SCALING RECURRENT NEURAL NETWORK LANGUAGE MODELS

Will Williams, Niranjani Prasad, David Mrva, Tom Ash, Tony Robinson

Cantab Research, Cambridge, UK
{willw,tonyr}@cantabResearch.com

ABSTRACT

This paper investigates the scaling properties of Recurrent Neural Network Language Models (RNNLMs). We discuss how to train very large RNNs on GPUs and address the questions of how RNNLMs scale with respect to model size, training-set size, computational costs and memory. Our analysis shows that despite being more costly to train, RNNLMs obtain much lower perplexities on standard benchmarks than n-gram models. We train the largest known RNNs and present relative word error rates gains of 18% on an ASR task. We also present the new lowest perplexities on the recently released billion word language modelling benchmark, 1 BLEU point gain on machine translation and a 17% relative hit rate gain in word prediction.

***Index Terms*—** recurrent neural network, language modelling, GPU, speech recognition, RNNLM

1. INTRODUCTION

Statistical language models form a crucial component of many applications such as automatic speech recognition (ASR), machine translation (MT) and prediction for mobile phone text input. One such class of models, Recurrent Neural Network Language Models (RNNLMs), provide a rich and powerful way to model sequential language data. Despite an initial flurry of interest in RNNs in the early '90s for acoustic modelling [1, 2, 3], the computational cost and memory overheads of training large RNNs proved prohibitive. Since the earliest days of large vocabulary speech recognition, n-gram language models have been the dominant paradigm. However, recent work by Mikolov [4] on RNNLMs has shown that modestly sized RNNLMs can now be trained and have been shown to be competitive with n-grams. Mikolov trained RNNLMs with 800 hidden state units; Google's language modelling benchmark [5] subsequently established baseline RNNLM results with 1024 hidden state units. Many of the most recent ASR evaluations involve RNNLMs, highlighting their widespread popularity.

Although RNNs have replaced n-grams as state-of-the-art, the question remains whether these architectures will scale. Single CPU-based implementations have suffered from computational limitations and have been unable to scale to the large number of parameters now commonplace in the neural net literature. Concretely, we address the questions of how RNNLMs scale with respect to model size, training-set size, processing power and memory. Our primary performance metric here is perplexity and therefore all discussion on performance will relate specifically to the ability to reduce perplexity. The results presented here show the largest reductions in perplexity reported so far over KN 5-grams.

This paper summarises the results of SMART award "Large scale neural network language models" which ran from March 2013 to August 2014.

2. DATA SETS

2.1. Training Data

The main training corpus we use throughout this paper is an internally collated and processed data set. All data predates Nov. 2013 and totals approximately **8bn words**, comprising:

1. **[880m]** ***Spoken News***: Derived from transcripts of radio and news shows.
2. **[1.7bn]** ***Wikipedia***: Copy of Wikipedia articles.
3. **[6.1bn]** ***Written News***: Derived from a web crawl of many popular news sites.

We used a rolling buffer of minimum size of 140 characters split on sentences to deduplicate the corpora. We used multiple hash functions to store past occurrences efficiently and excluded a negligibly small proportion of text due to false positives. We then put the corpus through a typical ASR normalisation and tokenisation pipeline - expanding out numbers into digit strings, removing most punctuation, casting to lower case and inserting sentence markers.

2.2. Test Data

Unless otherwise stated, all our testing was done on a standard test benchmark: TED test data IWSLT14.SLT.tst2010[1]. We chose this out-of-domain test set to exclude any possibility of text overlap between training and test sets, and also because of the availability of a publicly available KALDI recipe[2].

