Natural Language Engineering **17** (2): 243–264. © Cambridge University Press 2011
doi:10.1017/S1351324911000040

Finite-state models for speech-based search on mobile devices

TANIYA MISHRA and SRINIVAS BANGALORE
AT&T Labs-Research, 180 Park Ave
Florham Park, NJ 07932, USA
email: {taniya,srini}@research.att.com

(*Received 23 May 2010; revised 24 September 2010; accepted 22 October 2010*)

Abstract

In this paper, we present techniques that exploit finite-state models for voice search applications. In particular, we illustrate the use of finite-state models for encoding the search index in order to tightly integrate the speech recognition and the search components of a voice search system. We show that the tight integration mutually benefits Automatic Speech Recognition and improves the search. In the second part of the paper, we discuss the use of finite-state techniques for spoken language understanding, in particular, to segment an input query into its component semantic fields so as to improve search as well as to extend the functionality of the system and be able to execute the user's request against a backend database.

1 Introduction

Mobile devices have revolutionized access to information. Information access has moved from desktop and laptop computers in office and home environments to be an any place, any time activity due to mobile devices. Mobile devices are pervasive due to their ability to access unlimited amount of information, almost everywhere, through the Internet.

Even so, information access using text input on mobile devices with small screens and soft/small keyboards is tedious and unnatural. In addition, by the *mobile* nature of these devices, users often like to use them in hands-busy environments, ruling out the possibility of typing text. Therefore, it might be expected that spoken utterances provide a more natural and less cumbersome modality for accessing information using mobile devices.

A second issue we address in this paper is to shorten the time needed to satisfy the user's information need. The current approach of information access involves a user typing in a query using keywords to a search engine, browsing the returned results on the small screen to select a potentially relevant document, suitably magnifying the screen to view the document and searching for the answer to her question in the document. By providing a method for the user to pose her query in natural language and presenting the relevant answer(s) to her question, we expect the user's information need to be fulfilled in a shorter period of time.

https://doi.org/10.1017/S1351324911000040 Published online by Cambridge University Press

As a solution towards addressing the aforementioned two issues, we present a speech-driven search application, Qme!. The system provides a natural input modality – spoken language input – for the users to pose their information need and presents a collection of *answers* that potentially addresses the information need directly. In this paper, we provide a speech driven access to two different repositories: (a) business listings containing the name, address and phone number of businesses; and (b) short question and answer pairs.

It is evident that using spoken language to access information comes at a cost – speech recognition is error prone and this may lead to lower search relevance. In this paper, we address this issue by tightly coupling speech recognition and search so as to alleviate some of the errors that might percolate from the speech recognizer to the search component. We model this interaction using finite-state transducers which allows us a clean mathematical framework for integrating the constraints across the two components. We show an improvement in speech recognition and search results by exploiting the co-constraints from the two components. Further, in order to improve the precision of search as well as to execute user's request against a backend database, it is imperative to parse a user's utterance to extract relevant attributes from the user's request. We show that the parsing process can also be represented as a finite-state transducer that can also be tightly integrated with the speech recognizer.

The layout of the paper is as follows. In Section 2, we review the related literature. We present the retrieval methods we used to implement Qme! in Section 3. In Section 4, we discuss our approach to tight coupling of speech recognition and search components and present the results of our approach in Section 5. In Section 6, we present an finite-state transducer-based method for extracting the relevant parameters of a user's question which can transparently scale to input word lattices from speech recognition. In Section 7, we present the results of the parsing experiments and conclude in Section 8.

2 Related work

Early question-answering (QA) systems, such as Baseball (Green *et al.* 1961) and Lunar (Woods 1973) were carefully hand-crafted to answer questions in a limited domain, similar to the QA components of ELIZA (Weizenbaum 1966) and SHRDLU (Winograd 1972). However, there has been a resurgence of QA systems following the TREC conferences with an emphasis on answering factoid questions. This work on text-based question-answering which is comprehensively summarized in Maybury (2004), range widely in terms of linguistic sophistication. At one end of the spectrum, there are linguistically motivated systems (Katz 1997; Waldinger *et al.* 2004) that analyze the user's question and attempt to synthesize a coherent answer by aggregating the relevant facts. At the other end of the spectrum, there are data intensive systems (Dumais *et al.* 2002) that attempt to use the redundancy of the web to arrive at an answer for factoid style questions. There are also variants of such QA techniques that involve an interaction and use context to resolve ambiguity (Yang, Feng and DiFabbrizio 2006). In contrast to these approaches, our method matches

https://doi.org/10.1017/S1351324911000040 Published online by Cambridge University Press

the user's query against the *questions* in a large corpus of question-answer pairs and retrieves the associated answer.

In the information retrieval community, QA systems attempt to retrieve precise segments of a document instead of the entire document. In Tomuro and Lytinen (2004), the authors match the user's query against a frequently asked questions (FAQ) database and select the answer whose question matches most closely to the user's question. An extension of this idea is explored in Xue, Jeon and Croft (2008) and in Jeon, Croft and Lee (2005), where the authors match the user's query to a community collected QA archive such as Yahoo! (2009) and MSN-QnA (2009). Our approach is similar to both these lines of work in spirit, although the user's query for our system originates as a spoken query, in contrast to the text queries in previous work. We also address the issue of noisy speech recognition and assess the value of tight integration of speech recognition and search in terms of improving the overall performance of the system. A novelty in this paper is our method for addressing dynamic questions[1] as a seamless extension to answering static questions.

Also related is the literature on voice-search applications (Google 2009; Microsoft 2009; YellowPages 2009; vlingo.com 2009) that provide a spoken language interface to business directories and return phone numbers, addresses and web sites of businesses. User input is typically not a free flowing natural language query and is limited to expressions with a business name and a location. In our system, users can avail of the full range of natural language expressions to express their information need.

And finally, our method of retrieving answers to dynamic questions has relevance to the database and meta search community. There is growing interest in this community to mine the 'hidden' web – information repositories that are behind web forms – and provide a unified meta-interface to such information sources, for example, web sites related travel, or car dealerships. Dynamic questions can be seen as providing a natural language interface (NLI) to such web forms, similar to early work on NLI to databases (Androutsopoulos 1995).

3 Speech-driven question retrieval system

We describe the speech-driven query retrieval application (`Qme!`) in this section. The user of this application provides a spoken language query to a mobile device intending to find an answer to her question. User input could be either a short business listing query (e.g. *pizza hut near urbana illinois*) or a complete question (e.g. *how do I fix a leaky dishwasher*, *what movies are playing in los angeles california*).

As shown in Figure 1, we classify questions to `Qme!` into three categories and apply different retrieval methods.

