A Binarized Neural Network Joint Model for Machine Translation

Jingyi Zhang[1,2], **Masao Utiyama**[1], **Eiichro Sumita**[1]
Graham Neubig[2], **Satoshi Nakamura**[2]
[1]National Institute of Information and Communications Technology,
3-5Hikaridai, Keihanna Science City, Kyoto 619-0289, Japan
[2]Graduate School of Information Science, Nara Institute of Science and Technology,
Takayama, Ikoma, Nara 630-0192, Japan
jingyizhang/mutiyama/eiichiro.sumita@nict.go.jp
neubig/s-nakamura@is.naist.jp

Abstract

The neural network joint model (NNJM), which augments the neural network language model (NNLM) with an m-word source context window, has achieved large gains in machine translation accuracy, but also has problems with high normalization cost when using large vocabularies. Training the NNJM with noise-contrastive estimation (NCE), instead of standard maximum likelihood estimation (MLE), can reduce computation cost. In this paper, we propose an alternative to NCE, the binarized NNJM (BNNJM), which learns a binary classifier that takes both the context and target words as input, and can be efficiently trained using MLE. We compare the BNNJM and NNJM trained by NCE on various translation tasks.

1 Introduction

Neural network translation models, which learn mappings over real-valued vector representations in high-dimensional space, have recently achieved large gains in translation accuracy (Hu et al., 2014; Devlin et al., 2014; Sundermeyer et al., 2014; Auli et al., 2013; Schwenk, 2012; Sutskever et al., 2014; Bahdanau et al., 2015).

Notably, Devlin et al. (2014) proposed a neural network joint model (NNJM), which augments the n-gram neural network language model (NNLM) with an m-word source context window, as shown in Figure 1a. While this model is effective, the computation cost of using it in a large-vocabulary SMT task is quite expensive, as probabilities need to be normalized over the entire vocabulary. To solve this problem, Devlin et al. (2014) presented a technique to train the NNJM to be self-normalized and avoided the expensive normalization cost during decoding. However, they also note that this self-normalization technique sacrifices neural network accuracy, and the training process for the self-normalized neural network is very slow, as with standard maximum likelihood estimation (MLE).

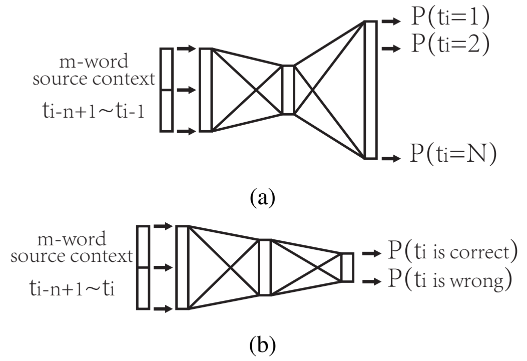

Figure 1: (a) the traditional NNJM and (b) the proposed BNNJM

To remedy the problem of long training times in the context of NNLMs, Vaswani et al. (2013) used a method called noise contrastive estimation (NCE). Compared with MLE, NCE does not require repeated summations over the whole vocabulary and performs nonlinear logistic regression to discriminate between the observed data and artificially generated noise.

This paper proposes an alternative framework of binarized NNJMs (BNNJM), which are similar to the NNJM, but use the current target word not as the output, but as the input of the neural network, estimating whether the target word under examination is correct or not, as shown in Figure 1b. Because the BNNJM uses the current target word as input, the information about the current target word can be combined with the context word information and processed in the hidden layers.

The BNNJM learns a simple binary classifier, given the context and target words, therefore it can be trained by MLE very efficiently. "Incorrect" target words for the BNNJM can be generated in the same way as NCE generates noise for the NNJM. We present a novel noise distribution based on translation probabilities to train the NNJM and the BNNJM efficiently.

