Improved Speech Representations with Multi-Target Autoregressive Predictive Coding

Yu-An Chung, James Glass
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
{andyyuan,glass}@mit.edu

Abstract

Training objectives based on predictive coding have recently been shown to be very effective at learning meaningful representations from unlabeled speech. One example is Autoregressive Predictive Coding (Chung et al., 2019), which trains an autoregressive RNN to generate an unseen future frame given a context such as recent past frames. The basic hypothesis of these approaches is that hidden states that can accurately predict future frames are a useful representation for many downstream tasks. In this paper we extend this hypothesis and aim to enrich the information encoded in the hidden states by training the model to make more accurate future predictions. We propose an auxiliary objective that serves as a regularization to improve generalization of the future frame prediction task. Experimental results on phonetic classification, speech recognition, and speech translation not only support the hypothesis, but also demonstrate the effectiveness of our approach in learning representations that contain richer phonetic content.

1 Introduction

Unsupervised speech representation learning, which aims to learn a function that transforms surface features, such as audio waveforms or spectrograms, to higher-level representations using only unlabeled speech, has received great attention recently (Baevski et al., 2019, 2020; Liu et al., 2020; Song et al., 2019; Jiang et al., 2019; Schneider et al., 2019; Chorowski et al., 2019; Pascual et al., 2019; Oord et al., 2018; Kamper, 2019; Chen et al., 2018; Chung and Glass, 2018; Chung et al., 2018; Milde and Biemann, 2018; Chung et al., 2016; Hsu et al., 2017). A large portion of these approaches leverage self-supervised training, where the learning target is generated from the input itself, and thus can train a model in a *supervised* manner.

Chung et al. (2019) propose a method called Autoregressive Predictive Coding (APC), which trains an RNN to predict a future frame that is n steps ahead of the current position given a context such as the past frames. The training target can be easily generated by right-shifting the input by n steps. Their intuition is that the model is required to produce a good summarization of the past and encode such knowledge in the hidden states so as to accomplish the objective. After training, the RNN hidden states are taken as the learned representations, and are shown to contain speech information such as phonetic and speaker content that are useful in a variety of speech tasks (Chung and Glass, 2020).

Following their intuition, in this work we aim to improve the generalization of the future frame prediction task by adding an auxiliary objective that serves as a regularization. We empirically demonstrate the effectiveness of our approach in making more accurate future predictions, and confirm such improvement leads to a representation that contains richer phonetic content.

The rest of the paper is organized as follows. We start with a brief review of APC in Section 2. We then introduce our approach in Section 3. Experiments and analysis are presented in Section 4, followed by our conclusions in Section 5.

2 Autoregressive Predictive Coding

Given a context of a speech signal represented as a sequence of acoustic feature vectors $(x_1, x_2, \ldots, x_t)$, the objective of Autoregressive Predictive Coding (APC) is to use the context to infer a future frame x_{t+n} that is n steps ahead of x_t. Let $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ denote a full utterance, where N is the sequence length, APC incorporates an RNN to process each frame x_t sequentially and update its hidden state h_t accordingly. For $t = 1, \ldots, N - n$, the RNN produces

