Searching the Web by Voice

Alexander FRANZ
Google Inc.
2400 Bayshore Parkway
Mountain View, CA 94043
alex@google.com

Brian MILCH
Computer Science Division
University of California at Berkeley
Berkeley, CA 94720
milch@cs.berkeley.edu

Abstract

Spoken queries are a natural medium for searching the Web in settings where typing on a keyboard is not practical. This paper describes a speech interface to the Google search engine. We present experiments with various statistical language models, concluding that a unigram model with collocations provides the best combination of broad coverage, predictive power, and real-time performance. We also report accuracy results of the prototype system.

1 Introduction

Web search has a number of properties that make it a particularly difficult speech recognition problem. First, most queries are very short: typical queries range between one and five or six words, with a median length of two words. Second, search engine queries use a very large vocabulary. Even a vocabulary of 100,000 words covers only about 80% of the query traffic. Third, recognition must be done in close to real time. By contrast, the systems that achieved good accuracy on the 2000 NIST conversational telephone speech task required from 250 to 820 times real time (Fiscus et al., 2000). In this paper, we describe the language modeling techniques that we used to address these problems in creating a prototype voice search system (setting aside the question of how to browse the search results).

2 Trade-Offs in Language Modeling

A speech recognition system uses a language model to determine the probability of different recognition hypotheses. For our application, there is a trade-off among three considerations: What fraction of the query traffic is covered by the vocabulary of the language model? How much predictive power does the language model provide? And what is the observed computational complexity of applying the language model during hypothesis search?

At one extreme, a language model that simply used a list of the most frequent queries in their entirety would have the lowest coverage, but would provide the best predictive power within the covered queries (have the lowest per-query perplexity), and would be the least computationally expensive. At the other extreme, (Lau, 1998; Ng, 2000) report on experiments with sub-word n-gram language models, which have very high coverage, but rather low predictive power (high per-query perplexity).

We experimented with various configurations of back-off word n-gram models (Katz, 1987; Jelinek, 1997). In our experience with commercially available speech recognition systems, we found that for a vocabulary size of 100,000 items, unigram models were the only computationally feasible choice, yielding close to real-time performance. When using the bigram model, the recognizer needed to spend several minutes processing each utterance to achieve accuracy as high as it achieved with the unigram model. Recognition with a bigram model was unacceptably slow even when we pruned the model by removing bigrams that provided little improvement in perplexity (Stolcke, 1998). For this reason, we explored a method to increase the predictive power of the unigram model by adding collocations to its vocabulary.

3 Collocations

A collocation is ”an expression of two or more words that corresponds to some conventional way of saying things” (Manning and Schütze, 1999). Sometimes, the notion of collocation is defined in terms of syntax (by possible part-of-speech patterns) or in terms of semantics (requiring collocations to exhibit non-compositional meaning) (Smadja, 1993). We adopt an empirical approach and consider any sequence of words that co-occurs more often than chance a potential collocation.

3.1 The Likelihood Ratio

We adopted a method for collocation discovery based on the likelihood ratio (Dunning, 1993). Suppose we wish to test whether two words $w_1 w_2$ form a collocation. Under the independence hypothesis we assume that the probability of observing the second word w_2 is independent of the first word: $P(w_2|w_1) = P(w_2|\neg w_1)$. The alternative is that the two words form a collocation: $P(w_2|w_1) > P(w_2|\neg w_1)$. The likelihood ratio λ is calculated by dividing the likelihood of observing the data under the hypothesis of independence, $L(H_i)$, by the likelihood of observing the data under the hypothesis that the words form a collocation, $L(H_c)$:

$$\lambda = \frac{L(H_i)}{L(H_c)}$$

After counting how many times the word w_2 and the sequence $w_1 w_2$ occur in training data, we derive maximum likelihood estimates for $P(w_2|w_1)$ and P$(w_2|\neg w_1)$, and compute the two likelihoods using the binomial distribution (see (Manning and Schütze, 1999) for details). If the likelihood ratio is small, then H_c explains the data much better than H_i, and so the word sequence is likely to be a collocation.

3.2 Discovering Longer Collocations

Two-word collocations can be discovered by carrying out the calculations described above for all frequent two-word sequences, ranking the sequences according to their likelihood ratios, and selecting all sequences with ratios below a threshold. Collocations are not limited to two words, however. We have extended Dunning's scheme to discover longer collocations by performing the likelihood ratio tests iteratively. The algorithm for this is shown below.