2 Neural Network Joint Model

Let $T = t_1^{|T|}$ be a translation of $S = s_1^{|S|}$. The NNJM (Devlin et al., 2014) defines the following probability,

$$P(T|S) = \prod_{i=1}^{|T|} P\left(t_i | s_{a_i-(m-1)/2}^{a_i+(m-1)/2}, t_{i-n+1}^{i-1}\right) \tag{1}$$

where target word t_i is affiliated with source word s_{a_i}. Affiliation a_i is derived from the word alignments using heuristics[1]. To estimate these probabilities, the NNJM uses m source context words and $n-1$ target history words as input to a neural network and performs estimation of unnormalized probabilities $p(t_i|C)$ before normalizing over all words in the target vocabulary V,

$$\begin{aligned} P(t_i|C) &= \frac{p(t_i|C)}{Z(C)} \\ Z(C) &= \sum_{t_i{}' \in V} p(t_i{}'|C) \end{aligned} \tag{2}$$

where C stands for source and target context words as in Equation 1.

The NNJM can be trained on a word-aligned parallel corpus using standard MLE, but the cost of normalizing over the entire vocabulary to calculate the denominator in Equation 2 is quite large. Devlin et al. (2014)'s self-normalization technique can avoid normalization cost during decoding, but not during training.

NCE can be used to train NNLM-style models (Vaswani et al., 2013) to reduce training times. NCE creates a noise distribution $q(t_i)$, selects k noise samples $t_{i1}, ..., t_{ik}$ for each t_i and introduces a random variable v which is 1 for training examples and 0 for noise samples,

$$\begin{aligned} P(v=1, t_i|C) &= \frac{1}{1+k} \cdot \frac{p(t_i|C)}{Z(C)} \\ P(v=0, t_i|C) &= \frac{k}{1+k} \cdot q(t_i). \end{aligned}$$

NCE trains the model to distinguish training data from noise by maximize the conditional likelihood,

$$L = \log P(v=1|C, t_i) + \sum_{j=1}^{k} \log P(v=0|C, t_{ik}).$$

The normalization cost can be avoided by using $p(t_i|C)$ as an approximation of $P(t_i|C)$.[2]

[1]If t_i aligns to exactly one source word, a_i is the index of this source word; If t_i aligns to multiple source words, a_i is the index of the aligned word in the middle; If t_i is unaligned, they inherit its affiliation from the closest aligned word.

[2]The theoretical properties of self-normalization techniques, including NCE and Devlin et al. (2014)'s method, are investigated by Andreas and Klein (2015).

3 Binarized NNJM

In this paper, we propose a new framework of the binarized NNJM (BNNJM), which is similar to the NNJM but learns not to predict the next word given the context, but solves a binary classification problem by adding a variable $v \in \{0, 1\}$ that stands for whether the current target word t_i is correctly/wrongly produced in terms of source context words $s_{a_i-(m-1)/2}^{a_i+(m-1)/2}$ and target history words t_{i-n+1}^{i-1},

$$P\left(v | s_{a_i-(m-1)/2}^{a_i+(m-1)/2}, t_{i-n+1}^{i-1}, t_i\right).$$

The BNNJM is learned by a feed-forward neural network with $m+n$ inputs $\left\{s_{a_i-(m-1)/2}^{a_i+(m-1)/2}, t_{i-n+1}^{i-1}, t_i\right\}$ and two outputs for $v = 1/0$.

Because the BNNJM uses the current target word as input, the information about the current target word can be combined with the context word information and processed in the hidden layers. Thus, the hidden layers can be used to learn the difference between correct target words and noise in the BNNJM, while in the NNJM the hidden layers just contain information about context words and only the output layer can be used to discriminate between the training data and noise, giving the BNNJM more power to learn this classification problem.

