

A Language Model Combining Trigrams and Context-Free Grammars

John Gillett and Wayne Ward
Carnegie Mellon University

ABSTRACT

We present a class trigram language model in which each class is specified by a probabilistic context-free grammar. We show how to estimate the parameters of the model, and how to smooth these estimates. Experimental perplexity and speech recognition results are presented.

1. INTRODUCTION

The two language models most commonly used in speech recognition systems are the trigram model and the Stochastic Context-Free Grammar (SCFG). The trigram model is simple and powerful, but it doesn't use longer span information or any prior human knowledge of language. It does not do a good job of estimating probabilities for rare words and unseen sequences. SCFGs can model longer-range effects and work well on limited-domain tasks of low perplexity with experienced users. However, large-vocabulary general-purpose SCFGs have not worked well. It is difficult to get good rule-based coverage of spontaneous speech. The grammars are either too brittle to handle unseen data or too ambiguous to be useful. SCFGs and are smoothed differently than trigrams. Trigrams assign a probability for all possible sequences of words. SCFGs smooth across allowed rule expansions, which is much more constrained than all possible word sequences. Since their strengths are complementary, we combine the two mechanisms in a way to take advantage of the best properties of each.

We propose a class trigram model in which each class is specified by a stochastic context-free grammar. While our algorithms permit the specification of complex classes, our intent is to use only simple classes which capture straightforward concepts, and to leave the rest of the modeling problem to the trigrams. The class grammars are used to model closed form expressions specific to the domain. It is on the extraction and interaction of these phrases that the task depends. The trigram provides a stochastic model of how the phrases are embedded in word strings. This work is similar to the ideas presented in Ward & Young [5]. That system used a trigram of slot-level classes where each class was expanded by a Context-Free Grammar. The grammar circumscribed the word sequences for a class, but probabilities were assigned by a word bigram rather than a stochastic grammar. The slot-level classes used in that system were high-level tokens matching longer strings of words. In the current work, most classes are degenerate (single word) with a fairly small number of low level tokens matching fairly short strings. This strategy puts much less burden on the grammar as it does not require very complete coverage. It provides a graceful way to tailor the model to the coverage of the class grammar. The model matches the patterns when it can, but always assigns non-zero probability to any string of words. As the coverage of the grammar gets better, the class grammars can become more complex giving better constraint. This model

has some similarities to the Chronus system [2] and the HUM system [4] in that it tries to derive a statistical relation between parsed elements. The mechanism here is quite different. Those systems used Hidden Markov Models to segment the input into classes and estimate its likelihood. Our mechanism uses a combination of SCFGs and second-order Markov models.

Section 2 presents the model and an example. Section 3 describes an estimation-maximization (EM) algorithm to train the model's parameters, and section 4 describes a smoothing algorithm. Section 5 presents experimental perplexity and speech recognition results.

2. THE MODEL

We propose a class trigram model in which each class is specified by a SCFG. In this model, the probability of the next word is computed as $P(C_n | C_{n-1}, C_{n-2}) * P(W | C_n)$, the probability of class n given the two previous classes times the probability of word string W given class C_n . $P(C_n | C_{n-1}, C_{n-2})$ is given by the trigram, $P(W | C_n)$ is given by the SCFG for class C_n . The SCFGs can range from quite complex to degenerate grammars representing single words. They are intended to represent those phrases which can be modeled well by grammars: times, dates, locations, etc. They especially should represent those domain-specific quantities to be extracted. We use a training corpus and the EM algorithm to estimate model parameters.

We introduce our model, with the help of an example. The parameters of the model are class trigram probabilities $P(C | A, B)$ and grammar rule probabilities $P(X \rightarrow \square)$. X is a non-terminal and \square is a sequence of terminals and non-terminals.

Define a grammar for the [date] class as:

[date] \rightarrow [month] [day]

[month] \rightarrow "January", ... \rightarrow "December"

[day] \rightarrow [digit]
 \rightarrow [digit] [digit]

[digit] \rightarrow "0", ... \rightarrow "9"

Here [month], [day], and [digit] are auxiliary non-terminals for the [date] grammar, only [date] defines a class. Next we define a trivial class for each word in our word vocabulary ([apple] \rightarrow "apple", ... , [zebra] \rightarrow "zebra"). We assign uniform class trigram probabilities by setting $P(C | A, B) = 1/|V|$, where $|V|$ is the number of classes. We assign uniform grammar rule probabilities by setting $P(X \rightarrow \gamma) = 1/n_X$, where n_X is the number of rewrite rules for X . A parse of a sentence s

$C([///], [///], e.to.C; s)$ by
 $(1/P(s)) * P(e.to.C | [///], [///]) * e$
 The class of the graph's last vertex is also $[///]$. For each remaining vertex v , and for each edge e into v and each edge f out of v , we increase
 $C(e.from.C, v.C, f.to.C; s)$ by
 $(1/P(s)) * e * P(f.to.C | e.from.C, v.C) * f$
 and, for each rule $X \rightarrow \square$ in v 's rule sequence, we increase $C(X \rightarrow \square; s)$ by

$$\frac{1}{P(s)} \cdot \sum_{e.to=v} \left(e\alpha \cdot \sum_{f.from=v} f\beta \cdot P(f.to.C | e.from.C, v.C) \right)$$

