An Empirical Evaluation of Noise Contrastive Estimation for the Neural Network Joint Model of Translation

Colin Cherry
National Research Council Canada
Colin.Cherry@nrc-cnrc.gc.ca

Abstract

The neural network joint model of translation or NNJM (Devlin et al., 2014) combines source and target context to produce a powerful translation feature. However, its softmax layer necessitates a sum over the entire output vocabulary, which results in very slow maximum likelihood (MLE) training. This has led some groups to train using Noise Contrastive Estimation (NCE), which side-steps this sum. We carry out the first direct comparison of MLE and NCE training objectives for the NNJM, showing that NCE is significantly outperformed by MLE on large-scale Arabic-English and Chinese-English translation tasks. We also show that this drop can be avoided by using a recently proposed translation noise distribution.

1 Introduction

The Neural Network Joint Model of Translation, or NNJM (Devlin et al., 2014), is a strong feature for statistical machine translation. The NNJM uses both target and source tokens as context for a feed-forward neural network language model (LM). Unfortunately, its softmax layer requires a sum over the entire output vocabulary, which slows the calculation of LM probabilities and the maximum likelihood estimation (MLE) of model parameters.

Devlin et al. (2014) address this problem at runtime only with a self-normalized MLE objective. Others advocate the use of Noise Contrastive Estimation (NCE) to train NNJMs and similar monolingual LMs (Mnih and Teh, 2012; Vaswani et al., 2013; Baltescu and Blunsom, 2015; Zhang et al., 2015). NCE avoids the sum over the output vocabulary at both train- and run-time by wrapping the NNJM inside a classifier that attempts to separate real data from sampled noise, greatly improving training speed. The training efficiency of NCE is well-documented, and will not be evaluated here. However, the experimental evidence that NCE matches MLE in terms of resulting model quality is all on monolingual language modeling tasks (Mnih and Teh, 2012). Since cross-lingual contexts provide substantially stronger signals than monolingual ones, there is reason to suspect these results may not carry over to NNJMs.

To our knowledge there is no published work that directly compares MLE and NCE in the context of an NNJM; this paper fills that gap as its primary contribution. We measure model likelihood and translation quality in large-scale Arabic-to-English and Chinese-to-English translation tasks. We also test a recently-proposed translation noise distribution for NCE (Zhang et al., 2015), along with a mixture of noise distributions. Finally, we test a widely known, but apparently undocumented, technique for domain adaptation of NNJMs, demonstrating its utility, as well as its impact on the MLE-NCE comparison.

2 Methods

The NNJM adds a bilingual context window to the machinery of feed-forward neural network language models, or NNLMs (Bengio et al., 2003). An NNLM calculates the probability $p(e_i|e_{i-n+1}^{i-1})$ of a word e_i given its $n-1$ preceding words, while an NNJM assumes access to a source sentence F and an aligned source index a_i that points to the most influ-

Proceedings of NAACL-HLT 2016, pages 41–46,
San Diego, California, June 12-17, 2016.

ential source word for the next translation choice. It calculates $p(e_i|e_{i-n+1}^{i-1}, f_{a_i-m}^{a_i+m})$, which accounts for $2m+1$ words of source context, centered around f_{a_i}. The two models differ only in their definition of the conditioning context, which we will generalize with the variable $c_i = e_{i-n+1}^{i-1}, f_{a_i-m}^{a_i+m}$. When unambiguous, we drop the subscript i from e and c.

The feed-forward neural network that powers both models takes a context sequence c as input to its network, which includes an embedding layer, one or more hidden layers, and a top-level softmax layer that assigns probabilities to each word in the vocabulary V. Let $s_c(e)$ represent the unnormalized neural network score for the word e. The softmax layer first calculates $Z_c = \sum_{e' \in V} \exp s_c(e')$, which allows it to then normalize the score into a log probability $\log p(e|c) = s_c(e) - \log Z_c$. Given a training set of word-context pairs, MLE training of NNJMs minimizes the negative log likelihood $\sum_{e,c} -\log p(e|c)$.