We can use the BNNJM probability in translation as an approximation for the NNJM as below,

$$\begin{aligned} &P\left(t_i | s_{a_i-(m-1)/2}^{a_i+(m-1)/2}, t_{i-n+1}^{i-1}\right) \\ &\approx P\left(v=1 | s_{a_i-(m-1)/2}^{a_i+(m-1)/2}, t_{i-n+1}^{i-1}, t_i\right). \end{aligned}$$

As a binary classifier, the gradient for a single example in the BNNJM can be calculated efficiently by MLE without it being necessary to calculate the softmax over the full vocabulary. On the other hand, we need to create "positive" and "negative" examples for the classifier. Positive examples can be extracted directly from the word-aligned parallel corpus as $\left\langle s_{a_i-(m-1)/2}^{a_i+(m-1)/2}, t_{i-n+1}^{i-1}, t_i\right\rangle$; Negative examples can be generated for each positive example in the same way that NCE generates noise data as $\left\langle s_{a_i-(m-1)/2}^{a_i+(m-1)/2}, t_{i-n+1}^{i-1}, t_i{}'\right\rangle$, where $t_i{}' \in V \backslash \{t_i\}$.

4 Noise Sampling

4.1 Unigram Noise

Vaswani et al. (2013) adopted the unigram probability distribution (UPD) to sample noise for train-

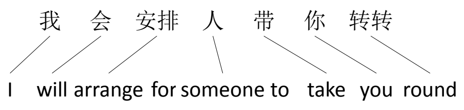

Figure 2: A parallel sentence pair.

ing NNLMs with NCE,

$$q\left(t_i{}'\right) = \frac{occur(t_i{}')}{\sum\limits_{t_i{}'' \in V} occur(t_i{}'')}$$

where $occur\left(t_i{}'\right)$ stands for how many times $t_i{}'$ occurs in the training corpus.

4.2 Translation Model Noise

In this paper, we propose a noise distribution specialized for translation models, such as the NNJM or BNNJM.

Figure 2 gives a Chinese-to-English parallel sentence pair with word alignments to demonstrate the intuition behind our method. Focusing on s_{a_i}="安排", this is translated into t_i ="*arrange*". For this positive example, UPD is allowed to sample any arbitrary noise, such as $t_i{}'$ = "*banana*". However, in this case, noise $t_i{}'$ = "*banana*" is not useful for model training, as constraints on possible translations given by the phrase table ensure that "安排" will never be translated into "*banana*". On the other hand, noise $t_i{}'$ = "*arranges*" and "*arrangement*" are both possible translations of "安排" and therefore useful training data, that we would like our model to penalize.

Based on this intuition, we propose the use of another noise distribution that only uses $t_i{}'$ that are possible translations of s_{a_i}, i.e., $t_i{}' \in U\left(s_{a_i}\right) \setminus \{t_i\}$, where $U\left(s_{a_i}\right)$ contains all target words aligned to s_{a_i} in the parallel corpus.

Because $U\left(s_{a_i}\right)$ may be quite large and contain many wrong translations caused by wrong alignments, "*banana*" may actually be included in U("安排"). To mitigate the effect of uncommon examples, we use a translation probability distribution (TPD) to sample noise $t_i{}'$ from $U\left(s_{a_i}\right) \setminus \{t_i\}$ as follows,

$$q\left(t_i{}' | s_{a_i}\right) = \frac{align\left(s_{a_i}, t_i{}'\right)}{\sum_{t_i{}'' \in U\left(s_{a_i}\right)} align\left(s_{a_i}, t_i{}''\right)}$$

where $align\left(s_{a_i}, t_i{}'\right)$ is how many times $t_i{}'$ is aligned to s_{a_i} in the parallel corpus.

