



Efficient Machine Translation Decoding with Slow Language Models

Ahmad Emami

IBM T.J. Watson Research Center
 1101 Kitchawan Rd, Yorktown Heights, NY 10598
 emami@us.ibm.com

Abstract

Efficient decoding has been a fundamental problem in machine translation research. Usually a significant part of the computational complexity is found in the language model cost computations. If slow language models, such as neural network or maximum-entropy models are used, the computational complexity can be so high as to render decoding impractical. In this paper we propose a method to efficiently integrate slow language models in machine translation decoding. We specifically employ neural network language models in a hierarchical phrase-based translation decoder and achieve more than 15 times speed-up versus directly integrating the neural network models. The speed-up is achieved without any noticeable drop in machine translation output quality, as measured by automatic evaluation metrics. Our proposed method is general enough to be applied to a wide variety of models and decoders.

1. Introduction

Statistical Machine Translation (SMT) systems work by searching through a large space of possible translation hypotheses and choosing the most plausible translation. This search is guided by using models, mainly the Translation Model (TM) and the Language Model (LM), which score each translation hypothesis.

The space of all possible translations for a given sentence f is very large; in fact the decoding problem for word-based, as well as phrase-based, models is NP-complete [1]. All decoders work by searching only through a smaller subset of possible translations.

Translation model costs are usually local and monotonic, meaning that the total translation cost of a hypothesis is simply the sum of the translation costs of the steps used to generate that given hypothesis. However the language model cost is not monotonic, requiring language model states to be maintained which results in a significant expansion of the search space.

In practice, a significant portion of decoding time is spent in computing the language model costs. This becomes problematic for very large or slow language models. In the case of neural probabilistic language models [2] this can be an inhibiting factor.

In this paper, we propose a method that enables the use of slow and complex language models for decoding in machine translation. Our approach is to slightly delay the model cost computations, and compute and assign the language model costs only once the set of translations for a subset of source words is finalized. In order to avoid severe search errors, a baseline language model is used to guide the decoding search.

We evaluate our approach using a hierarchical phrase-based translation system [3] augmented with tree-to-string rules, with a neural probabilistic language model acting as the slow model.

However our approach is not limited to this setup and can be applied to different decoders (eg. phrase-based decoders [4]), as well as other types of slow models, for instance large back-off language models, or maximum-entropy models such as ModelM [5, 6].

Experimental results show more than a 15 times speed-up in decoding without a noticeable loss in translation quality.

This paper is organized as follows: Sections 2 and 3 provide background information on neural network language models and hierarchical phrase-based translation systems respectively. In Section 4 we describe our algorithm for efficiently using slow models in MT decoding. Experimental results are given in Section 5, followed by comparisons to earlier work and a discussion.

2. Neural Probabilistic Language Models

Neural probabilistic language models project the word history into a continuous space and use a multi-layered network to compute a probability distribution over all possible future words [2]. The neural network is composed of one or more hidden layers followed by a *softmax* output layer:

$$p_k = \frac{e^{z_k}}{\sum_j e^{z_j}} \quad k = 1, 2, \dots, |V_o| \quad (1)$$

where z_j is the j^{th} input to the softmax layer, V_o is the *output vocabulary* (set of items we are predicting, usually target language words), and p_k is the conditional probability of the k^{th} item of this vocabulary given the specified input (history). Computing the softmax layer is computationally expensive since it involves a normalization $Z = \sum_j e^{z_j}$ over all the items in the output vocabulary.

3. Hierarchical Phrase-Based Translation

While our approach is applicable to most decoders, we have chosen to apply it to a Tree-to-String Hierarchical Phrase-Based Translation system [7]. Hierarchical phrase-based translation has been proven to be a simple and powerful translation system [3].

The hierarchical phrase-based system uses hierarchical phrases, meaning it uses phrases that themselves contain other phrases. The system formally works by employing a weighted *Synchronous Context-Free Grammar (SCFG)* [8]. The decoding process is basically a parsing of the source sentence according to the given grammar. Target translations are synchronously built by following the target side of the applied SCFG rules. The parsing is performed using the CYK+ algorithm [9] which allows a CYK-style parsing to be carried out without the need to modify the original synchronous grammar.

