Interspeech 2018
2-6 September 2018, Hyderabad

Improving Language Modeling with an Adversarial Critic for Automatic Speech Recognition

Yike Zhang[1,2], *Pengyuan Zhang*[1,2], *Yonghong Yan*[1,2,3]

[1]Institute of Acoustics, Chinese Academy of Sciences, China
[2]University of Chinese Academy of Sciences, China
[3]Xinjiang Technical Institute of Physics and Chemistry, Chinese Academy of Sciences
{zhangyike,zhangpengyuan}@hccl.ioa.ac.cn

Abstract

Recurrent neural network language models (RNN LMs) trained via the maximum likelihood principle suffer from the exposure bias problem in the inference stage. Therefore, potential recognition errors limit their performance on rescoring N-best lists of the speech recognition outputs. Inspired by the generative adversarial net (GAN), this paper proposes a novel approach to alleviate this problem. We regard the RNN LM as a generative model in the training stage. And an auxiliary neural critic is used to encourage the RNN LM to learn long-term dependencies by forcing it generating valid sentences. Since the vanilla GAN has limitations when generating discrete sequences, the proposed framework is optimized through the policy gradient algorithm. Experiments were conducted on two mandarin speech recognition tasks. Results show the proposed method achieved lower character error rates on both datasets compared with the maximum likelihood method, whereas it increased perplexities slightly. Finally, we visualised the sentences generated from RNN LMs. Results demonstrate the proposed method really helps the RNN LM to learn long-term dependencies and alleviates the exposure bias problem partly.

Index Terms: speech recognition, language modeling, generative adversarial networks, policy gradients

1. Introduction

Recurrent neural network language models (RNN LMs) have achieved impressive results in speech recognition. They are usually trained via the maximum likelihood principle. However, log-likelihood training methods are limited by the discrepancy between training and inference. More specifically, the RNN LM takes ground-truth tokens as inputs to predict later outputs in the training stage. Whereas in the inference stage, N-best lists of speech recognition outputs are fed into the RNN LM iteratively. However, the N-best lists have been already corrupted by recognition errors, and they are usually never seen in the training data. This can lead to poor rescoring performance as recognition errors accumulate in the RNN LM's hidden states. To be more specific, the RNN LM might assign high probabilities to undesired tokens rather than the ground-truth one conditioned on corrupted contexts.

The discrepancy between training and inference is also called exposure bias in the sequence prediction [1, 2]. Lamb *et al.* used the generative adversarial net (GAN) [3] and adversarial domain adaptation [4, 5] to force the RNN behaving similarly during training and inference [6]. However, the GAN is designed for generating continuous data and has difficulties in directly generating discrete outputs. The actor-critic algorithm is another way to alleviate the exposure bias problem [7], and it can deal with discrete outputs. However, one drawback of the actor-critic method is well designed rewards are required. In practice, appropriate rewards are usually unavailable. Currently, Yu *et al.* adopted the gradient policy algorithm to optimize the GAN with discrete outputs [8, 9]. And they bypassed the crafted rewards in the actor-critic method by using the discriminator's outputs as rewards.

Inspired by the GAN, this paper proposes a novel method for language modeling, which aims to alleviate the exposure bias problem during rescoring. We regard the RNN LM as a stochastic policy in the training stage. And a convolutional neural network (CNN) is used as an auxiliary critic to distinguish whether a given sequence is generated via the RNN LM or sampled from the training data. Similar to Refs. [8, 9], we take the critic's output as rewards. But in order to prevent the RNN LM from diverging from the maximum likelihood estimation, we take the rewards as a regularization term of the log-likelihood loss.

Like the maximum likelihood method, the proposed method still forces the RNN LM to learn patterns in the training data. But in addition, the proposed method also encourages the RNN LM to learn long-term dependencies by guiding the RNN LM to generate valid sentences. And thereby the proposed method can alleviate the exposure bias problem during rescoring. Namely, the proposed method can improve the RNN LM's robustness for recognition errors in N-best lists.