Note that t_i could be unaligned, in which case we assume that it is aligned to a special $null$ word. Noise for unaligned words is sampled according to the TPD of the $null$ word. If several target/source words are aligned to one source/target word, we choose to combine these target/source words as a new target/source word.[3]

5 Experiments

5.1 Setting

We evaluated the effectiveness of the proposed approach for Chinese-to-English (CE), Japanese-to-English (JE) and French-to-English (FE) translation tasks. The datasets officially provided for the patent machine translation task at NTCIR-9 (Goto et al., 2011) were used for the CE and JE tasks. The development and test sets were both provided for the CE task while only the test set was provided for the JE task. Therefore, we used the sentences from the NTCIR-8 JE test set as the development set. Word segmentation was done by BaseSeg (Zhao et al., 2006) for Chinese and Mecab[4] for Japanese. For the FE language pair, we used standard data for the WMT 2014 translation task. The training sets for CE, JE and FE tasks contain 1M, 3M and 2M sentence pairs, respectively.

For each translation task, a recent version of Moses HPB decoder (Koehn et al., 2007) with the training scripts was used as the baseline (Base). We used the default parameters for Moses, and a 5-gram language model was trained on the target side of the training corpus using the IRSTLM Toolkit[5] with improved Kneser-Ney smoothing. Feature weights were tuned by MERT (Och, 2003).

The word-aligned training set was used to learn the NNJM and the BNNJM.[6] For both NNJM and BNNJM, we set $m = 7$ and $n = 5$. The NNJM was trained by NCE using UPD and TPD as noise distributions. The BNNJM was trained by standard MLE using UPD and TPD to generate negative examples.

The number of noise samples for NCE was set to be 100. For the BNNJM, we used only one negative example for each positive example in each training epoch, as the BNNJM needs to calculate

[3]The processing for multiple alignments helps sample more useful negative examples for TPD, and had little effect on the translation performance when UPD was used as the noise distribution for the NNJM and the BNNJM in our preliminary experiments.

[4]http://sourceforge.net/projects/mecab/files/

[5]http://hlt.fbk.eu/en/irstlm

[6]Both the NNJM and the BNNJM had one hidden layer, 100 hidden nodes, input embedding dimension 50, output embedding dimension 50. A small set of training data was used as validation data. The training process was stopped when validation likelihood stopped increasing.

		CE		JE		FE	
		E	T	E	T	E	T
NNJM	UPD	20	22	19	49	20	28
	TPD	4		6		4	
BNNJM	UPD	14	16	12	34	11	22
	TPD	11		9		9	

Table 1: Epochs (E) and time (T) in minutes per epoch for each task.

		CE	JE	FE
Base		32.95	30.13	24.56
NNJM	UPD	34.36+	**31.30+**	24.68
	TPD	34.60+	**31.50+**	24.80
BNNJM	UPD	32.89	30.04	24.50
	TPD	**35.05+***	**31.42+**	**25.84+***

Table 2: Translation results. The symbol + and * represent significant differences at the $p < 0.01$ level against Base and NNJM+UPD, respectively. Significance tests were conducted using bootstrap resampling (Koehn, 2004).

the whole neural network (not just the output layer like the NNJM) for each noise sample and thus noise computation is more expensive. However, for different epochs, we resampled the negative example for each positive example, so the BNNJM can make use of different negative examples.

5.2 Results and Discussion

Table 1 shows how many epochs these two models needed and the training time for each epoch on a 10-core 3.47GHz Xeon X5690 machine.[7] Translation results are shown in Table 2.

We can see that using TPD instead of UPD as a noise distribution for the NNJM trained by NCE can speed up the training process significantly, with a small improvement in performance. But for the BNNJM, using different noise distributions affects translation performance significantly. The BNNJM with UPD does not improve over the baseline system, likely due to the small number of noise samples used in training the BNNJM, while the BNNJM with TPD achieves good performance, even better than the NNJM with TPD on the Chinese-to-English and French-to-English translation tasks.

From Table 2, the NNJM does not improve translation performance significantly on the FE task. Note that the baseline BLEU for the FE task is lower than CE and JE tasks, indicating that learning is harder for the FE task than CE and JE tasks. The validation perplexities of the NNJM with UPD for CE, JE and FE tasks are 4.03, 3.49 and 8.37. Despite these difficult learning circumstances and lack of large gains for the NNJM, the BNNJM improves translations significantly for the FE task, suggesting that the BNNJM is more robust to difficult translation tasks that are hard for the NNJM.

