The Sogou-TIIC Speech Translation System for IWSLT 2018

Yuguang Wang[*], *Liangliang Shi*[*], *Linyu Wei*[*], *Weifeng Zhu*[*], *Jinkun Chen*[*]
Zhichao Wang[*], *Shixue Wen*[*], *Wei Chen*[*], *Yanfeng Wang*[*], *Jia Jia*[†]

[*]Voice Interaction Technology Center, Sogou Inc., Beijing, China
{wangyuguang,shiliangliang}@sogou-inc.com
[†]Tiangong Institute for Intelligent Computing, Tsinghua University, Beijing, China
{jjia}@tsinghua.edu.cn

Abstract

This paper describes our speech translation system for the IWSLT 2018 Speech Translation of lectures and TED talks from English to German task. The pipeline approach is employed in our work, which mainly includes the Automatic Speech Recognition (ASR) system, a post-processing module, and the Neural Machine Translation (NMT) system. Our ASR system is an ensemble system of Deep-CNN, BLSTM, TDNN, N-gram Language model with lattice rescoring. We report average results on tst2013, tst2014, tst2015. Our best combination system has an average WER of 6.73. The machine translation system is based on Google's Transformer architecture. We achieved an improvement of 3.6 BLEU over baseline system by applying several techniques, such as cleaning parallel corpus, fine tuning of single model, ensemble models and re-scoring with additional features. Our final average result on speech translation is 31.02 BLEU.

1. Introduction

We have participated in the Speech Translation of lectures and TED talks from English to German task. The goal of this task is to translate fully un-segmented talks or lectures from English to German.

A pipeline approach is employed in our work. It consists of segmentation of audio data, ASR system, punctuation restoration and NMT system. A two pass decoding is used in the ASR system. In the first pass, we use several different neural network, such as Deep-CNN [1] [2], BLSTM and TDNN [3] to generate ensemble results. Then the decoding lattices of the ensemble system are sent to a second pass decoder for lattice rescoring. In order to bridge the gap between the output of ASR system and training data of NMT system, punctuation restoration, disfluency detection and inverse text normalization are necessary in our pipeline. Our NMT system is based on the Transformer architecture [4], which is based solely on attention mechanisms. Several techniques are adopted to improve our system, such as parallel corpus cleaning, fine tuning, model ensembling and re-scoring with additional features.

The rest of this paper is structured as follows. Section 2 describes the details of our ASR system, and Section 3 describes our NMT system. Our results in the speech translation task are presented in Section 4. We conclude this paper in Section 5.

2. Automatic speech recognition

2.1. Audio Segmentation

In this evaluation, the test set is provided without manual sentence segmentation, thus automatic segmentation of the final test set is essential. We utilize an approach to automatic segment audio data based on the signal energy. We set a threshold to split the audio between 8 and 15 seconds and then concatenate utterances that are shorter than 8 seconds to its neighboring utterances.

2.2. Audio Data Preparation and Feature Extraction

2.2.1. Data Cleaning

Our acoustic data comes from two sources. The first is the TED-LIUM [5], which contains 340 hours of well transcribed data. The second part comes from Speech-Translation TED corpus, which is about 270 hours of data with some bad segments, e.g. music or transcriptions not comparing the wav files. We follow the way in kaldi toolkit [6] to do the cleaning[1]. This aimed to cut the bad part off and only retrain the segments that can be compared with the transcripts. And we got about 220 hours in this part.

2.2.2. Dereverbration

For speech dereverbration, we calculate the RT60 [7] of the speech firstly. The speech whose RT60 is longer than 400ms is filtered with the Kalman filtering algorithm [8] to dereverberate the speech. Thus we get about 11,000 kalman filtered utterances and add them to the original data.

2.2.3. Speed Perturbation

Speed perturbation is done with 1.1 and 0.9 times for all the data above. Finally we obtain about total 1700 hours acoustic data to get robust performance in the end.

