INTERSPEECH 2009 BRIGHTON

Measuring Tagging Performance of a Joint Language Model

Denis Filimonov[1], *Mary Harper*[1,2]

[1]Laboratory for Computational Linguistics and Information Processing
Institute for Advanced Computer Studies
University of Maryland, College Park
[2]Human Language Technology Center of Excellence
Johns Hopkins University
{den,mharper}@umiacs.umd.edu

10.21437/Interspeech.2009-120

Abstract

Predicting syntactic information in a joint language model (LM) has been shown not only to improve the model at its main task of predicting words, but it also allows this information to be passed to other applications, such as spoken language processing. This raises the question of just how accurate the syntactic information predicted by the LM is. In this paper, we present a joint LM designed not only to scale to large quantities of training data, but also to be able to utilize fine-grain syntactic information, as well as other features, such as morphology and prosody. We evaluate the accuracy of our model at predicting syntactic information on the POS tagging task against state-of-the-art POS taggers and on perplexity against the ngram model.
Index Terms: language modeling, joint language model, part-of-speech tagging

1. Introduction

Part-of-speech (POS) tags are helpful in a number of NLP applications: named entity recognition, sentence boundary detection, word sense disambiguation, to name a few. The HMM approach, historically the first to be applied to the task of POS tagging, has been largely superseded by sophisticated algorithms [1, 2] that make better use of scarce human-annotated data. Recently, some systems have added large bodies of unannotated or automatically annotated text to assist in the tagging task, reporting significant improvements in acuracy [3, 4].

It has also been shown that augmenting output of a language model with predicted POS tags, i.e., predicting joint word-tag events, can improve performance of ASR applications, such as detecting speech repairs and intonational phrase boundaries [5], thus giving an additional incentive to develop joint language models. In this paper, we present a joint language model that can predict syntactic tags as by-product, and compare our model against state-of-the-art POS taggers representing four different approaches: HMM [6], MaxEnt [1], CRF [3], and perceptron [4]. Note, that all these high-performing taggers are bi-directional to some extent, ranging from simple word look-ahead to varying the order of tag prediction, which renders them inappropriate to certain tasks requiring a strict left-to-right approach.

Designing a scalable joint model is notoriously hard. For example, the maximum amount of data that the model in [5] (arguably the closest model to ours) was trained on was only 1M words. In this paper, we present our model trained on 170M words of data, with fine-grain syntactic tags.

While there are other joint language models that use syntactic information in various forms that could potentially be used to predict POS tags [5, 7, 8, 9], none of them has been evaluated on the POS tagging task, which is an important measure if the tags are to be used in a speech recognition task[1]. In Section 2 of this paper, we outline the design of our language model and evaluate it on the POS tagging task and on perplexity in Section 3.

2. Language Model

A typical application of a language model in Automatic Speech Recognition (ASR) is to provide the prior probability in the noisy channel framework:

$$w_1^n = \underset{w_1^n}{argmax}\ p(w_1^n|A) = \underset{w_1^n}{argmax}\ p(A|w_1^n) \cdot p(w_1^n) \quad (1)$$

where $p(A|w_1^n)$ is the acoustic model and $p(w_1^n)$ is the language model. The latter is typically factored as[2]:

$$p(w_1^n) = \prod_{i=1}^{n} p(w_i|w_1^{i-1})$$

A joint language model predicts joint events $p(w_i t_i|w_1^{i-1}t_1^{i-1})$, where t_i is a *tag* assigned to the word w_i, such as part-of-speech tag. If the joint model is used, the distribution is marginalized:

$$p(w_1^n) = \prod_{i=1}^{n} p(w_i|w_1^{i-1}) = \sum_{t_1 \ldots t_n} \prod_{i=1}^{n} p(w_i t_i|w_1^{i-1}t_1^{i-1})$$

However it is also possible (and beneficial [5]) to augment Equation 1 to allow the acoustic model to use tags predicted by the LM:

$$w_1^n t_1^n = \underset{w_1^n t_1^n}{argmax}\ p(A|w_1^n t_1^n) \cdot p(w_1^n t_1^n) \quad (2)$$

The size and sparsity of the parameter space of the joint model necessitate the use of dimensionality reduction measures

[1]While [5] presents raw POS error rate, the results were obtained on tiny 58k-word Trains corpus and were not compared to any other system.

