A LANGUAGE MODEL COMBINING TRIGRAMS AND STOCHASTIC CONTEXT-FREE GRAMMARS

John Gillett and Wayne Ward
Carnegie Mellon University

gillett@cs.cmu.edu, whw@cs.cmu.edu

ABSTRACT

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

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 can't use long-range information or prior knowledge of language, and it estimates rare-word and unseen-sequence probabilities poorly. The SCFG can model long-range effects and works well on limited-domain tasks of low perplexity, but large-vocabulary, general-purpose SCFGs work poorly because they are either too brittle to handle unseen data or too ambiguous to be useful. SCFGs are smoothed differently than trigram models: where trigrams assign a probability for all possible sequences of words, SCFGs smooth across tightly-constrained rule expansions. Since their strengths are complementary, we combine the two models to take advantage of the best properties of each.

We propose a class trigram model [1] in which each class is specified by a stochastic context-free grammar [2]. While our algorithms permit the specification of complex classes, our intent is to use simple classes that capture straightforward concepts, and to leave the rest of the modeling problem to the trigrams. We use class grammars to model closed-form expressions specific to a task domain, such as dates or times; several current speech applications depend on the extraction and interaction of such phrases. The trigram provides a stochastic model of how these content-bearing phrases are embedded in sentences.

This work was developed from ideas presented by Ward & Young [3]. Their system used a trigram of slot-level classes in which each class was expanded by a context-free grammar. A grammar circumscribed the word sequences for a class, and a word bigram model assigned probabilities within a class. The slot-level classes in their system matched longer strings of words. In the current work, most classes are degenerate (single-word), with a small number of classes matching short strings. This puts less burden on the grammars as it does not require very complete coverage. It provides a graceful way to tailor the model to the coverage of the grammars. The model matches patterns when it can, but still assigns non-zero probability to every word string.

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 results.

2. THE MODEL

The parameters of our model are class trigram probabilities $P(C \mid A, B)$, where each of A, B, and C is a class specified by a SCFG; and grammar rule probabilities $P(X \rightarrow \gamma)$, where X is a non-terminal and γ is a sequence of terminals and non-terminals.

To score a sample sentence, we define a *[DATE]* class by the rules

[DATE] → *{month} {day}*

{month} → "january"
...
→ "december"

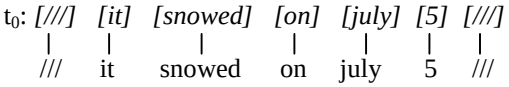
{day} → *{digit}*
→ *{digit} {digit}*

{digit} → "0"
...
→ "9"

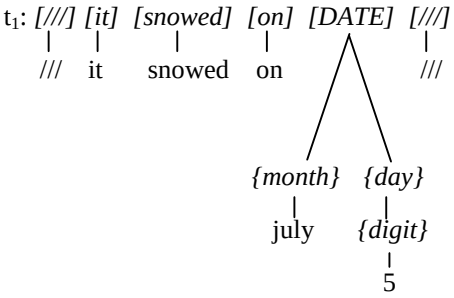
Next we define a trivial class for each word in our vocabulary: *[apple]* → "apple", ... , *[zebra]* → "zebra". We assign uniform class trigram probabilities $P(C \mid A, B) = 1/N$, where N is the number of classes, and uniform grammar rule probabilities $P(X \rightarrow \gamma) = 1/n_X$, where n_X is

the number of rewrite rules for *X*. We define a parse of a sentence *s* to be a sequence of classes and a corresponding set of grammar rules which together generate the sequence of words in *s*. We use an artificial word "///" to indicate the boundary between sentences; "///" begins an utterance with probability 1.