2.2.4. Feature Extraction

Our acoustic feature engineering is not complicated. The system is built using several different features including 39-dimensional MFCC for GMM, 40-dimensional static MFCC and 80-dimensional filter banks for neural networks. These features can be augmented with i-vectors to train speaker

[1]https://github.com/kaldi-asr/kaldi/blob/master/egs/ami/s5b/local/run_cleanup _segmentation.sh

adapted networks. The dimension of i-vectors is chosen to be 200 which is extracted from every 50 frames. I-vectors and features are combined in the input layer for TDNNs and BLSTMs. While for CNN, input layer only contains filter bank and i-vectors are combined with the fully-connected layer before the output. We also extracted fMLLR transformed feature on 340hr TED-LIUM data and found fMLLR features contribute no significant improvement compared to i-vector augmented system. So we only use i-vector to train our final speaker adapted systems.

2.3. Acoustic Modeling

We use DNN-HMM hybrid acoustic model for all our ASR experiments. All NN systems were trained using Lattice-free MMI (LF-MMI) [9] loss function with low frame rate (LFR) equals to 3 to predict context dependent phones (bi-phone). We mainly used three different types of neural net architecture, including deep convolutional neural network (DCNN), bidirectional LSTM (BLSTM) and time delayed deep neural network (TDNN). In GMM-HMM part, we use 13-dimension mel frequency cepstral coefficient (MFCC) with first and second derivatives with 500 hours data without speed perturbation. The dictionary provided in the TED-LIUM dataset is used for our GMM training. The final GMM has totally 150,193 Gaussian mixtures, correspond to 4056 states. This 500hr GMM-HMM was used to align all the 1500h data to generate state alignments for clustering bi-phone labels used in LF-MMI training. After clustering we finally get 3144 bi-phones which equal the output nodes number of all our neural networks.

2.3.1. Deep CNN

We were inspired by the VGG net [10] and the deep CNN architecture used in [1] [2] to design our CNN model. We train our DCNN model with 80 dimension filter bank feature without first and second derivatives. We use batch normalization (BN) and ReLU nonlinear activations following each convolution layer. We stack 31 layer of such conv-BN-ReLU block with residual connections around every two of them. Most of the two dimensional time-frequency convolution kernels are all set to 3 x 3 with stride 1x1. We set kernel size to 5 x 5 with stride 2 x 1 in the 6th, 12th, 24th, 30th convolution layer to reduce the frequency dimension from 80 to 5. Every time we reduce the frequency dimension we double our kernel number. So as we go deeper, the kernel number is set to 32, 64, 128, 256, and 384. We train such DCNN with all the 1500h data to obtain system *dcnn*.

2.3.2. BLSTM

Our BLSTM model consists of 5 layers that has two unidirectional LSTM with 1024 cells and 512 projections. 256 of the projections are recurrent units and the other 256 projections are non-recurrent ones. 40 dimensional static MFCC feature is extracted for BLSTM training. By concatenating the two previous and the two following frames of MFCC, we use 40 * 5 = 200 dimension feature to train two BLSTM with different random seeds using all of the 1500h data. We call the two BLSTM with *blstm1* and *blstm2*.

2.3.3. TDNN

For TDNN neural acoustic model, we use factored form of TDNN [3] to design our own network. The factorized TDNN (TDNN-F) is reported to beat common TDNN with deeper architecture [3], in order to identify some hyper parameter configuration, we first train TDNN-F models with only the TED-LIUM data and decode with a relatively small n-gram language model. We summarize the intermediate result in table 1. Here we only report average WER of tst2013, tst2014 and tst2014.

As can be seen from the table, it is beneficial to use i-vector or fMLLR transformed feature to train speaker adapted networks. Comparing the third row and the forth row, we find WER of fMLLR system is 15.09 which is worse than i-vector augmented system of 14.23. As we gradually increase the number of layers from 16 to 26, a steady performance improvement is obtained. TDNNs that is deeper than 26 may decrease in performance as the 31 layer net is worse than 26 layer net. The 200 dimensional i-vector augmented 26 layer TDNN reaches a WER of 13.93. Additional discriminative training (DT) also help, it helps to decrease WER from 14.03 to 13.62. When adding the cleaned 220 hours of data, we lower the WER from 14.03 to 13.35.

