USING SEMANTIC CLASS INFORMATION FOR RAPID DEVELOPMENT OF LANGUAGE MODELS WITHIN ASR DIALOGUE SYSTEMS

Eric Fosler-Lussier and Hong-Kwang Jeff Kuo

Bell Labs, Lucent Technologies, 600 Mountain Ave., Murray Hill, NJ 07974-0636, U.S.A.
email: {fosler,kuo}@research.bell-labs.com

ABSTRACT

When dialogue system developers tackle a new domain, much effort is required; the development of different parts of the system usually proceeds independently. Yet it may be profitable to coordinate development efforts between different modules. Here, we focus our efforts on extending small amounts of language model training data by integrating semantic classes that were created for a natural language understanding module. By converting finite state parses of a training corpus into a probabilistic context free grammar and subsequently generating artificial data from the context free grammar, we can significantly reduce perplexity and ASR word error for situations with little training data. Experiments are presented using data from the ATIS and DARPA Communicator travel corpora.

1. INTRODUCTION

Building natural spoken dialogue systems for new domains has traditionally required great effort; developers usually train an automatic speech recognition (ASR) engine, natural language components, and a speech generation system for the specific domain. One would prefer automatic (or semi-automatic) generation of dialogue system components for database-driven tasks; this requires the incorporation of domain knowledge extracted from the database into the individual system modules.

Domain knowledge is often easier to codify for some modules than for others. We often can foresee the types of knowledge required for a database application (*e.g.*, city names and dates in the DARPA Communicator travel task). Designers of a natural language understanding module can use this knowledge in building semantic parsers. On the other hand, pre-specified domain knowledge is more difficult to integrate into the statistical models of ASR systems. System designers typically require a good deal of "Wizard of Oz" data — data collected from users interacting with a person who mimics the operations of a computerized system — in order to bootstrap an automatic dialogue system. The ASR language model (LM) usually requires much more domain adaptation (in terms of number of sentences) than acoustic models since there are many fewer observations (*cf.* number of words vs. number of acoustic frames). The research we present here uses domain knowledge from the semantic parser in order to reduce the amount of Wizard of Oz data needed for LM training.

Class-based grammars have been used to reduce the perplexity of the language model by introducing categories of words and phrases; the LM contexts for the entire class are collected to smooth out estimates of the n-gram probabilities across the class. A probabilistic context free grammar (PCFG) extends this notion by allowing hierarchical clustering of classes. Jurafsky *et al.* [2] showed that, given a PCFG, one could generate a pseudo-corpus that could be used to smooth the original corpus for the training of a bigram grammar. Such a system had the advantage of connecting the expected input of the parser to the LM of the ASR system. However, producing a complete top-down grammar with good coverage for a domain is a laborious procedure.

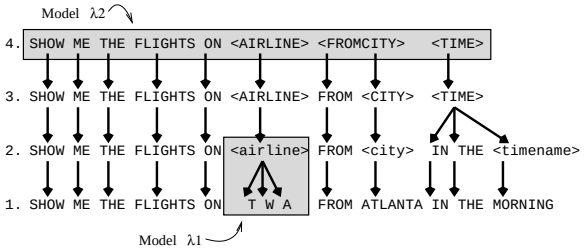


Fig. 1. Example best-path parse from the finite state parser, starting from the input on line 1 to the completed parse on line 4. Highlighted are the two domains of the statistical models: vertical rule probabilities represented by model λ_1, and horizontal n-gram probabilities represented by model λ_2.

Semantic island parsers, on the other hand, can be constructed relatively quickly since the only needed rules are ones that parse substrings of input into relevant semantic classes. In this work, we extend the finite state island parser of Potamianos and Kuo [4] to generate new smoothing sentences for the construction of n-gram grammars. A discussion of their parser follows in the next section. To augment the pre-specified semantic classes provided by the understanding component of the system, we also investigate the automatic induction of syntactic patterns [3, 6, 1] within a system for generating new LMs. In these initial experiments, we follow on the work of Potamianos and Kuo using data from ATIS travel domain [5], and then apply the models developed to the DARPA Communicator travel task.

2. FINITE STATE PARSER

The recursive finite state parser [4] uses a set of parsing rules encoded as a finite state transducer (FST) to iteratively rewrite strings from the input (Figure 1, line 1) to a maximal parse (line 4). The FST searches for potential substrings (islands) that match rule patterns, converting the pattern into its parent node. For example, the FST contained the pattern <timename> → morning, licensing the conversion of morning to <timename> in line 2. In a subsequent iteration, the pattern is further reduced by the rule <TIME> → in the <timename>.