[7]The decoding time for the NNJM and the BNNJM were similar, since the NNJM trained by NCE uses $p(t_i|C)$ as an approximation of $P(t_i|C)$ without normalization and the BNNJM only needs to be normalized over two output neurons.

S: 该(this) 移动(movement) 持续(continued) 到(until) 寄生虫(parasite) 由(by) 两(two) 个 舌(tongues) 部 21 彼此(each other) 接触(contact) 时(where) 的 点(point) 接触(touched) 。
R: *this movement is continued until the parasite is touched by the point where the two tongues 21 contact each other .*
T_1**:** *the mobile continues to the parasite from the two tongue 21 contacts the points of contact with each other .*
T_2**:** *this movement is continued until the parasite by two tongue 21 contact points of contact with each other .*

Table 3: Translation examples. Here, S: source; R: reference; T_1 uses NNJM; T_2 uses BNNJM.

	NNJM	BNNJM
该− >the	1.681	-0.126
移动− >mobile	-4.506	-3.758
持续− >continues	-1.550	-0.130
到− >to	2.510	-0.220
SUM	-1.865	-4.236
该− >this	-2.414	-0.649
移动− >movement	-1.527	-0.200
$null$− >is	0.006	-0.055
持续− >continued	-0.292	-0.249
到− >until	-6.846	-0.186
SUM	-11.075	-1.341

Table 4: Scores for different translations.

Table 3 gives Chinese-to-English translation examples to demonstrate how the BNNJM (with TPD) helps to improve translations over the NNJM (with TPD). In this case, the BNNJM helps to translate the phrase "该 移动 持续 到" better. Table 4 gives translation scores for these two translations calculated by the NNJM and the BNNJM. Context words are used for predictions but not shown in the table.

As can be seen, the BNNJM prefers T_2 while the NNJM prefers T_1. Among these predictions, the NNJM and the BNNJM predict the translation for "到" most differently. The NNJM clearly predicts that in this case "到" should be translated into "to" more than "until", likely because this example rarely occurs in the training corpus. However, the BNNJM prefers "until" more than "to", which demonstrates the BNNJM's robustness to less frequent examples.

5.3 Analysis for JE Translation Results

Finally, we examine the translation results to explore why the BNNJM with TPD did not outperform the NNJM with TPD for the JE translation task, as it did for the other translation tasks. We found that using the BNNJM instead of the NNJM on the JE task did improve translation quality significantly for infrequent words, but not for frequent words.

First, we describe how we estimate translation quality for infrequent words. Suppose we have a test set S, a reference set R and a translation set T with I sentences,

$$S_i\left(1 \leq i \leq I\right), R_i\left(1 \leq i \leq I\right), T_i\left(1 \leq i \leq I\right)$$

T_i contains J individual words,

$$W_{ij} \in Words\left(T_i\right)$$

$T_o\left(W_{ij}\right)$ is how many times W_{ij} occurs in T_i and $R_o\left(W_{ij}\right)$ is how many times W_{ij} occurs in R_i.

The general 1-gram translation accuracy (Papineni et al., 2002) is calculated as,

$$P_g = \frac{\sum\limits_{i=1}^{I}\sum\limits_{j=1}^{J}\min\left(T_o\left(W_{ij}\right), R_o\left(W_{ij}\right)\right)}{\sum\limits_{i=1}^{I}\sum\limits_{j=1}^{J}T_o\left(W_{ij}\right)}$$

This general 1-gram translation accuracy does not distinguish word frequency.

