INTERSPEECH 2009 BRIGHTON

Recognition and Correction of Voice Web Search Queries

Keith Vertanen, Per Ola Kristensson

University of Cambridge, Cavendish Laboratory, United Kingdom

{kv227,pok21}@cam.ac.uk

Abstract

In this work we investigate how to recognize and correct voice web search queries. We describe our corpus of web search queries and show how it was used to improve recognition accuracy. We show that using a search-specific vocabulary with automatically generated pronunciations is superior to using a vocabulary limited to a fixed pronunciation dictionary. We conducted a formative user study to investigate recognition and correction aspects of voice search in a mobile context. In the user study, we found that despite a word error rate of 48%, users were able to speak and correct search queries in about 18 seconds. Users did this while walking around using a mobile touch-screen device.

Index Terms: speech recognition, voice search, error correction, mobile web search

1. Introduction

Recognition of spoken search queries is a challenging problem. First, search queries tend to be brief, lack context, and lack strong word order. Second, queries often involve a diverse vocabulary that may be difficult to capture—even using a language model with a very large vocabulary. Third, a proper language model for voice search requires appropriate training material.

Here we outline the steps we took to create a voice search system. We describe a search query corpus that we collected over a period of four years. We show how we handled the large and diverse vocabularies indicative of web search queries. We argue that voice search is made more effective and pleasant by introducing a correction interface based on a word confusion network. Finally, we test the performance of our mobile voice search system by letting users enter queries while walking around indoors.

2. Search Query Corpus

To our knowledge, no publicly available search query corpus currently exists. Therefore, we resorted to collecting our own. As a basis, we used past user queries mined from various web search engine "spy" pages. These spy pages display recent or popular searches made by users of a particular search engine. Over a period of four years, we collected about one million queries. We also used the "related search" feature of the *Yahoo!* search engine [1] to further expand the collection. The *Yahoo!* related search feature returns a list of other possible queries based on the original query. For example, the original query "german steins" returns related queries such as "beer mugs" and "beer glasses".

We added the *Yahoo!* related search variants to our collection. We also used *Yahoo!* to filter out any of our original queries that had no related results. This eliminated garbage in the original spy results. After expansion and filtering we ended up with 7.1M search queries.

	Mean	Median	Max	Std dev
Words	2.9 (2.6)	3 (2)	37 (39)	1.4 (1.7)
Chars	18.9 (16.8)	10 (15)	168 (224)	8.5 (9.2)

Table 1: Statistics about the queries in our corpus. Numbers in parentheses are from a *Google* corpus of searches (see text).

To collect more data, we fed individual "seed" words to the *Yahoo!* related search feature. For some seed words, *Yahoo!* would respond with a list of related searches. The following sources were used for seed words: the CMU dictionary, the titles of *Wikipedia* articles, and the top words in the *Google* n-gram corpus. From approximately 500K unique words, we obtained an additional 3.1M search queries. This resulted in a total corpus of 10.2M queries. We reserved 10% of these queries as test data.

Table 1 summarizes our search query corpus. Our corpus statistics were broadly similar to the published statistics on search queries submitted to *Google* from mobile phones and PDAs [2]. Obviously, our collection procedure may have resulted in a smaller and lower-quality data collection than what might be available to commercial search engine companies. However, as we'll see, even our modest corpus provided major gains over merely using readily-available newswire training text. We'll also see that models built using our corpus were successful at recognizing spoken search queries.

3. Handling Large Vocabularies

A challenge when recognizing queries is the large and diverse vocabulary used in the queries. One way to handle this is to simply increase the size of the recognizer's vocabulary. In the simplest case, words are added from sources that have pronunciations for each word. But such sources are limited in size and may not match the types of words used in web search queries. For example, the CMU dictionary has 123K English words but lacks many common search terms such as "ebay" and "firefox".

To demonstrate the problem, we compared the out-of-vocabulary (OOV) rates of vocabularies limited to the CMU dictionary and vocabularies that were free to use any word. Both vocabularies used the most frequent words in our corpus of search queries. We measured the OOV rate of the vocabularies using a set of 3K queries collected from a search engine spy [3]. This search engine spy was not used in the original collection of our corpus. We proofread the queries to correct obvious typos. We also removed garbage and expanded abbreviations. As shown in Table 2, using the vocabularies based on the web search corpus substantially reduced the OOV rate.