Finally, we conducted experiments on two mandarin speech recognition datasets, THCHS30 [10] and AISHELL [11]. Compared with the maximum likelihood method, the proposed method achieved better results on character error rate (CER). but increased perplexities slightly. Additionally, we visualized the sentences sampled from the RNN LMs. Results demonstrate the proposed method helps to alleviate the exposure bias problem partly.

In the remaining part, we first describe the details of the proposed method in Section 2. Then, experiments and discussions are shown in Section 3. Finally, Section 4 concludes the paper.

2. The proposed method

2.1. Formulations

Given a sequence $y_1, y_2, \cdots, y_T$, the RNN LM decomposes the probability of the sentence into a product of conditional distribution over tokens

$$P(y_1, y_2, \cdots, y_T) = \prod_{t=1}^{T} P(y_t | y_{<t}) \tag{1}$$

Under the maximum likelihood principle, the RNN LM are

10.21437/Interspeech.2018-1111

trained to minimize the negative log probability of the given sequence

$$\text{NLL} = -\sum_{t=1}^{T} \log P(y_t|y_{<t}) \tag{2}$$

The RNN LM can be also regarded as a generative model. It generates sequences in an iterative way, in which the current token is predicted conditioned on previous generated tokens. We can also interpret the sequence generation problem from the perspective of reinforcement learning. At timestep t, the state s is the tokens $y_1, \cdots, y_{t-1}$ that have been produced at present, and the action a is the next token y_t to select. In this paper, the θ-parameterized RNN LM R_θ acts as the policy to determine how to select a proper action conditioned on the current state. And a ϕ-parameterized critic C_ϕ is used to distinguish whether a given sequence is sampled from the training data or generated via the policy. Given a sequence l, the critic's output $C_\phi(l)$ stands for a probability that l comes from real data rather than the policy. And $C_\phi(l)$ is used as the reward to update R_θ.

On the basis of reinforcement learning, the objective of the policy R_θ is to produce a sequence with T tokens to maximize its expected reward

$$L(\theta) = \mathbb{E}[\sum_{t=1}^{T} R_\theta(y_t|s_t)Q(y_t, s_t)] \tag{3}$$

where $Q(y, s)$ stands for the expected reward to produce token y given state s. When $s = y_1, ..y_{T-1}$, the output of C_ϕ can be used as the reward directly

$$Q(y = y_T, s = y_1, ..., y_{T-1}) = C_\phi(y_{1:T}) \tag{4}$$

Since C_ϕ can only evaluate completed sequences, it cannot be used to estimate rewards of intermediate states. However, intermediate states also play an important part because they impact on the future tokens to be generated and thereby affect the entire sequence.

To deal with the problem, we apply Monte Carlo search [12] with a rollout policy R_γ to sample the unknown last $T - t$ tokens. In this paper, R_θ and R_γ have the same architecture, but one can use any variants of R_θ as the rollout policy. Formally, a N-time Monte Carlo search can be represented as follow

$$\{y^1_{1:T}, .., y^N_{1:T}\} = \text{MC}_{R_\gamma}(y_{1:t}; N) \tag{5}$$

where $y^n_{1:t}$ represents the current state and $y^n_{t+1:T}$ is sampled from R_γ based on $y^n_{1:t}$. In order to reduce the variance, we sample based on the rollout policy from the current state to the end of the sequence for N times. Thus, the intermediate reward can be represented as

$$Q(a = y_t, s = y_{1:t-1}) = \begin{cases} \frac{1}{N}\sum_{n=1}^{N} C_\phi(y^n_{1:T}), y^n_{1:T} \in \text{MC}_{R_\gamma}(y_{1:t}; N), & t < T \\ C_\phi(y_{1:T}), & t = T \end{cases} \tag{6}$$

In order to keep the policy stay around the maximum likelihood estimation, we adopt the expected rewards as a regularization term. And the joint objective function is defined as follow

$$\begin{aligned} J(\theta) &= \text{NLL} - \mu L(\theta) \\ &= -\sum_{t=1}^{T} \log P(y_t|y_{<t}) - \mu\mathbb{E}[\sum_{t=1}^{T} R_\theta(\widetilde{y}_t|\widetilde{y}_{<t})Q(\widetilde{y}_t, \widetilde{y}_{<t})] \end{aligned} \tag{7}$$

where $\widetilde{y}$ represents the token generated via R_θ, and y is sampled from the training data. μ is a scalar to balance the effect between the log-likelihood and the expected rewards, and its value is dependent on the length of training sequences and the number of the Monte Carlo search for each intermediate state.