[2]Although models that require the entire sentence up front do exist, e.g., [10], their applicability to ASR tasks is limited.

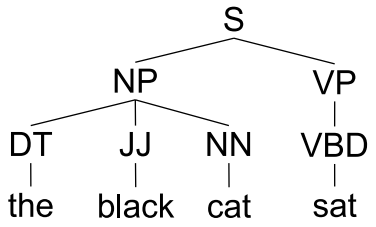

Figure 1: A parse tree example

in order to make the model computationally tractable and to allow for accurate estimation of the model's parameters. We also want the model to be able to easily accommodate additional sources of information, such as morphological features and prosody. In the rest of this section we discuss avenues we have taken to address these problems.

2.1. Fine-grain Syntactic Tags

The model presented in this section was designed to accommodate large tagsets. In this paper we evaluate the impact of extending POS tags with additional syntactic information derived from parse trees, namely:

- A POS tag with its immediate parent in the parse tree, along with the tag's relative position among its siblings. We refer to this tagset as *parent*. For example, the tree in Figure 1 will be tagged: *the/DT-NP-start black/JJ-NP-mid cat/NN-NP-end sat/VB-VP-single*.
- A POS tag of a word with the POS tag of the word's lexical head, later referred to as *head*. The example in Figure 1 is tagged: *the/DT-NN black/JJ-NN cat/NN-VBD sat/VBD-root*.

The English POS tag set has 45 tags, while the fine-grain tagsets have approximately 1,500 tags.

2.2. Decision Tree Clustering

Binary decision tree clustering has been shown to be effective for reducing the parameter space in language modeling [5, 11] and other language processing applications, e.g., [12]. Like any clustering algorithm, it can be represented by a function H that maps the space of histories to a set of equivalence classes.

$$p(w_i t_i | w_{i-n+1}^{i-1} t_{i-n+1}^{i-1}) \approx p(w_i t_i | H(w_{i-n+1}^{i-1} t_{i-n+1}^{i-1})) \quad (3)$$

Since optimal tree construction is intractable, greedy algorithms are universally used. While the procedure is standard – to recursively select binary questions about the history optimizing some function – the greedy nature of the approach makes the details, such as which questions to ask and which function to optimize, crucial to achieve good results. In the remainder of this section, we discuss the decisions we made regarding these issues.

2.3. Factors

The Factored Language Model (FLM) [13] offers a convenient view of the input data: it represents every word in a sentence as a tuple of factors. This allows us to extend the language model with additional parameters to build a more effective model. In an FLM, however, all factors have to be deterministically computed; whereas, we need to distinguish between the factors that are given or computed and the factors that the model must predict stochastically. We call these types of factors *overt* and *hidden*, respectively. Examples of overt factors include surface words, morphological features such as suffixes, case information when available, etc.; the hidden factors are POS tags or other syntactic variables.

In this work, we used suffixes obtained using the Snowball stemmer[3] and a simple regular expression-based feature identifying whether the word is a number, a punctuation, an abbreviation, a foreign word, etc. Henceforth, we will use *word* to represent the set of overt factors and *tag* to represent the set of hidden factors.

2.4. Hidden Factors Tree

Similarly to [5], we construct a binary tree where each tag is a leaf; we refer to this tree as the *Hidden Factors Tree* (HFT). We use Minimum Discriminative Information (MDI) algorithm [14] to build the tree. The HFT represents a hierarchical clustering of the tag space. One of the reasons for doing this is to allow questions about subsets of tags rather than individual tags alone[4].

Unlike [5], where the tree of tags was only used to create questions, this representation of the tag space is, in addition, a key feature of our decoding optimizations to support model scalability. We refer the reader to [15] for the details on the decoding algorithm.

2.5. Questions

We use different types of questions for hidden and overt factors.

