"Almost Parsing" Technique for Language Modeling*

B. Srinivas

Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
srini@linc.cis.upenn.edu

ABSTRACT

In this paper we present an approach that incorporates structural information into language models without really parsing the utterance. This approach brings together the advantages of a n-gram language model – speed, robustness and the ability to integrate with the speech recognizer with the need to model syntactic constraints, under a uniform representation. We also show that our approach produces better language models than language models based on part-of-speech tags.

1. Introduction

N-gram language models have proved to be very successful in the language modeling task. They are fast, robust and provide state-of-the-art performance that is yet to be surpassed by more sophisticated models. Further, n-gram language models can be seen as weighted finite state automata which can be tightly integrated within speech recognition systems [10]. However, these models fail to capture even relatively local dependencies that exist beyond the order of the model. We expect that the performance of a language model could be improved if these dependencies can be exploited. However, extending the order of the model to accommodate these dependencies is not practical since the number of parameters of the model is exponential in the order of the model, reliable estimation of which needs enormous amounts of training material.

In order to overcome the limitation of the n-gram language models and to exploit syntactic knowledge, many researchers have proposed structure-based language models [15, 7]. Structure-based language models employ grammar formalisms richer than weighted finite-state grammars such as Probabilistic LR grammars, Stochastic Context-Free Grammars (SCFG), Probabilistic Link Grammars (PLG) [8] and Stochastic Lexicalized Tree-Adjoining Grammars (SLTAGs) [11, 12]. Formalisms such as PLGs and SLTAGs are more readily applicable for language modeling than SCFGs due to the fact that these grammars encode lexical dependencies directly. Although all these formalisms can encode arbitrarily long-range dependencies, tightly integrating these models with a speech recognizer is a problem since most parsers of these formalisms only accept and rescore N-best sentence lists. A recent paper [5] attempts at a tight integration of syntactic constraints provided by a domain-specific SCFG in order to work with lattice of word hypothesis. However, the integration is computationally expensive and the word lattice pruning is sub-optimal. Also, most often the utterances in a spoken language system are ungrammatical and may not yield a full parse spanning the complete utterance.

In this paper, we present an alternate technique of integrating structural information into language models without really parsing the utterance. It brings together the advantages of a n-gram language model – speed, robustness and the ability to closely integrate with the speech recognizer with the need to model syntactic constraints. Our approach relies on Lexicalized Tree-Adjoining Grammars whose primitive structures, supertags, incorporate both lexical dependency and syntactic and semantic constraints in a uniform representation. The paper is laid out as follows. In Section 2 we discuss the representation of supertags with an example. We present the details of the language model in Section 3 and in Section 3.1 present the performance results of this model. We discuss some issues for the future in Section 4.

2. Supertags

Supertags are the elementary structures of Lexicalized Tree Adjoining Grammars (LTAGs) [3, 13]. They are of two kinds: (a) Initial supertags, representing language constructs that contain no recursion such as simple noun phrases and, (b) Auxiliary supertags representing recursive constructs of the language such as modifiers and adjuncts. Supertags are combined by the operations of substitution and adjunction to yield a parse for the sentence. The process of combining the supertags to yield a parse is represented by the *derivation tree*. The derivation tree can also be interpreted as a *dependency tree* of the sentence.

*This work is partially supported by NSF grant NSF-STC SBR 8920230, ARPA grant N00014-94 and ARO grant DAAH04-94-G0426. We thank Aravind Joshi, Stephen Isard and R. Chandrasekar for providing valuable comments.