The joint objective function consists of two components. The first one stands for the maximum likelihood estimation on the training data, while the second one is designed to reduce the discrepancy of the RNN policy between training and inference. These two components are obtained in different ways. The negative log-likelihood is equivalent to the cross entropy between the ground-truth tokens and the policy's outputs. The expected rewards can be computed according to Eq.(3) by regarding the policy as a generative model. The key difference is that ground-truth tokens are fed into the policy when the negative log-likelihood is computed, while the policy takes previous generated tokens as input when we compute the expected rewards. More details are presented in Section 2.3.

After training the policy for several iterations, it can produce higher quality sequences. Then, we re-train the critic by maximizing

$$L(\phi) = \mathbb{E}[\log C_\phi(y_{1:T})] + \mathbb{E}[\log(1 - C_\phi(\widetilde{y}_{1:T}))] \tag{8}$$

Since the critic is dynamically updated during the training stage, using it as the reward function can improve the policy iteratively. Finally, we update the rollout policy by

$$\gamma^{t+1} = \gamma^t + \alpha\theta^t \tag{9}$$

where γ^t and θ^t represent the parameters of the rollout policy and the policy in iteration t respectively, and α is the update rate.

2.2. Model architectures

The RNN with long short-term memory (LSTM) cells [13] dominates many fields including language modeling. Therefore, this paper adopts the LSTM-RNN as the policy, and the LSTM has the same architecture as that used in Ref. [14].

Additionally, we adopt a CNN as the critic, since CNNs have achieved great success in text classification tasks recently [15, 16]. The critic takes a sequences as input, and outputs a probability that stands for the input coming from the training data rather than the policy. Given a sequence $y_1, y_2, \cdots, y_T$, a 2-D input matrix for the critic is obtained by

$$\varepsilon_{1:T} = e_1 \oplus e_2 \oplus ... \oplus e_T \tag{10}$$

where $e_t \in \mathbf{R}^k$ is the word embedding of the token y_t. $\oplus$ represents the concatenation operator. Then, a kernel $r \in \mathbf{R}^{l\times k}$ applies a convolutional operation to a windows size of l words

$$c_i = \rho(r * \varepsilon_{i:i+l-1} + b) \tag{11}$$

where $*$ represents the convolutional operation, ρ stands for a non-linear function and b denotes the bias. We can get different feature maps by applying various number of kernels with different window sizes to ε. Subsequently, a max-over-time pooling operation is applied to each feature map

$$\tilde{c} = \max\{c_1, ..., c_{T-l+1}\} \tag{12}$$

In addition, we put a highway layer [17] over the pooling layer to improve the critic's performance. Finally, a dense layer followed by a sigmoid function outputs a single value which represents the probability that the input is from the training data.

2.3. Training strategies

We find that a warm start has an important effect on the proposed method. A warm start for the policy can efficiently reduce the variance of the expected rewards and speed up convergence. Furthermore, a warm start also helps the critic to rapidly reach optimum in each adversarial training iteration, which can provide a more efficient guidance for the policy. Algorithm 1 shows the details of the proposed method.