Clearly a search-specific vocabulary is preferable. But to use such a vocabulary, pronunciations are required for words

Vocab	20K	40K	80K	120K	200K	300K
CMU	12.2%	8.6%	7.5%	7.2%	–	–
Search	11.7%	7.3%	4.7%	3.7%	3.0%	2.6%

Table 2: OOV rates in a test set of queries varying the vocabulary size. *CMU* was limited to the CMU pronunciation dictionary. *Search* could use any word.

not in the dictionary. One way to generate these pronunciations is to use the joint multigram model [4]. The joint multigram model is a statistical model that uses an existing dictionary to learn sub-word units called graphones. Graphones consist of likely pairings of letters and phones in a language. The dependency between graphones is modeled using a standard n-gram language model (LM). In past work [5], we found accurate letter-to-phone conversion (6.5% phone error rate) could be obtained using small graphones (0–1 letters/phones per graphone) and a 6-gram LM. Our graphone model can produce the best phone sequence for a word, or it can produce an n-best list of phone sequences (and their associated probabilities under the model). For further details about using graphones for letter-to-phone conversion, see [6].

4. Recognition Experiments

Our recognition experiments used HTK v3.4, HDecode, and the acoustic model training recipe from [7]. We trained a speaker-independent acoustic model using cross-word triphones on WSJ (211 hours). We used 12 MFCCs plus deltas and delta-deltas, 16 Gaussians/state, and 8K tied-states. Except where noted, we used a bigram LM for decoding, rescoring lattices with a trigram. We report decoding time as a proportion of the utterance audio time (×RT) on a 3.3 GHz desktop computer.

4.1. Development Test Set

To provide a development audio test set, one of the authors recorded 755 queries. Audio was recorded at 16 kHz using a wired headset microphone. The queries were drawn from a search engine spy [3] that was not used for the original corpus collection. The test set had an OOV rate of 7.9% using a 86K vocabulary that used all CMU dictionary words that occurred in the search query corpus. The average query length was 3.0 words (19.5 characters). The test set had a per-word perplexity of 296 using a 86K trigram LM trained on the search corpus.

4.2. Mixture Language Model

When building the language model, there is a question of the relative importance of using search-specific training text versus easily-available newswire text. In addition, since our search query corpus was somewhat small (27M words), we hypothesized it might help to use additional non-search training text.

To investigate these issues, we trained a range of LMs that mixed a search LM and a newswire LM. For the newswire LM, we segmented the original training sentences of the CSR-III newswire corpus (222M words) into short pseudo-sentences of between 1–5 words. We trained the search and newswire LMs separately with no count cutoffs. We then created a mixture LM using SRILM. The final mixture LM was entropy-pruned to reduce its size. As a vocabulary, we used the 86K words in the CMU dictionary that appeared in our search query corpus.

As shown in Table 3, an LM trained on only search data ($\lambda = 0.0$) was much better than an LM trained on only newswire data ($\lambda = 1.0$). Mixing in a small amount of newswire data ($\lambda = 0.05$) improved the WER slightly. Except where noted, all remaining experiments used mixture LMs with $\lambda = 0.05$.

λ:	0.0	0.05	0.1	0.15	0.2	0.5	1.0
WER:	28.3	28.2	28.4	28.6	28.6	29.6	41.0

Table 3: WER varying the mixture weight between LMs built with search data ($\lambda = 0.0$) and newswire data ($\lambda = 1.0$).

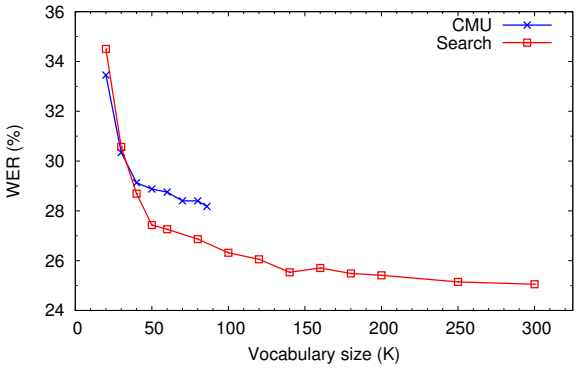

Figure 1: WER using the different vocabulary types and sizes.