Table 1: TDNN results on TED-LIUM corpus

#layer	configuration	average
16	40MFCC	15.76
16	40MFCC + 100ivec	15.53
21	40MFCC + 100ivec	14.23
21	40MFCC + fMLLR	15.09
26	40MFCC + 100ivec	14.03
26	40MFCC + 200ivec	13.93
26	40MFCC + 100ivec + DT	13.62
31	40MFCC + 100ivec	14.3
26	40MFCC + 100ivec + 220h data	13.35

Conclude from Table 1, our final TDNN use a 26 layers TDNN architecture. We abandon fMLLR and use 200 dimensional i-vector augmented to MFCC to train speaker adapted net. Each hidden layer contains 1024 units and 160-dimension bottleneck. The input to TDNN is 5 frames of 40-dimension static MFCC. The other TDNN layer has an input context equals to 3 which has different time stride. We constructed 3 consecutive layers with time stride 1, 4 consecutive layers with time stride 2 and 15 consecutive layers with time stride 3. Each of these consecutive layers with the same time stride is followed by a fully connected layer. We train two of them with different random seeds with total data, and the third TDNN with 80% data. After TDNN training we got *tdnn1*, *tdnn2* and *tdnn3*.

2.4. Language Model

2.4.1. Data Preparation and the Vocabulary

For the data preparation, number normalization and lowercasing are adopted to formatting the all-corpora. Next, punctuations are removed and the paragraphs are split into sentences. We choose 152217 English words to build the vocabulary and replace all the out-of-vocabulary (OOV) words in the corpora with the symbol "<unk>".

2.4.2. *N-gram Language Models*

The constrained all-corpora consists of various text resources, such as news, TED subtitles, film subtitles, Europarl dataset and some web crawled materials. The Table 2 shows the details of the cleaned sub-corpora and their interpolation coefficients in n-gram language modeling. We estimate a series of sub-corpora 5-gram language models using the SRILM toolkit [11] with the modified Kneser-Ney smoothing. And then, the development datasets are used for the perplexities and the interpolation weights tuning. By linearly interpolating the different sub-corpora 5-gram models, the final back-off language model is estimated and adopted to the speech recognition system. The perplexities of the development datasets are listed in the Table 3.

Table 2: English language modelling datasets and interpolation coefficients.

Text corpus	**# Words**	**Interpolation**
TED	5.747 M	0.131
OpenSubtitles	144.1 M	0.064
Para WIT	3.263 M	0.029
ParaCrawl + Common crawl	765.1 M	0.048
News discussions	4638 M	0.397
News articles	4004 M	0.331

Table 3: The perplexities (PPL) of the English dev corpora.

Dev set	**5-gram LM**
tst2013	112.31
tst2014	143.11
tst2015	121.80

2.4.3. *LSTM based Neural Language Model*

To improve the computation efficiency in the neural language model, the vocabulary needs to be downsized. We select the top 30000 frequent words from the cleaned corpora to construct a small vocabulary, and replace the out-of-vocabulary words in the cleaned corpora with the symbol "<oos>" according to the customized vocabulary.

The LSTM based language model are trained with TensorFlow. The model contains two stacked dropout wrapped LSTM layers [12] with the hidden size of 256. The word embedding size is 256 and the initial learning rate is 0.1. After the training, we apply the LSTM based language model in the lattice rescoring and n-best rescoring with the Kaldi toolkit [6]. The pruned lattice-rescoring algorithm in [13] helps to achieve lower word error rate (WER) in ASR. Both in the lattice rescoring and n-best rescoring stages, interpolating the 5-gram language model with the LSTM based language model can further improve the ASR accuracies.

2.5. System combination

In the first pass, we use 6 neural network systems described in section 2.3, e.g. *dcnn, blstm1, blstm2, tdnn1, tdnn2, tdnn3* and 5-gram language model described in section 2.4.2. We combine the system in the posterior level and generate the first pass ensemble results. We also select the best single network system *tdnn1* to perform discriminative training, but found no performance gain when combine with the above 6 systems. The decoding lattices of the ensemble system are sent to a second pass decoder for lattice rescoring.

3. Neural Machine Translation

In this section, some post-processing details of ASR output and the architecture of our neural machine translation system are described.

