

# Estimation of Language Models for New Spoken Language Applications

*Sunil Issar*

School of Computer Science  
Carnegie Mellon University  
Pittsburgh, Pennsylvania 15213-3890, USA

## ABSTRACT

Spoken language interfaces can provide natural communication for many database retrieval tasks. The CMU ATIS system provides an example of accessing airline information using spoken natural language queries. However, a lot of training data is needed to develop a spoken language application. For example, we need training data to generate a language model that can be used by the recognizer to reduce the search space. In this paper, we will address some issues arising from small amount of training data available for a new spoken language application.

We are working on a spoken language interface to access information from a library catalogue. The catalogue contains around 13,000 titles, 6000 authors and 19000 subjects. There are more than 20,000 words in the dictionary. The user can seek information about books, authors, subjects, publishers, etc. For example, "I'd like to see books dealing with science fiction by Clarke." We will describe some language modelling experiments for this task.

We will briefly describe a speech interface [5] for a library catalogue. We will also review class-based language models and describe their limitations. Finally, we will present our approach to building statistical language models for new spoken language applications. This is important because a lot of training data is normally needed to generate a language model. However, it is not practical to have or collect a large corpus of data for each new spoken language application.

## 1. Introduction

Many businesses provide automated database access over the telephones. In most of these applications, the customer uses the telephone touch pad to communicate with the system. Telephone banking, credit card inquiries, flight inquiry system, and getting mutual fund quotes are some examples of these systems. However, it can be extremely tedious to seek information. It may require navigating through multiple menus, listening to a long list of options (for example, if you don't know the code for a mutual fund) and using the telephone touch pad to enter non numeric data (for example, the first three letters of the departure city). A spoken language interface can provide a more natural communication approach in these applications. Our goal is to allow the users to communicate in a

natural way, because it reduces the amount of learning by the user. The CMU ATIS [9, 3] system and other similar systems [1] demonstrate the feasibility of using spoken natural language queries for a database retrieval task. However, we still need to address several issues before we can build robust spoken language applications. For example, we need to generate a language model that can be used by the recognizer to reduce the search space. Finite-state language models may work for simple speech enabled menu commands. However, they may not be the best choice [3] for database retrieval tasks where the user can ask a large number of questions. We are conducting experiments [11] with a recognizer that uses both stochastic and finite-state language models. In this paper, we will look at the issues that arise in generating stochastic language models for a new spoken language application.

The speech recognizer is essentially a search algorithm, which searches over all possible word sequences (based on the words in the dictionary). A language model constrains this search by defining acceptable word sequences. The Sphinx-II [2, 7] system uses a trigram (or bigram) language model, which estimates the probability of a word based on the last two (one) words. It is trained from large amounts of domain specific data. For example, the language model used in our CSR dictation system, which has a vocabulary of 20,000 words, was based on 40 million words of WSJ data. Our ATIS language model is based on 24,000 sentences containing about 250,000 words.

We will first briefly describe a speech interface [5] for a library catalogue. We use the Sphinx-II [2, 7] recognizer for transcribing spoken queries. We will also review class-based language models and describe their limitations. We will next discuss our approach to building statistical language models for new spoken language applications. This is important because a lot of training data is normally needed to generate a language model. However, it is not practical to have or collect a large corpus of data for each new spoken language application. Finally, we will discuss using multiple language models for improving recognition accuracy.

## 2. Library Information System

We are working on a spoken language interface to access information from a library catalogue [5]. The user can seek information about books, authors, subjects, publishers, etc. For example, "I'd

like to see books dealing with science fiction by Clarke.” This is a collaborative project with the speech group at NIST. We are using the database [5] that was extracted from the Library of Congress catalogue. This catalogue contains around 13000 titles, 6000 authors and 19000 subjects. There are around 20,000 words in the dictionary.

The Sphinx-II decoder is used for speech recognition in this system. We use the acoustic models that were trained for the CMU CSR dictation system. However, around 5000 triphones (generated from the dictionary) were missing from the mapping table. We regenerated the mapping table using the CSR training data.