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 2353–2358

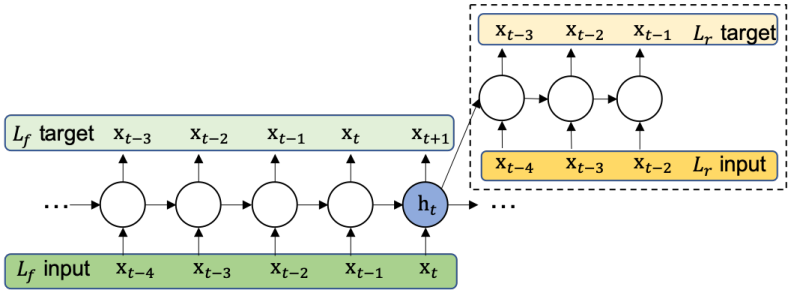


Figure 1: Overview of our method. L_f is the original APC objective that aims to predict x_{t+n} given a context $(x_1, x_2, \ldots, x_t)$ with an autoregressive RNN. Our method first samples an anchor position, assuming it is time step t. Next, we build an auxiliary loss L_r that computes L_f of a past sequence $(x_{t-s}, x_{t-s+1}, \ldots, x_{t-s+\ell-1})$ (see Section 3.1 for definitions of s and ℓ), using an auxiliary RNN (dotted line area). In this example, $(n, s, \ell) = (1, 4, 3)$. In practice, we can sample multiple anchor positions, and averaging over all of them gives us the final L_r.

an output $y_t = \mathbf{W} \cdot h_t$, where $\mathbf{W}$ is an affinity matrix that maps h_t back to the dimensionality of x_t. The model is trained by minimizing the frame-wise L1 loss between the predicted sequence $(y_1, y_2, \ldots, y_{N-n})$ and the target sequence $(x_{1+n}, x_{2+n}, \ldots, x_N)$:

$$L_f(\mathbf{x}) = \sum_{t=1}^{N-n} |x_{t+n} - y_t|. \quad (1)$$

When $n = 1$, one can view APC as an *acoustic* version of neural LM (NLM) (Mikolov et al., 2010) by thinking of each acoustic frame as a token embedding, as they both use a recurrent encoder and aim to predict information about the future. A major difference between NLM and APC is that NLM infers tokens from a closed set, while APC predicts frames of real values.

Once an APC model is trained, given an utterance $(x_1, x_2, \ldots, x_N)$, we follow Chung et al. (2019) and take the output of the last RNN layer $(h_1, h_2, \ldots, h_N)$ as its extracted features.

3 Proposed Methodology

Our goal is to make APC's prediction of x_{t+n} given h_t more accurate. In Section 4 we will show this leads to a representation that contains richer phonetic content.

3.1 Remembering more from the past

An overview of our method is depicted in Figure 1. We propose an auxiliary loss L_r to improve the generalization of the main objective L_f (Equation 1).

The idea of L_r is to refresh the current hidden state h_t with the knowledge learned in the past. At time step t, we first sample a past sequence $\mathbf{p}_t = (x_{t-s}, x_{t-s+1}, \ldots, x_{t-s+\ell-1})$, where s is how far the start of this sequence is from t and ℓ is the length of $\mathbf{p}_t$. We then employ an auxiliary RNN, denoted as RNN_{aux}, to perform predictive coding defined in Equation 1 conditioning on h_t. Specifically, we initialize the hidden state of RNN_{aux} with h_t, and optimize it along with the corresponding $\mathbf{W}_{\text{aux}}$ using $L_f(\mathbf{p}_t)$, which equals to $\sum_{t'=t-s}^{t-s+\ell-1} |x_{t'+n} - y_{t'}|$. Such a process reminds h_t of what has been learned in $h_{t-s}, h_{t-s+1}, \ldots, h_{t-s+\ell-1}$.

For a training utterance $\mathbf{x} = (x_1, x_2, \ldots, x_N)$, we select each frame with probability P as an anchor position. Assume we end up with M anchor positions: $a_1, a_2, \ldots, a_M$. Each a_m defines a sequence $\mathbf{p}_{a_m} = (x_{a_m-s}, x_{a_m-s+1}, \ldots, x_{a_m-s+\ell-1})$ before x_{a_m}, which we use to compute $L_f(\mathbf{p}_{a_m})$. Averaging over all anchor positions gives the final auxiliary loss L_r:

$$L_r(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} L_f(\mathbf{p}_{a_m}). \quad (2)$$