10.21437/Interspeech.2015-514

Algorithm 1 Deferred-Cost Decoding

```
1: function DECODE( $C$ ) ▷ input is a Chart
2:   for  $c \in C$  in topological order do
3:     build  $c$  using cube pruning (or similar)
4:     COLLECTEVENTS( $c, t$ )
5:     SCORECELL( $c, t$ )
6:     re-rank hypotheses in  $c$  ▷ with new total costs
7:   procedure COLLECTEVENTS( $c, t$ )
8:     for  $h \in c$  do ▷  $h$  is a hypothesis
9:       for  $e \in h$  do ▷  $e$  is an event
10:        if  $e \notin t$  then
11:          add  $e$  to  $t$ 
12:   procedure SCORECELL( $c, t$ )
13:     for  $h \in c$  do ▷  $h$  is a hypothesis
14:        $nc \leftarrow 0$  ▷  $nc$  is the new cost
15:       for  $e \in h$  do ▷  $e$  is an event
16:          $nc \leftarrow nc + t(e)$ 
17:       append  $nc$  to the list of costs of  $h$ 
18:       recompute total cost for  $h$ 
```

The search space is organized as a *chart* where each cell of the chart holds the translations for a given span (a sub-phrase) of the source sentence. Decoding is carried out in a hierarchical bottom-up manner, where shorter spans (lower cells) are fully built before utilizing them to construct translations in the upper cells.

To find the optimum translation for a given grammar rule applied to a given cell, ideally all the possible translations reachable by that rule should be constructed and scored by the language model, which is computationally infeasible. *Cube Pruning* [10] addresses this issue by evaluating only a small number of combinations of hypotheses from predecessors cells. This can be repeated for each rule, or all the rules applicable to the cell in question can be folded into the cube pruning search itself.

While cube pruning or similar algorithms make decoding with a language model possible, they still construct and compute costs for a relatively large number of partial translations. This becomes problematic when using slow language models. For example when a neural network language model is used in the decoder (with cube pruning) the decoding speed drops to values much lower than what is acceptable for practical purposes. In the next section we describe our approach to solving this problem.

4. Deferred-Cost Decoding

Since the underlying problem is the time complexity of the language model score computation, it seems logical to try to reduce the number of language model lookups as much as possible. Cube pruning for example, attempts to reduce computation by limiting the number of hypotheses that are generated and scored. Our proposed algorithm instead attempts to decrease the number of LM lookups by slightly delaying LM cost computations to a more convenient stage. We observe that in typical decoding algorithms, each newly constructed hypothesis is scored independently of others, which impedes sharing of common LM computations among translations. Furthermore in cube pruning a significant number of generated hypotheses end up not being added to their destination cell, meaning that it was not necessary to carry out their LM computations.

We approach the problem of reducing the number of calls

to the slow LM by deferring it from the hypothesis generation (cube pruning) stage until after the cell is fully populated. Only when a cell is fully constructed and populated, the deferred costs (for the slow model) are computed and the cell hypotheses are re-ranked according to the new total costs. To further reduce the number of calls to the slow LM, the set of all n -grams contained in the current cells hypotheses is first computed, then the slow LM is called only once for each n -gram.

Algorithm 1 provides the details of our deferred-cost method. The main idea is to defer the slow model cost computation from cube pruning (step 3) to after the cell is fully populated. Procedures at steps 4 and 6 are to make sure each unique n -gram in the cell is scored by the slow LM only once. Steps 7-11 compute all the n -grams contained in the cell and unique them, while steps 12-18 used the scored n -grams to compute the slow LM cost for each hypothesis in the cell. Once the slow LM cost computations are finalized, the cell hypotheses are re-ranked based on their new total costs.

While quite simple, the proposed algorithm reduces LM computations in a few ways. Firstly, LM costs are computed only for hypotheses that are stored in a cell, and not for all the hypotheses that are generated (many of which are usually discarded). Secondly, every unique LM query in a cell is scored only once. Thirdly, and this is specific to neural network models, queries can be sent to the model in mini-batches which results in significant speed-ups over single query calls [11, 12].¹ The latter two optimizations, while very effective, introduce no additional search errors.

5. Experiments

We report results on a Arabic-to-English translation system. Our decoder is a syntax-based hierarchical [3] decoder that also allows for tree-to-string rules in addition to a hierarchical phrase-based (Hiero) grammar. [7]. Excluding language models, we use 16 feature functions, most of which are similar to those used in phrase-based translation systems such as Moses [4]. These include a brevity penalty, IBM Model-1 alignment probabilities in both directions, relative frequencies in both directions, word/rule counts, content/function word mismatch features, along with features for tree-to-string rules. To this a total of 16 *provenance* features were added by splitting the training data into 8 genres and computing provenance conditioned lexical features in both directions for each genre [13]. The model (log-linear) weights were tuned using PRO [14] on a held-out set. Our deferred-cost method was used to find a weight and involve the neural network model in the tuning process.