The focus of our work has been on improving the language model used in the recognition system. We started with a class based language model. However, we needed to modify the algorithms because of limitations discussed later.

### 3. Class Based Language Models

We use a class-based language model [6], when there is insufficient data to generate a language model. In this approach,

1. Words are clustered into semantic classes. For example,

- City names
- Airport codes
- Cities not in database
- Time intervals

Words can be in their own semantic class (singleton classes), but in this case we will not distinguish between a word and its class. Although words can belong to multiple classes, we will assume for simplicity that each word occurs in a unique class. It is not a good idea to cluster the articles (*A, An, The*) into the same class.

2. The training data is mapped using these classes. For example, the sentence “*I’d like to find a flight from Pittsburgh to Boston in the morning*” would be mapped to “**I’d like to find a flight from [city] to [city] in the [time\_interval]**”

3. An n-gram backoff class language model [8] for the training corpus is generated.

4. We next generate a word based language model from this class based language model. For simplicity, we will assume that we are using bigram language models. We will describe below a recursive procedure (**classbg\_to\_wdbg**) that generates the appropriate word bigrams based on a bigram  $Pr(c_2|c_1)$  in the class based language model. Then for each bigram  $Pr(y|x) = p$  in the class based language model generated above, the procedure **classbg\_to\_wdbg** ( $x, y, p$ ) will generate the corresponding word based bigrams.

**classbg\_to\_wdbg** ( $c_1, c_2, p$ )

- If  $c_1$  is a word,

– and if  $c_2$  is a word,  $Pr(c_2|c_1) = p$

– otherwise ( $c_2$  is a class)

for each  $wd_{2i}$  in the class  $c_2$ ,  
 $Pr(wd_{2i}|c_1) = p * Pr(wd_{2i}|c_2)$

- Otherwise ( $c_1$  is a class)

for each  $wd_{1i}$  in the class  $c_1$ ,  
**classbg\_to\_wdbg** ( $wd_{1i}, c_2, p$ )

We still need a mechanism to compute the  $Pr(wd_{2i}|c_2)$ , which denotes the probability of a word  $wd_{2i}$  occurring in a class  $c_2$ . This can be done in several ways, for example:

– Uniform distribution

$$Pr(wd_{2i}|c_2) = 1/|c_2|$$

where  $|c_2|$  denotes the number of words in the class  $c_2$ .

– Based on the training set

$$Pr(wd_{2i}|c_2) = Pr(wd_{2i})/\sum_j Pr(wd_{2j})$$

where  $Pr(w)$  denotes the unigram probability of the word  $w$  occurring in the training set.

5. We also need to regenerate the unigram probabilities for the words. This can be done in several ways, for example,

- Compute them directly from the training corpus
- for each word  $wd_i$  in a class  $c$ ,

$$Pr(wd_i) = Pr(c) * Pr(wd_i|c)$$

6. Finally, we need to compute the backoff weights for the words. Since we have already computed the bigrams and unigrams, this is straight forward [4, 8].

We will refer to the language model generated by expanding the bigrams as described above as a class based language model. We can further interpolate this class based language model with a word based language model generated directly from the training set. Experimental results [10] show that the interpolated language model performs better than the class based language model.

### 4. Limitations of Class Based Language Model

We can generate a class based language model for the library information system. The classes would be the major categories in the library database, for example, [author], [author’s] (eg., Tom Clancy’s), [subject] and [titles]. One problem with this approach is the large number of entries in some of the classes. However, there are many other limitations, for example,

- Class bigrams of the form [author’s] [title] (for example, Tom Clancy’s Hunt for Red October) generate too many (6000 author’s x 13,000 title = 78 million) word bigrams. This does not take advantage of the relations inherent in the database.