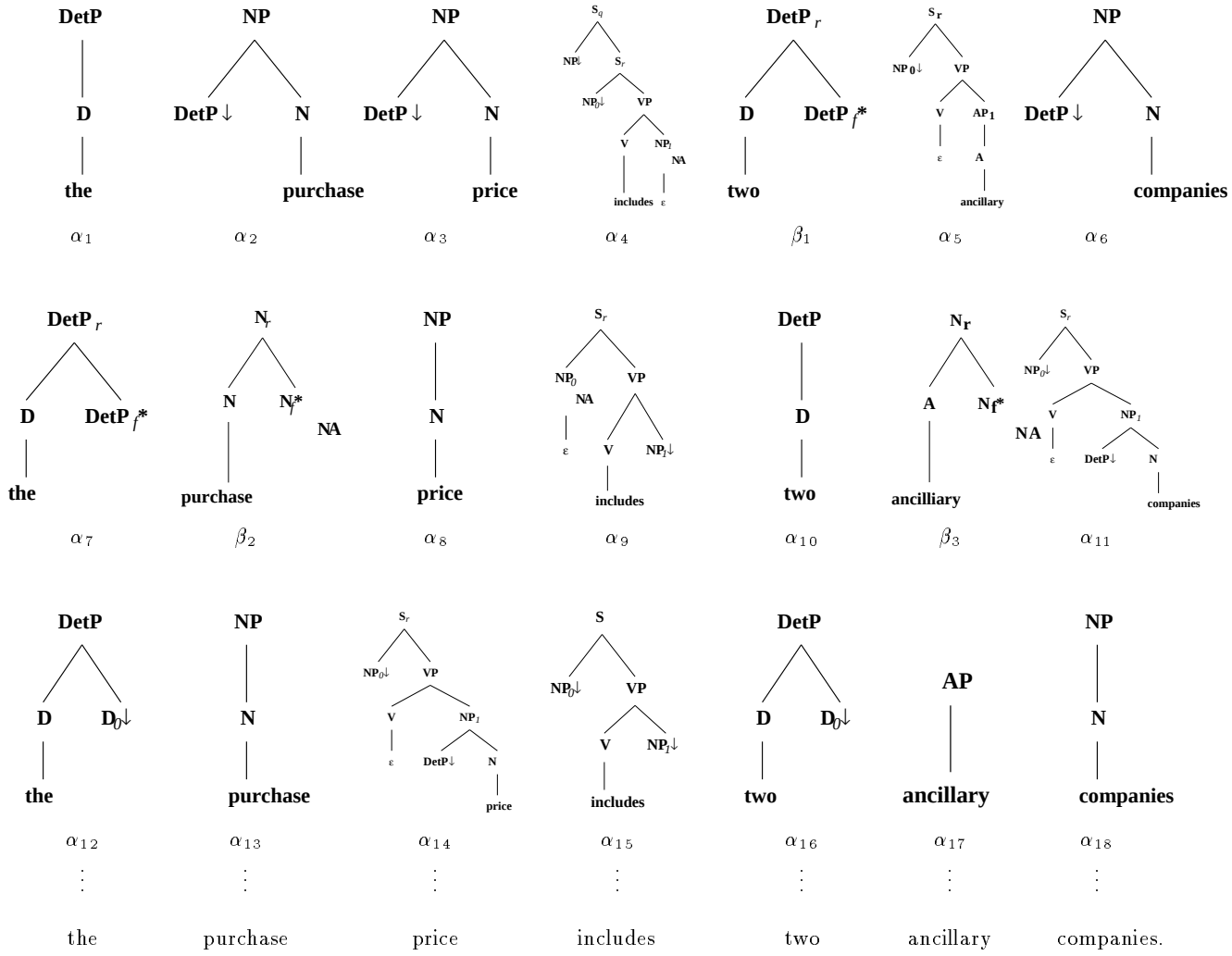

Figure 1: A sample set of supertags with each word of the sentence *the purchase price includes two ancillary companies.* For example, the three supertags shown for the word *includes* represent the supertag for object extraction (α_4), the supertag for imperative construction (α_9) and the supertag for the canonical indicative construction (α_{15}), in that order.

Each supertag is lexicalized, i.e., associated with at least one lexical item – the anchor. This provides a direct method of encoding lexical sensitivity in the grammar which is not possible in the case of Context-Free Grammars (CFGs) since CFGs are not lexicalized. Further, all the arguments of the anchor of a supertag are localized within the same supertag which allows the anchor to impose syntactic and semantic (predicate-argument) constraints directly on its arguments.

As a result of localization of arguments within a supertag, a word is typically associated with one supertag for each syntactic configuration the word may appear in, for example, indicative, imperative, relative clause on each of its arguments and so on. Figure 1 shows a few supertags associated with each word in the sentence *the purchase price includes two ancillary companies.* Furthermore, each supertag is associated with a number of attribute-value matrices, the values of which may be different even if two supertags are structurally identical. Thus supertags can be seen as providing a much more refined set of classes than do part-of-speech tags and hence we expect supertag-based language models to be better than part-of-speech based language models.

The task of a parser for the LTAG formalism is two-fold: (a) disambiguate the supertags associated with each word so as to assign (ideally) one supertag to each word, and (b) link the assigned supertags. Identifying the appropriate supertag for each word given the context of the sentence results in an *almost parse.* We have discussed two models for disambiguating supertags in [4] – an n-gram based model and a dependency-based model. The n-gram based model is appealing for speech understanding systems since it can be seen as a weighted finite-state transducer [10], which can be directly integrated into a speech recognizer and the syntactic

constraints provided by the *almost parse* can be utilized in the construction of the word lattice.