We trained our system on all of the parallel training data available to us (United Nations data was excluded since doing so results in a stronger baseline) which consisted of about 2 million parallel sentences. We chose the standard test sets of combined *newswire* and *weblog* genres from the NIST MT08 and MT09 evaluations. Our training and test data is summarized in Table 1.

Our baseline language model is a 6-gram modified Knneser-Ney model trained on around 20 billion words of English data. We evaluated our idea of decoding with slow models by adding a neural network language model (NNLM) which is prohibitively slow (see Section 2).

¹Due to the fact that on current CPUs, matrix-matrix operations can be optimized much more aggressively than matrix-vector operations, even though the total number of computations are the same

Table 1: Training and Test Data

data	sentences	source tokens	target tokens
Training	2.06M	38M	40M
MT08	1,360	45K	-
MT09	1,313	41K	-

Table 2: MT08 Results

MT08	BLEU	TER	time	lookups
fully integrated	51.34	42.46	100%	3325
deferred-cost	51.14	42.45	5.795%	140

Table 3: MT09 Results

MT09	BLEU	TER	time	lookups
fully integrated	54.48	39.21	100%	1885
deferred-cost	54.41	39.12	6.300%	124

The NNLM was trained on 336 million words of monolingual English data with a sampling rate of 0.1, meaning a random sample of about 34M words was chosen for each epoch. The neural network model consisted of two hidden layers, (with 1024 and 384 units respectively) and it employed 320 dimensional feature (word representation) vectors. The input and output vocabularies consisted of 120K and 32K words respectively.

Tables 2 and 3 show BLEU [15] and TER [16] scores, along with decoding times for MT08 and MT09 test sets. Decoding time is shown as a percentage of time needed to decode with fully integrated NNLMs. The deferred-cost algorithm results in a 15-17 times speed-up in decoding without any considerable loss in scores: a drop of 0.1 points and no change in (TER-BLEU)/2 on MT08 and MT09 respectively. Baseline (no NN) MT08 systems achieved BLEU and TER scores of 50.32 and 43.08, while the baseline MT09 scores were 53.67 and 39.68 respectively. Therefore almost all the gains over the baseline system from using NNLM models were kept while achieving a significant speed up in decoding time.

We also compare the total number of NNLM lookups (in millions) for fully integrated versus deferred-cost decoding. We observe a significant drop in the number of queries to the NNLM when using the deferred-cost decoding approach.

Figure 1 shows decoding time (in log base-10 second) for varying sentence lengths. As the sentence length increases, the gap between fully integrated and deferred-cost decoding increases even more.

An obvious comparison would be to run the decoder with fully integrated NNLM but with tighter search parameters for a faster decoding time. We ran one such experiment on the MT08 test set which still had a decoding time 3 times slower than the deferred-cost system, while being worse by about 1.1 and 0.6 points in BLEU and TER respectively.

6. Comparison to Earlier Work

A common approach to using slow and/or complex models is to employ them in an N -best re-ranking framework. Our approach can be thought of as a generalization of N -best re-ranking; If we limit Lines 4-6 in Algorithm 1 to the final cell only, that would be exactly equivalent to N -best re-ranking. Therefore our method maintains the generality and ease of use of N -best re-ranking without suffering from less search errors. Also since

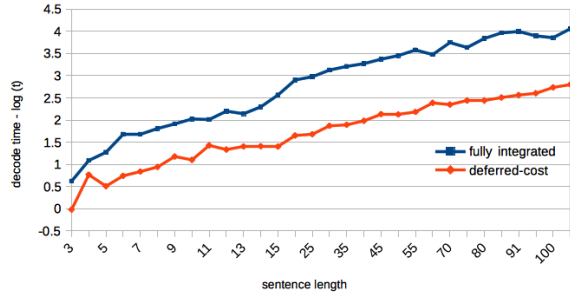


Figure 1: Decoding time per sentence lengths

our deferred-cost actually integrates the slow model in the decoding it can be used in iterative weight tuning algorithms such as MERT [17] or PRO [14]. N -best re-ranking methods are limited to one iteration only since the actual N -best lists are not going to change from one iteration to the other.

In order to use very large distributed language models efficiently, [18] propose a *batch querying* method. However, for them the bottleneck is the network communication and not the cost computation itself. Therefore, in a trade-off, they actually query more n -grams than required so as to reduce the number of network calls. In contrast our goal is (and our method aims) to reduce the number of model calls.