4.3. Vocabulary Type

We tested two types of vocabularies. Each used the most frequent words seen in the search query corpus. The first (*CMU*) was limited to words that occurred in the CMU pronunciation dictionary. The other (*Search*) was free to use any word. For words that were not in the CMU dictionary, the best phone sequence was found using a letter-to-phone graphone model (0–1 graphones with a 6-gram LM). As shown in Figure 1, for smaller vocabulary sizes (20–40K) *CMU* and *Search* performed similarly. For vocabularies larger than 40K, the *Search* vocabularies were superior. The *Search* vocabulary continued to noticeably reduce WER until about 140K words.

We also investigated including multiple pronunciations for words in the *Search* vocabularies. This was done by performing an n-best search using the letter-to-phone model. We found no significant reduction of WER and the additional pronunciations substantially increased decoding time.

4.4. Language Model Size

Prior work on web search query recognition has used a unigram LM [8, 9]. As shown in Table 4, we found that using a bigram was much more accurate and did not slow down decoding. Further, rescoring the bigram lattices with a trigram improved accuracy slightly and required little additional processing time.

4.5. Testing on Realistic Audio

In the user study (to be described shortly) we collected 776 queries from 4 users. The users spoke the queries while walking around indoors. They used a mobile device to see the recognition results and perform corrections. We used the collected audio in offline experiments using the recognition techniques and setup described previously. As shown in Table 5, we observed gains by using a mixture LM, using a *Search* vocabulary, and increasing the vocabulary size to 100K. Using the 100K *Search*

LM size	×RT	Memory	WER
Unigram	1.4	152 MB	41.7
Bigram	1.3	172 MB	27.1
Trigram†	1.3	291 MB	26.3

Table 4: Real-time factor, memory use, and WER for different LM sizes. Results are on our dev. test set using a 100K *Search* vocab. †Bigram used for recognition, rescored with a trigram.

Vocab type	Vocab size	Mixture λ	×RT	WER
CMU	86K	1.00	2.3	54.5
CMU	86K	0.00	1.7	37.7
CMU	86K	0.05	1.9	37.4
Search	86K	0.05	2.3	35.7
Search	100K	0.05	2.3	35.1
Search	140K	0.05	2.7	35.3

Table 5: Performance of different LMs on the audio recorded during our user study.

vocabulary, the WER was 35% on our users' audio. This was higher than the 26% we saw on our development test set. Artifacts introduced by the wireless microphone as well as breath noises from our users may have contributed to this increase.

5. Correction Interface

Spoken web search queries are difficult to recognize. Further, voice search is likely to be used in noisy environments. Therefore, recognition errors are quite likely. We hypothesize that a correction interface is required to make voice search practical. We have previously developed a mobile speech recognition system called Parakeet [10]. Parakeet runs on mobile touch-screen devices like the Nokia N800 (Figure 2).

Parakeet works as follows. First, the user speaks a query into a wireless microphone. When recognition is complete, the best hypothesis is displayed. If a word is a recognition error, the user can change it to one of up to four alternative words by touching the desired word. Words can be deleted altogether by touching the "X" box (Figure 2). The interface is based on a word confusion network [11] built from the recognition lattice.

In addition to selecting alternative words, the user can also perform corrections using an on-screen keyboard (Figure 3). To open the keyboard interface, the user double-taps a word. The

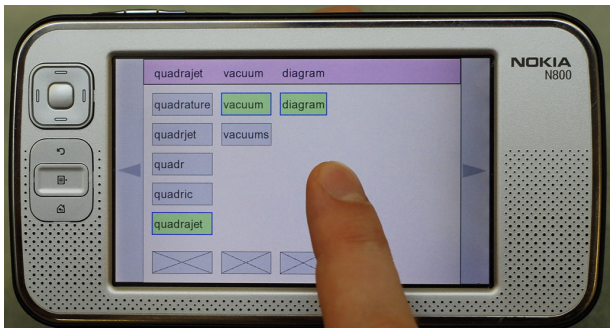

Figure 2: Parakeet's word confusion network interface. The user has spoken the query "quadrajet vacuum diagram". Errors are corrected by selecting alternative words from each column.