3.1. Punctuation Restoration

The automatic speech recognition system only generates a stream of words without any punctuation symbols. In our work, we model the punctuation using the sequence to sequence architecture. Our punctuation restoration model is based on the Transformer architecture, which is based on attention only. In our work, given a sequence of words as our inputs, we label each word based on the punctuation after the word. Specifically, we label each word with comma, period, question mark, exclamation mark and non-punctuation.

The training dataset contains 41.5M sentences in total. Sentences were encoded using byte-pair encoding [15] with source vocabulary of about 30k tokens. We evaluate the performance of our punctuation restoration model by precision, recall and F1 score. We present the results in table.

Table 4: The result of our Punctuation Restoration model

Dev set	Precision	Recall	F1 value
tst13	88.01%	82.18%	85.00%
tst14	88.62%	84.28%	86.40%
tst15	91.51%	86.26%	88.81%
average	89.38%	84.24%	86.73%

3.2. Disfluency Detection and Inverse Text Normalization

Since the automatic speech recognition outputs often contain various disfluencies. In this paper, a simple but efficient detection approach is employed to identify and repair these disfluencies. At first, we remove the filled pauses, such as "uh" and "um". Then we define a window to identify and remove the repetitions in the output of ASR system.

After disfluency detection, the inverse text normalization is necessary for machine translation, because the corpus of machine translation are in written form, but the output of the automatic speech recognition are generally in spoken form, especially in figure, data and the amount of money. As shown in Figure 1, the word stream generated by ASR system is transformed into the standard form after punctuation restoration, disfluency detection and inverse text normalization.

Figure 1: Post-processing for ASR output

The output of ASR system:
and the results from the **twenty twenty two *uh* point five million** sentences we selected **sixteen point eight** and which let us to throw like **twenty two percent** of the corpus

After our post-process:
and the results from the **22.5 million** sentences, we selected **16.8** and which let us to throw like **22%** of the corpus.

3.3. Data Preparation and Cleaning for NMT

The parallel text data consists of four parts: Speech-Translation TED corpus, TED corpus (Web Inventory of Transcribed and Translated Talks, WIT), WMT2018 and OpenSubtitles2018.

We tokenize both the English and Germany text data by the Moses tokenizer[2]. Then the English data is transformed to lower case. To simplify the post processing of translation, we do not transform the German data to lower case. Finally, we use BPE subword segmentation tool to process the English data and German data.

We have observed some noise data, which cause a lot of translation errors. In order to improve the quality of parallel text data, we have cleaned the data.

- The samples whose number of tokens are over 100 will be removed.
- For one sentence pair, if the length rate of source/target is less than 1/2 or large than 2, they will be removed.
- We use SRILM Toolkit [11] to train an English ngram language model and a German ngram language model respectively with the parallel text data. The two LMs are used to evaluate the perplexity (PPL) for the sentences. For one source sentence (English side) and target sentence (German), they are removed if they meet the following two conditions: (1) we use source LM to calculate PPL. The PPL of source sentence is larger than that of target sentence; (2) we use target LM to calculate PPL. The PPL of target sentence is larger than that of the source sentence.

3.4. NMT Architecture

Our model follows the Transformer architecture which is solely based on attention mechanisms [4]. In our setup, the encoder has six layers. Each layer is consist of two parts: multi-head self-attention network and position-wise fully connected feed-forward network. The two parts employ both residual connection and layer-normalization. In the decoder, we employ masking to ensure that the prediction for the current word only depends on the previous words.

The dimension of word embedding is set to 512. The hidden state size is set to 1024. The vocabulary sizes of English and German are set of 60,000.

The sentences which have the similar number of tokens are grouped together. During training, the batches of size is set by the number of tokens which is set to 8000. We use the Adam optimizer to train the model.

3.5. Fine-tune

A large part of the training data comes from WMT, whose domain is news. But the test sets come from oral domain. After the systems are trained, we continue to train the systems by 5000 steps with the WIT parallel text data.

3.6. Ensemble

It's common to avoid over-fitting by using ensemble of several systems. There are two methods we have adopted. For one system training, we always average all of parameters across the last 20 checkpoints. For several system trainings, we compute the output tokens' possibilities by averaging the systems' output possibilities.
In the final system, we choose six systems to apply the second ensemble.