(1) *Business listing questions:* These questions typically contains a business name, optionally followed by a city and state to indicate the location of the business, such as *pizza hut near urbana illinois*. User input with a business category

[1] The answer to a dynamic question changes based on the time when the question is asked.

https://doi.org/10.1017/S1351324911000040 Published online by Cambridge University Press

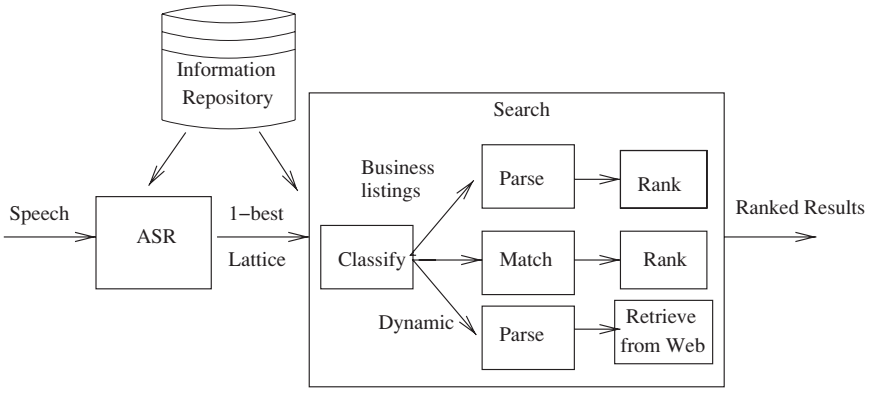


Fig. 1. The architecture of the speech-driven question–answering system.

(*laundromats in madison*) and without location information (*hospitals*) are some variants supported by this application. The result from speech recognition is parsed to identify the business name (aka SearchTerm) and the city/state segments that the user is interested in. The parsed information is used to search a business listing database of over 10 million entries to retrieve the entries pertinent to the user query.

(2) *Static questions:* These are general user queries that are not constrained to be of any specific question type such as *what, where, when, how* questions. The answers to these questions remain mostly the same irrespective of when and where the questions are asked. Some examples of such queries include *what is the fastest animal in water?*, *how do you make a bloody mary mix?*, *when is George Washington's birthday?*, *why is the sky blue?*. The result of the speech recognizer is used to search a large corpus of human generated question-answer pairs to retrieve the answers pertinent to the user's static questions. This ensures high accuracy for the answers retrieved (if found in the archive) and also allows us to retrieve related questions on the user's topic of interest.

(3) *Dynamic questions:* The answers to these questions depend on when and where they are asked. For such questions, the above method results in less than satisfactory and sometimes inaccurate answers. Examples of such questions are *what is the stock price of General Motors?*, *who won the baseball game last night?*, *what is playing at the theaters near me?* For the dynamic questions, the answers are retrieved by querying a web form from the appropriate web site (for example, www.fandango.com for movie information).

For all user queries, the result from the speech recognizer can be a single-best string or a weighted word lattice. The retrieved results are ranked using different metrics discussed in the next section. In Table 1, we show the results that the system returns for business listings query *audi automobiles*. In Figure 2, we illustrate the answers that the system returns for the static question *what are the symptoms of a heart attack* and the dynamic question *what movies are playing on sunday*. Note

https://doi.org/10.1017/S1351324911000040 Published online by Cambridge University Press

Table 1. *Listings retrieved for query* audi automobiles

Search Results for the query *audi automobiles*	
1	auburn audi repair auburn wa
2	audi bellevue repair bellevue wa
3	university audi seattle wa
4	beverly hills audi los angeles ca
5	audi independent repairs by eurotech livermore ca

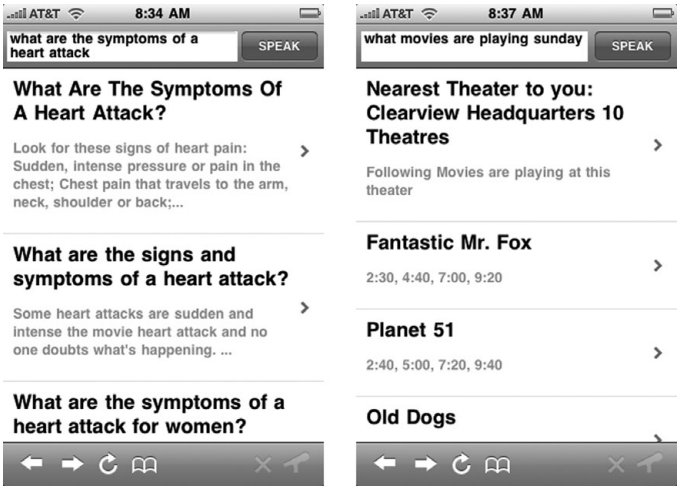


Fig. 2. Retrieval results for static and dynamic questions using `Qme!`.

that the system returns all the movie listings and timing information for shows on Sunday for the dynamic question.

4 Tightly coupling speech recognition and search

Most of the speech-driven search systems (Acero *et al.* 2008; Bacchiani *et al.* 2008; vlingo.com 2009) use the 1-best output from the speech recognizer (ASR) as the query for the search component. Given that ASR 1-best output is likely to be erroneous, this serialization of the ASR and search components might result in suboptimal search accuracy. As will be shown in our experiments, the oracle word/phrase accuracy using n-best hypotheses is far greater than the 1-best output. However, using each of the n-best hypothesis as a query to the search component is computationally suboptimal since the strings in the n-best hypotheses usually share large subsequences with each other. A lattice representation of the ASR output, in particular, a word-confusion network (WCN) transformation of the lattice, compactly encodes the n-best hypothesis with the flexibility of pruning alternatives at each word position. Two examples of WCNs are shown in Figures 3 and 4. The weights on the arcs are to be interpreted as costs and the best path in the WCN is the lowest cost path from the start state to the final state. Note that the 1-best path in Figure 4 is *how old is*

https://doi.org/10.1017/S1351324911000040 Published online by Cambridge University Press

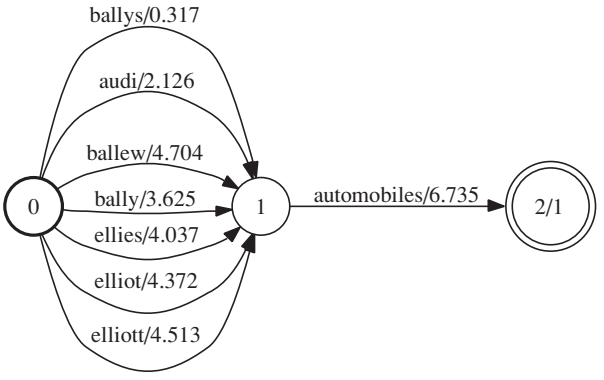


Fig. 3. A sample word confusion network for a business listing query.