2.3. Entropy Filtering

To filter our corpus we used cross-entropy difference scoring [6, 7]. Typically, sentences are used as the natural choice for segments in this style of filtering. However, for the composite corpus, we wanted to maintain between-sentence context for the benefit of the RNNLM training, and so did not want to filter our corpus on a sentence by sentence basis. We instead implemented a rolling-buffer solution: a cross-entropy difference score was calculated across rolling batches of 16 sentences. If a sentence was ever part of a rolling buffer with

[1]http://hltshare.fbk.eu/IWSLT2014/IWSLT14.SLT.tst2010.en-fr.en.tgz

[2]Kaldi Revision 4084, recipe: http://sourceforge.net/p/kaldi/code/HEAD/tree/trunk/egs/tedlium/s5/

cross-entropy difference below the threshold, it was maintained in the output corpus. This led to output that was more likely to be drawn from consecutive sentences, maintaining between-sentence information. The entropy threshold was then set at empirically determined levels to produce filtered corpora of the desired size, i.e. $\frac{1}{2}$, $\frac{1}{4}$ etc. of the original corpus size, as determined by word count. For typical in-domain data we used IWSLT13.ASR.train.en[3].

3. SCALING PROPERTIES OF N-GRAMS

n-gram language models have maintained their dominance in statistical language modelling largely due to their simplicity and ease of computation. Although *n*-grams can be modelled and queried comparatively fast, they exhibit diminishing gains when built on more data.

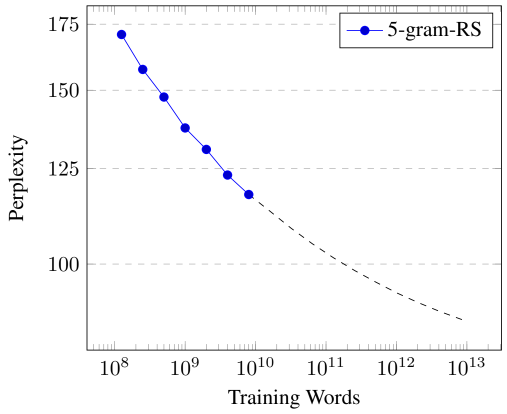

Fig. 1. Effect of increasing training data size on a randomly selected (RS) subsets of our training corpus with a 5-gram. The dashed line is extrapolation.

n-grams can be computed on larger data sets more easily than RNNLMs but as the extrapolation in Figure 1 indicates, each order of magnitude increase in the training data above 10^{12} words gives a reduction in perplexity of less than 6%. The largest current text corpora such as Google *n*-grams[4] and CommonCrawl [8] are about size 10^{12}. The asymptote of an exponential curve fitted to Figure 1 is approximately perplexity 73; these values represent a hard limit on the performance of 5-grams on this test set. *n*-grams also scale very poorly with respect to memory. At 8bn words the KN 5-gram already takes up 362GB in ARPA format and 69GB in KenLM [9] binary trie format - already impractically large for current commercial ASR. In comparison to RNNLMs, *n*-grams take up massively more space for modest entropy improvements and therefore do not scale well with respect to data size.

Additionally, increasing the order of the *n*-gram model gives vanishingly small gains for anything above a 5-gram trained on currently available corpora. The number of parameters of an *n*-gram increases significantly with its order, making attempts to increase the order impractical.

4. RNNLMS ON GPUS

4.1. Background

Recent attempts to train very large networks (on the order of billions of parameters) have required many CPU cores to work in concert [5, 10]. High-end Graphics Processing Units (GPUs) represent a viable alternative, being both affordable and capable of extremely high computational throughput. GPUs have therefore been one of the key elements in the resurgence of neural networks over the last decade. RNNLMs are highly amenable to parallelisation on GPUs due to their predominant reliance on linear algebra operations (such as matrix multiplies) for which highly optimised GPU-based libraries are publicly available.

In contrast to *n*-gram models, which simply count the occurrences of particular combinations of words, RNNs learn a distributed hidden representation for each context from which a probability distribution over all words in a given vocabulary can be obtained.

We use a standard RNN architecture [11] but dispense with bias units to maximise efficiency on the GPU and because, in our experience, they do not provide any practical benefit. Recent work has improved our understanding of how to effectively train RNNs [12]. However, many of these improvements - such as optimisation and regularisation techniques - make large claims on resources. Very large RNNs with a large number of hidden state units, *nstate*, can only be trained efficiently on current generation GPU hardware if one simplifies both the architecture and training algorithm as far as possible. We believe our setup from 2013 in this paper gives better speedups than previously reported elsewhere [13, 14, 15, 16, 17].