3.7. Re-scoring with NMT Variants

In order to get better translation result, we test different NMT variant models in re-scoring n-best list.

Target right-to-left NMT Model: When the target words are decoded by the NMT system, the later words will depend on the previous words decisions in the beam search decoder. So the word decision at time step t is much harder than that of time step t-1[16]. In order to alleviate this imbalance problem, a variant NMT model, which decodes the target words from right-to-left (R2L), is trained. The R2L model is used to re-score the n-best list which produced by the main NMT model. The scores represents the conditional probabilities of the reversed translations given the source sentences.

Target-to-source NMT Model: Moreover, the translations may be inadequate: the translations may repeat or miss out some words [17]. In order to cope with the inadequateness, we also test the target-to-source (T2S) model, which is trained with the source and target swapped.

We first produce one n-best list with an ensemble of serval models. Then we do force decoding with target right-to-left, target-to-source NMT models. We treat each models scores as an individual feature. We use k-batched MIRA [18] to tune weights for all the features. In order to get more diverse n-best list, we also try to increase the size of beam from 10 to 100 for re-scoring.

4. Results

4.1. Results for ASR

Table 5 shows our systems built for the ASR submission. In the first pass, we use 6 neural network system described in section 2.3, e.g. *dcnn, blstm1, blstm2, tdnn1, tdnn2, tdnn3* and 5-gram language model described in section 2.4.2. We combine the system in the posterior level and generate the first pass ensemble results. The decoding lattices of the ensemble system are sent to a second pass decoder for lattice rescoring.

Table 5: The WER result of our ASR model

System	tst2013	tst2014	tst2015	average
dcnn	11.15	8.86	7.77	9.26
blstm1	8.65	7.84	8.02	8.17
blstm2	8.78	8.07	8.03	8.29
tdnn1	8.5	7.35	6.24	7.36
tdnn2	8.52	7.42	6.15	7.36
tdnn3	8.47	7.55	6.18	7.4
+ensemble	8.01	7.08	6.54	7.21
+rescoring	7.49	6.76	5.95	6.73

4.2. Results for NMT

Table 6 shows the machine translation results on validation sets. All the results are cased BLEU evaluate by multi-bleu.perl script in Moses[3]. Our data cleaning technique

[2] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/tokenizer.perl

[3] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl

improves the baseline by 0.74 BLEU. Due to the domain of training data is not very consistent with that of test data, we continue training the system with the WIT parallel text data. This fine-tune technique get an improvement of 1.43 BLEU. In order to get more diverse models and better ensemble results, we train 6 models independently with different random initializations. The ensemble result gives an improvement of 0.53 BLEU over best single system. By increasing the beam size from 10 to 100 during decoding, we achieve another improvement of 0.05 BLEU. We add six right-to-left and six target-to-source NMT models as re-scoring features. It improved the system by 0.85 BLEU. The test2013 set is used as development set to tune the weights of re-scoring features.

Table 6: The English→Germany NMT results on three development sets. Submitted system is the last system.

system	tst2013	tst2014	tst2015	average
baseline	34.73	29.09	33.02	32.28
+data cleaning	35.4	30.03	33.62	33.02
+fine-tune	37.13	31.28	34.93	34.45
+ensemble	37.79	31.56	35.58	34.98
+beam(10 → 100)	37.92	31.32	35.86	35.03
+rescore(6*R2L,6*T2S)	38.90	32.36	36.38	35.88

4.3. Results for Speech Translation

Table 7 shows the final speech translation results on three test set. In order to tune the ASR and NMT system individually. We first segment the full utterance, and then align the utterance into segments with the correct English text segments and German translations. The transcript of the best ASR system was then passed to disfluency detection, Punctuation Restoration and text normalization module. Finally, ASR outputs with punctuations were translated into German. The average result of three test set for our Speech Translation is 31.02 BLEU.

Table 7: The English→Germany speech translation results on three sets.

system	tst2013	tst2014	tst2015	average
final system	32.95	28.28	31.82	31.02

5. Conclusions

This paper describes our pipeline system for the IWSLT 2018 Speech Translation task from English to German. The whole pipeline are consist of the wav utterance segmentation module, the ASR system, the punctuation restoration and the NMT system.