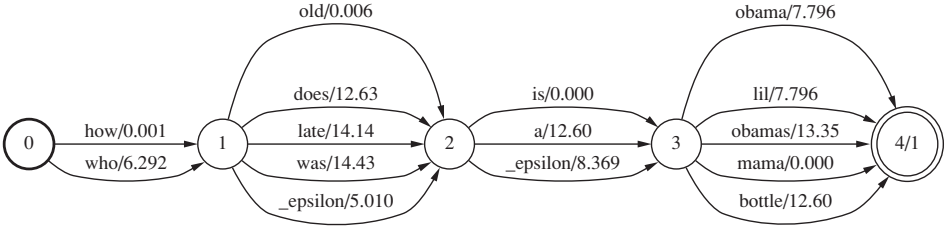


Fig. 4. A sample word confusion network with arc costs as negative logarithm of the posterior probabilities.

mama, while the input speech was *how old is obama* which also is in the WCN, but at a higher cost.

In order to obtain a measure of the ambiguity per word position in the WCN, we define the (average) *arc density* of a WCN as the ratio of the total number of arcs to the number of states in the WCN. As can be seen, with very small increase in arc density, the number of paths that are encoded in the WCN can be increased exponentially. In Figure 5, we show the improvement in oracle-path word and phrase accuracies as a function of the arc density for our data set. Oracle-path is a path in the WCN that has the least edit-distance (Levenshtein 1966) to the reference string. It is interesting to note that the oracle accuracies can be improved by almost 10% absolute over the 1-best accuracy with small increase in the arc density.

4.1 Representing search index as an FST

In order to exploit WCNs for Search, we have implemented our own search engine instead of using an off-the-shelf search engine such as Lucene (Hatcher and Gospodnetic 2004). We index each business listing (d) in our data that we intend to search using the words (w_d) in that listing. The pair (w_d, d) is assigned a weight ($c_{(w_d,d)}$) using different metrics, including the standard $tf * idf$, as explained below. This index is represented as a weighted finite-state transducer (*SearchFST*) as shown in Figure 6 where w_d is the input symbol, d is the output symbol and $c_{(w_d,d)}$ is the weight of that arc.

https://doi.org/10.1017/S1351324911000040 Published online by Cambridge University Press

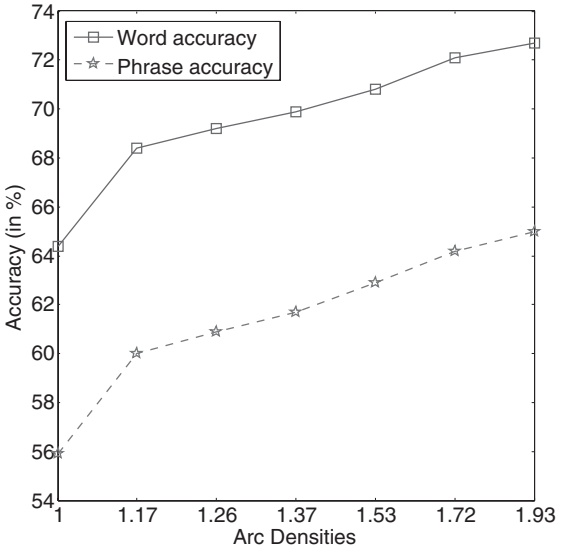


Fig. 5. Oracle accuracy graph for the WCNs at different arc densities.

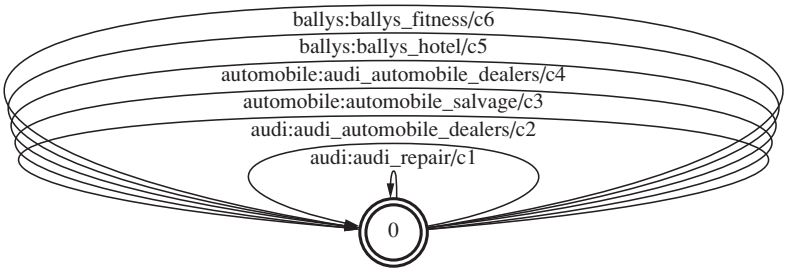


Fig. 6. An example of an FST representing the search index related to the business listings repository.

The FST search index for the question answer repository is built as follows. We index each question-answer (QA) pair from our repository ((q_i, a_i), qa_i for short) using the words (w_{q_i}) in question q_i. This index is represented as a weighted finite-state transducer (*SearchFST*) as shown in Figure 7. Here a word w_{q_i} (e.g *old*) is the input symbol for a set of arcs whose output symbol is the index of the QA pairs where *old* appears in the question. The weight of the arc $c_{(w_{q_i}, q_i)}$ is one of the similarity based weights discussed in Section 4.1. As can be seen from Figure 7, the words *how*, *old*, *is* and *obama* contribute a score to the question-answer pair *qa25*; while other pairs, *qa150, qa12, qa450* are scored by only one of these words.

4.2 Query relevance metrics

For a given user query, we retrieve documents (i.e. either business listings or question-answer pairs) from the information repository based on the similarity of match between the user's query and each of the documents (d) in the repository.

https://doi.org/10.1017/S1351324911000040 Published online by Cambridge University Press

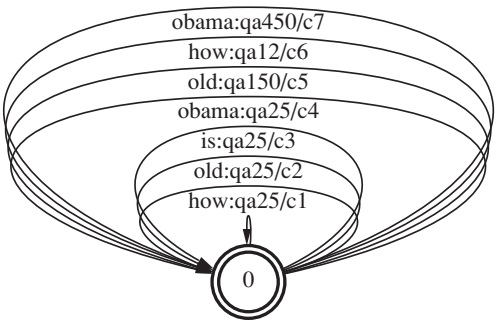


Fig. 7. An example of an FST representing the search index related to the QA repository.

The problem of answering user queries that are classified by the system as static questions is formulated as follows.[2] Given a question-answer archive $\mathbf{QA} = \{(q_1, a_1), (q_2, a_2), \ldots, (q_N, a_N)\}$ of N question-answer pairs, and a user's question q_u, the task is to retrieve a subset $\mathbf{QA^r} = \{(q_1^r, a_1^r), (q_2^r, a_2^r), \ldots, (q_M^r, a_M^r)\}$ $M \ll N$ using a selection function *Select* and rank the members of $\mathbf{QA^r}$ using a scoring function *Score* such that $Score(q_u, (q_i^r, a_i^r)) > Score(q_u, (q_{i+1}^r, a_{i+1}^r))$. Here, we assume $Score(q_u, (q_i^r, a_i^r)) = Score(q_u, q_i^r)$, that is, the relevance score of a question-answer pair to a user's question is approximated as the similarity score of the question in the corpus to the user's question.

The *Select* function is intended to select the matching questions that have high "semantic" similarity to the user's question. However, given there is no objective function that measures semantic similarity, we approximate it using different metrics discussed below.

Ranking of the members of the retrieved set can be based on the scores computed during the selection step or can be independently computed based on other criteria such as popularity of the question, credibility of the source, temporal recency of the answer, geographical proximity to the answer origin.