Another approach in reducing the computational cost of NNLMs is to train *un-normalized* models [19, 20]. These models are trained such that during decoding, computing the softmax denominator in Equation 1 is not required. However the un-normalized models are only an approximation and there is usually some degradation in systems results. Furthermore, our approach still applies to un-normalized models and combining the two will take advantage of both speed-ups. It should also be noted that the un-normalized method applies to neural network (or maximum-entropy) models only while our approach is general and can be used for any type of models.

7. Conclusion

In this paper we proposed a method to efficiently use a slow model, in our case a neural network language model, for MT decoding. Our algorithm achieves more than 15 times speed-up in decoding versus using the NNLM in the usual cube pruning setup, without any noticeable loss in decoding results as measured by automatic MT evaluation metrics.

Our approach can be thought of as a generalized N -best re-ranking method, and its generality allows it to be used on a wide range of models (eg. maximum-entropy, random forests) and decoders (eg. phrase-based or even speech recognition systems). Our approach does not suffer from as severe search errors as N -best re-ranking, and unlike the N -best re-ranking method it allows for iterative weight tuning algorithms such as MERT and PRO.

8. Acknowledgments

We would like to acknowledge the support of DARPA under Grant HR0011-12-C-0015 for funding part of this work. The views, opinions, and/or findings contained in this article/presentation are those of the author/ presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the DARPA.

9. References

- [1] K. Knight, "Decoding complexity in word-replacement translation models," *Comput. Linguist.*, vol. 25, no. 4, pp. 607–615, Dec. 1999.
- [2] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.
- [3] D. Chiang, "A hierarchical phrase-based model for statistical machine translation," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ser. ACL '05. Stroudsburg, PA, USA: Association for Computational Linguistics, 2005, pp. 263–270.
- [4] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst, "Moses: Open source toolkit for statistical machine translation," in *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ser. ACL '07. Stroudsburg, PA, USA: Association for Computational Linguistics, 2007, pp. 177–180.
- [5] S. Chen, "Shrinking exponential language models," in *The Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Boulder, Colorado, June 2009, pp. 468–476.
- [6] A. Emami, S. F. Chen, A. Ittycheriah, H. Soltau, and B. Zhao, "Decoding with shrinkage-based language models," in *INTER-SPEECH*, 2010, pp. 1033–1036.
- [7] B. Zhao and Y. Al-onaizan, "Generalizing local and non-local word-reordering patterns for syntax-based machine translation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '08. Stroudsburg, PA, USA: Association for Computational Linguistics, 2008, pp. 572–581. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1613715.1613785>
- [8] A. V. Aho and J. D. Ullman, "Syntax directed translations and the pushdown assembler," *J. Comput. Syst. Sci.*, vol. 3, no. 1, pp. 37–56, Feb. 1969.
- [9] J.-C. Chappelier and M. Rajman, "A generalized cyk algorithm for parsing stochastic cfg," in *TAPD*, 1998, pp. 133–137.
- [10] D. Chiang, "Hierarchical phrase-based translation," *Comput. Linguist.*, vol. 33, no. 2, pp. 201–228, Jun. 2007.
- [11] J. Bilmes, K. Asanović, C.-W. Chin, and J. Demmel, "Using PHiPAC to speed error back-propagation learning," in *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, April 1997.
- [12] H. Schwenk, "Continuous space language models," *Comput. Speech Lang.*, vol. 21, no. 3, pp. 492–518, Jul. 2007. [Online]. Available: <http://dx.doi.org/10.1016/j.csl.2006.09.003>
- [13] D. Chiang, S. DeNeeffe, and M. Pust, "Two easy improvements to lexical weighting," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, ser. HLT '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 455–460.
- [14] M. Hopkins and J. May, "Tuning as ranking," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 1352–1362.
- [15] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 311–318.
- [16] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul, "A study of translation edit rate with targeted human annotation," in *In Proceedings of Association for Machine Translation in the Americas*, 2006, pp. 223–231.
- [17] F. J. Och, "Minimum error rate training in statistical machine translation," in *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ser. ACL '03. Stroudsburg, PA, USA: Association for Computational Linguistics, 2003, pp. 160–167.
- [18] T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean, "Large language models in machine translation," in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007, pp. 858–867.
- [19] A. Mnih and Y. W. Teh, "A fast and simple algorithm for training neural probabilistic language models," in *Proceedings of the 29th International Conference on Machine Learning*, 2012, pp. 1751–1758.
- [20] J. Devlin, R. Zbib, Z. Hunag, T. Lamar, R. Schwartz, and J. Makhoul, "Fast and robust neural network joint models for statistical machine translation," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. Baltimore, MD: Association for Computational Linguistics, June 2014.