For the sentence *s* = "/// it snowed on july 5 ///", two parses are possible. The first consists entirely of trivial classes:



The second parse uses the *[DATE]* class:



The probability of *s* is

$$P(s) = P(s,t_0) + P(s,t_1)$$
$$= \left(\frac{1}{N}\right)^6 + \left(\frac{1}{N}\right)^5 \bullet 1 \bullet \frac{1}{12} \bullet \frac{1}{2} \bullet \frac{1}{10}$$

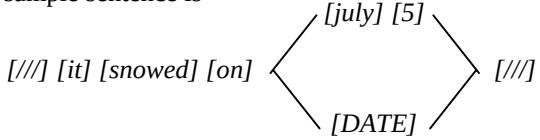
More generally, if we let $c(X \rightarrow \gamma; t, s)$ denote the number of times the rule $X \rightarrow \gamma$ is used in generating the sentence *s* via the parse *t*, and $C_1, \ldots, C_n$ denote the class sequence used in *t*, then

$$P(s) = \sum_t P(s,t)$$

where

$$P(s,t) = \prod_{i=1}^{n} P(C_i \mid C_{i-2}, C_{i-1}) \bullet \prod_{X \rightarrow \gamma} P(X \rightarrow \gamma)^{c(X \rightarrow \gamma; t, s)}$$

To calculate this sum efficiently, we represent the set of parses by a directed graph, each vertex of which represents a class generating a sub-string of *s*; and whose edges are such that the set of paths through the graph corresponds to the set of parses of *s*. The graph for our sample sentence is



(If a class can match a sub-string in more than one way, we make one vertex for each SCFG parse. This eliminates the need for an inside-outside algorithm [2], but risks inefficiency for ambiguous grammars.)

To get *P(s, t)* from our graph, we follow the path corresponding to *t* from its start to its end: we set the start vertex's probability to 1, and then at each vertex we multiply in the trigram probability of the current vertex's class given the previous two vertices' classes, and the vertex's SCFG probability. To calculate

$$P(s) = \sum_t P(s,t)$$

we sum the probabilities over all paths from the start vertex to the end vertex. A forward-backward algorithm [4] stores probabilities on the graph's edges to calculate this sum efficiently. (Only forward probabilities are required to calculate *P(s)*; backward probabilities are needed for the training described in the next section.)

3. TRAINING

We use the EM algorithm [5] to set the model parameters to (local) maximum likelihood estimates over a training text. We choose an initial parameter set, and then repeat the process of collecting expected counts over the text according to the current model, and normalizing these counts to create a new model, until the sequence of likelihoods of the text according to these successive models converges.

To facilitate collecting the required class trigram counts, let *c(A, B, C; t, s)* denote the number of times the class sequence *(A, B, C)* appears in the parse *t* of the sentence *s*; let *count(A, B, C; s)* denote the expected number of times the class sequence *(A, B, C)* appears in all parses of *s*; and let *count(A, B, C)* denote the expected number of times the class sequence *(A, B, C)* appears in all parses of all sentences *s* in the training text. Then

$$count\ (A, B, C; s) = \sum_t P(t \mid s) \bullet c(A, B, C; t, s)$$

and

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

Similarly, to facilitate collecting the required grammar rule counts, let $count(X \rightarrow \gamma; s)$ denote the expected number of times $X \rightarrow \gamma$ is used in all parses of *s*; and let $count(X \rightarrow \gamma)$ denote the expected number of times that $X \rightarrow \gamma$ is used in all parses of all sentences in the text. Then

$$count\ (X \rightarrow \gamma; s) = \sum_t P(t \mid s) \bullet c(X \rightarrow \gamma; t, s)$$

and

$$count\ (X \rightarrow \gamma) = \sum_s count\ (X \rightarrow \gamma; s)$$

To collect the expected counts for *s*, we build its graph and run the forward-backward algorithm. We set *count(A, B, C; s) = 0* and *count(X → γ; s) = 0* for all *A, B, C, X,* and γ. Then a second forward pass through the graph visits each vertex to increment these counts.

As we process each sentence *s* in the training text in this manner, we accumulate

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

and

$$count(X \rightarrow \gamma) = \sum_s count(X \rightarrow \gamma; s)$$

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

$$P'(C \mid A, B) = \frac{count(A, B, C)}{\sum_D count(A, B, D)}$$

and

$$P'(X \rightarrow \gamma) = \frac{count(X \rightarrow \gamma)}{\sum_\theta count(X \rightarrow \theta)}$$