To decide between ambiguous parse structures, the probability

of a parse structure T is determined by two models, $P(T|\lambda_1)$ and $P(T|\lambda_2)$. Model λ_1 is the PCFG probability model: the probability of a parse tree headed by parent node P is the joint probability of the parent node P and all recursive expansions of that node down to the terminal nodes. In Figure 1, for example, the probability of the tree headed by the <AIRLINE> node in the top line is computed as follows:[1]

$$\begin{aligned} P(T|\lambda_1) \quad = \quad & P(\texttt{<AIRLINE>}) * \\ & P(\texttt{<AIRLINE>} \rightarrow \texttt{<AIRLINE>}) * \\ & P(\texttt{<AIRLINE>} \rightarrow \texttt{<airline>}) * \\ & P(\texttt{<airline>} \rightarrow \texttt{T W A}) \end{aligned}$$

Rules are assumed to fire independently of each other; however, this assumption likely does not hold in practice. To compensate for this, a second model, λ_2, computes the probability of each line of the parse, estimating the joint probability of the words and semantic classes on the line with an n-gram grammar. This assumes an independence between different levels of the parse. The two models are integrated by interpolating with exponential weights:

$$P(T|\lambda) = P(T|\lambda_1)^{\gamma_1} P(T|\lambda_2)^{\gamma_2}$$

This system is advantageous for rapid development of dialogue systems, since one needs to encode only the relevant semantic attributes by hand. While a complete PCFG could be used as a grammar within the system, a full-blown syntactic grammar of natural language is unnecessary as local syntactic knowledge is encoded via n-gram probabilities.

3. SENTENCE GENERATION

In contrast to the bottom-up parsing approach described above, generating sentences from a PCFG typically requires a top-down approach. The generation algorithm takes a random walk through the set of all possible parse trees:

$W =$ <START>
while W contains nonterminals
  foreach nonterminal $w_i \in W$
    choose random rule $w_i \rightarrow w_{i1} w_{i2} \ldots$
    replace w_i with $w_{i1} w_{i2} \ldots$ in W

The rules that fire are chosen randomly according to the probability distribution $P(w_i \rightarrow w_{i1} w_{i2} \ldots)$.

3.1. Converting bottom-up parses into PCFGs

The bottom-up parser in our system need not produce complete parses (*i.e.*, it will allow grammars that do not reduce the input to a single start symbol). In order to use the PCFG generation algorithm, we must produce a probalistic model that connects a single <START> symbol to the maximal bottom-up parse. We have experimented with two models for the probability of start rules.

Method 1 (explicit start rules): for every maximal parse in our training corpus with symbols $w_1 w_2 \ldots w_n$, suggest a new rule <START> $\rightarrow w_1 w_2 \ldots w_n$. Figure 2 shows a new rule being suggested for the example in Figure 1. The probability of the start rule is determined empirically from the corpus.

[1]The probability $P(\texttt{<AIRLINE>} \rightarrow \texttt{<AIRLINE>})$ may seem strange for those used to a PCFG framework, but parsing with our finite state grammar incorporates a "self-rule" probability, since each word from line $n-1$ must be represented on line n either by itself, or by a parent found by a reduction pattern.

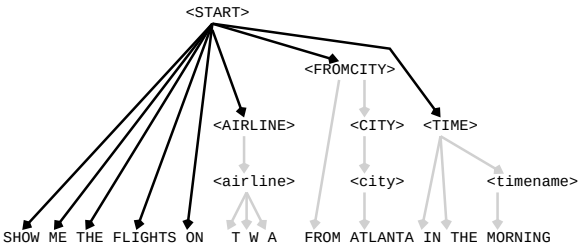


Fig. 2. Start rule induction, method 1: postulate a new start rule connecting all words in the maximal parse (shown on the example from Figure 1).

