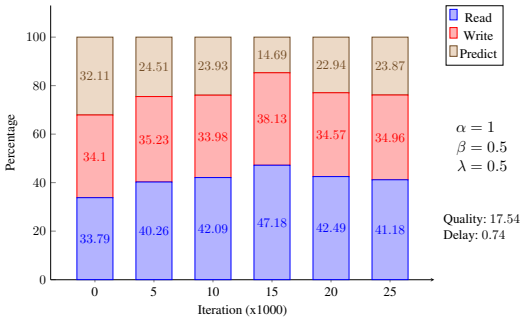
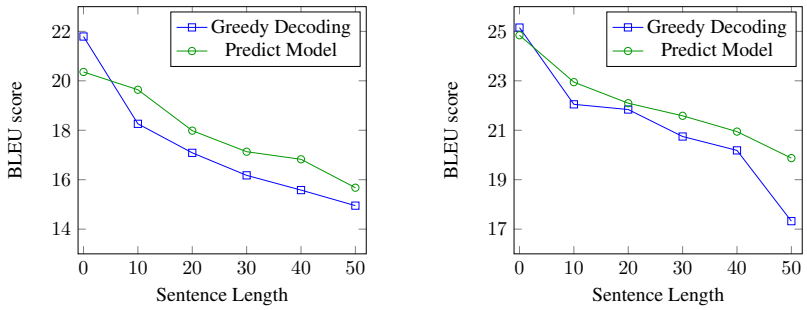


Figure 3: Action distribution for English to German translation in the first 25000 iterations. The numbers in each bar are the average action percentage over the previous 5000 iterations.

Model Configuration For a fair comparison, we follow the settings that worked the best for the greedy decoding model in (Gu et al., 2017) and set the target delay d^* for the AGENT to 0.7. The ENVIRONMENT consists of two unidirectional layers with 1028 GRU units for encoder and decoder. We train the network using AdaDelta optimizer, a batch of size 32 and a fixed learning rate of 0.0001 without decay. We use softmax policy via recurrent networks with 512 GRU units and a softmax function for the AGENT and train it using Adam optimizer (Kingma and Ba, 2014). The batch size for the AGENT is 10, and the learning rate is 2e-6. The word predictor is a two layer RNN Language model which consists of two layers of 1024 units, followed by a softmax layer. The batch size is 64 with a learning rate of 2e-5. The predictor has been trained on the WMT'16 dataset and tested on Newstest'16 corpora for both languages. The perplexity of our language model is reported in Table 1. We set $\alpha = 1$, $\beta = 0.5$ and $\lambda = 0.5$. We tried different settings for these hyperparameters during training and picked values that gave us the best Quality and Delay on the training data.

Results and Analysis Figure 4 shows that as the sentence length increases, prediction helps translation quality due to complex reordering and multi-clausal sentences; However, for shorter samples where the structure of the sentences are simpler, the prediction action cannot improve translation quality. Table 2 compares our model with the Greedy Decoding (GD) model in terms of translation quality and latency. It shows that the prediction mechanism outperforms the GD model in terms of BLEU and average proportion (AP).

The delay evaluation measure (AP) counts about the same number of READs and WRITEs. It



(a) EN→DE (b) DE→EN

Figure 4: Comparing translation quality between prediction model and greedy decoding method for various sentence lengths.

	Perplexity	
	Test	Train
English	17.5917	8.1350
German	19.8532	11.1808

Table 1: Performance of our predictor for both English and German languages.

	EN→DE			DE→EN		
	BLEU	AP	μ	BLEU	AP	μ
GD	16.75	0.79	5.6	21.43	0.77	6
PM	**17.54**	**0.74**	**4.7**	**21.83**	**0.70**	**5.2**

Table 2: Comparison of translation quality (BLEU), Average Proportion (AP), and average segment length (μ) for Greedy Decoding (GD) model with Prediction Mechanism (PM) model.

does not account for less delay as longer sentences are produced. A better measure than AP might be needed to emphasize delay differences. Therefore we also report the average segment length (μ), which is computed as the average number of consecutive READs in each sentence. In both EN→DE and DE→EN experiments, our model constantly decreases the segment length by around 1 word which results in less latency.