The problem with this objective is that calculating Z_c requires a sum over the entire vocabulary, which is very expensive. This problem has received much recent study, but Devlin et al. (2014) proposed a novel solution for their NNJM, which we refer to as *self-normalization*. Assume we are willing to incur the cost of calculating Z_c during training, which might be mitigated by special-purpose hardware such as graphical processing units (GPUs). One can modify the MLE objective to encourage $\log Z_c$ to be small, so that the term can be safely dropped at run-time:

$$\sum_{e,c} \left[-\log p(e|c) + \alpha \left(\log Z_c\right)^2 \right]$$

where α trades self-normalization against model likelihood. Devlin et al. (2014) have shown that self-normalization has minimal impact on model quality and a tremendous impact on run-time efficiency.

2.1 Noise Contrastive Estimation

Introduced by Gutmann and Hyvärinen (2010) and first applied to language modeling by Mnih and Teh (2012), NCE allows one to train self-normalized models without calculating Z. It does so by defining a noise distribution q over words in V, which is typically a **unigram noise** distribution q_u. It samples k noise words $\hat{e}_1^k$ for each training word e, and wraps the NNJM inside a binary classifier that attempts to separate true data from noise. Let D be a binary variable that is 1 for true data and 0 for noise. We know the joint noise probability $p(D=0, e|c) = \frac{k}{k+1} q(e)$, and we can approximate the joint data probability using our neural network $p(D=1, e|c) \approx \frac{1}{k+1} p(e|c) \approx \frac{1}{k+1} \exp s_c(e)$. Note that the final approximation drops Z_c from the calculation, improving efficiency and forcing the model to self-normalize. With these two terms in place, and a few manipulations of conditional probability, the NCE training objective can be given as:

$$-\sum_{e,c} \left[\log p(D=1|e,c) + \sum_{j=1}^{k} \log p(D=0|\hat{e}_j, c) \right]$$

which measures the probability that data is recognized as data, and noise is recognized as noise.

Note that q ignores the context c. Previous work on monolingual language modeling indicates that a unigram proposal distribution is sufficient for NCE training (Mnih and Teh, 2012). But for bilingual NNJMs, Zhang et al. (2015) have shown that it is beneficial to have q condition on *source* context. Recall that $c_i = e_{i-n+1}^{i-1}, f_{a_i-m}^{a_i+m}$. We experiment with a **translation noise** distribution $q_t(\hat{e}|f_{a_i})$. We estimate q_t by relative frequency from our training corpus, which implicitly provides us with one e_i, f_{a_i} pair for each training point e_i, c_i. Conditioning on f_{a_i} drastically reduces the entropy of the noise distribution, focusing training on the task of differentiating between likely translation candidates.

As our experiments will show, under NCE with translation noise, the NNJM no longer provides meaningful scores for the entire vocabulary. Therefore, we also experiment with a novel **mixture noise** distribution: $q_m(\hat{e}|f_{a_i}) = 0.5 q_u(\hat{e}) + 0.5 q_t(\hat{e}|f_{a_i})$.

3 Implementation details

We implement our NNJM and all candidate training objectives described above in a shared codebase in Theano (Bergstra et al., 2010). To ensure a fair comparison between MLE and NCE, the various systems share code for model structures and algorithms, differing only in their training objectives. A GeForce GTX TITAN GPU enables efficient MLE training. Following Devlin et al. (2014), all NNJMs use 3 tokens for target context, a source context window with $m = 5$, and a 192-node embedding layer.

We deviate from their configuration by using a single 512-node hidden layer, motivated by our internal development experiments. All NCE variants use $k = 100$ noise samples.

NNJM training data is pre-processed to limit vocabularies to 16K types for source or target inputs, and 32K types for target outputs. We build 400 deterministic word clusters for each corpus using `mkcls` (Och, 1999). Any word not among the 16K / 32K most frequent words is replaced with its cluster.