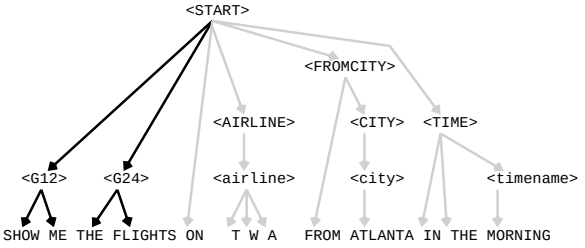


Fig. 3. Introducing syntactic generalizations into parse trees

Method 2 (semantic n-grams): instead of calculating explicit rule probabilities, we can compute an n-gram grammar on the maximal parses of the entire training corpus (similar to model λ_2 in the Section 2). To generate a new sentence, we generate a random sequence from this semantic n-gram grammar, and then recursively expand nonterminals as in the PCFG algorithm. The novel maximal parse sequences generated by this method are usually somewhat aberrant at the global level (for instance, two destination cities can be generated), but produce relatively good local word collocations for n-gram statistics.

Perplexity experiments on the ATIS corpus show that generation from explicit rules and semantic n-grams is roughly equivalent. For ease of implementation, we have conducted the experiments described here with explicit start rules (Method 1), but the method of start rule generation is an important area for future research.

4. GENERALIZATION

There may exist additional structure in the data beyond the predefined classes that should be exploited by the sentence generator. For example, in ATIS `show me` and `tell me` have the same *syntactic* properties, as well as `the flights` and `a flight`. Our understanding module does not interpret these structures semantically (otherwise they would be classified by the semantic grammar), but by introducing new rules we can syntactically generalize the data; for example, in Figure 3, `show me` has been generalized into class <G12>. The PCFG rule for <G12> might have productions including the phrases `show me`, `tell me about`, or `list`.

There has been substantial work in the field on learning phrases and phrase rules; in this paper, we follow the work of McCandless and Glass [3] as well as Siu and Meng [6] in determining syntactic classes automatically from data. The algorithm, described below,

has two parts.

4.1. Chunking into phrases

To find phrases within our training corpus, we use an extended version of the mutual information criterion for phrase selection. We examine all frequent n-grams[2] to find the mutual information between a cluster and any of its subclusters:

$$MI(w_1 \dots w_n) = P(w_1 \dots w_n) \log \frac{P(w_1 \dots w_n)}{\max_{m \in 1,\dots,n-1} P(w_1 \dots w_m) P(w_{m+1} \dots w_n)}$$

This criterion allows larger clusters to form while still discriminating against them in favor of smaller clusters; if a smaller cluster is favored in an initial pass, larger clusters can be formed later by clustering hierarchically. Clustering continues until a predetermined minimum mutual information threshold is reached.

4.2. Generalizing

In the generalization process, we group together words or phrases that share similar syntactic contexts. We utilize the Kullback-Leibler (KL) distance between bigram contexts as a measure of the similarity of two words. We train two bigram probability models, P_l and P_r, corresponding to the probability of a word given its *left* or *right* context, respectively. To obtain a symmetric measure, we estimate the distance between two words w_1 and w_2 as:

$$Dist(w_1, w_2) = D_l(w_1||w_2) + D_l(w_2||w_1) + D_r(w_1||w_2) + D_r(w_2||w_1)$$

with KL distances D_l and D_r defined over the vocabulary V as

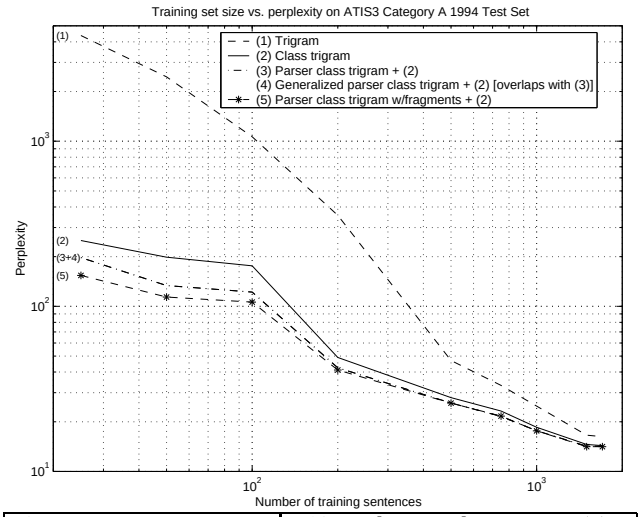
$$D_x(w_1||w_2) = \sum_{v \in V} P_x(v|w_1) \log \frac{P_x(v|w_1)}{P_x(v|w_2)}.$$

We search over all pairs of words/phrases that occur sufficiently frequently in the corpus and find the pair (w_1, w_2) that minimizes the distance measure. We then add two new generalization rules, $<G_i> \rightarrow w_1$ and $<G_i> \rightarrow w_2$, to the corpus. The optimal number of generalized clusters formed by iterating this process is determined through experimentation.