4.2. Implementation

To achieve high throughput and utilisation on GPUs we train a standard RNN with stochastic gradient descent and rmsprop[18]. Where possible we use CUDA's highly optimised linear algebra library cuBLAS to make SGEMM calls. Where this wasn't possible we wrote and optimised our own custom CUDA kernels. We use floats everywhere; the precision which doubles offer is unnecessary to learn good representations. We pack our input data using a data offset scheme which indexes the input corpus at a number of different points, denoted *noffset*. This entails managing *noffset* by minibatch size different hidden states throughout training. Typically we use *noffset* 128, similar to [16]. Empirically, we found a minibatch size of $\sim$ 8-10 represents a good trade-off between memory usage and perplexity. It also allows us to marshal a large and very efficient matrix multiply to perform all the hidden state-to-state and gradient calculations in one cuBLAS SGEMM operation. We use a very large rmsprop smoothing constant (0.9995). For the input-to-state and state-to-output matrices we approximate the majority

[3]`http://hltshare.fbk.eu/IWSLT2013/IWSLT13.ASR.train.en`

[4]`http://googleresearch.blogspot.co.uk/2006/08/all-our-n-gram-are-belong-to-you.html`

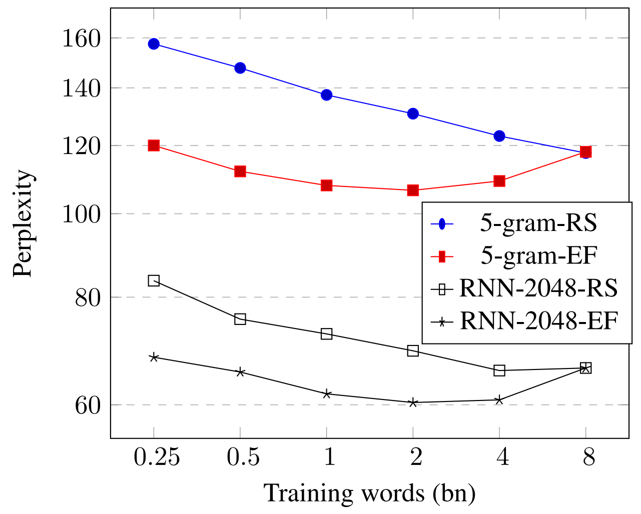

Fig. 2. Performance of 5-grams against *nstate* 2048 RNNs with increasing training data size. We test on Randomly Selected (RS) splits and Entropy Filtered (EF) splits of the 8bn corpus.

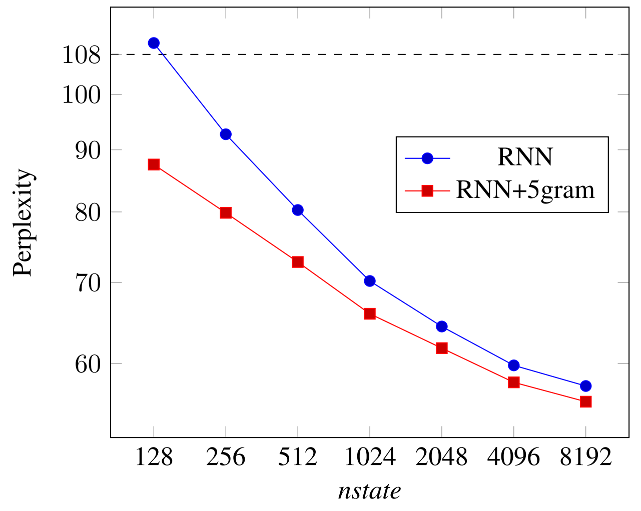

Fig. 3. Scaling *nstate* trained on 1bn words of the entropy filtered 8bn corpus. Dashed line is the 5-gram baseline.

of the rmsprop values by averaging over *nstate* units such that each word has just one rmsprop value rather than n values. This almost halves the total memory usage. We train using Noise Contrastive Estimation [19]. When reporting perplexities we explicitly normalise the output distribution. When performing lattice rescoring we find empirically that the small amount of normalisation noise does not significantly change the accuracy, whilst considerably faster than standard class based modelling.