We train our models with mini-batch stochastic gradient descent, with a batch size of 128 words, and an initial learning rate of 0.3. We check our training objective on the development set every 20K batches, and if it fails to improve for two consecutive checks, the learning rate is halved. Training stops after 5 consecutive failed checks or after 60 checks. As NCE may take longer to converge than MLE, we occasionally let NCE models train to 90 checks, but this never resulted in improved performance. Finally, after training finishes on the complete training data, we use that model to initialize a second training run, on a smaller in-domain training set known to better match the test conditions.[1] This in-domain pass uses a lower initial learning rate of 0.03.

Our translation system is a multi-stack phrase-based decoder that is quite similar to Moses (Koehn et al., 2007). Its features include standard phrase table probabilities, KN-smoothed language models including a 6-gram model trained on the English Gigaword and a 4-gram model trained on the target side of the parallel training data, domain-adapted phrase tables and language models (Foster and Kuhn, 2007), a hierarchical lexicalized reordering model (Galley and Manning, 2008), and sparse features drawn from Hopkins and May (2011) and Cherry (2013). It is tuned with a batch-lattice variant of hope-fear MIRA (Chiang et al., 2008; Cherry and Foster, 2012).

4 Experiments

We test two translation scenarios drawn from the recent BOLT evaluations: Arabic-to-English and Chinese-to-English. The vital statistics for our corpora are given in Table 1. The training set mixes NIST data with BOLT-specific informal genres. The development and test sets are focused specifically on the web-forum genre, as is the in-domain subset of the training data (*In-dom*). The Arabic was segmented with MADA-ARZ (Habash et al., 2013), while the Chinese was segmented with a lexicon-based approach. All data was word-aligned with IBM-4 in GIZA++ (Och and Ney, 2003), with grow-diag-final-and symmetrization (Koehn et al., 2003).

Lang.	Train	In-dom	Dev	Test1	Test2
Arabic	38.6M	1.8M	72K	38K	40K
Chinese	29.2M	1.9M	77K	38K	36K

Table 1: Corpus sizes in terms of number of target tokens. Dev and Test sets have 3 references for Arabic and 5 for Chinese.

4.1 Comparing Training Objectives

Our main experiment is designed to answer two questions: (1) does training NNJMs with NCE impact translation quality? and (2) can any reduction be mitigated through alternate noise distributions? To this end, we train four NNJMs.

- **MLE**: Maximum likelihood training with self-normalization $\alpha = 0.1$
- **NCE-U**: NCE with unigram noise
- **NCE-T**: NCE with translation noise
- **NCE-M**: NCE with mixture noise

and compare their performance to that of a system with no NNJM. Each NNJM was trained as described in Section 3, varying only the learning objective.[2] To measure intrinsic NNJM quality, we report average negative log likelihoods (NLL) and average $|\log Z|$, both calculated on Dev. Lower NLL scores indicate better prediction accuracy, while lower $|\log Z|$ values indicate more effective self-normalization. We also provide average BLEU scores and standard deviations for Test1 and Test2, each calculated over 5 random tuning replications. Statistical significance is calculated with MultEval (Clark et al., 2011).

Our results are shown in Table 2. By comparing MLE to no NNJM, we can confirm that the NNJM is a very effective translation feature, showing large

[1] Recommended by Jacob Devlin, personal communication.

[2] The only exception was the Arabic NCE-M system, which showed some instability during optimization, leading us to reduce its initial learning rate to 0.2.

	Arabic-English						Chinese-English					
Method	NLL	$\lvert \log Z \rvert$	test1	std	test2	std	NLL	$\lvert \log Z \rvert$	test1	std	test2	std
No NNJM	–	–	39.2	0.1	39.9	0.1	–	–	31.6	0.2	27.8	0.1
MLE	1.76	0.50	41.7	0.1	42.0	0.1	2.35	0.49	32.9	0.0	29.1	0.0
NCE-U	1.85	0.42	40.9	0.2	41.5	0.1	2.54	0.42	32.2	0.1	28.3	0.1
NCE-T	3.87	2.36	**41.6**	0.1	**42.4**	0.2	3.93	1.70	**32.7**	0.1	28.7	0.2
NCE-M	1.85	0.30	41.4	0.1	**42.1**	0.1	2.40	0.30	32.6	0.1	28.8	0.1

Table 2: Comparing various NNJM training objectives on two translation scenarios. BLEU results that are statistically better than NCE-U are underlined ($p \leq 0.05$). Those statistically equivalent to or better than MLE are in **bold** ($p \leq 0.05$).