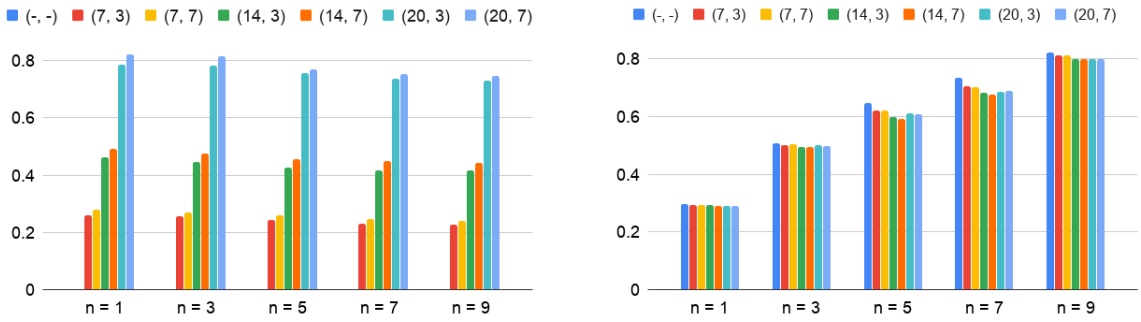
The final APC objective combines Equations 1 and 2 with a balancing coefficient λ:

$$L_m(\mathbf{x}) = L_f(\mathbf{x}) + \lambda L_r(\mathbf{x}). \quad (3)$$

We re-sample the anchor positions for each $\mathbf{x}$ during each training iteration, while they all share the same RNN_{aux} and $\mathbf{W}_{\text{aux}}$.

4 Experiments

We demonstrate the effectiveness of L_r in helping optimize L_f, and investigate how the improvement is reflected in the learned representations.



(a) L_r (auxiliary objective, Equation 2)

(b) L_f (main objective, Equation 1)

Figure 2: Validation loss of L_r (left) and L_f (right) on LibriSpeech `dev-clean` when training APC using different (n, s, ℓ) combinations. Each bar of the same color represents one (s, ℓ) combination. We use $(-, -)$ to denote an APC optimized only with L_f. Bars are grouped by their n's with different (s, ℓ) combinations within each group.

4.1 Setup

We follow Chung et al. (2019) and use the audio portion of the LibriSpeech (Panayotov et al., 2015) `train-clean-360` subset, which contains 360 hours of read speech produced by 921 speakers, for training APC. The input features are 80-dimensional log Mel spectrograms, i.e., $x_t \in \mathbb{R}^{80}$. Both RNN and RNN_{aux} are a 3-layer, 512-dim unidirectional GRU (Cho et al., 2014) network with residual connections between two consecutive layers (Wu et al., 2016). Therefore, $\mathbf{W}, \mathbf{W}_{\text{aux}} \in \mathbb{R}^{512\times80}$. λ is set to 0.1 and the sampling probability P is set to 0.15, that is, each frame has a 15% of chance to be selected as an anchor position. P and λ are selected based on the validation loss of L_f on a small data split. All models are trained for 100 epochs using Adam (Kingma and Ba, 2015) with a batch size of 32 and a learning rate of 10^{-3}.

4.2 Effect of L_r

We first validate whether augmenting L_r improves L_f. As a recap, n is the number of time steps ahead of the current position t in L_f, and s and ℓ denote the start and length, respectively, of a past sequence before t to build L_r. We consider $(n, s, \ell) \in \{1, 3, 5, 7, 9\} \times \{7, 14, 20\} \times \{3, 7\}$. Note that each phone has an average duration of about 7 frames.

Figures 2a and 2b present L_r (before multiplying λ) and L_f of the considered APC variants on the LibriSpeech `dev-clean` subset, respectively. Each bar of the same color represents one (s, ℓ) combination. We use $(-, -)$ to denote an APC optimized only with L_f. Bars are grouped by their n's with different (s, ℓ) combinations within each group.