Semantic similarity between a user query that is a complete question and the question-answer pairs in our question-answer repository was measured using with the following metrics.

(1) *TF-IDF metric:* The user input query and the document (in our case, questions in the repository) are represented as bag-of-n-grams (aka terms). The term weights are computed using a combination of term frequency (tf) and inverse document frequency (idf) (Robertson 2004). If $Q = q_1, q_2, \ldots, q_n$ is a user query, then the aggregated score for a document d using a unigram model of the query and the document is given as in (1). For a given query, the documents with the highest total term weight are presented as retrieved results. Terms can also be defined as n-gram sequences of a query and a document. In our experiments, we have

[2] User queries that are classified as dynamic questions are answered by querying the Deep web using web forms.

https://doi.org/10.1017/S1351324911000040 Published online by Cambridge University Press

used up to 4-grams as terms to retrieve and rank documents.

$$Score(d) = \sum_{w \in Q} tf_{w,d} idf_w. \quad (1)$$

(2) *Levenshtein distance:* The Levenshtein distance (or edit distance) is the minimum number of operations needed to transform the query string into the document string, where an operation is an insertion, deletion, or substitution of a single word. In our evaluation, the penalty for insertions, deletions and substitutions is identical.

(3) *BLEU-score:* The Levenshtein metric overly penalizes differing but valid word orders between the query and document strings. The BLEU-score metric alleviates this penalty by considering a weighted geometric mean of the n-gram precisions (up to 4-grams) between the query and the document (Papineni *et al.* 2002), as shown in (2). Here p_n refers to the modified n-gram precision; w_n is weight for corresponding n-gram; and BP is the brevity penalty that penalizes length mismatches between a query and a document.

$$BLEU = BP \left(\sum_{n=1}^{N} w_n \ log \ p_n \right). \quad (2)$$

For queries relating to business listings, we experimented with the following six difference relevance metrics.

(1) idf_w: idf_w refers to the inverse document frequency of the word, w, which is computed as $ln(D/d_w)$, where D refers to the total number of documents in the collection, and d_w refers to the total number of documents in the collection that contain the word, w (Robertson and Jones 1997; Robertson 2004).

(2) atf_w: atf_w refers to average term frequency, which is computed as cf_w/d_w (Pirkola, Leppänen and Järvelin 2002).

(3) $cf_w \times idf_w$: Here cf_w refers to the collection frequency, which is simply the total number of occurrences of the word, w in the collection.

(4) $atf_w \times idf_w$: (Each term as described above).

(5) $\sum \frac{f_{w,d}}{|d_w|} \times idf_w$: Here $f_{w,d}$ refers to the frequency of the word, w, in the document, d, whereas $|d_w|$ is the length of the document, d, in which the word, w, occurs.

(6) $\frac{cf_w}{\sum |d_w|} \times idf_w$: (Each term as described above).

4.3 Search process using FSTs

Given a user's speech query, the process of obtaining search results is defined by (3) to (8). In these equations, $\circ$ indicates composition of two finite state transducers (or one transducer and one acceptor); and π_2 indicates that the output of the given transducer is projected (or preserved) to obtain a new automaton.

The user's speech query, after speech recognition (either 1-best or WCN), is represented as an FSA Q. The n-grams of Q and their counts are constructed using the function *CountNgrams* and the result is represented as another FSA (N_Q) as shown in (3). For this construction, we follow the algorithm presented in Allauzen,

https://doi.org/10.1017/S1351324911000040 Published online by Cambridge University Press

Mohri and Roark (2003). The result is that, if Q is an unweighted FSA with one path (e.g. a 1-best ASR output), N_Q contains the set of n-grams of Q as paths and the frequencies of those n-grams as the path weights. If Q is a weighted FSA (e.g. a WCN), the paths in the N_Q represent the set of n-grams in the WCN and the path weights correspond to the weighted frequencies of the n-grams of Q.

The weighted FSA N_Q is then composed with the *SearchFST* (4). For numerical stability, the weights are converted as negative logarithm scores, and the composition operation adds these scores. We retrieve the 1-best path (R_1) from the result of the composition D_1 (5). R_1 represents that particular question with the best score (a combination of acoustic, language model and search score) in the repository that also contains the n-grams in N_Q. We use R_1 as the rescored output from the ASR, compute its n-grams N_{R_1} (6) and compose N_{R_1} with the $SearchFST$ (7) to obtain all the arcs $(w_q, d_{w_q}, c_{(w_q,d_{w_q})})$ where w_q is a query term, d_{w_q} is a either a QA index or a business listing index containing the query term and, $c_{(w_q,d_{w_q})}$ is the weight associated with that pair. Using this information, we aggregate the weight for a document (which is either a QA pair or a business listing) (d_q) across all query words (using $fsmdeterminize$) and rank the retrieved QAs in the descending order of this aggregated weight.[3] We select the top N QA pairs (8) from this ranked list.

$$N_Q = CountNgrams(Q) \tag{3}$$

$$D_1 = \pi_2(N_Q \circ SearchFST) \tag{4}$$

$$R_1 = fsmbestpath(D_1, 1) \tag{5}$$

$$N_{R_1} = CountNgrams(R_1) \tag{6}$$

$$D_2 = \pi_2(N_{R_1} \circ SearchFST) \tag{7}$$

$$TopN = fsmbestpath(fsmdeterminize(D_2), N). \tag{8}$$

While the computational complexity of the n-gram counting is $O(|q|)$, where $|q|$ is the number of arcs in q, the composition operation on two FSTs is $O(m * n)$ where m is the number of states of the first FST and n is the number of states of the second FST. Determinization of FSTs in general is $O(2^p)$ where p is the number of states, however, in our approach, the FST $D2$ has $n+1$ states for an n-gram based model. We have built this model on over 30 million documents in the *SearchFST*, without any scalability problems. Extensions to this model using lazy evaluation techniques (Mohri, Pereira and Riley 1998) can be used in cases where scalability becomes an issue.

In Figure 8, we illustrate the result of reranking the WCN shown in Figure 3 using the search relevance weights of each word in the WCN. It must be noted that the least cost path[4] for the WCN in Figure 3 is *ballys automobiles* while the reranked 1-best output in Figure 8 is *audi automobiles*. Given that the user's query was *audi*

[3] The FSA is converted into the *real* semiring before $fsmdeteriminze$ to allow for aggregation of weights.