In order to evaluate the effectiveness of our proposed reward function for PREDICT action, we have explored various values for its hyper-parameters (α, β, and λ). Our empirical results show that the best trade-off between quality and delay is achieved when around 20 percent of the actions are PREDICT for both EN→DE and DE→EN translation tasks (Figure 3). When there is no reward for PREDICT action ($\lambda = 0$), the AGENT prefers other actions, and the number of PREDICT actions turns into zero immediately after training the AGENT. If the reward for prediction is valued too highly, the ENVIRONMENT depends more on predicted words and the translation quality decreases[2].

5 Related work

Early work in SNMT was done in speech, where the incoming signals were segmented based on acoustic or statistical cues (Bangalore et al., 2012; Fügen et al., 2007). (Sridhar et al., 2013; Matusov et al., 2007; Yarmohammadi et al., 2013; Siahbani et al., 2014) use a separate segmentation step and incrementally translate each segment using a standard phrase-based MT system. (Matsubara et al., 2000) applied pattern matching to predict target-side verbs in Japanese to English translation. (Grissom II et al., 2014) used reinforcement learning to predict the next word and the sentence-final verb in a statistical MT model. These models reduce the delay but are not trained end-to-end like our agent-based SNMT system. (Cho and Esipova, 2016) proposed a non-trainable heuristic agent which is not able to trade-off quality with delay. It always prefers to read more words from the input and this approach does not work well in practice. (Satija and Pineau, 2016) introduced a trainable agent which they trained using Deep Q networks (Mnih et al., 2015). We modified the SNMT trainable agent in (Gu et al., 2017) and added a new non-trivial PREDICT action to the agent. We compare to their model and show better results in delay and quality.

6 Conclusion

We introduce a new prediction action in a trainable agent for simultaneous neural machine translation. With prediction, the agent can be informed about future time steps in the input stream. Compared to a very strong baseline our results show that prediction can lower delay and improve the

[2]See Figure 6 in supplementary materials for more numerical results.

translation quality, especially for longer sentences and translating from an SOV (subject-object-verb) language (DE) to an SVO language (EN).

Acknowledgments

We would like to thank the anonymous reviewers for their helpful remarks. The research was also partially supported by the Natural Sciences and Engineering Research Council of Canada grants NSERC RGPIN-2018-06437 and RGPAS-2018-522574 and a Department of National Defence (DND) and NSERC grant DGDND-2018-00025 to the third author.

References

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*, NAACL HLT '12, pages 437–445, Stroudsburg, PA, USA. Association for Computational Linguistics.

Boxing Chen and Colin Cherry. 2014. A systematic comparison of smoothing techniques for sentence-level bleu. In *WMT*, pages 362–367.

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

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

Alvin Grissom II, He He, Jordan Boyd-Graber, John Morgan, and Hal Daumé III. 2014. Don't until the final verb wait: Reinforcement learning for simultaneous machine translation. In *Empirical Methods in Natural Language Processing*.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O.K. Li. 2017. Learning to translate in real-time with neural machine translation. In *15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Valencia, Spain.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shigeki Matsubara, Keiichi Iwashima, Nobuo Kawaguchi, Katsuhiko Toyama, and Yasuyoshi Inagaki. 2000. Simultaneous japanese-english interpretation based on early prediction of english verb. *Proceedings of the 4th Symposium on Natural Languauge Processing(SNLP-2000)*, pages 268–273.

Evgeny Matusov, Dustin Hillard, Mathew Magimai-Doss, Dilek Hakkani-Tür, Mari Ostendorf, and Hermann Ney. 2007. Improving speech translation with automatic boundary prediction. In *Eighth Annual Conference of the International Speech Communication Association*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature*, 518:529 EP –.

Harsh Satija and Joelle Pineau. 2016. Simultaneous machine translation using deep reinforcement learning. *Abstraction in Reinforcement Learning Workshop, ICML 2016*, (33).

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)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Maryam Siahbani, Ramtin Mehdizadeh Seraj, Baskaran Sankaran, and Anoop Sarkar. 2014. Incremental translation using hierarchichal phrase-based translation system. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 71–76. IEEE.

Vivek Kumar Rangarajan Sridhar, 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*, pages 230–238.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*, pages 229–256.

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*, pages 1032–1036.