4.3. Analysis

Figure 3 shows that as we scale *nstate* we observe a near linear reduction in log perplexity with log training words. Given that n-grams scale very poorly with respect to their order it is clear that, on a fixed-size data set, RNNs scale much better with model size. For network sizes over *nstate* 1024, our

Model	# Params [millions]	Training Time	Perplexity
KN 5-gram	1,740	30m	66.9
RNN - 128	16.4	6h	60.8
RNN - 256	32.8	16h	57.3
RNN - 512	65.8	1d2h	53.2
RNN - 1024	132	2d2h	48.9
RNN - 2048	266	4d7h	45.2
RNN - 4096	541	14d5h	42.4

Table 1. Perplexities on shard-0 ('news.en.heldout-00000-of-00050') from [5].

implementations on an Nvidia GeForce GTX Titan give 100x speedups against the baseline single core Mikolov implementation[5] on a 3.4GHz Intel i7 CPU. Despite much improved training times on the GPU, our larger RNNs take on the order of days to train rather than hours which n-grams require. However, more compute power will favour RNNs since larger *nstate* RNNs can leverage the extra computation to scale the model size with *nstate* - something which yields a much smaller performance gain for n-grams. With respect to scaling the training set size, Figure 2 shows that the *nstate* 2048 saturates (i.e. can not make good use of more data) on 4bn words when trained on a randomly chosen splits. We can, however, increase the model size (i.e. *nstate*) to mitigate this problem - an approach which, as already discussed, is impractical for n-grams. With an *nstate* 8192 RNN from Figure 3 we have already reduced perplexity to 57.5, which is below the 73 we believe to be the asymptotic minimum for a 5-gram on this task. It therefore seems unlikely that even a 5-gram with unlimited data and a further 15% reduction from entropy filtering could ever outperform an RNNLM. In addition, the *nstate* 8192 RNN corresponds to a 48% reduction over the 5-gram. Despite being so much better in terms of perplexity on the same amount of data, the RNN uses only 886M parameters - approximately half the number of the 5-gram. Moreover, we can match the perplexity of the 5-gram with an *nstate* 128 RNN which uses under 1% of the parameters. In summary, although training time is much larger for an RNN and we have to increase *nstate* when scaling the training set size, RNNs make much better use of the additional data than n-grams and use far fewer parameters to do so.

5. RESULTS

5.1. Language Modelling

We present our RNNLM results on the recently released billion word language modelling benchmark[5]. The vocabulary size for this task is very large (770K) - we therefore train the RNNs on a much smaller 64k vocabulary and interpolate them with a full size 5-gram to fill in rare word probabilities. RNN perplexities in Table 1 are interpolated with the KN 5-gram. The *nstate* 4096 RNN is larger in state size than those in [5] and

[5]`http://rnnlm.org/`

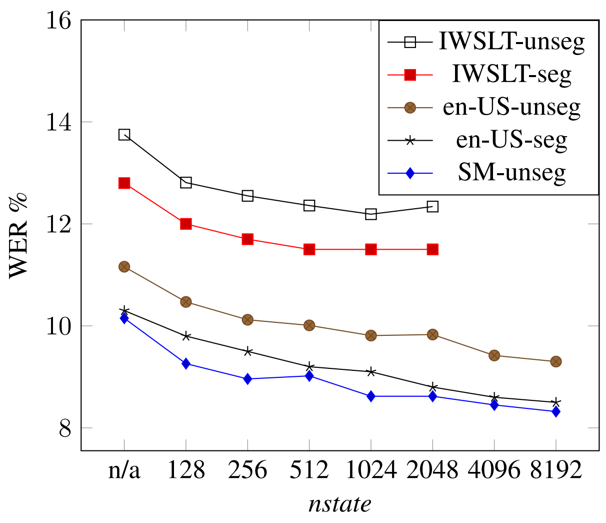

Fig. 4. Rescoring Kaldi Lattices with RNNLMs. 'n/a' refers to rescoring with n-gram only and is the n-gram baseline.

we achieve a better perplexity (42.4) with that one single model than the combination of interpolated models in [5] whilst also using only 3% of the total parameters.