[4] We transform the scores into costs and search for minimum cost paths.

https://doi.org/10.1017/S1351324911000040 Published online by Cambridge University Press

Table 2. *Listings retrieved for query* audi automobiles *before and after ASR WCNs were rescored using search relevance weights*

Before rescoring	After rescoring
ballys intl	auburn audi repair
los angeles ca	auburn wa
ballys las vegas	audi bellevue repair
las vegas nv	bellevue wa
ballys las health spa	university audi seattle wa
las vegas nv	
ballys cleaners	beverly hills audi
palm desert ca	los angeles ca
ballys brothers	audi independent repairs
yorba linda ca	by eurotech livermore ca

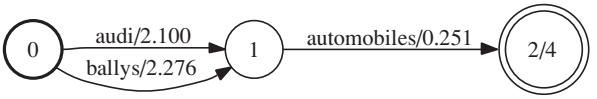


Fig. 8. A WCN rescored using word-level search relevance weights.

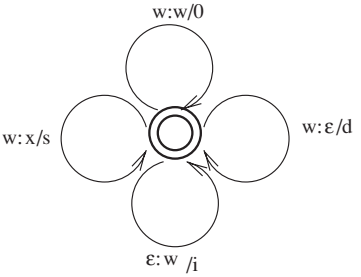


Fig. 9. An *EditFST* encoding the edit operations with substitution, deletion and insertion costs.

automobiles, the listings retrieved from the 1-best output after reranking are much more relevant than those retrieved before reranking, as shown in Table 2.

The process of retrieving documents using the Levenshtein-based string similarity metric can also be encoded as a composition of FSTs as shown in (9) and 10. The *SearchFST* encodes the corpus of question-answer pairs as shown in Figure 10. The *EditFST* (Figure 9) allows for substitution, insertion and deletion of terms and the composition operation ($\circ$) allows the matching of the user's query string to the questions in the corpus through edit operations.

$$D = \pi_2(Q \circ EditFST \circ SearchFST) \tag{9}$$

$$TopN = fsmbestpath(D, N) \tag{10}$$

https://doi.org/10.1017/S1351324911000040 Published online by Cambridge University Press

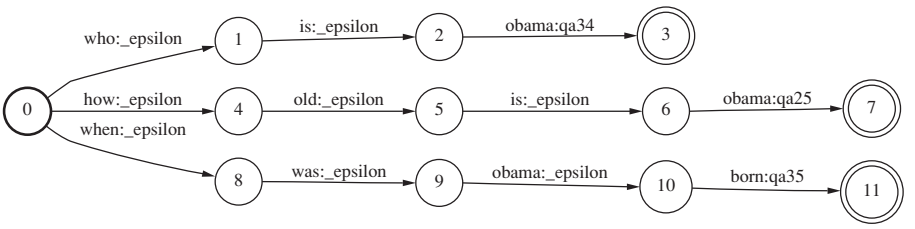


Fig. 10. An example of an FST representing the search index for static queries.

Table 3. *Performance of the metrics used for rescoring the WCNs output by ASR*

Metric	Word accuracy	Sentence accuracy
idf_w	63.1	57.0
$cf_w \times idf_w$	63.5	58.3
atf_w	63.6	57.3
$atf_w \times idf$	63.1	57.0
$\sum \frac{f_{w,d}}{\lvert df_w \rvert} \times idf$	63.9	58.3
$\frac{cf_w}{\sum \lvert df_w \rvert} \times idf_w$	63.5	57.3

5 Experiments and results

In this section, we present the results of our experiments on tight coupling the ASR and Search components. We show that we not only improve the relevance of search results, but we also improve the speech recognition accuracy. We illustrate this for both the business listing queries as well as the general static queries.

5.1 On business listings queries

We used 852 speech queries collected from users using a mobile device based speech search application. We ran the speech recognizer (Goffin *et al.* 2005) on these queries using a four-gram language model that is built from the corpus of user queries and created word-confusion networks such as those illustrated in Figures 3 and 4. These 852 utterances were divided into 300 utterances for the development set and 552 for the test set.

5.1.1 ASR experiments

The speech recognizer used to recognize these utterances incorporates an acoustic model adapted to speech collected from mobile devices and a four-gram language model that is built from the corpus of questions. The baseline ASR word and sentence (complete string) accuracies on the development set are 63.1% and 57.0% while those on the test set are 65.1% and 55.3%, respectively.

In Table 3, we summarize the improvements obtained by rescoring the ASR WCNs based on the different metrics used for computing the word scores according to the search criteria. The largest improvement in word and sentence accuracies is obtained

https://doi.org/10.1017/S1351324911000040 Published online by Cambridge University Press

Table 4. *The recall, precision and F-scores obtained from the baseline ASR output compared to those obtained by the reranked ASR output on the search task*

	Baseline ASR output	Reranked ASR output
Precision	70.8	72.8
Recall	72.8	74.2
F-Score	71.8	73.5

by using the rescoring metric: $\sum \frac{f_{w,d}}{|df_w|} \times idf$. The word-level accuracy improved from the baseline accuracy of 63.1–63.9% after rescoring while the sentence-level accuracy improved from 57.0% to 58.3%. Thus, this rescoring metric was used to rerank the 552 WCNs in the test set. After rescoring, on the test set, the word-level accuracy improved from 65.1% to 65.9% and sentence-level accuracy improved from 55.3% to 56.2%.

5.1.2 Search experiments

To analyze the Search accuracy of the baseline ASR output in comparison to the ASR output, reranked using the $\sum \frac{f_{w,d}}{|df_w|} \times idf$ reranking metric, we used each of the two sets of ASR outputs (i.e. baseline and reranked) as queries to our search engine, *SearchFST* (described in Section 3). For the search results produced by each set of queries, we computed the precision, recall, and F-score values of the listings retrieved against the listings retrieved by the set of human transcribed queries (*Reference*). The precision, recall, and F-scores for the baseline ASR output and the reranked ASR output, averaged across each set, is presented in Table 4. For the purposes of this experiment, we assume that the set returned by our *SearchFST* for the human transcribed set of queries is the reference search set. This is however an approximation for a human annotated search set and it does introduce a bias in the results. In on-going work, we are working on creating a query set annotated with human annotated relevancy judgments.

In Table 4, by comparing the search accuracy scores corresponding to the baseline ASR output to those corresponding to the reranked ASR output, we see that reranking the ASR output using the information repository improves both the precision and recall of the search.

5.2 On general static queries

We have a fairly large data set consisting of over a million question-answer pairs collected by harvesting the web. In order to evaluate the retrieval methods discussed earlier, we use two test sets of QA pairs: a *Seen* set of 450 QA pairs and an *Unseen* set of 645 QA pairs. The queries in the *Seen* set have an exact match with some question in the database, while the queries in the *Unseen* set may not match any

https://doi.org/10.1017/S1351324911000040 Published online by Cambridge University Press

question in the database exactly.[5] The questions in the *Unseen* set, however, like those in the *Seen* set, also have a human generated answer that is used in our evaluations.