BLEU improvements on all tests. By comparing MLE to NCE-U, we can see that NCE training does reduce translation quality. NCE-U outperforms having no NNJM, but lags behind MLE considerably, resulting in significantly worse performance on all tests. This is mitigated with translation noise: NCE-T and NCE-M both perform significantly better than NCE-U. Furthermore, in 3 out of 4 tests, NCE-T matches or exceeds the performance of MLE. The one Arabic-to-English case where NCE-T exceeds the performance of MLE is particularly intriguing, and warrants further study.

Though NCE-T performs very well as a translation feature, it is relatively lousy as a language model, with abnormally large values for both NLL and $\lvert \log Z \rvert$. This indicates that NCE-T is only good at predicting the next word from a pool of reasonable translation candidates. Scores for words drawn from the larger vocabulary are less accurate. However, the BLEU results for NCE-T show that this does not matter for translation performance. If model likelihoods over the complete vocabulary are needed, one can repair these estimates by mixing in unigram noise, as shown by NCE-M, which achieves the same or better likelihoods than NCE-U, with comparable BLEU scores to those of NCE-T.

Devlin et al. (2014) suggest that one drawback of NCE with respect to self-normalized MLE is NCE's lack of an α hyper-parameter to control the objective's emphasis on self-normalization. However, the $\lvert \log Z \rvert$ values for NCE-U are only slightly lower than those of MLE, and are larger than those of the superior NCE-M. This suggests that we could not have improved NCE-U's performance by adjusting its emphasis on self-normalization.

	General		Adapted	
Method	BLEU	Δ	BLEU	Δ
No NNJM	39.6	-1.4	39.6	-2.2
MLE	41.0	—	41.8	—
NCE-U	40.7	-0.3	41.2	-0.6
NCE-T	41.0	0.0	42.0	+0.2
NCE-M	40.9	-0.1	41.7	-0.1

Table 3: Comparing NNJM training objectives with and without a domain adaptation step for Arabic-to-English task.

4.2 Impact of the Domain Adaptation Pass

We began this project with the hypothesis that NCE may harm NNJM performance. But NCE-U performed worse than we expected. In particular, the differences between NCE-U and NCE-T are larger than those reported by Zhang et al. (2015). This led us to investigate the domain adaptation pass, which was used in our experiments but not those of Zhang et al. This step refines the model with a second training pass on an in-domain subset of the training data. We repeated our comparison for Arabic without domain adaptation, reporting BLEU averaged over two test sets and across 5 tuning replications. We also report each system's BLEU differential Δ with respect to MLE. The results are shown under *General* in Table 3, while *Adapted* summarizes our results from Table 2 in the same format.

The domain adaptation step magnifies the differences between training objectives, perhaps because it increases performance over-all. The spread between the worst and best NNJM is only 0.3 BLEU under *General*, while it is 0.8 BLEU under *Adapted*. Therefore, groups training unadapted models may not see as large drops from NCE-U as we have reported above. Note that we experimented with several configurations that account specifically for this

domain-adaptation pass (noise distributions based on general versus in-domain corpora, alternate stopping criteria), so that NCE-U would be presented in the most positive possible light. Perhaps most importantly, Table 3 shows that the domain adaptation pass is quite effective, producing large improvements for all NNJMs.

4.3 Impact on Speed

MLE and NCE both produce self-normalized models, so they both have the same impact on decoding speed. With the optimizations described by Devlin et al. (2014), the impact of any single-hidden-layer NNJM is negligible.