- The points at which the branching factor is high are those where the real content words are. So in some sense the model is helping you least where you need it most.
- Class bigrams expand only to a set of single words. When a word string is needed, it is entered in the lexicon as a single “compound word” which must be pronounced as a unit. This is very unsatisfactory. While it might suffice for author names, (ie. make TomClancy a single word), it clearly won’t work for titles. Some options are:

– Use several “compound words” for a title. Some “compound words” corresponding to the title “A Casebook on Ken Kesey’s One flew over the cuckoo’s nest” are as follows:

- \* ACasebookOnKesey’sTheCuckoo’sNest
- \* ACasebookOnKenKesey’sTheCuckoo’sNest
- \* OneFlewOverTheCuckoo’sNest
- \* KenKesey’sOneFlewOverTheCuckoo’sNest
- \* KenKesey’sTheCuckoo’sNest

This approach generates too many dictionary entries. Also, it is very tedious and it is impossible to cover all possible ways that a user might refer to this title.

– Use classes based on word position, for example, “A Casebook on Ken Kesey’s One flew over the cuckoo’s nest” can be represented by “[Title\_1] [Title\_2] ... [Title\_11]”, where the word “A” is in the class [Title\_1], “Casebook” is in the class [Title\_2], etc.

However, these bigram lose most of the associations in a specific title and weakens the constraint considerably.

The language model research has so far been focused on generating language models from large amounts of data. Simply gathering more data will not improve the situation much. The basic class-based sequences will most likely have a low perplexity and will not require large amounts of data for training. It is the class expansions which are the problem, and we cannot expect to reduce this even by gathering large amounts of data. We need to study new techniques for generating language models aided both by frequency data and information from the database.

## 5. Modified Class Based Language Model

We extended the class-based language model as follows in order to address the limitations and problems described earlier.

1. Allow word strings in classes, for example, “Tom Clancy’s” and “Clancy’s” can both occur in the [author’s] class.
2. The training data is mapped using these classes. For example, “Do you have Tom Clancy’s Hunt for Red October” is mapped to “[list] [author’s] [title]”
3. Generate separate n-gram language models for each of the classes and for the task. For example, [author] n-gram, [author’s] n-gram, and task n-gram.

4. We next combine all these language models to generate a task language model. For simplicity, we will assume that we are using bigram language models. We will describe below a recursive procedure (**classlm\_to\_wordlm**) that generates the appropriate word bigrams based on a bigram  $Pr(c_2|c_1)$  in the task language model. Then for each bigram  $Pr(y|x) = p$  in the task language model generated above, the procedure  $classlm\_to\_wordlm(x, y, p)$  will generate the corresponding word based bigrams. We next add the bigrams that occur more than once.

Each language model has two special symbols:

- $\langle s \rangle$  for start symbol
- $\langle /s \rangle$  for end symbol

The goal is to expand the class tokens (for example, [author]) in the task language model into words.

**classlm\_to\_wordlm** ( $c_1, c_2, p$ )

- If  $c_1$  is a word,
  - and if  $c_2$  is a word,  $Pr(c_2|c_1) = p$
  - Otherwise ( $c_2$  is a class)
    - for each bigram  $Pr(wd_i | \langle s \rangle)$  in the language model for  $c_2$ ,  
 $Pr(wd_i|c_1) = p * Pr(wd_i | \langle s \rangle)$
- Otherwise ( $c_1$  is a class)
  - if  $c_2$  is a word,
    - for each bigram  $Pr(\langle /s \rangle | wd_i)$  in the language model for  $c_1$ ,  
 $Pr(c_2|wd_i) = p * Pr(\langle /s \rangle | wd_i)$
  - Otherwise ( $c_2$  is a class)
    - for each bigram  $Pr(\langle /s \rangle | wd_{1i})$  in the language model for  $c_1$ ,  
 $classlm\_to\_wdbg(wd_{1i}, c_2, p * Pr(\langle /s \rangle | wd_{1i}))$

## 6. Experimental Results

The focus of the work has been on building better language models and improving the recognition accuracy of the system. We need to collect more data for training, development and test purposes. In this section, we will describe some preliminary results based on the data that is currently available.