For each query, we retrieve the 20 most relevant QA pairs, ranked in descending order of the value of the particular metric under consideration. However, depending on whether the user query is a seen or an unseen query, the evaluation of the relevance of the retrieved question-answer pairs is different as discussed below.[6]

5.2.1 Evaluation metrics

For the set of *Seen* queries, we evaluate the relevance of the retrieved top-20 question-answer pairs in two ways:

(1) *Retrieval Accuracy of Top-N results:* We evaluate whether the question that matches the user query exactly is located in the top-1, top-5, top-10, top-20 or not in top-20 of the retrieved questions.
(2) *Coherence metric:* We compute the coherence of the retrieved set as the mean of the BLEU-score between the input query and the set of top-5 retrieved questions. The intuition is that we do not want the top-5 retrieved QA pairs to distract the user by not being relevant to the user's query.[7]

For the set of *Unseen* queries, since there are no questions in the database that exactly match the input query, we evaluate the relevance of the top-20 retrieved question-answer pairs in the following way. For each of the 645 *Unseen* queries, we know the human-generated answer. We manually annotated each unseen query with the *Best-Matched* QA pair whose answer was the closest semantic match to the human-generated answer for that unseen query. We evaluate the position of the Best-Matched QA in the list of top twenty retrieved QA pairs for each retrieval method.

5.2.2 Results

On the *Seen* set of queries, as expected the retrieval accuracy scores for the various retrieval techniques performed exceedingly well. The unigram based *tf.idf* method retrieved 93% of the user's query in the first position, 97% in one of top-5 positions and 100% in one of top-10 positions. All the other retrieval methods retrieved the user's query in the first position for all the *Seen* queries (100% accuracy).

In Table 5, we tabulate the results of the Coherence scores for the top-5 questions retrieved using the different retrieval techniques for the *Seen* set of queries. Here,

[5] There may however be semantically matching questions.
[6] The reason it is not a recall and precision curve is that, for the 'seen' query set, the retrieval for the questions is a zero/one boolean accuracy. For the 'unseen' query set there is no perfect match with the input question in the query database, and so we determine the closeness of the questions based on the closeness of the answers. Coherence attempts to capture the homogeneity of the questions retrieved, with the assumption that the user might want to see similar questions as the returned results.
[7] There are approaches in statistical machine translation which work on optimizing the lattice minimum error rate (Macherey *et al.* 2008) which may be relevant to our metric.

https://doi.org/10.1017/S1351324911000040 Published online by Cambridge University Press

Table 5. *Coherence metric results for top-5 queries retrieved using different retrieval techniques for the* seen *set*

Method		Coherence metric for top-5 results
TF-IDF	unigram	61.58
	bigram	66.23
	trigram	66.23
	4-gram	69.74
BLEU-score		66.29
Levenshtein		67.36

Table 6. *Retrieval results for the unseen queries*

Method		Top-1	Top-20
TFIDF	Unigram	69.13	75.81
	Bigram	62.46	67.41
	Trigram	61.97	65.93
	4-gram	56.54	58.77
	WCN	70.12	78.52
Levenshtein		67.9	77.29
BLEU-score		72.0	75.31

the higher the *n*-gram the more coherent is the set of the results to the user's query. It is interesting to note that the BLEU-score and Levenshtein similarity driven retrieval methods do not differ significantly in their scores from the *n*-gram *tf.idf* based metrics.

In Table 6, we present the retrieval results using different methods on the *Unseen* queries. For 240 of the 645 unseen queries, the human expert found that there was no answer in the data repository that could be considered semantically equivalent to the human-generated response to that query. So, these 240 queries cannot be answered using the current database. For the remaining 405 unseen queries, over 60% have their Best-Matched question-answer pair retrieved in the top-1 position. We expect the coverage to improve considerably by increasing the size of the QA archive.

5.2.3 Speech-driven query retrieval

For this experiment, we use the speech utterances corresponding to the *Unseen* set as the test set. We use a different set of 250 *speech* queries as the development set. In Table 7, we show the Word and Sentence Accuracy measures for the best path in the WCN before and after the composition of *SearchFST* with the WCN on the development and test sets. We note that by integrating the constraints from the search index, the ASR accuracies can be improved by about 1% absolute.

https://doi.org/10.1017/S1351324911000040 Published online by Cambridge University Press

Table 7. *ASR accuracies of the best path before and after (in parentheses) the composition of SearchFST*

Set	# of utterances	Word accuracy	Sentence accuracy
Dev Set	250	77.1(78.2)	54(54)
Test Set	645	70.8(72.1)	36.7(37.1)

In order to investigate the impact of WCNs on search accuracy, we computed the search results by integrating the ASR WCNs with the *SearchFST* on the speech utterances of the *Unseen* set. As shown in line 5 of Table 6, the integration of the ASR WCNs with the *SearchFST* produces higher search accuracy (70.12/78.52) compared to ASR 1-best (69.13/75.81). This indicates that the WCNs have information that can be utilized effectively to improve Search retrieval accuracy.

5.2.4 Role of stop words

An important aspect of query-to-question search is the role of stop words. In traditional search, stop words in the query and the document are often filtered out prior to search. However, since our queries are short, directed questions about ten words long, filtering out the stop words leaves very little information content in the query. For example, two questions varying only in tense such as *Who is the president of the US?* versus *Who was the president of the US?* are quite different questions, however, filtering out the stop words leaves behind only the words *president* and *US* in each of the two queries; hence, both queries may return the same question-answer pairs without any regard to the time-dependency factor inherent in the question. Prompted by such pitfalls of stripping out stop words, we experimented with the relative weighting of stop words and content words. The relative weighting mechanism is only applicable to the unigram based models.

In this experiment, prior to search, we weighted each of the *Seen* query words using a weighting parameter β as shown below using a predefined list to identify stop words.

$$weight(w) = \beta \quad if \quad w \quad \notin stop\ word \tag{11}$$

$$weight(w) = 1 - \beta \quad if \quad w \quad \in \quad stop\ word. \tag{12}$$

The parameter β stayed constant for each of the 450 queries in the set of *Seen* queries.

With this differential weighting, we retrieved the top-20 question-answer pairs using the unigram *tf.idf* based search. We then computed the rank of the user's query in the list of top twenty retrieved questions. The top-1 retrieval accuracy is shown in the Figure 11 below as we vary β from 0 to 1.