For training, the main benefit of NCE is that it reduces the cost of the network's output layer, replacing a term that was linear in the vocabulary size with one that is linear in the sample size. In our experiments, this is a reduction from 32K to 100. The actual benefit from this reduction is highly implementation- and architecture-dependent. It is difficult to get a substantial speedup from NCE using Theano on GPU hardware, as both reward dense matrix operations, and NCE demands sparse vector operations (Jean et al., 2015). Therefore, our decision to implement all methods in a shared codebase, which ensured a fair comparison of model quality, also prevented us from providing a meaningful evaluation of training speed, as the code and architecture were implicitly optimized to favour the most demanding method (MLE). Fortunately, there is ample evidence that NCE can provide large improvements to per-batch training speeds for NNLMs, ranging from a $2\times$ speed-up for 20K-word vocabularies on a GPU (Chen et al., 2015) to more than $10\times$ for 70K-word vocabularies on a CPU (Vaswani et al., 2013). Meanwhile, our experiments show that 1.2M batches are sufficient for MLE, NCE-T and NCE-M to achieve very high quality; that is, none of these methods made use of early stopping during their main training pass. This indicates that per-batch speed is the most important factor when comparing the training times of these NNJMs.

5 Conclusions

We have shown that NCE training with a unigram noise distribution does reduce NNJM performance with respect to MLE training, both in terms of model likelihoods and downstream translation quality. This performance drop can be avoided if NCE uses a translation-aware noise distribution. We have emphasized the importance of a domain-specific training pass, and we have shown that this pass magnifies the differences between the various NNJM training objectives. In a few cases, NCE with translation noise actually outperformed MLE. This suggests that there is value in only considering plausible translation candidates during training. It would be interesting to explore methods to improve MLE with this intuition.

Acknowledgments

Thanks to George Foster, Eric Joanis, Roland Kuhn and the anonymous reviewers for their valuable comments on an earlier draft.

References

Paul Baltescu and Phil Blunsom. 2015. Pragmatic neural language modelling in machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL (HLT-NAACL)*, pages 820–829, Denver, Colorado, May–June.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June.

X Chen, X Liu, MJF Gales, and PC Woodland. 2015. Recurrent neural network language model training with noise contrastive estimation for speech recognition. In *ICASSP*.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL (HLT-NAACL)*, pages 427–436, Montréal, Canada, June.

Colin Cherry. 2013. Improved reordering for phrase-based translation using sparse features. In *Proceedings of the Human Language Technology Conference of the NAACL (HLT-NAACL)*, pages 22–31, Atlanta, Georgia, June.

David Chiang, Yuval Marton, and Philip Resnik. 2008. Online large-margin training of syntactic and structural translation features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 224–233.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 176–181.

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 Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1370–1380, Baltimore, Maryland, June.

George Foster and Roland Kuhn. 2007. Mixture-model adaptation for SMT. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 128–135.

Michel Galley and Christopher D. Manning. 2008. A simple and effective hierarchical phrase reordering model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 848–856, Honolulu, Hawaii.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Artificial Intelligence and Statistics*, pages 297–304.

Nizar Habash, Ryan Roth, Owen Rambow, Ramy Eskander, and Nadi Tomeh. 2013. Morphological analysis and disambiguation for dialectal arabic. In *Proceedings of the Human Language Technology Conference of the NAACL (HLT-NAACL)*, pages 426–432, Atlanta, Georgia, June.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1352–1362.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China, July.

Philipp Koehn, Franz Joesef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the Human Language Technology Conference of the NAACL (HLT-NAACL)*, pages 127–133.

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 Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 177–180, Prague, Czech Republic, June.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1751–1758.

Franz Joesef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–52.

Franz Josef Och. 1999. An efficient method for determining bilingual word classes. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

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

Jingyi Zhang, Masao Utiyama, Eiichiro Sumita, Graham Neubig, and Satoshi Nakamura. 2015. A binarized neural network joint model for machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2094–2099, Lisbon, Portugal, September.