We use a modified 1-gram translation accuracy that weights infrequent words more heavily,

$$P_c = \frac{\sum\limits_{i=1}^{I}\sum\limits_{j=1}^{J}\min(T_o(W_{ij}), R_o(W_{ij}))\cdot\frac{1}{Occur\left(W_{ij}\right)}}{\sum\limits_{i=1}^{I}\sum\limits_{j=1}^{J}T_o(W_{ij})}$$

where $Occur\left(W_{ij}\right)$ is how many times W_{ij} occurs in the whole reference set. Note P_c will not be 1 even in the case of completely accurate translations, but it can approximately reflect infrequent word translation accuracy, since correct frequent word translations contribute less to P_c.

Table 5 shows P_g and P_c for different translation tasks. It can be seen that the BNNJM improves infrequent word translation quality similarly for all translation tasks, but improves general translation quality less for the JE task than the other translation tasks. We conjecture that the reason why the BNNJM is less useful for frequent word translations on the JE task is the fact that the JE parallel corpus has less accurate function word alignments than other language pairs, as the grammatical features of Japanese and English are quite different.[8] Wrong function word alignments will make noise sampling less effective and therefore lower the BNNJM performance for function word translations. Although wrong word alignments will also make noise sampling less effective for the NNJM, the BNNJM only uses one noise sample for each positive example, so wrong word alignments affect the BNNJM more than the NNJM.

	CE		JE		FE	
	P_g	P_c	P_g	P_c	P_g	P_c
NNJM	70.3	5.79	68.2	4.15	61.2	6.70
BNNJM	70.9	5.97	68.4	4.30	61.7	6.86
Imp. (%)	0.85	3.1	0.29	3.6	0.81	2.4

Table 5: 1-gram precisions and improvements.

6 Related Work

Xu et al. (2011) proposed a method to use binary classifiers to learn NNLMs. But they also used the current target word in the output, similarly to NCE. The BNNJM uses the current target word as input, so the information about the current target word can be combined with the context word information and processed in hidden layers.

Mauser et al. (2009) presented discriminative lexicon models to predict target words. They train a separate classifier for each target word, as these lexicon models use discrete representations of words and different classifiers do not share features. In contrast, the BNNJM uses real-valued vector representations of words and shares features, so we train one classifier and can use the similarity information between words.

7 Conclusion

This paper proposes an alternative to the NNJM, the BNNJM, which learns a binary classifier that takes both the context and target words as input and combines all useful information in the hidden layers. We also present a novel noise distribution based on translation probabilities to train the BNNJM efficiently. With the improved noise sampling method, the BNNJM can achieve comparable performance with the NNJM and even improve the translation results over the NNJM on Chinese-to-English and French-to-English translations.

[8]Infrequent words are usually content words and frequent words are usually function words.

References

Jacob Andreas and Dan Klein. 2015. When and why are log-linear models self-normalizing? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 244–249.

Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1044–1054.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1370–1380.

Isao Goto, Bin Lu, Ka Po Chow, Eiichiro Sumita, and Benjamin K Tsou. 2011. Overview of the patent machine translation task at the NTCIR-9 workshop. In *Proceedings of The 9th NII Test Collection for IR Systems Workshop Meeting*, pages 559–578.

Yuening Hu, Michael Auli, Qin Gao, and Jianfeng Gao. 2014. Minimum translation modeling with recurrent neural networks. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 20–29.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395.

Arne Mauser, Saša Hasan, and Hermann Ney. 2009. Extending statistical machine translation with discriminative and trigger-based lexicon models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 210–218.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Holger Schwenk. 2012. Continuous space translation models for phrase-based statistical machine translation. In *Proceedings of International Conference on Computational Linguistics : Posters*, pages 1071–1080.

Martin Sundermeyer, Tamer Alkhouli, Joern Wuebker, and Hermann Ney. 2014. Translation modeling with bidirectional recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 14–25.

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

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1387–1392.

Puyang Xu, Asela Gunawardana, and Sanjeev Khudanpur. 2011. Efficient subsampling for training complex language models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1128–1136.

Hai Zhao, Chang-Ning Huang, and Mu Li. 2006. An improved Chinese word segmentation system with conditional random field. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 162–165.