To find the β value that produced the best results (i.e. positioned the location of the exact match as close to the top of the list of retrieved questions), we systematically varied the parameter β from 0 to 1.

https://doi.org/10.1017/S1351324911000040 Published online by Cambridge University Press

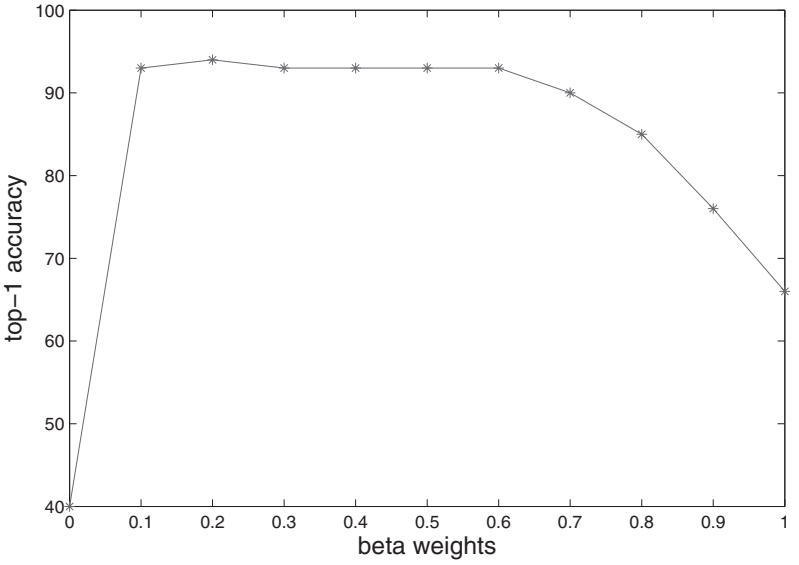


Fig. 11. The effect of varying the β value on *top-1* retrieval accuracy on the *Seen* data set.

5.3 Answering dynamic questions

Dynamic questions are questions whose answers vary based on the place and time of the question, for example, *what movies are playing tonight, where is the closest walmart, when is the next train to new york city*. In order to answer such questions, we first classify a question as either static or dynamic using semi-supervised methods. For the static questions, we use the finite-state transducer-based search technique described in the previous section. For the questions classified as dynamic, we first identify the topic of the question (for example, *movies*), and then we parse the query to obtain relevant parameters that are needed to query a content aggregator web site. For example, if the question is *what movies are playing in glendale California?*, we parse out the city and state information and use it to query a movie aggregator site like www.fandango.com. A detailed report of our work on answering dynamic questions is presented in Mishra and Bangalore (2010). The finite-state transducer-based parser is described in the following section.

6 Finite-state transducer-based parser

In this section, we present a method for parsing the segments of a user's utterance to identify the parameters to be passed to query the information repository. These methods work not only on 1-best ASR output but can transparently scale to take word lattices from ASR as input. We encode the problem of parsing as a weighted finite-state transducer (FST). This encoding allows us to apply the parser on ASR 1-best as well as ASR WCNs using the composition operation of FSTs.

We formulate the parsing problem as associating with each token of the input a label indicating whether that token belongs to one of a search term (*st*), location term (*lt*) or neither (*null*). Thus, given a word sequence ($W = w_1, \ldots, w_n$) output from

https://doi.org/10.1017/S1351324911000040 Published online by Cambridge University Press

Table 8. *A Sample set of annotated sentences*

1.	pizza_st hut_st new_lt york_lt new_lt york_lt
2.	home_st depot_st around_null san_lt francisco_lt
3.	please_null show_null me_null indian_st restaurants_st in_null chicago_lt
4.	pediatricians_st open_null on_null sundays_null
5.	hyatt_st regency_st in_null honolulu_lt hawaii_lt

ASR, we search of the most likely label sequence ($T = t_1, \ldots, t_n$), as shown in (13). We use the joint probability $P(W, T)$ and approximate it using an k-gram model as shown in (14) and (15). The sequence w_{i-1}^{i-k+1} represents the $k-1$ consecutive words of the history and t_{i-1}^{i-k+1} represents the $k-1$ consecutive tags of this history.

$$T^* = argmax_T \; P(T|W) \tag{13}$$

$$= argmax_T \; P(W, T) \tag{14}$$

$$= argmax_T \prod_i^n P(w_i, t_i \mid w_{i-1}^{i-k+1}, t_{i-1}^{i-k+1}) \tag{15}$$

A k-gram model can be encoded as a weighted finite-state acceptor (FSA; Allauzen *et al.* 2004). The states of the FSA correspond to the k-gram histories, the transition labels to the pair (w_i, t_i) and the weights on the arcs are $-log(P(w_i, t_i \mid w_{i-1}^{i-k+1}, t_{i-1}^{i-k+1}))$. The FSA also encodes back-off arcs for purposes of smoothing with lower order k-grams. An annotated corpus of words and labels is used to estimate the weights of the FSA. A sample corpus is shown in Table 8.

The FSA on the joint alphabet is converted into an FST. The paired symbols (w_i, t_i) are reinterpreted as consisting of an input symbol w_i and output symbol t_i. The resulting FST (M) is used to parse the 1-best ASR (represented as FSTs (I)), using composition of FSTs and a search for the lowest weight path, as shown in (16). The output symbol sequence (π_2) from the lowest weight path is T^*.

$$T^* = \pi_2(Bestpath(I \circ M)). \tag{16}$$

Equation (16) shows a method for parsing the 1-best ASR output using the FST. However, a similar method can be applied for parsing WCNs. The WCN arcs are associated with a posterior weight that needs to be scaled suitably to be comparable to the weights encoded in M. We represent the result of scaling the weights in WCN by a factor of λ as WCN^{λ}. The value of the scaling factor is determined empirically. Thus the process of parsing a WCN is represented by (17).

$$T^* = \pi_2(Bestpath(WCN^{\lambda} \circ M)). \tag{17}$$

7 Experiments

We have access to text query logs consisting of 18 million queries annotated with SearchTerm and LocationTerm labels. In addition to these logs, we have access to 11 million unique business listing names and their addresses. We use the combined data to train the parameters of the two parsing models as discussed in

https://doi.org/10.1017/S1351324911000040 Published online by Cambridge University Press

Table 9. *ASR performance on three data sets*

Data sets	Number of speech queries	Word accuracy
Test1	1484	70.1
Test2	544	82.9
Test3	658	77.3

Table 10. *SearchTerm extraction accuracy using the FST approach*

Data sets	SearchTerm Extraction accuracy			
Input	ASR 1-best	WCN	Oracle path 4	Transcription
Test1	56.9	57.4	65.6	92.2
Test2	69.5	71.0	81.9	98.0
Test3	59.2	60.6	69.3	96.1

Table 11. *LocationTerm extraction accuracy using the FST approach*

Data sets	LocationTerm extraction accuracy			
Input	ASR 1best	WCN	Oracle path 4	Transcription
Test1	79.8	79.8	83.8	95.1
Test2	89.4	89.4	92.7	98.5
Test3	87.1	87.1	89.3	97.3

the previous sections. We tested our approaches on three data sets, which in total include 2,686 speech queries. These queries were collected from users using mobile devices from different time periods. Labelers transcribed and annotated the test data using SearchTerm and LocationTerm tags.

We use an ASR with a trigram-based language model trained on the query logs. Table 9 shows the ASR word accuracies on the three data sets. The accuracy is the lowest on Test1, in which many users were non-native English speakers and a large percentage of queries are not intended for business search.