- Questions about surface words are constructed using the Exchange algorithm [16]. This algorithm takes the set of words that appear at a certain position in the training data associated with the current node in the history tree and divides the set into two complementary subsets greedily optimizing on some target function (we use the average entropy of the marginalized word distribution, the same as for question selection). Note that since the algorithm only operates on the words that appear in the training data, we need to do something more to account for the unseen words. Thus, to represent this type of question, we create the history tree structure depicted in Fig. 2. For other overt factors with smaller vocabularies, such as suffixes, we use equality questions[5].
- As we mentioned in 2.4, we use the Hidden Factors Tree to create questions about hidden factors. Note that every node in a binary tree can be represented by a binary path from the root with all nodes under an inner node sharing the same prefix. Thus, a question about whether a tag belongs to a subset of tags dominated by a node can be expressed as whether the tag's path matches the binary prefix.

[3] http://snowball.tartarus.org/

[4] Trying all possible subsets of tags is not feasible since there are $2^{|T|}$ of them, the tree allows us to reduce the number to $O(T)$ of the most meaningful (as per the clustering algorithm) subsets.

[5] E.g., suffix(w_{i-1}) = 'ed'.

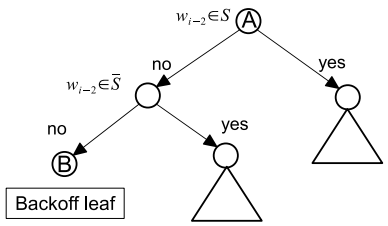

Figure 2: Backoff Nodes

2.6. Optimization Criterion and Stopping Rule

To select questions we use the average entropy of the marginalized word distribution. We found that this criterion significantly outperforms the entropy of the distribution of joint events. This is probably due to the increased sparsity of the joint distribution.

2.7. Smoothing

In order to estimate probability distributions at the leaves of the history tree, we use the following recursive formula:

$$\tilde{p}_n(w_i t_i) = \lambda_n p_n(w_i t_i) + (1 - \lambda_n)\tilde{p}_{n'}(w_i t_i) \qquad (4)$$

where n' is the n-th node's parent, $p_n(w_i t_i)$ is the ML distribution at node n. The root of the tree is interpolated with the distribution $p_{unif}(w_i t_i) = \frac{1}{|V|} p_{ML}(t_i|w_i)$[6]. To estimate interpolation parameters λ_n, we use the EM algorithm described in [12], however, rather than setting aside a separate development set of optimizing λ_n, we use 4-fold cross validation and take the geometric mean of the resulting coefficients[7]. We chose this approach because a small development set often does not overlap with the training set for low-count nodes, leading the EM algorithm to set $\lambda_n = 0$ for those nodes.

For backoff nodes (see Figure 2), we use a lower order model[8] interpolated with the distribution at the backoff node's grandparent (see node A in Figure 2):

$$\begin{aligned}\tilde{p}_B(w_i t_i|w_{i-n+1}^{i-1} t_{i-n+1}^{i-1}) &= \alpha_A \tilde{p}_{bo}(w_i t_i|w_{i-n+2}^{i-1} t_{i-n+2}^{i-1}) \\ &\quad + (1 - \alpha_A)\tilde{p}_A(w_i t_i)\end{aligned}$$

How to most effectively compute α_A is an open question. For this study, we use a simple heuristic based on observation that the further node A is from the root the more reliable the distribution $\tilde{p}_A(w_i t_i)$ is, and hence α_A is lower.

2.8. Tagging

When the model uses tags finer-grained than POS, the best POS sequence cannot be tractably inferred from the decoding lattice. We use the following variational approximation to estimate probability of POS tag sequences. Here, we denote POS tags T while t are the fine-grained tags used by the model. Note that, by construction, each t belongs to exactly one POS tag, thus T can be considered a set of fine-grain tags.

$$\begin{aligned}p(T_1^n w_1^n) &= \prod_{i=1}^{n} p(T_i w_i|T_1^{i-1} w_1^{i-1}) \\ &\approx \prod_{i=1}^{n} \sum_{\substack{t_i \in T_i \\ t_{i-1} \in T_{i-1} \\ t_{i-2} \in T_{i-2}}} p(t_i w_i|t_{i-2}^{i-1} w_{i-2}^{i-1})\end{aligned}$$

We refer the reader to [17] for the derivation of this approximation.