3. Language Modeling using Supertags

In this section, we discuss a supertag-based n-gram language model that is similar to a class-based n-gram language model except that the classes are defined by the supertags. A related work that employs part-of-speech categories as classes is presented in [14]. Since the supertags encode both lexical dependencies and syntactic and semantic constraints in a uniform representation, supertag-based classes are more fine-grained than part-of-speech based classes.

In a class-based language model, the probability of a n word sequence W $(= w_1, w_2 \ldots, w_n)$ is given by equation (1) which is the sum of the probabilities of all possible class sequences C $(= c_1, c_2 \ldots, c_n)$ that can be assigned to the n word sequence.

$$P(W) = \sum_{C \in C^k} P(W|C) * P(C) \quad (1)$$

where C^k is the set of all possible class sequences C that can be assigned to W.

To compute this using only local information, assume that the probability of a word depends only on its class:

$$P(w_1, w_2, \ldots, w_n|c_1, c_2, \ldots, c_n) \approx \prod_{i=1}^{n} P(w_i|c_i)$$

and also use an n-gram (trigram, in this case) approximation for the probability of the class sequence.

$$P(c_1, c_2, \ldots, c_n) \approx \prod_{i=1}^{n} P(c_i|c_{i-2}, c_{i-1})$$

Thus the class-based n-gram language model is given by

$$P(w_1, w_2 \ldots, w_n) = \sum_{C^k} \prod_{i=1}^{n} P(w_i|c_i) * P(c_i|c_{i-2}, c_{i-1})$$

The term $P(c_i|c_{i-2}, c_{i-1})$ is known as the *contextual probability* since it indicates the size of the context used in the model and the term $P(w_i|c_i)$ is called as the *word emit probability* since it is the probability of emitting the word w_i given the class c_i. These probabilities are estimated using a corpus where each word is tagged with its correct supertag. The contextual probabilities were estimated using Good-Turing discounting technique [2] combined with Katz's back-off model [6] given by:

$$\begin{aligned} p(c_3|c_1, c_2) &= p(c_3|c_1, c_2) \; if \; p(c_3|c_1, c_2) > 0 \\ &= \alpha(c_1, c_2) * p(c_3|c_2) \;\; if \; p(c_2|c_1) > 0 \\ &= p(c_3|c_2) \; otherwise \end{aligned}$$

$$\begin{aligned} p(c_2|c_1) &= p(c_1, c_2) \;\; if \; p(c_2|c_1) > 0 \\ &= \beta(c_1) * p_1(c_2) \; otherwise \end{aligned}$$

where $\alpha(c_i, c_j)$ and $\beta(c_k)$ are constants to ensure that the probabilities sum to one.

The word emit probability for the {word,class} pairs that appear in the training corpus is computed using the relative frequency estimates:

$$P(w_i|c_i) = \frac{N(w_i, c_i)}{N(c_i)}$$

The counts for the {word,class} pairs for the words that do not appear in the corpus is estimated using the leaving-one-out technique [14, 9]. A token "UNK" is associated with each class and its count N_{UNK} is estimated by:

$$P(UNK|c_j) = \frac{N_1(c_j)}{N(c_j) + \eta}$$

$$N_{UNK}(c_j) = \frac{P(UNK|c_j) * N(c_j)}{1 - P(UNK|c_j)}$$

where $N_1(c_j)$ is the number of words that are associated with the class c_j that appear in the corpus exactly once. $N(c_j)$ is the frequency of the class c_j and $N_{UNK}(c_j)$ is the estimated count of UNK in c_j. The constant η is introduced so as to ensure that the probability is not greater than one, especially for classes that are sparsely represented in the corpus.

3.1. Performance Evaluation

The performance of a language model is ideally measured in terms of its ability to decrease the word error rate of a recognizer. However, to avoid this expensive metric, an alternate measure of performance, *perplexity (PP)* [1] is used:

$$PP = 2^{(-1/n) * log_2(Pr(W))} \quad (2)$$

Perplexity, roughly speaking, measures the average size of the set from which the next word is chosen from. The smaller the perplexity measure the better the language model at predicting the next word. The perplexity measure for a language model depends on the domain of discourse.

In this section, we present perplexity results for three models on the the Wall Street Journal corpus: a word-based, part-of-speech based and a supertag-based trigram language model. A set of 200,000 words from the Wall Street Journal was randomly split into a set of 180,000 words for training and 20,000 words for the test corpus. The corpus was tagged with the correct part-of-speech tags[1] and supertags to train the part-of-speech based language model and supertagger based language model respectively. Then the performance of the three models was measured on the test corpus. The results are shown in Table 1.