Algorithm 1 Improving RNN LM with an adversarial critic

Require: training data D; policy R_θ; rollout policy R_γ; critic C_ϕ;
Pre-train R_θ on D by Eq.(2)
$R_\gamma \leftarrow R_\theta$
Pre-train C_ϕ by Eq.(8) with negative examples from R_θ and positive examples from D
repeat
  for $i = 0; i < n; i++$ **do**
    Sample negative examples from R_θ
    Sample positive examples from D
    Train critic C_ϕ by Eq.(8)
  end for
  for $j = 0; j < m; j++$ **do**
    Sample positive examples from D
    Compute negative log-likelihood on positive examples
    Sample negative examples from R_θ
    Compute (intermediate) rewards by Eq.(6)
    Train policy R_θ by Eq.(7)
  end for
  $\gamma \leftarrow \gamma + \alpha\theta$
until R_θ converges

3. Experiments

3.1. Experimental settings

We experimented on two mandarin news speech databases. THCHS30 contains 30 hours speech and there are 10000/893/2459 utterances in the training/development/test set. AISHELL has a total of 170 hours speech and the training/development/test set contains 120098/14326/7176 utterances. In experiments, we limited the vocabulary to the most frequent 55590 words in transcriptions for LSTM LMs and trained ngram LMs with the unlimited vocabulary (more than 130k tokens). For each corpus, a Kneser-Ney smoothed tri-gram LM (KN3) trained with transcriptions was used in the first-pass decoding. Since transcriptions are too little to train LSTM LMs, we crawled some news articles (about 6M tokens) from the internet. Then we augmented the transcriptions with the crawled data as the training data for the LSTM LM and the critic. The baseline speech recognition systems were built by the Kaldi toolkit [18], ngram LMs were estimated by the SRILM toolkit [19], and the LSTM LM and the critic were implemented based on TensorFlow [20].

The LSTM LM included two recurrent layers, each had 150 LSTM units. As for comparison, another LSTM LM with the same architecture was trained via the maximum likelihood principle on the same dataset. As for the critic, we used various kernel sizes $\{1, 2, 3, 4, 5, 10\}$ to extract different information. For each kernel size, 50 different kernels were used. In addition, a highway layer with 150 units was adopted.

After pre-training the policy for 10 epochs and the critic for 5 epochs, we totally conducted the adversarial training for 1,000 iterations. In each iteration, the critic was trained for three times, and the policy was trained for one time. And for each iteration, we sampled different negative examples and positive examples. The Monte Carlo search was conducted 10 times for each intermediate state. The scalar μ was set to 0.1. The update rate of the rollout policy was set to 0.5. The training sequences were clipped to a maximum length of 20, and batch size was 35. The learning rate for the LSTM LM and the critic was set to 0.0003 and 0.0001 respectively. In order to avoid over-fitting, a dropout of 0.25 was used for both the LSTM LM and the critic.

3.2. Perplexities

We first evaluated the LSTM LMs trained via the proposed method and the maximum likelihood method on perplexities. Since the LSTM LMs are also generative models, we can also indirectly evaluated the proposed method from the generated text. For each LSTM LM, we sampled 1M sentences, and estimated a Kneser-Ney smoothed tri-gram LM (sKN3) with the generated data.

Since ngram LMs and LSTM LMs had different vocabularies, it is invalid to directly compare their perplexities. In fact, we just want to compare the proposed method with the maximum likelihood method rather than to compare different LMs. Therefore, we calculated the relative/absolute improvement of the proposed method compared to the maximum likelihood method for each pair of LMs. It is valid since the two LMs to be compared always shared the same vocabulary. Full results are shown in table 1. And a negative value represents the proposed method achieved in a higher perplexity compared with the maximum likelihood method.

Table 1: *Relative/Absolute perplexity improvement of the proposed method compared to the maximum likelihood method.*

	THCHS30	**AISHELL**
sKN3	9.1%/37.4	-1.7%/-9.0
LSTM	-3.8%/-6.9	-2.8%/-5.7

Unlike the maximum likelihood method, the proposed method is not designed to reduce the perplexity. Hence, we are not surprised to see the proposed method achieved slightly higher perplexities on both datasets. During inference, we also found the proposed method assigned infinitesimal probabilities to some words in both test sets. This also degraded the performance of the proposed method, even if we clipped the infinitesimal probabilities to -99 (log-zero) in experiments. This phenomenon indicates the proposed method resulted in a sharper distribution over the vocabulary, and degraded the smoothing nature and generalization of LSTM LMs. This may suffer from the biased samples from the policy. In particular, if a certain token is encouraged by the policy gradients, it is more likely to be sampled later. And if the token is sampled again, it will be encouraged by the policy gradients once more. Experimental results also show that falling back to tri-gram LMs (sKN3) can alleviate the problem, since tri-gram LMs take the advantages of additional smoothing algorithms.