We start with analyzing Figure 2a. Note that L_r does not exist for $(-, -)$ and is set to 0 in the figure. We see that under the same n, the performance of L_r is mainly decided by how far (s) the past sequence is from the current position rather than the length (ℓ) to generate: when we keep ℓ fixed and increase s from 7 (red), 14 (green), to 20 (blue), we observe the loss surges as well.

From Figure 2b, we have the following findings.

For a small n, the improvement in L_f brought by L_r is minor. By comparing $(-, -)$ with other bars, we see that when $n \leq 3$, which is smaller than half of the average phone duration (7 frames), adding L_r does not lower L_f by much. We speculate that when $n \leq 3$, x_{t+n} to be inferred is usually within the same phone as x_t, making the task not challenging enough to force the model to leverage more past information.

L_r becomes useful when n gets larger. We see that when n is close to or exceeds the average phone duration ($n \geq 5$), an evident reduction in L_f after adding L_r is observed, which validates the effectiveness of L_r in assisting with the optimization of L_f. When $n = 9$, the improvement is not as large as when $n = 5$ or 7. One possible explanation is that x_{t+9} has become almost independent from the previous context h_t and hence is less predictable.

By observing the validation loss, we have shown that L_r indeed helps generalize L_f.

4.3 Learned representation analysis

Next, we want to examine whether an improvement in L_f leads to a representation that encodes more useful information. Speech signals encompass a rich set of acoustic and linguistic properties. Here

Feature	Time shift						
	-15	-10	-5	0	+5	+10	+15
log Mel	83.3	80.3	67.6	49.9	65.5	77.9	82.7
APC trained with L_f (Equation 1)							
$n = 1$	56.1	45.8	36.1	33.7	56.5	73.7	81.6
$n = 3$	50.8	41.8	34.8	33.4	56.0	73.5	81.1
$n = 5$	48.7	38.2	32.5	31.9	54.8	73.0	80.5
$n = 7$	44.6	38.6	32.9	32.1	56.3	73.8	80.4
$n = 9$	51.0	41.8	35.7	36.9	58.4	74.6	81.0
APC trained with L_m (Equation 3)							
$n = 1$	50.6	42.2	35.1	33.1	54.4	73.4	81.4
$n = 3$	46.4	38.0	34.1	32.4	54.1	71.4	80.5
$n = 5$	41.8	35.1	29.8	28.1	49.6	64.6	76.8
$n = 7$	39.8	33.8	28.7	27.8	46.8	60.6	74.4
$n = 9$	42.3	35.3	30.3	29.7	50.0	63.3	76.6

Table 1: Phonetic classification results using different types of features as input to a linear logistic regression classifier. The classifier aims to correctly classify each frame into one of the 48 phone categories. Frame error rates (↓) are reported. Given a time shift $w \in \{0, \pm5, \pm10, \pm15\}$, the classifier is asked to predict the phone identity of x_{t+w} given x_t.

we will only focus on analyzing the phonetic content contained in a representation, and leave other properties such as speaker for future work.

We use phonetic classification on TIMIT (Garofolo et al., 1993) as the probing task to analyze the learned representations. The corpus contains 3696, 400, and 192 utterances in the train, validation, and test sets, respectively. For each $n \in \{1, 3, 5, 7, 9\}$, we pick the (s, ℓ) combination that has the lowest validation loss. As described in Section 2, we take the output of the last RNN layer as the extracted features, and provide them to a linear logistic regression classifier that aims to correctly classify each frame into one of the 48 phone categories. During evaluation, we follow the protocol (Lee and Hon, 1989) and collapse the prediction to 39 categories. We report frame error rate (FER) on the test set, which indicates how much phonetic content is contained in the representations. We also conduct experiments for the task of predicting x_{t-w} and x_{t+w} given x_t for $w \in \{5, 10, 15\}$. This examines how contextualized h_t is, that is, how much information about the past and future is encoded in the current feature h_t. We simply shift the labels in the dataset by $\{\pm5, \pm10, \pm15\}$ and retrain the classifier. We keep the pre-trained APC RNN fixed for all runs. Results are shown in Table 1.