The training and testing data was collected by the NIST speech group [5] using a prototype system. The training set consists of 418 sentences. Let’s look at some utterances that were used for testing:

- Do you have any books on playing the piano?
- Help find joke books
- I need some publishing information about Gone with the Wind
- Books about making speeches

We first generated a traditional class-based language model. We only used three classes: [author], [subject] and [title]. The tokens in each class were all the words that occurred in that database category. We next generated a new language model as described above (using the same classes), but this time the classes included the relevant database entries. We compared the performance of these language models on a test set containing 100 utterances. The new language model reduced the error rate from 54.9% to 43.0%.

We are trying to understand the reasons for the high error rate on the test set.

- Some errors were caused by out of vocabulary (oov) words. There were 22 utterances which had one or more oov words.
- Some errors seem to occur in regions that contain a long silence.

## 7. Future Work

The goal of the spoken language interface is to assist the user in accessing database information in a natural conversational way. We will continue experiments aimed at reducing the error rate. However, we want to decide if the user was successful in his goal and the recognition error rate may not be a very good measure.

We are working on a number of improvements to the language model:

1. Using more classes in the language model, for example, a separate class [list] for initial phrases, like “I want”, “Show me”, “Do you have”, “Are there any”
2. Using other corpuses, for example, ATIS or NAB news, to obtain language model information about domain independent phrases and many of the database phrases.
3. The language model should also represent many words and phrases which are similar to database items. For example, “Abstracting and Indexing” is a subject in the database, but “Abstraction or Indexing”, “Abstraction” and “Indexing” should also be represented in the language model.

Similarly, there may be many different ways of specifying an author, for example, a user can ask about “Bob Allen” instead of “Robert Allen”.

We have considered some completion techniques earlier [3], but we need to extend them and automate them as much as possible.

Spontaneous speech applications pose several challenges. We are also looking at dialogue issues that can constrain the search space. In particular, we are also trying to use multiple language models and use language models generated dynamically from the given context.

## 8. Acknowledgments

This research was supported in part by grants from NIST and from the Department of the Navy, Naval Research Laboratory. The views

and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

We thank Raj Reddy, Dave Pallett, the NIST speech group and the CMU speech group for their contributions to this work.

## 9. REFERENCES

1. Deborah A. Dahl, Madeline Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Proceedings of the DARPA Human Language Technology Workshop*, March 1994.
2. Xuedong Huang, Fileno Allewa, Mei-Yuh Hwang, and Ronald Rosenfeld. An overview of the SPHINX-II speech recognition system. In *Proceedings of the DARPA Human Language Technology Workshop*, March 1993.
3. Sunil Issar and Wayne Ward. CMU’s robust spoken language understanding system. In *Proceedings of Eurospeech*, pages 2147–2150, September 1993.
4. Slava Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. In *IEEE Transactions on Acoustics, Speech and Signal Processing*, volume ASSP-35, pages 400–401, March 1987.
5. Bruce Lund, William M. Fisher, John S. Garofolo, David S. Pallett, Mark Przybocki, and R. Allen Wilkinson. A spoken natural language interface to libraries. In *Proceedings of the Spoken Language Systems Technology Workshop*, January 95.
6. P. Placeway, R. Schwartz, P. Fung, and L. Nguyen. The estimation of powerful language models from small and large corpora. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, pages II33–II36, April 1993.
7. Mosur K Ravishankar. *Efficient Algorithms for Speech Recognition*. PhD thesis, Computer Science, Carnegie Mellon University, May 1996.
8. Ronald Rosenfeld. *Adaptive Statistical Language Modeling: A Maximum entropy approach*. PhD thesis, Computer Science, Carnegie Mellon University, April 1994.
9. Wayne Ward. Understanding spontaneous speech: the Phoenix system. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, pages 365–367, May 1991.
10. Wayne Ward and Sunil Issar. Recent improvements in the CMU spoken language understanding system. In *Proceedings of the ARPA Human Language Technology Workshop*, pages 213–216, March 1994.
11. Wayne Ward and Sunil Issar. A class based language model for speech recognition. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, page ???, May 1996.