System	Gold data	Additional data	Accuracy
Parser	1M	5M	97.24
MaxEnt	1M	0	97.24
Perceptron	1M	200M	97.44
CRF	1M	1000M	97.40
HMM	0	170M	97.15
LM	0	170M	97.21

Table 1: Tagging accuracy

3. Experiments

To evaluate the tagging accuracy of our model we chose the Wall Street Journal Penn Treebank since it is the most widely used corpus for this task. While this is a newswire corpus we believe it is reasonable to assume that if our model performs well relative to the state-of-the-art taggers on this task, it will also port to other genres, including broadcast news and conversational speech (with an appropriately trained parser to generate tags). We used WSJ sections 22-24 for testing, both for tagging and perplexity experiments.

To train our model and the HMM tagger [6], we parsed 40M words from BLLIP WSJ and 130M words from NYT section of the English Gigaword corpus. We used an extension of the Berkeley parser [18] trained on WSJ sections 2-21 with 210,000 sentences from BLLIP WSJ for self-training. The F-score of the parser on section 23 is 91.65.

The tagging accuracies of the joint LM and other systems are presented in Table 1. In the first column, *Parser* refers to the Berkeley parser used to generate training data for LM and HMM. *MaxEnt* is described in [1], *Perceptron* in [4][9], *CRF* in [3], and *HMM* in [6]. *LM* refers to this work; note that *head* and *parent* both have the same score. The second and the third columns show the amounts of human labeled and automatically labeled (or unlabeled in the case of CRF) data the models used. Note that while LM does not outperform the state-of-the-art bidirectional algorithms, it does do significantly better than the left-to-right HMM ($p < 0.0001$ in sign test), and does not differ significantly from the parser used to label the training data.

In Figure 3, we present a detailed breakdown of the performance of the joint LM using various tags and trained on different amounts of data. When trained on 40M words, sparsity prevents the systems from exhibiting significantly different performances, including the HMM tagger. With more data, mod-

[6] We use this distribution rather than uniform joint distribution $\frac{1}{|V||T|}$ because we do not want to allow word-tag pairs that have never been observed. The idea is similar to [6].

[7] To avoid a large number of zeros due to the product, we set a minimum for λ to be 10^{-7}.

[8] The lower order model is constructed by the same algorithm, although with smaller context. Note that the lower order model can back off on words or tags, or both. In the experiments in this paper, we backoff both on words and tags, i.e., $p(w_i t_i|w_{i-2}^{i-1} t_{i-2}^{i-1})$ backs off to $p(w_i t_i|w_{i-1} t_{i-1})$, which in turn backs off to the unigram $p(w_i t_i)$.

[9] Note that *Perceptron* used higher quality text for training, labeled by a combination of taggers with resulting accuracy of 97.44.

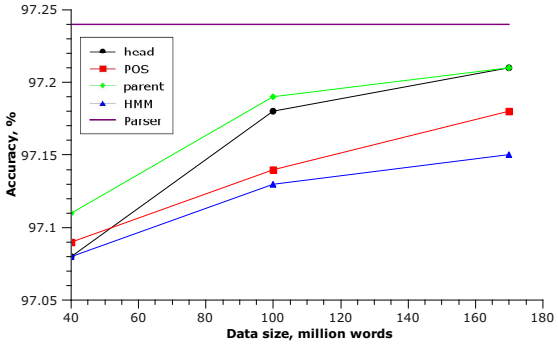

Figure 3: Tagging accuracy: POS vs. fine-grain

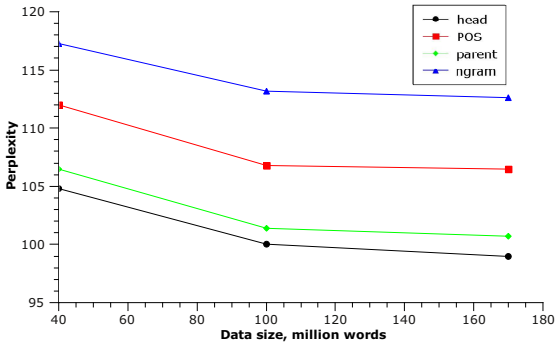

Figure 4: Perplexity: POS vs. fine-grain

els with fine-grain tags outperform the joint POS model significantly, although they begin to saturate as they approach performance of the parser used to generate their training data. The perplexity study in Figure 4 also shows the importance of fine-grain tags: both fine-grain models have 6-7% lower perplexity than the POS model and 12-13% lower than the standard trigram model[10]. For the evaluation of this model in the ASR rescoring task, we refer the reader to [15].