We emphasize that our hyperparameters are chosen based on L_f and are never selected based on their performance on any downstream task, including phonetic classification, speech recognition, and speech translation to be presented next. Tuning hyperparameters towards a downstream task defeats the purpose of unsupervised learning.

Phonetic classification We first study the standard phonetic classification results, shown in the column where time shift is 0. We see that APC features, regardless of the objective (L_f or L_m), achieve lower FER than log Mel features, showing that the phonetic information contained in the surface features has been transformed into a more accessible form (defined as how linearly separable they are). Additionally, we see that APC features learned by L_m outperform those learned by L_f across all n. For $n \geq 5$ where there is a noticeable improvement in future prediction after adding L_r as shown in Figure 2b, their improvement in phonetic classification is also larger than when $n \leq 3$. Such an outcome suggests that APC models that are better at predicting the future do learn representations that contain richer phonetic content. It is also interesting that when using L_f, the best result occurs at $n = 5$ (31.9); while with L_m, it is when $n = 7$ that achieves the lowest FER (27.8).

Predicting the past or future We see that it is harder to predict the nearby phone identities from a log Mel frame, and the FER gets higher further away from the center frame. An APC feature h_t contains more information about its past than its future. The result matches our intuition as the RNN generates h_t conditioning on h_i for $i < t$ and thus their information are naturally encoded in h_t. Furthermore, we observe a consistent improvement in

both directions by changing L_f to L_m across all n and time shifts. This confirms the use of L_r, which requires the current hidden state h_t to recall what has been learned in previous hidden states, so more information about the past is encoded in h_t. The improvement also suggests that an RNN can forget the past information when training only with L_f, and adding L_r alleviates such problem.

4.4 Speech recognition and translation

The above phonetic classification experiments are meant for analyzing the phonetic properties of a representation. Finally, we apply the representations learned by L_m to automatic speech recognition (ASR) and speech translation (ST) and show their superiority over those learned by L_f.

We follow the exact setup in Chung and Glass (2020). For ASR, we use the Wall Street Journal corpus (Paul and Baker, 1992), use `si284` for training, and report the word error rate (WER) on `dev93`. For ST, we use the LibriSpeech En-Fr corpus (Kocabiyikoglu et al., 2018), which aims to translate an English speech to a French text, and report the BLEU score (Papineni et al., 2002). For both tasks, the downstream model is an end-to-end, sequence-to-sequence RNN with attention (Chorowski et al., 2015). We compare different input features to the same model. Results, shown in Table 2, demonstrate that the improvement in predictive coding brought by L_r not only provides representations that contain richer phonetic content, but are also useful in real-world speech applications.[1]

Feature	ASR (WER ↓)	ST (BLEU ↑)
log Mel	18.3	12.9
APC w/ L_f	15.2	13.8
APC w/ L_m	14.2	14.5

Table 2: Automatic speech recognition (ASR) and speech translation (ST) results using different types of features as input to a seq2seq with attention model. Word error rates (WER, ↓) and BLEU scores (↑) are reported for the two tasks, respectively.

5 Conclusions

We improve the generalization of Autoregressive Predictive Coding by multi-target training of future prediction L_f and past memory reconstruction L_r, where the latter serves as a regularization. Through phonetic classification, we find the representations learned with our approach contain richer phonetic content than the original representations, and achieve better performance on speech recognition and speech translation.

[1] According to Chung and Glass (2020), when using a Transformer architecture (Vaswani et al., 2017; Liu et al., 2018) as the autoregressive model, representations learned with L_f can achieve a WER of 13.7 on ASR and a BLEU score of 14.3 on ST.

References

Alexei Baevski, Michael Auli, and Abdelrahman Mohamed. 2019. Effectiveness of self-supervised pre-training for speech recognition. *arXiv preprint arXiv:1911.03912*.