3.3. N-best rescoring

We first generated 100 best lists from each baseline system. Then, we rescored the candidate hypotheses with the LMs listed in Table 1. During rescoring, we also clipped the infinitesimal

probabilities to -99. And we found this almost had no effect on the CER, since the infinitesimal probabilities hardly appeared in top candidates. In experiments, we interpolated different LMs log-linearly, since LSTM LMs and ngram LMs had different vocabularies. And the interpolation weights were chosen to minimize the CER on each development set. Results are reported in Table 2. The superscript † represents the model was trained via the maximum likelihood method.

Table 2: *Character error rate comparison with different LMs adopted for rescoring.*

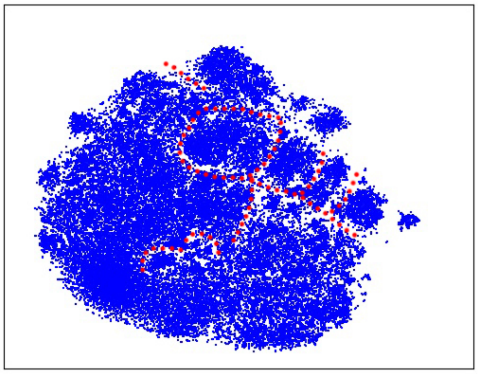
	THCHS30	**AISHELL**
KN3	42.5%	16.9%
KN3 + LSTM†	40.8%	15.7%
KN3 + sKN3†	40.8%	16.0%
KN3 + LSTM† + sKN3†	40.7%	15.5%
KN3 + LSTM	40.8%	15.3%
KN3 + sKN3	40.4%	15.7%
KN3 + LSTM + sKN3	40.4%	15.2%

From Table 2, we can see that the proposed method outperformed the maximum likelihood method on both test sets in all cases, excepted the LSTM LM on THCHS30. This is probably because many sentences in the THCHS30 test set are longer than the maximum sequence length used to train the LSTM LM. Experimental results also show the proposed method were complementary to the baseline models. Finally, by combining the LSTM LM with the sampled tri-gram LM (sKN3), the proposed method achieved the best performance on both test sets.

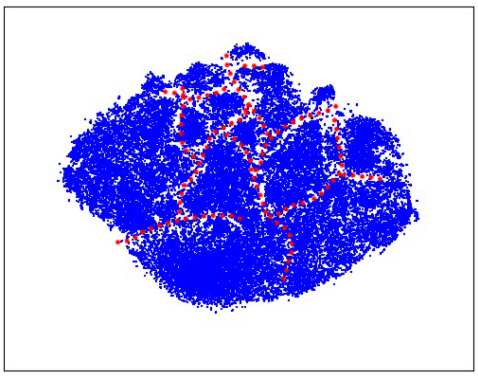
3.4. Visualizing generated text

We sampled 50k sentences from the two LSTM LMs respectively. Each sentence was represented by a 100-dimensional vector according to Ref. [21]. Then, the sentence vectors were reduced to 2-D via the t-SNE algorithm [22] and plotted in Figure 1. In practice, sentences with similar meanings always cluster together. And we outlined the boundaries of clusters with red dotted lines in Figure 1.

In Figure 1(a), the maximum likelihood method only yields a few clusters in the upper right part of the figure, while most sentences evenly spread throughout the lower left area. This is probably because the maximum likelihood method usually fails to generate sentences with long-term semantics. In other words, the generated sentences usually have no specific meanings, and there are no clear similarities among them. As a result, such sentences fail to gather into clusters. While in Figure 1(b), the proposed method yields more clusters in the entire figure area. This indicates there are more clear similarities among the generated sentences. It also hints the proposed method can alleviate the exposure bias problem and encourage the RNN LM to learn long-term dependencies.



(a) Sentences sampled from the LSTM LM trained via the maximum likelihood method.