1. Count occurrences of sequences of tokens (initially, words) for lengths of up to n tokens.
2. For each sequence $S = w_1, \ldots, w_n$ of n tokens in the training data, let $\lambda(S)$ be the greatest likelihood ratio found by considering all possible ways to split the n-token sequence into two contiguous parts.
3. Sort the n-token sequences S by $\lambda(S)$, and designate the K_n sequences with the lowest $\lambda(S)$ values as collocations.
4. Re-tokenize the data by treating each collocation as a single token.
5. Set $n = n - 1$.
6. Repeat through $n = 2$.

The constants K_n, which represent the number of desired collocations of length n, are chosen manually. This algorithm solves two key problems in discovering longer collocations. The first problem concerns long word sequences that include shorter collocations. For example, consider the sequence *New York flowers*: this sequence does indeed occur together more often than chance, but if we identify *New York* as a collocation then including *New York flowers* as an additional collocation provides little additional benefit (as measured by the reduction in per-query perplexity).

To solve this problem, step 2 in the collocation discovery algorithm considers all $n - 1$ possible ways to divide a potential collocation of length n into two parts. For the case of *New York flowers*, this means considering the combinations *New York* + *flowers* and *New* + *York flowers*. The likelihood ratio used to decide whether the word sequence should be considered a collocation is the maximum of the ratios for all possible splits. Since *flowers* is close to independent from *New York*, the potential collocation is rejected.

The second problem concerns subsequences of long collocations. For example, consider the collocation *New York City*. *New York* is a collocation in its own right, but *York City* is not. To distinguish between these two cases, we need to note that *York City* occurs more often than chance, but usually as part of the larger collocation *New York City*, while *New York* occurs more often than chance outside the larger collocation as well.

The solution to this problem is to find larger collocations first, and to re-tokenize the data to treat collocations as a single token (step 4 above). In this way, after *New York City* is identified as a collocation, all instances of it are treated as a single token, and do not contribute to the counts for *New York* or *York City*. Since *New York* occurs outside the larger collocation, it is still correctly identified as a collocation, but *York City* drops out.

4 Implementing Voice Search

4.1 Training and Test Data

To create the various language models for the voice search system, we used training data consisting of 19.8 million query occurrences, with 12.6 million distinct queries. There were 54.9 million word occurrences, and 3.4 million distinct words. The

evaluation data consisted of 2.5 million query occurrences, with 1.9 million distinct queries. It included 7.1 million word occurrences, corresponding to 750,000 distinct words.

We used a vocabulary of 100,000 items (depending on the model, the vocabulary included words only, or words and collocations). The word with the lowest frequency occurred 31 times.

4.2 Constructing the Language Model

The procedure for constructing the language model was as follows:

1. Obtain queries by extracting a sample from Google's query logs.
2. Filter out non-English queries by discarding queries that were made from abroad, requested result sets in foreign languages, etc.
3. Use Google's spelling correction mechanism to correct misspelled queries.
4. Create lists of collocations as described in Section 3 above.
5. Create the vocabulary consisting of the most frequent words and collocations.
6. Use a dictionary and an automatic text-to-phonemes tool to obtain phonetic transcriptions for the vocabulary, applying a separate algorithm to special terms (such as acronyms, numerals, URLs, and filenames).
7. Estimate n-gram probabilities to create the language model.

4.3 System Architecture

Figure 1 presents an overview of the voice search system. The left-hand side of the diagram represents the off-line steps of creating the statistical language model. The language model is used with a commercially available speech recognition engine, which supplies the acoustic models and the decoder.

The right-hand side of the diagram represents the run-time flow of a voice query. The speech recognition engine returns a list of the n-best recognition hypotheses. A disjunctive query is derived from this n-best list, and the query is issued to the Google search engine.