Alexei Baevski, Steffen Schneider, and Michael Auli. 2020. vq-wav2vec: Self-supervised learning of discrete speech representations. In *ICLR*.

Yi-Chen Chen, Sung-Feng Huang, Chia-Hao Shen, Hung-Yi Lee, and Lin-Shan Lee. 2018. Phonetic-and-semantic embedding of spoken words with applications in spoken content retrieval. In *SLT*.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *NIPS*.

Jan Chorowski, Ron Weiss, Samy Bengio, and Aäron van den Oord. 2019. Unsupervised speech representation learning using wavenet autoencoders. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(12):2041–2053.

Yu-An Chung and James Glass. 2018. Speech2vec: A sequence-to-sequence framework for learning word embeddings from speech. In *Interspeech*.

Yu-An Chung and James Glass. 2020. Generative pre-training for speech with autoregressive predictive coding. In *ICASSP*.

Yu-An Chung, Wei-Ning Hsu, Hao Tang, and James Glass. 2019. An unsupervised autoregressive model for speech representation learning. In *Interspeech*.

Yu-An Chung, Wei-Hung Weng, Schrasing Tong, and James Glass. 2018. Unsupervised cross-modal alignment of speech and text embedding spaces. In *NeurIPS*.

Yu-An Chung, Chao-Chung Wu, Chia-Hao Shen, Hung-Yi Lee, and Lin-Shan Lee. 2016. Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder. In *Interspeech*.

John Garofolo, Lori Lamel, William Fisher, Jonathan Fiscus, David Pallett, and Nancy Dahlgren. 1993. DARPA TIMIT acoustic-phonetic continuous speech corpus. Technical Report NISTIR 4930, NIST.

Wei-Ning Hsu, Yu Zhang, and James Glass. 2017. Unsupervised learning of disentangled and interpretable representations from sequential data. In *NIPS*.

Dongwei Jiang, Xiaoning Lei, Wubo Li, Ne Luo, Yuxuan Hu, et al. 2019. Improving Transformer-based speech recognition using unsupervised pre-training. *arXiv preprint arXiv:1910.09932*.

Herman Kamper. 2019. Truly unsupervised acoustic word embeddings using weak top-down constraints in encoder-decoder models. In *ICASSP*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Ali Kocabiyikoglu, Laurent Besacier, and Olivier Kraif. 2018. Augmenting LibriSpeech with French translations: A multimodal corpus for direct speech translation evaluation. In *LREC*.

Kai-Fu Lee and Hsiao-Wuen Hon. 1989. Speaker-independent phone recognition using hidden markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(11):1641–1648.

Andy Liu, Shu-Wen Yang, Po-Han Chi, Po-Chun Hsu, and Hung-Yi Lee. 2020. Mockingjay: Unsupervised speech representation learning with deep bidirectional Transformer encoders. In *ICASSP*.

Peter Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In *ICLR*.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*.

Benjamin Milde and Chris Biemann. 2018. Unspeech: Unsupervised speech context embeddings. In *Interspeech*.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An ASR corpus based on public domain audio books. In *ICASSP*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *ACL*.

Santiago Pascual, Mirco Ravanelli, Joan Serrà, Antonio Bonafonte, and Yoshua Bengio. 2019. Learning problem-agnostic speech representations from multiple self-supervised tasks. In *Interspeech*.

Douglas Paul and Janet Baker. 1992. The design for the wall street journal-based CSR corpus. In *Speech and Natural Language Workshop*.

Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. 2019. wav2vec: Unsupervised pre-training for speech recognition. In *Interspeech*.

Xingchen Song, Guangsen Wang, Zhiyong Wu, Yiheng Huang, Dan Su, et al. 2019. Speech-XLNet: Unsupervised acoustic model pretraining for self-attention networks. *arXiv preprint arXiv:1910.10387*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc Le, Mohammad Norouzi, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.