4. SMOOTHING

We use interpolated estimation [1] to smooth our model. We form smoothed class trigram probabilities *S(C | A, B)* by interpolating uniform, unigram, bigram, and trigram probabilities:

$$\begin{aligned} S(C \mid A, B) = {} & \lambda_0(A, B) \cdot \left(\frac{1}{N}\right) \\ & + \lambda_1(A, B) \cdot P(C) \\ & + \lambda_2(A, B) \cdot P(C \mid B) \\ & + \lambda_3(A, B) \cdot P(C \mid A, B) \end{aligned}$$

where the lambdas 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 *S(X → γ)* by interpolating uniform and trained rule probabilities:

$$S(X \rightarrow \gamma) = \lambda_X \cdot (1/n_X) + (1 - \lambda_X) \cdot P(X \rightarrow \gamma)$$

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 the language model described for two different speech-recognition tasks: a medical records transcription system and an automated travel agent. In each case, we compared the perplexity and recognition error rate of our model to those of a standard word trigram model [6]. For comparing recognition rates, we rescored word lattices produced by CMU's Sphinx II speech decoder [7]. For the new model, we did a Viterbi search through a graph of parses built onto the word lattice to find that path for which $P(s, t) \cdot A(a \mid s)$ is a maximum, where $A(a \mid s)$ is the probability assigned by the acoustic model to the acoustic information *a* given the sentence *s*. We would like to do a forward search and find the path for which

$$P(s) \cdot A(a \mid s) = \left(\sum_t P(s, t)\right) \cdot A(a \mid s)$$

is a maximum, but we don't know how to do this efficiently.

For the medical transcription task, we had several hundred transcripts of dictated physical exams. Of 20,358 sentences (150,116 words), we used 18,326 for training and 2,032 for smoothing. We gathered 392 new sentences for testing. We created classes to match blood pressures ("blood pressure is 125 over 80"), temperatures, pulse rates, and similar things. We trained the word trigram model over all non-test sentences.

	Perplexity	Word Error
Standard	27.7	12.4%
New	23.1	12.1%

Figure 1: Medical transcription results

Figure 1 indicates that for the medical transcription task our model gave a 17% improvement in perplexity without significantly improving the word recognition error rate.

For the automated travel agent, our data consisted of spontaneous interaction, via a telephone call, with a reservations system. Our training set contained 1,400 utterances (8,036 words) and the test set contained 300. We created classes to match things like dates, times, and airport names. This travel agent is in an early stage of development.

	Perplexity	Word Error
Standard	32.3	45.9%
New	26.2	48.2%

Figure 2: Automatic travel agent results

Figure 2 indicates that for the travel agent task our model gave a 19% improvement in perplexity, but increased the word recognition error rate.

We are continuing to experiment with this new model, and we hope to extend it to capitalize on knowledge of the dialog state in a spoken language understanding system.

6. REFERENCES

1. Brown, P., della Pietra, V., de Souza, P., Lai, J., Mercer, R., "Class-Based n-gram Models of Natural Language," *Computational Linguistics* 18, pp. 467-479, 1992.

2. Baker, J., "Trainable Grammars for Speech Recognition," *Speech Communication: Papers Presented at the 97th Meeting of the Acoustical Society of America*, pp. 547-550, 1979.

3. Ward, W. and Young, S.R., "Flexible Use of Semantic Constraints in Speech Recognition," *Proceedings of ICASSP93 II*, pp. 49-50, 1993.

4. Baum, L., "An Inequality and Associated Maximization Technique in Statistical Estimation for Probabilistic Functions of Markov Processes," *Inequalities—III (Proceedings of the Third Symposium on Inequalities Held at the University of California, Los Angeles, 1969)*, pp. 1-8, 1972.

5. Demptser, A., Laird, N., and Rubin, D., "Maximum Likelihood from Incomplete Data via the *EM* Algorithm," *Journal of the Royal Statistical Society* 39 B, pp. 1-38, 1977.

6. Clarkson, P., and Rosenfeld, R., "Statistical Language Modeling Using the CMU—Cambridge Tool Kit," *Eurospeech '97 Proceedings*, pp. 2707-2710, 1997.

7. Ravishankar, M.K., "Efficient Algorithms for Speech Recognition," Ph.D. Dissertation CMU-CS-96-143, School of Computer Science, Carnegie Mellon University, 1996.