5. EXPERIMENTAL DESIGN

For our first set of experiments, we chose increasing amounts of data from the ATIS3 training set (from 25 to the full 1703 sentences) as our LM training set. The model selected as a "straw man" was a **trigram** LM trained on the first k sentences of the training set (with appropriate frequency cutoffs). It is clear, however, that training an n-gram on the data sets with few sentences would perform quite poorly, so a second baseline was developed: a **class-based trigram**, where the classes are determined by the semantic word classes (represented in our parse trees with lowercase letters). This is equivalent to training an n-gram on the second lines of the finite-state parse of the corpus (*cf.* line 2 of Figure 1). The best baseline performance was obtained by linearly interpolating the trigram and class trigram models, optimizing the weights based on the perplexity of a held-out development set.

In the **parser class** model, we parsed the training corpus of k sentences with the predefined semantic classes[3] and calculated probabilities for all the rules. We then generated 15 corpora of 1000 sentences using the learned rule probabilities and trained 15 separate trigram models. A combined model was produced by linearly interpolating these 15 models, tuning interpolation parameters on the development set. This tends to work better than pooling all data together and building one model; since the data is randomly generated, interpolation can de-emphasize poorly estimated n-grams — averaging several models reduces the variance of the combined estimator. In addition, the smoothing of trigram probabilities in one model with bigram backoffs in other models makes the estimate of the probability more robust. This combined model was subsequently interpolated with the class trigram, since one should always incorporate as much real data as possible.



Grammar	% perplexity reduction over (2)				
↓ # of sentences →	25	50	100	200	all
Parser class trigram (3)	21.2	32.6	30.6	13.4	0.8
w/fragments (5)	38.5	42.6	39.7	16.1	1.4
Generalized parser cl. (4)	20.6	32.2	30.6	14.9	1.6

Fig. 4. Perplexity scores and perplexity reduction percentages over baseline class trigram model. The generalized parser class trigram model is not shown in the graph as it closely overlaps the semantic class trigram scores.

The **generalized parser class** model was trained in a similar fashion, except that the chunking and generalization algorithms described in Section 4 were run over the parsed corpus. By optimizing on the development set, we defined a maximum of 50 new classes in the generalization process.

We found that, particularly for small training set sizes, many classes were not being generated because they did not appear in the training set. The **parser class model with fragments**, is trained identically to the parser class model, except that the <START> symbol is additionally allowed to expand to any single parser fragment. In the travel domain, for example, <START> → <TIME> would be one fragment rule added to the rule corpus. Both of the above two class models were also interpolated with the baseline class trigram grammar.

6. EVALUATION

6.1. Perplexity

We evaluated the two baseline models and three experimental models on the category A sentences from the 1994 ATIS test set; Figure 4 shows the perplexity of all models except the generalized

[2] For these experiments, we use a maximum n of 5.

[3] Since we cannot assume that we have rule probabilities, the parsing was accomplished with a greedy strategy [4].

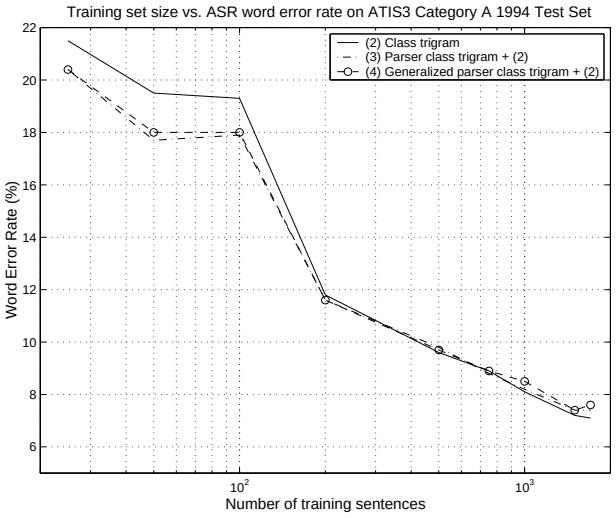


Fig. 5. ASR results for the 1994 ATIS test set with varying amounts of LM training data. Results including fragments were not available.

parser class trigram, which was indistinguishable on the graph from the ungeneralized parser class trigram model. As expected, using class information greatly decreases the perplexity over a plain trigram, especially with very few training sentences. The use of semantic class information gives an additional boost in this region, particularly when fragment rules are added to the grammar.