(b) Sentences sampled from the LSTM LM trained via the proposed method.

Figure 1: *Visualizing sampled sentences via t-SNE algorithm.*

4. Conclusions

This paper proposes a GAN-based training method for RNN LMs to alleviate the exposure bias problem during rescoring. We model the RNN LM as a stochastic policy. And an auxiliary CNN critic is used to guide the RNN LM generating valid sentences in the training stage. By combining the log-likelihood loss with the rewards from the critic, the proposed method not only forces the RNN LM to learn patterns in the training data, but also encourages the RNN LM to learn long-term dependencies. We conducted experiments on two mandarin speech corpora. Results show the proposed method outperforms the maximum likelihood method in term of CER. Since the proposed method is not designed to optimize the perplexity, it results in slightly higher perplexities. However, a better model is usually characterized by a lower perplexity. Hence, designing a reward which can optimize the perplexity indirectly is a direction for further study.

5. Acknowledgements

The authors want to thank the anonymous reviewers for their helpful comments and suggestions. This work is partially supported by the National Natural Science Foundation of China (Nos.11590770-4, U1536117), the National Key Research and Development Plan (Nos.2016YFB0801203, 2016YFB0801200), the Key Science and Technology Project of the Xinjiang Uygur Autonomous Region (No.2016A03007-1), and the Pre-research Project for Equipment of General Information System (No.JZX2017-0994/Y306).

6. References

[1] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," *arXiv preprint arXiv:1511.06732*, 2015.

[2] M. Norouzi, S. Bengio, Z. Chen, N. Jaitly, M. Schuster, Y. Wu, and D. Schuurmans, "Reward augmented maximum likelihood for neural structured prediction," in *Advances in Neural Information Processing Systems 29*, 2016, pp. 1723–1731.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[4] H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, and M. Marchand, "Domain-adversarial neural networks," *arXiv preprint arXiv:1412.4446*, 2014.

[5] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[6] A. M. Lamb, A. G. A. P. GOYAL, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio, "Professor forcing: A new algorithm for training recurrent networks," in *Advances In Neural Information Processing Systems*, 2016, pp. 4601–4609.

[7] D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio, "An actor-critic algorithm for sequence prediction," *arXiv preprint arXiv:1607.07086*, 2016.

[8] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient." in *AAAI*, 2017, pp. 2852–2858.

[9] K. Lin, D. Li, X. He, Z. Zhang, and M.-T. Sun, "Adversarial ranking for language generation," in *Advances in Neural Information Processing Systems*, 2017, pp. 3155–3165.

[10] D. Wang and X. Zhang, "Thchs-30: A free chinese speech corpus," *arXiv preprint arXiv:1512.01882*, 2015.

[11] H. Bu, J. Du, X. Na, B. Wu, and H. Zheng, "Aishell-1: An open-source mandarin speech corpus and a speech recognition baseline," *arXiv preprint arXiv:1709.05522*, 2017.

[12] J. B. Chaslot, M. H. M. Winands, H. J. V. D. Herik, J. W. H. M. Uiterwijk, and B. Bouzy, "Progressive strategies for monte-carlo tree search," *New Mathematics & Natural Computation*, vol. 4, no. 03, pp. 343–357, 2008.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] M. Sundermeyer, H. Ney, and R. Schluter, "From feedforward to recurrent lstm neural networks for language modeling," *IEEE/ACM Transactions on Audio Speech & Language Processing*, vol. 23, no. 3, pp. 517–529, 2015.

[15] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[16] X. Zhang and Y. LeCun, "Text understanding from scratch," *arXiv preprint arXiv:1502.01710*, 2015.

[17] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.

[18] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on Automatic Speech Recognition and Understanding*, no. EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

[19] A. Stolcke, "Srilm-an extensible language modeling toolkit," in *Seventh International Conference on Spoken Language Processing*, 2002.

[20] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[21] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International Conference on Machine Learning*, 2014, pp. 1188–1196.

[22] L. Van Der Maaten, "Accelerating t-sne using tree-based algorithms," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3221–3245, 2014.