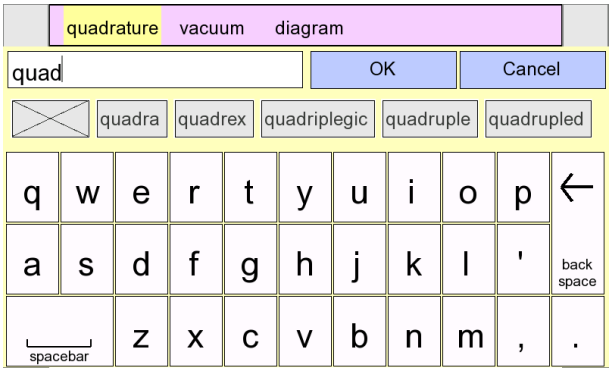

Figure 3: Parakeet's keyboard interface. As the user types, word predictions appear in the boxes above the keyboard.

user then types letters or selects words directly from a set of word predictions proposed by the system. In this work, we used Parakeet to study how users spoke queries and corrected the recognition results. For further details about Parakeet's design, computational experiments, and user studies, see [10].

To see how good Parakeet would be at allowing users to correct errors, we simulated an ideal "oracle" user. First, we performed recognition on each utterance in our development test set and generated the words that would be displayed in Parakeet's word confusion network interface. The oracle user selected the words in this interface that minimized WER. The oracle user did not use the keyboard interface. Our computational experiment showed that the word confusion network interface allowed correction of over half the recognition errors, reducing WER from 26% to 12% (using a 100K *Search* vocabulary).

6. Formative User Study

Having observed theoretical benefits of introducing an error correction interface to voice search, we set out to investigate how real users would perform using our system. Our user study had three goals. First, to collect externally valid speech data from users walking around and performing web search queries on a mobile device. Second, to validate the design of our voice search system in a realistic setting. Third, to get an estimate of the envelope of voice search performance.

We recruited four participants from the university campus. All participants were female native speakers of American English. Participants were asked to speak a set of past queries obtained from a search engine spy [3]. We had participants use our Parakeet system [10] running on a Nokia N800 mobile device (Figure 2). Participants only spoke and corrected queries. They did not execute actual web search queries or browse results.

Audio was recorded at 16 kHz using a Jabra M5390 wireless microphone. We used a 100K *Search* vocabulary, automatically generating a single pronunciation for words not in the CMU dictionary. We used PocketSphinx [12] for recognition, using the setup described in [10] and a speaker-independent acoustic model. While Parakeet can perform recognition on the mobile device, this was not practical for this difficult recognition task. Instead, recognition was performed on a nearby laptop that was wirelessly connected to the N800. However, to the user, it appeared just as if the device was doing the recognition.

In our study, users had to wait on average 2.7 s ± 0.8 s for the recognition result to appear. The average recognition WER was 48% (Table 6). This WER was higher than in offline experiments using the users' audio (see Section 4.5). This increase re-

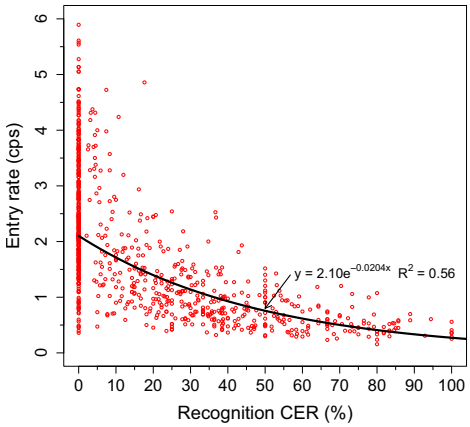

Figure 4: The entry rate and recognition character error rate (CER) for each query entered during our user study.

Error rate	Before correction	After correction
Character (CER)	22.9	1.7
Word (WER)	48.0	8.3
Sentence (SER)	61.6	17.4

Table 6: Error rates before and after correction in our user study.

flects the PocketSphinx recognition setup used during the study as well as compromises made to provide fast recognition.

In total, participants completed 776 queries (2434 words). The OOV rate was 3.8% using a 100K *Search* vocabulary. In 43% of the queries, participants reviewed the 1-best result and went directly to the next query. In only 7% of the queries did participants respeak a query. Participants spent 47% of their correction time using the confusion network interface and 53% of their time using the on-screen keyboard.

Including recording, recognition delay, and correction time, it took participants on average $18\,\text{s} \pm 15\,\text{s}$ to enter a query. Participants' mean entry rate was $1.7\,\text{cps} \pm 1.2\,\text{cps}$ (characters per second). As a reference point, Kamvar and Baluja [2] report that *Google*'s mobile web search queries (typically typed using a telephone keypad or a QWERTY thumb keyboard) had a mean entry rate of 0.42 cps (calculated from Table 2 in [2]).