5.2. ASR

We evaluated the RNNLMs by rescoring lattices on three different systems, using the IWSLT14.SLT.tst2010 data both with and without the supplied segmentation. The 'IWSLT' system uses the Kaldi TEDLIUM [20] recipe for acoustic models and language models built on [5][6]. The *nstate* 2048 models overtrain on the small entropy filtered corpus. The 'en-US' system uses the same framework but with the RNNLMs from Figure 3 and internal acoustic models. The 'SM' system also uses the RNNLMs from Figure 3, but within the commercial service available at `speechmatics.com`. Lattices were rescored using a highly efficient internal lattice rescoring tool which operates in considerably less than real time. Over both the 'en-US' and 'SM' task we see an average reduction in WER of 18% relative to rescoring with the n-gram alone by using *nstate* 8192 RNNLMs.

5.3. Machine Translation

In our MT experiments we rescored the WFST lattices from the WMT13 EN-RU system developed at Cambridge University Engineering Department [21]. The evaluation measure commonly used in statistical machine translation, BLEU score, was 32.34 with our baseline n-gram model and increased to 33.45 with an *nstate* 3520 RNNLM (bigger is better for BLEU scores). The improvement of 1 BLEU point over the baseline demonstrates the utility of RNNLMs in not just ASR but also statistical MT systems.

[6]The 12.8% IWSLT result is a competitive baseline that complies with IWSLT rules, using freely redistributable sources and can be recreated from the Kaldi TEDLIUM s5 recipe and `http://cantabResearch.com/cantab-TEDLIUM.tar.bz2`.

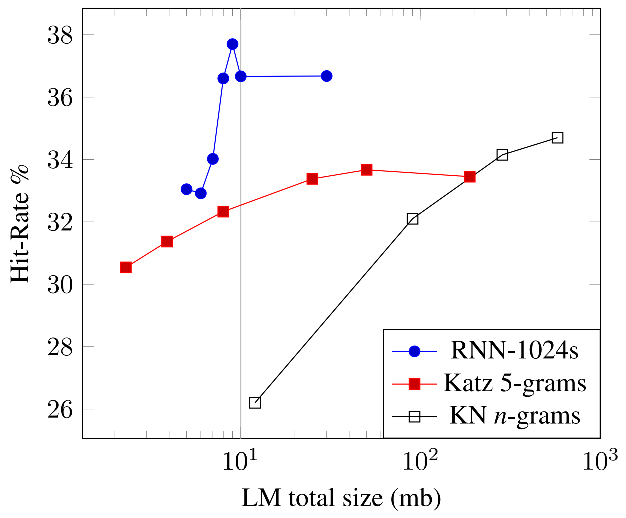

Fig. 5. Hit rates of Katz n-grams with different prune thresholds, RNNs quantised to different numbers of bits per weight and KN n-grams of increasing order. 7-bit quantisation produces an anomalously high hit-rate - we are unsure what is interacting to product this effect.

5.4. Word prediction

Word prediction involves predicting the next word in a string, with performance determined by percentage of times the target word is in the top 3 words predicted (known as 'hit-rate'). For this task we train on 100M words from the BNC corpus [22] using a vocabulary size of 10K and the RNN using a shortlist of 100 candidate words generated from a heavily pruned 2 MB n-gram. For our task of word prediction on mobile phones we have tight memory restrictions and therefore choose *nstate* 1024 as an appropriate RNN size. We compress the RNNs by inserting layers with 512 units above and below the hidden state and then train end-to-end. Additionally, we tie the input-state and state-output weights as their transpose and quantise all the weights. At 10 MB the RNN achieves a 17% relative hit-rate gain over the Katz-5 n-gram, proving the utility of RNNs in memory constrained settings.