[1] The UPenn treebank tagset with 40 tags was used for this purpose

Model	Perplexity	Change from trigram word perplexity
Trigram word model	163	
Part-of-speech based trigram model	203	+24.5%
Supertag based trigram model	101	−38%

Table 1: Word perplexities for the Wall Street Journal Corpus

It is interesting to note that the performance of the supertag model is better than that of the word model. This may be due to the fact that the training material used is fairly small to reliably estimate the parameters of the word model. However, that is exactly the strength of class based models; they have far fewer parameters compared to a word based model and hence require far less training material to reliably estimate those parameters. A second observation is that the performance of the supertag model is better than the part-of-speech model which suggests that the supertags define a better level of contextual generalization and provide more fine-grained classes than part-of-speech tags. Table 2 presents class perplexity results for the part-of-speech tag model and the supertag model. Class perplexity measures the average size of the set from which the next class is chosen from, given the previous history of the sentence. As can be seen, the perplexity for supertags is much higher than that for part-of-speech tags which clearly illustrates the increase in local ambiguity for supertags.

Model	Perplexity
Part-of-speech	3
Supertag	8

Table 2: Class perplexities for the Wall Street Journal Corpus

4. Discussion and Future Work

In this paper we have presented an approach that integrates syntactic, semantic and lexical dependencies in the class-based n-gram modeling framework, using supertags, without compromising the features of n-gram models – speed, robustness and the ability to tightly integrate with a recognizer. We have also shown that it serves as a better language model when compared with part-of-speech based language model. Further, we have shown that it is a better model than the word-based model when the training material is relatively small.

We propose to extend this work to incorporate more long-distance constraints by exploiting the distinction between recursive and non-recursive supertags. Further, the supertags can be organized in a hierarchy which could be utilized for more sophisticated back-off models than the one used in this paper.

5. REFERENCES

1. L.R. Bahl, J.K. Baker, F. Jelinek, and R.L. Mercer. Perplexity - a measure of the difficulty of speech recognition tasks. *Program of the 94th Meeting of the Acoustical Society of America J. Acoust. Soc. Am.*, 62, 1977.
2. I.J. Good. The population frequencies of species and the estimation of population parameters. *Biometrika 40 (3 and 4)*, 1953.
3. Aravind K. Joshi, L. Levy, and M. Takahashi. Tree Adjunct Grammars. *Journal of Computer and System Sciences*, 1975.
4. Aravind K. Joshi and B. Srinivas. Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing. In *Proceedings of the* 17^{th} *International Conference on Computational Linguistics (COLING '94)*, Kyoto, Japan, August 1994.
5. D. Jurafsky, Chuck Wooters, Jonathan Segal, Andreas Stolcke, Eric Fosler, Gary Tajchman, and Nelson Morgan. Using a Stochastic CFG as a Language Model for Speech Recognition. In *Proceedings, IEEE ICASSP*, Detroit, Michigan, 1995.
6. Salva M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, speech and SignalProcessing*, 35(3):400–401, 1987.
7. Kita Kenji and Wayne Ward. Incorporating LR Parsing into SPHINX. In *Proceedings, IEEE ICASSP*, 1991.
8. John Lafferty, Daniel Sleator, and Davy Temperley. Grammatical Trigrams: A Probabilistic Model of Link Grammar. Technical Report CMU-CS-92-181, School of Computer Science, Carnegie Mellon University, 1992.
9. Herman Ney, Ute Essen, and Reinhard Kneser. On the estimation of 'small' probabilities by leaving-one-out. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2), 1995.
10. Fernando C.N. Pereira and Michael D. Riley. Speech recognition by composition of weighted finite automata. In *Proceedings of the ARPA Workshop on Human Language Technology Workshop*, March 1994.
11. Philip Resnik. Probabilistic tree-adjoining grammar as a framework for statistical natural language processing. In *Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING '92)*, Nantes, France, July 1992.
12. Yves Schabes. Stochastic lexicalized tree-adjoining grammars. In *Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING '92)*, Nantes, France, July 1992.
13. Yves Schabes, Anne Abeillé, and Aravind K. Joshi. Parsing strategies with 'lexicalized' grammars: Application to Tree Adjoining Grammars. In *Proceedings of the* 12^{th} *International Conference on Computational Linguistics (COLING'88)*, Budapest, Hungary, August 1988.
14. T.R. Niesler and P.C. Woodland. A variable-length category-based n-gram language model. In *Proceedings, IEEE ICASSP*, 1996.
15. Victor Zue, James Glass, David Godine, Hong Leung, Michael Phillips, Joseph Polifroni, and Stephanie Seneff. Integration of Speech and Natural Language Processing in MIT Voyager System. In *Proceedings, IEEE ICASSP*, 1991.