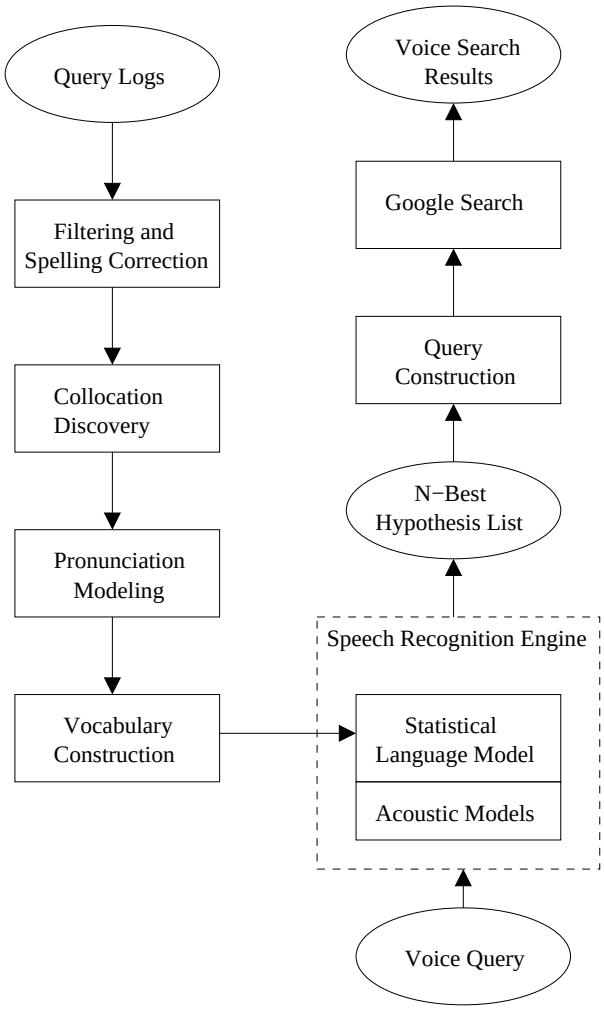

Figure 1: Voice Search Architecture

5 Coverage and Perplexity Results

We evaluated the coverage and perplexity of different language models. In our experiments, we varied the language models along two dimensions:

Context. We evaluated unigram, bigram, and trigram language models to see the effect of taking more context into account.

Collocations. We evaluated language models whose vocabulary included only the 100,000 most frequent words, as well as models whose vocabulary included the most frequent words and collocations. Specifically, we ran the algorithm in Section 3.2 to obtain 5000 three-word collocations, and then 20,000 two-token collocations (which could contain two, four, or six words). To obtain the final vocabulary of 100,000 words and collocations, we tokenized the training corpus using a vocabulary with all 25,000 collocations, and then selected the 100,000 most frequent tokens. Most of the collocations were included in the final vocabulary.

5.1 Query Coverage

We say that a vocabulary covers a query when all words (and collocations, if applicable) in the query are in the vocabulary. Table 1 summarizes the coverage of different-sized vocabularies composed of words, words + collocations, or entire queries.

	Words	Collocations	Queries
25k	62.2%	50.0%	12.4%
50k	72.2%	65.2%	15.3%
75k	76.7%	72.8%	17.1%
100k	79.2%	76.9%	18.4%
200k	83.9%	83.2%	21.5%
300k	85.9%	85.5%	23.2%
400k	87.1%	86.8%	24.3%
500k	87.9%	87.7%	25.2%

Table 1: Percent of Query Occurrences Covered

At a vocabulary size of 100,000 items, there is only a difference of 2.7% between an all-word vocabulary, and a vocabulary that includes words and collocations. Thus, using collocations does not result in a large loss of coverage.

5.2 Perplexity Results

We compared the perplexity of different models with a 100,000 item vocabulary in two ways: by measuring the per-token perplexity, and by measuring the per-query perplexity. Per-token perplexity measures how well the language model is able to predict the next word (or collocation), while per-query perplexity measures the contribution of the language model to recognizing the entire query. To avoid complications related to out-of-vocabulary words, we computed perplexity only on queries covered by the vocabulary (79.2% of the test queries for the all-word vocabulary, and 76.9% for words plus collocations). The results are shown in Table 2.

Model	Per-token	Per-query
Word unigram	1614	3.5×10^{12}
Word bigram	409	1.6×10^{10}
Word trigram	340	7.9×10^{9}
Collocation unigram	2019	2.5×10^{11}
Collocation bigram	763	8.8×10^{9}
Collocation trigram	696	6.4×10^{9}

Table 2: Language Model Perplexity

These results show that there is a large decrease in perplexity from the unigram model to the bigram model, but there is a much smaller decrease in perplexity in moving to a trigram model. Furthermore, the per-token perplexity of the unigram model with collocations is about 25% higher than that of the word-based unigram model. This shows that the distribution of the word plus collocation vocabulary is more random than the distribution of words alone. The bigram and trigram models exhibit the same effect.