6. CONCLUSION

We have shown that large RNNLMs can be trained efficiently on GPUs by exploiting data parallelism and minimising the number of extra parameters required during training. Such RNNs reduce the perplexity on standard benchmarks by over 40% against 5-grams, whilst using a fraction of the parameters. We believe RNNs now offer a lower perplexity than 5-grams for any amount of training data. In addition, we showed that state of the art ASR systems can be trained with Kaldi and high-end GPUs by rescoring with an RNNLM. Despite being both compute and GPU memory bound, RNNLMs are comfortably ahead of n-grams at present. We believe that future developments in compute power and memory capacity will further favour them.

7. REFERENCES

[1] Anthony J. Robinson, "An application of recurrent nets to phone probability estimation," *Neural Networks, IEEE Transactions on*, vol. 5, no. 2, pp. 298–305, 1994.

[2] Ronald J. Williams, "Complexity of exact gradient computation algorithms for recurrent neural networks," Tech. Rep., 1989.

[3] Mike Schuster and Kuldip K. Paliwal, "Bidirectional recurrent neural networks," *Signal Processing, IEEE Transactions on*, vol. 45, no. 11, pp. 2673–2681, 1997.

[4] Tomáš Mikolov, *Statistical language models based on neural networks*, Ph.D. thesis, Brno University of Technology, 2012.

[5] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson, "One Billion Word Benchmark for measuring progress in statistical language modeling," .

[6] Dietrich Klakow, "Selecting articles from the language model training corpus," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*. IEEE, 2000, vol. 3, pp. 1695–1698.

[7] Robert C. Moore and William Lewis, "Intelligent selection of language model training data," in *Proceedings of the ACL 2010 Conference Short Papers*. Association for Computational Linguistics, 2010, pp. 220–224.

[8] Christian Buck, Kenneth Heafield, and Bas van Ooyen, "N-gram counts and language models from the common crawl," LREC, 2014.

[9] Kenneth Heafield, "KenLM: Faster and smaller language model queries," in *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, 2011, pp. 187–197.

[10] Quoc V Le, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Jeff Dean, and Andrew Y. Ng, "Building high-level features using large scale unsupervised learning," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8595–8598.

[11] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur, "Recurrent neural network based language model," in *Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2010, pp. 1045–1048.

[12] Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu, "Advances in optimizing recurrent networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8624–8628.

[13] Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukar Burget, and J Cernocky, "Rnnlm-recurrent neural network language modeling toolkit," .

[14] Ilya Sutskever, James Martens, and Geoffrey E Hinton, "Generating text with recurrent neural networks," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 1017–1024.

[15] Zhiheng Huang, Geoffrey Zweig, Michael Levit, Benoit Dumoulin, Barlas Oguz, and Shawn Chang, "Accelerating recurrent neural network training via two stage classes and parallelization," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 326–331.

[16] Xie Chen, Yongqiang Wang, Xunying Liu, Mark JF Gales, and Philip C Woodland, "Efficient GPU-based training of recurrent neural network language models using spliced sentence bunch," *Proc. ISCA Interspeech*, 2014.

[17] Boxun Li, Erjin Zhou, Bo Huang, Jiayi Duan, Yu Wang, Ningyi Xu, Jiaxing Zhang, and Huazhong Yang, "Large scale recurrent neural network on gpu," in *Neural Networks (IJCNN), 2014 International Joint Conference on*. IEEE, 2014, pp. 4062–4069.

[18] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, vol. 4, 2012.

[19] Andriy Mnih and Koray Kavukcuoglu, "Learning word embeddings efficiently with noise-contrastive estimation," in *Advances in Neural Information Processing Systems*, 2013, pp. 2265–2273.

[20] Anthony Rousseau, Paul Deléglise, and Yannick Estève, "TED-LIUM: an Automatic Speech Recognition dedicated corpus," in *LREC*, 2012, pp. 125–129.

[21] J. Pino, A. Waite, T. Xiao, A. de Gispert, F. Flego, and W. Byrne, "The University of Cambridge Russian-English system at WMT13," in *Proceedings of the Eighth Workshop on Statistical Machine Translation*, 2013, pp. 198–203.

[22] BNC Consortium et al., "The British National Corpus, version 3 (BNC XML Edition), distributed by Oxford University Computing Services on behalf of the BNC Consortium," 2007.