It is somewhat surprising that the additional machinery of the automatic generalization techniques described in Section 4 do not improve the LM over the semantic classes alone. Siu and Meng [6] note that their method is *semi-automatic*; they hand-correct the postulated CFG rules. We suspect that this method may assist in developing rules for a new domain; one could iteratively apply the entire training procedure presented here to simultaneously induce rules for the parser and ASR LM in a supervised fashion.

While it appears from Figure 4 that the models are roughly comparable when trained on the full ATIS test set, adding parser class information does improve robustness on other test sets. When the ATIS-trained models are evaluated on DARPA Communicator travel data from Bell Labs and the University of Colorado, the parser classes reduce the perplexity of non out-of-vocabulary words by 8% over the class trigram (86.8 $\rightarrow$ 80.0); adding the fragments into the grammar results in a 18% reduction in perplexity (86.8 $\rightarrow$ 71.3). This suggests that the parser class-based grammars might be a good starting point for domain adaptation.

6.2. ASR

We also evaluated the ATIS models by performing speech recognition on the 1994 test set. The acoustic models for this experiment were trained on acoustic data independent of the ATIS domain and were held constant for all language model training set sizes.

Figure 5 illustrates that the ASR word error rate parallels the perplexity results of Figure 4. For a training set size of 50–100 sentences, the parser class grammar reduces word error over the class trigram by 7–9% relative (significant at $p < 0.05$). As the number of available LM training sentences increases, however, the difference in error rates becomes indistinguishable. Including the generalization techniques did not improve ASR results over the plain parser class grammar. Results for the parser class grammar with fragments were not available as of the writing of this paper.

7. SUMMARY

The techniques proposed in this paper attempt to reduce the amount of required Wizard of Oz data for building language models. Using the semantic classes defined for the understanding module of the dialogue system can improve the performance of language models trained only on a limited number of sentences. We have also found, using the DARPA Communicator data, that integrating semantic class information in a model trained on a related (but different) domain can provide a good starting point for domain adaptation.

The automatic generalization techniques found in the literature did not improve performance of the parser class model. However, we hope that we can use generalization to help build semantic classes in a new domain, and iterate this technique as new domain data becomes available. One drawback to the generalization techniques presented here is that they require a sufficient number of examples to generalize, at which point there is probably enough data to train the n-gram anyway; an important future direction for this work is to find ways to cluster rare events.

Rapid dialogue system development is a continuing focus within our lab. The parser used in this work was designed to require the encoding of a minimal amount of domain information (in the form of regular expressions) to produce a grammar for natural language understanding. We have extended this work by incorporating this encoded information into the ASR system. While it is premature to declare success in this area, the initial results presented in this study are a promising step towards quick system development.

8. ACKNOWLEDGMENTS

The authors would like to thank Alexandros Potamianos, Chin-Hui Lee, Andrew Pargellis, and Hongyan Jing for their helpful discussion of this work; Alexandros Potamianos provided the acoustic models for the ATIS ASR experiments. Thanks also to Andreas Stolcke and SRI for providing the SRI LM toolkit for use in these experiments. This work was conducted as part of the Bell Labs DARPA Communicator effort, grant #N66001-00-C-8013.

9. REFERENCES

[1] K. Arai, J. Wright, G. Riccardi, and A. Gorin. Grammar fragment acquisition using syntactic and semantic clustering. *Speech Communication*, 27:43–62, 1999.

[2] D. Jurafsky, C. Wooters, J. Segal, A. Stolcke, E. Fosler, G. Tajchman, and N. Morgan. Using a stochaistic context-free grammar as a language model for speech recognition. In *ICASSP*, Detroit, MI, 1995.

[3] M. McCandless and J. Glass. Empirical acquisition of word and phrase classes in the ATIS domain. In *Eurospeech*, volume 2, pages 981–984, Berlin, Germany, 1993.

[4] A. Potamianos and H. Kuo. Statistical recursive finite state machine parsing for speech understanding. In *ICSLP*, Bejing, China, 2000.

[5] P. Price. Evaluation of spoken language systems: The ATIS domain. In *Proc. ARPA Human Language Technology Workshop*, pages 91–95, 1990.

[6] K. Siu and H. Meng. Semi-automatic acquisition of domain-specific semantic structures. In *Eurospeech*, Budapest, Hungary, 1999.