5.3 Per-Query Perplexity

Per-query perplexity shows the gains from including collocations in the vocabulary. Using collocations means that the average number of tokens (words or collocations) per query decreases, which leads to less uncertainty per query, making recognition of entire queries significantly easier. For the unigram model, collocations lead to a reduction of per-query perplexity by a factor of 14. We can see that the per-query perplexity of the unigram model with collocations is about halfway between the word-based unigram and bigram models. In other words, collocations seem to give us about half the effect of word bigrams.

Similarly, the per-query perplexity of the bigram model with collocations is very close to the perplexity of the word-based trigram model. Furthermore, moving from a collocation bigram model to a collocation trigram model only yields a small additional per-query perplexity decrease.

6 Recall Evaluation

We also evaluated the recall of the voice search system using audio recordings that we collected for this purpose. Since only unigram models yielded close to real-time performance for the speech recognizer, we limited our attention to comparing unigram models with a vocabulary size of 100,000 items consisting of either words, or words and collocations. With these unigram models, the recognizer took only 1-2 seconds to process each query.

6.1 Data Collection

We collected voice query data using a prototype of the voice search system connected to the phone network. In total, 18 speakers made 809 voice queries. The collected raw samples exhibited a variety of problems, such as low volume, loud breath sounds, clicks, distortions, dropouts, initial cut-off, static, hiccups, and other noises. We set aside all samples with insurmountable problems and speakers with

very strong accents. This left 581 good samples. These good samples include a variety of speakers, various brands of cell phones as well as desktop phones, and different cell phone carriers. The average length of the utterances was 2.1 words.

6.2 Recall Results

We used the 581 good audio samples from the data collection to evaluate recognition recall, for which we adopted a strict definition: disregarding singular/plural variations of nouns, did the recognizer return the exact transcription of the audio sample as one of the top n (1, 5, 10) hypotheses? Note that this recall metric incorporates coverage as well as accuracy: if a query contains a word not in the vocabulary, the recognizer cannot possibly recognize it correctly. The results are shown in Table 3.

Recall	Words only	Words + Collocations
@1	27.5%	43.4%
@5	42.3%	56.8%
@10	45.8%	60.4%

Table 3: Recall Results on 581 Queries

These results show that adding collocations to the recognition vocabulary leads to a recall improvement of 14-16 percentage points.

7 Conclusion

We have shown that a commercial speech recognition engine, using a unigram language model over words and collocations, can return the correct transcription of a spoken search query among its top 10 hypotheses about 60% of the time. Because we were not able to use a bigram model without sacrificing real-time performance, including collocations in the language model was crucial for attaining this level of recall.

Still, there is a lot of room for improvement in the recall rate. One idea is to rescore the recognizer's top hypotheses with a bigram or trigram language model in a postprocessing step. However, there are many cases where the correct transcription is not among the recognizer's top 100 hypotheses. Another approach would be to adapt the acoustic and language models to individual users, but such personalization would require a different system architecture. We might also improve our language models by training on documents as well as queries (Fujii, 2001).

The language models described in this paper were trained from typed queries, but queries made by voice in different settings might have quite different characteristics. For example, our data consisted of *keyword queries*, but voice search users might prefer to ask questions or make other types of *natural language queries* (which would actually be easier to model and recognize). The voice search system is currently available at `labs.google.com`; the data from this demonstration system could lead to improved language models in the future.

References

T. Dunning. 1993. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.

J. Fiscus, W. M. Fisher, A. Martin, M. Przybocki, and D. S. Pallett. 2000. 2000 NIST evaluation of conversational speech recognition over the telephone. In *Proceedings of the 2000 Speech Transcription Workshop*.

A. Fujii. 2001. Speech-driven text retrieval: Using target IR collections for statistical language model adaptation in speech recognition. In *SIGIR '01 Workshop on IR Techniques for Speech Applications*, New Orleans, LA.

F. Jelinek. 1997. *Statistical Methods for Speech Recognition*. MIT Press, Cambridge, MA.

S. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3):400–401.

R. Lau. 1998. *Subword Lexical Modelling for Speech Recognition*. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA.

C. Manning and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.

K. Ng. 2000. *Subword-based Approaches for Spoken Document Retrieval*. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA.

F. Smadja. 1993. Retrieving collocations from text: Xtract. *Computational Linguistics*, 19(1):143–177.

A. Stolcke. 1998. Entropy-based pruning of backoff language models. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pages 270–274, Lansdowne, VA.