As for the ASR system, we adopted an ensemble system of Deep-CNN, BLSTM, TDNN, n-gram Language model with lattice rescoring. According to our experiments, TDNN achieved the lowest WER among these three acoustic modeling network for this task. For our tdnn acoustic modeling, we found adding layers, i-vector, cleaned data are effective. We have achieved average WER of 6.73 over three test sets using the combination system. For the NMT system, we also use an ensemble of Transformer system with n-best rescoring. And we use various techniques in our system, such as data cleaning, fine-tune, ensemble of models and n-best rescoring. These techniques help our system achieve 3.6 BLEU better than baseline. We use the outputs of the best ASR system as input of our NMT system, and we achieved average BLEU score of 31.02 over three development sets.

How to use document-level information to improve the ASR and NMT system performance and build a robust NMT system will be our future work.

6. References

[1] Zhang Y, Pezeshki M, Brakel P, et al. Towards end-to-end speech recognition with deep convolutional neural networks[J]. arXiv preprint arXiv:1701.02720, 2017.
[2] Qian, Yanmin, and Philip C. Woodland. "Very deep convolutional neural networks for robust speech recognition." Spoken Language Technology Workshop (SLT), 2016 IEEE. IEEE, 2016.
[3] Povey, D., Cheng, G., Wang, Y., Li, K., Xu, H., Yarmohamadi, M., & Khudanpur, S. (2018). Semi-orthogonal low-rank matrix factorization for deep neural networks. INTERSPEECH 2018.
[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. CoRR, 2017.
[5] A. Rousseau, P. Deléglise, and Y. Estève, “Enhancing the TED-LIUM Corpus with Selected Data for Language Modeling and More TED Talks”, in Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC’14), May 2014.
[6] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., ... & Silovsky, J. (2011). The Kaldi speech recognition toolkit. In IEEE 2011 workshop on automatic speech recognition and understanding (No. EPFL-CONF-192584). IEEE Signal Processing Society.
[7] Nakatani, T., Yoshioka, T., Kinoshita, K., Miyoshi, M., & Juang, B. H. (2008, March). Blind speech dereverberation with multi-channel linear prediction based on short time Fourier transform representation. In Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on (pp. 85-88). IEEE.
[8] Schwartz, B., Gannot, S., & Habets, E. A. (2015). Online speech dereverberation using Kalman filter and EM algorithm. IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP), 23(2), 394-406.
[9] Povey, D., Peddinti, V., Galvez, D., Ghahremani, P., Manohar, V., Na, X., & Khudanpur, S. (2016, September). Purely Sequence-Trained Neural Networks for ASR Based on Lattice-Free MMI. In Interspeech (pp. 2751-2755).
[10] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.
[11] A. Stolcke, “SRILM-an extensible language modeling toolkit,” in Proceedings of Interspeech, September 2002, pp. 901–904.
[12] Y. Gal, and G. Zoubin, "A theoretically grounded application of dropout in recurrent neural networks," in Advances in neural information processing systems, pp. 1019-1027. 2016.
[13] H. Xu, T. Chen, D. Gao, Y. Wang, K. Li, N. Goel, Y. Carmiel, D. Povey and S. Khudanpur, “A Pruned

RNNLM Lattice-Rescoring Algorithm for Automatic Speech Recognition," in Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on. IEEE, 2017.

[14] E. Cho, J. Niehues, and A. Waibel, "NMT-based segmentation and punctuation insertion for real-time spoken language translation," Proc. Interspeech 2017, pp. 2645–2649, 2017.

[15] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Neural machine translation of rare words with subword units. In Proceedings of ACL 2016.

[16] Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Agreement on Target-bidirectional Neural Machine Translation. In NAACL HLT 16, San Diego, CA.

[17] Zhaopeng Tu, Yang Liu, Lifeng Shang, Xiaohua Liu, and Hang Li. 2016a. Neural machine translation with reconstruction. arXiv URL: https://arxiv.org/abs/1611.01874.

[18] Colin Cherry and Gorge Foster. 2012. Batch Tuning Strategies for Statistical Machine Translation, In NAACL, 2012.