As we process each sentence s in T in this manner, we accumulate

$$C(A, B, C) = \sum_s C(A, B, C; s)$$

and

$$C(X \Rightarrow \gamma) = \sum_s C(X \Rightarrow \gamma; s)$$

Finally, we normalize the expected counts to get our next model:

$$P(C | A, B) = \frac{C(A, B, C)}{\sum_x C(A, B, x)}$$

and

$$P(X \Rightarrow \gamma) = \frac{C(X \Rightarrow \gamma)}{\sum_x C(X \Rightarrow x)}$$

4. SMOOTHING

We use interpolated estimation [1] to smooth our model. We form smoothed class trigram probabilities $\hat{P}(C | A, B)$ by interpolating uniform, unigram, bigram, and trigram probabilities:

$$\hat{P}(C | A, B) = \lambda_0(A, B) * (1/V) + \lambda_1(A, B) * P(C) + \lambda_2(A, B) * P(C | B) + \lambda_3(A, B) * P(C | A, B),$$

where the λ s sum to 1 and depend on A and B only through the count of the bigram (A, B) and the count of the unigram B . We form smoothed rule probabilities $\hat{P}(X \rightarrow \square)$ by interpolating uniform and trained rule probabilities:

$$\hat{P}(X \rightarrow \square) = \lambda_X * P(X \rightarrow \square) + (1 - \lambda_X) * (1/n_X)$$

We choose $\lambda_i(A, B)$ and λ_X by the EM algorithm to maximize the likelihood our smoothed model assigns to a held-out training text.

5. EXPERIMENTAL RESULTS

We created language models as described for two different tasks, 1) patient medical record entry by voice and 2) an automated travel agent task. These language models were

compared to standard class based trigrams for both perplexity and recognition error rate. For comparing recognition rate, we used our model to rescore word lattices produced by CMU's Sphinx II speech decoder. The Sphinx II speech decoder normally produces a word lattice and then finds the best path through the lattice using a class trigram language model [3]. Our procedure does a Viterbi search through a graph of parses built from the word lattice to find the path for which $P(s, t)$ is a maximum.

For the medical records task, we had several hundred transcripts of dictated physical exams. From these 20,358 sentences (170,473 words), 18,326 were used for training and 2,032 for smoothing. A set of 392 new sentences was gathered for testing. Classes were created to match blood pressures, temperatures, pulse rates, and similar things. There were 30 non-trivial classes in the system tested. For comparison we also built a word trigram model using the CMU/Cambridge Tool Kit (Rosenfeld 1995), trained over all the non-test transcripts.

	Perplexity	Word Error
Baseline	27.7	12.4
Class	23.1	12.1

Results for Medical Reporting Task

The results show that the new model had a perplexity 14% lower than the baseline model, but only had a marginal effect on recognition word error rate.

The second task was an automated travel agent. The data was spontaneous interaction with a system, rather than dictation. Subjects were interacting with a speech recognition system to make travel reservations. For this task, the training set was 1400 utterances and the test set was 300. Classes were created for concepts like date, time, number, etc.

	Perplexity	Word Error
Baseline	32.3	54.1
Class	26.2	51.8

Results for Travel Agent Task

The performance of the models on the second task was very similar to the performance on the first. The new model showed a 19% reduction in perplexity over the old, but only a 4% reduction in word error rate.

6. DISCUSSION

In both tasks, the new model gave a moderate reduction in perplexity but only a marginal improvement in word error rate. While the overall word error rates of the two models were similar, many of the word strings were different, ie. they made different errors. The new model can augment a word based trigram, instead of replace it. When we interpolated the two models (equally weighted) the perplexity was not further reduced. Looking at the detailed perplexity comparisons provided some insight into the properties of the two models. The two models assigned similar probabilities in most

situations, with the word based trigram having a slightly lower perplexity. However, when the word based trigram made a really bad prediction the new model often assigned a much better probability. This is encouraging because it indicates that we can combine the models to good effect. But fixed interpolation is not the best way to combine them. We are constructing a decision tree to combine the models in a situation specific fashion.

7. REFERENCES

1. Bahl, L., Jelinek, F., and Mercer, R., "A Maximum Likelihood Approach to Continuous Speech Recognition," in *Readings in Speech Recognition*, ed. Waibel and Lee, Morgan Kaufmann, 1990, pp. 308-319.
2. Levin, E. and Pieraccini, R., "CHRONUS - The Next Generation," *Proceedings of ARPA HLT*, Jan. 1995, pp. 269-271.
3. Ravishankar, M.K., "Efficient Algorithms for Speech Recognition," Unpublished PhD Dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA May 1996.
4. Schwartz, R., Miller, S., Stallard, D. And Makhoul, J., "Language Understanding Using Hidden Understanding Models," *Proceedings ICSLP*, Oct 1996, pp. 997-1000
5. Ward, W. and Young, S.R., "Flexible Use of Semantic Constraints in Speech Recognition," *Proceedings of ICASSP93*, April 1993
6. Dempster, A., Laird, N., and Rubin, D., "Maximum Likelihood from Incomplete Data via the *EM* Algorithm," *Journal of the Royal Statistical Society* 39 (1977) B, 1-38.
7. Baker, J., Trainable Grammars for Speech Recognition, *Proceedings of the Spring Conference of the Acoustical Society of America* (Boston, MA), pp. 547-550, 1979.