As expected, participants' entry rate was heavily influenced by the recognition error rate (Figure 4). We measured recognition error rate using character error rate (CER) to provide a finer grain error measure than WER. Given the short nature of queries, WER can be high due to only minor differences such as spacing or pluralization. Figure 4 gives us an estimate of the envelope of voice search performance. It can be used to predict the performance of voice search at different recognition error levels. Notably, even at a 40% CER, the text entry rate is still at an acceptable 1 cps (12 wpm). At low character error rates in the range 0–10%, we observed a range of entry speeds from 0.4 to 5.9 cps (4 to 71 wpm).

There were two limitations to our study. First, the number of participants was small. Second, participants did not perform a full search task in which they formulated their own queries and browsed the results. Nevertheless, our results suggest spoken web search queries can potentially be very fast.

7. Conclusions

In this paper we described how we built a research system that enables users to speak and correct web search queries. We described how we collected a corpus of web search queries. We showed that a search-specific LM is important for accurate recognition. We also showed that using a vocabulary with automatically generated pronunciations was superior to limiting a vocabulary to a fixed pronunciation dictionary.

We found that our users were on average able to speak and correct a web search query in about 18 seconds. This was despite a high overall WER of 48%. We gave an estimate of the performance envelope of voice search. This estimate was obtained from users walking around and speaking web search queries to a mobile device. Assuming similar recognition latency and correction interface, we predict that an average entry rate of 1.9 cps (23 wpm) is possible when CER is in the range 0–10%. This shows that, despite limitations such as latency and recognition errors, a voice search system with a good correction interface may be a useful addition to mobile devices.

8. Acknowledgements

We would like to express our gratitude to our study participants. This research was in part funded by a donation from Nokia. The following applies to P.O.K. only: The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement number 220793.

9. References

[1] "Yahoo! developer network home," http://developer.yahoo.com, accessed December 8, 2008.

[2] M. Kamvar and S. Baluja, "Deciphering trends in mobile search," *IEEE Computer*, vol. 40, no. 8, pp. 58–62, 2007.

[3] "Dogpile SearchSpy," http://www.dogpile.com/dogpile/ws/searchspy/rfcid=4101/rfcp=InternalNavigation/_iceUrlFlag=11?_IceUrl=true, accessed December 8, 2008.

[4] S. Deligne, F. Yvon, and F. Bimbot, "Variable-length sequence matching for phonetic transcription using joint multigrams," *Proceedings of European Conference on Speech Communication and Technology*, pp. 2243–2246, 1995.

[5] K. Vertanen, "Combining open vocabulary recognition and word confusion networks," in *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, March 2008, pp. 4325–4328.

[6] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Communications*, vol. 50, no. 5, pp. 434–451, 2008.

[7] K. Vertanen, "Baseline WSJ acoustic models for HTK and Sphinx: Training recipes and recognition experiments," http://www.inference.phy.cam.ac.uk/is/papers/baseline_wsj_recipes.pdf, Cavendish Lab., Tech. Rep., 2006.

[8] A. Franz and B. Milch, "Searching the web by voice," in *Proceedings of the Conference on Computational Linguistics*, 2002, pp. 1213–1217.

[9] J. Sherwani, D. Yu, T. Paek, M. Czerwinski, Y. C. Ju, and A. Acero, "Voicepedia: Towards speech-based access to unstructured information," in *Proceedings of European Conference on Speech Communication and Technology*, 2007, pp. 146–149.

[10] K. Vertanen and P. O. Kristensson, "Parakeet: A continuous speech recognition system for mobile touch-screen devices," in *IUI '09: Proceedings of the 14th International Conference on Intelligent User Interfaces*. ACM, 2009, pp. 237–246.

[11] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus in speech recognition: Word error minimization and other applications of confusion networks," *Computer Speech and Language*, vol. 14, no. 4, pp. 373–400, 2000.

[12] D. Huggins-Daines, M. Kumar, A. Chan, A. W. Black, M. Ravishankar, and A. I. Rudnicky, "PocketSphinx: A free, real-time continuous speech recognition system for hand-held devices," in *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2006, pp. 185–188.