4. Conclusions and Future Work

In this paper, we presented a joint language model designed for large tagsets and large amounts of training data[11]. While there are other joint LMs with syntactic tags, this work is the first to evaluate the accuracy of tag prediction. We consider this an important contribution of this paper. Although our LM falls short of the best tagging algorithms (not surprisingly, since they have the advantage of accessing words in the future), it does outperform the HMM tagger, and, unlike bi-directional taggers, the LM can be used in ASR applications where strict left-to-right decoding is required.

We also established that models with fine-grain tags outperform the POS model, both in tagging accuracy and perplexity. This is a novel contribution since none of the previously reported joint models compared their performance on different tagsets. We would like to underscore that it is the flexibility of our model that allowed us to employ fine-grain tagsets and other features, such as suffixes, that proved important for predicting tags accurately.

The tagging accuracy of the parser used to generate training data limits the performance of our model, we believe the issue can be alleviated by mixing output of different parsers. We leave this for future work. We also intend to perform similar experiments in Chinese and continue to improve the scalability of the model to accommodate Gigaword-scale data.

[10] We used interpolated modified KN-smoothing.

[11] While 170M words is smaller than the volumes used for training ngram models, it is much larger than other joint models typically use.

5. Acknowledgments

This work was supported by NSF IIS-0703859 and DARPA HR0011-06-C-0023. Any opinions, findings and/or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

6. References

[1] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL*, 2003.

[2] Libin Shen, Giorgio Satta, and Aravind K. Joshi. Guided learning for bidirectional sequence classification. In *Proceedings of the ACL*, 2007.

[3] Jun Suzuki and Hideki Isozaki. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *Proceedings of the ACL-HLT*, June 2008.

[4] Drahomíra Spoustová, Jan Hajič, Jan Raab, and Miroslav Spousta. Semi-supervised training for the averaged perceptron POS tagger. In *Proceedings of the EACL*, March 2009.

[5] P. Heeman. POS tagging versus classes in language modeling. Technical report, 1998.

[6] Scott M. Thede and Mary P. Harper. A second-order hidden Markov model for part-of-speech tagging. In *Proceedings of the ACL*, 1999.

[7] Aravind K. Joshi and B. Srinivas. Disambiguation of super parts of speech (or Supertags): Almost parsing. In *Proceedings of the COLING*, 1994.

[8] Ciprian Chelba and Frederick Jelinek. Structured language modeling for speech recognition. *CoRR*, 2000.

[9] Wen Wang, Mary P. Harper, and Andreas Stolcke. The robustness of an almost-parsing language model given errorful training data. In *Proceedings of the IEEE ICASSP*, 2003.

[10] Eugene Charniak. Immediate-head parsing for language models. In *Proceedings of the ACL*, 2001.

[11] L.R. Bahl, P.F. Brown, P.V. de Souza, and R.L. Mercer. A tree-based statistical language model for natural language speech recognition. *IEEE Transactions on ASSP*, 1989.

[12] David M. Magerman. *Natural language parsing as statistical pattern recognition*. PhD thesis, Stanford, CA, USA, 1994.

[13] Jeff Bilmes and Katrin Kirchhoff. Factored language models and generalized parallel backoff. In *Proceedings of HLT/NACCL*, 2003.

[14] Imed Zitouni. Backoff hierarchical class n-gram language models: Effectiveness to model unseen events in speech recognition. *Computer Speech & Language*, 21(1):88–104, 2007.

[15] Denis Filimonov and Mary Harper. A joint language model with fine-grain syntactic tags. In *Proceedings of the EMNLP*, 2009.

[16] Sven Martin, Jorg Liermann, and Hermann Ney. Algorithms for bigram and trigram word clustering. In *Speech Communication*, pages 1253–1256, 1998.

[17] Slav Petrov, Adam Pauls, and Dan Klein. Learning structured models for phone recognition. In *Proceedings of the EMNLP-CoNLL*, 2007.

[18] Zhongqiang Huang and Mary Harper. Self-Training PCFG grammars with latent annotations across languages. In *Proceedings of the EMNLP*, 2009.