We measure the parsing performance in terms of extraction accuracy on the two non-filler slots: SearchTerm and LocationTerm. Extraction accuracy computes the percentage of the test set where the string identified by the parser for a slot is exactly the same as the annotated string for that slot.

In Tables 10 and 11, we report the parsing performance for the FST-based approach. We note that the FST-based parser on a WCN improves the SearchTerm and LocationTerm extraction accuracy over ASR 1-best, an improvement of about 1.5%. The performance under 'Oracle path' shows the upper bound for the parser using the

https://doi.org/10.1017/S1351324911000040 Published online by Cambridge University Press

Table 12. *Search performances using the FST approach*

Data sets		Precision	Recall	F1
ASR 1-best	Test1	71.6	64.3	67.8
	Test2	79.6	76.0	77.7
	Test3	72.9	67.2	70.0
WCN	Test1	70.5	64.7	67.5
	Test2	80.3	77.3	78.8
	Test3	72.9	68.1	70.3

oracle path[8] from the pruned WCN. We pruned WCN by only retaining arcs that are within *cthresh* (=4 in our experiments) of the lowest cost arc between two states.

We evaluated the impact of parsing performance on search accuracy. In order to measure search accuracy, we first collected a reference set of search results for our test utterances. For this purpose, we submitted the human annotated two-field data to the search engine (http://www.yellowpages.com/) and extracted the top 5 results from the returned pages. The returned search results are either business categories such as 'Chinese Restaurant' or business listings including business names and addresses. We considered these results as the reference search results for our test utterances.

In order to evaluate our voice search system, we submitted the two fields resulting from the query parser on the ASR output (1-best/WCN) to the search engine. We extracted the top 5 results from the returned pages and we computed the Precision, Recall and F1 scores between this set of results and the reference search set. In Table 12, we report the search performance. The overall improvement in search performance is not as large as the improvement in the slot accuracies between using ASR 1-best and WCNs.

8 Conclusions

In this paper, we have presented a novel approach to tightly couple speech recognition and search components of a voice-driven search system. We use finite-state models for both ASR and Search to achieve this tight integration. We have shown experimentally that both the accuracy of the speech recognition and the relevance of search results improve due to the co-constraints from the two modules. We have also illustrated a finite-state transducer approach to parsing users' utterances in order to extract terms that are relevant for querying a backend database in the context of dynamic questions.

References

Acero, A., Bernstein, N., Chambers, R., Ju, Y., Li, X., Odell, J., Nguyen, O. S. P., and Zweig , G. 2008. Live search for mobile: web services by voice on the cellphone. In *Proceedings of ICASSP 2008*, Las Vegas, NV, USA, pp. 5256–9.

[8] Oracle text string is the path in the WCN that is closest to the reference string in terms of Levenshtein edit distance.

https://doi.org/10.1017/S1351324911000040 Published online by Cambridge University Press

Allauzen, C., Mohri, M., Riley, M., and Roark , B. 2004. A generalized construction of speech recognition transducers. In *Proceedings of ICASSP 2004*, Montreal, Canada, pp. 761–4.

Allauzen, C., Mohri, M., and Roark, B. 2003. Generalized algorithms for constructing statistical language models. In *Proceedings of Association for Computational Linguistics*, Sapporo, Japan, pp. 40–7.

Androutsopoulos, L. 1995. Natural language interfaces to databases - an introduction. *Journal of Natural Language Engineering* **1**: 29–81.

Bacchiani, M., Beaufays, F., Schalkwyk, J., Schuster, M., and Strope, B. 2008. Deploying GOOG-411: early lesstons in data, measurement and testing. In *Proceedings of ICASSP 2008*, Las Vegas, NV, USA, pp. 5260–3.

Dumais, S., Banko, M., Brill, E., Lin, J., and Ng, A. 2002. Web question answering: is more always better? In *SIGIR '02: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, USA. ACM, pp. 291–8.

Goffin, V., Allauzen, C., Bocchieri, E., Hakkani-Tur, D., Ljolje, A., Parthasarathy, S., Rahim, M., Riccardi, G., and Saraclar, M. 2005. The AT&T WATSON Speech Recognizer. In *Proceedings of ICASSP 2005*, Philadelphia, PA.

Google 2009. http://www.google.com/mobile. Google.

Green, B., Wolf, A., Chomsky, C., and Laughery, K. 1961. Baseball, an automatic question answerer. In *Proceedings of the Western Joint Computer Conference*, Los Angeles, CA, pp. 219–24.

Hatcher, E., and Gospodnetic, O. 2004. *Lucene in Action (In Action series)*. Greenwich, CT, USA: Manning Publications Co.

Jeon, J., Croft, W. B., and Lee, J. H. 2005. Finding similar questions in large question and answer archives. In *CIKM '05: Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, New York, NY, USA, pp. 84–90. ACM.

Katz, B. 1997. Annotating the world wide web using natural language. In *Proceedings of the 5th RIAO Conference on Computer Assisted Information Searching on the Internet (RIAO)*, McGill University, Montreal, Quebec, Canada.

Levenshtein, V. 1966. Binary codes capable of correcting deletions, insertion and reversals. *Soviet Physics Doklady* **10**: 707–10.

Macherey, W., Och, F., Thayer, I., and Uszkoreit, J. 2008. Lattice-based minimum error rate training for statistical machine translation. In *Proceedings of Empirical Methods in Natural Language Processing*, Hawaii, USA, pp. 725– 34.

Maybury, M. T. 2004. *New Directions in Question Answering*. Menlo Park, CA, USA: The AAAI Press.

Microsoft 2009. http://www.live.com. Microsoft.

Mishra, T., and Bangalore, S. 2010. Qme! : a speech-based question-answering system on mobile devices. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Los Angeles, CA, USA, pp. 55–63.

Mohri, M., Pereira, F., and Riley, M. 1998. *A rational design for a weighted finite-state transducer library*, pp. 144–58. Number 1436 in Lecture notes in computer science. Berlin Heidelberg, Germany: Springer-Verlag.

MSN-QnA (2009). http://qna.live.com/. MSN-QnA.

Papineni, K., Roukos, S., Ward, T., and Zhu, W. 2002, July. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association of Computational Linguistics*, Philadelphia, PA, USA, pp. 313–18.

Pirkola, A., Leppänen, E., and Järvelin, K. 2002. The "RATF" formula (kwok's formula): exploiting average term frequency in cross-language retrieval. *Information Research* **7**(2).

Robertson, S. (2004). Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of Documentation* **60**: 503–20.

https://doi.org/10.1017/S1351324911000040 Published online by Cambridge University Press

Robertson, S. E., and Jones, K. S. 1997. Simple proven approaches to text retrieval. Technical report, Cambridge University, Cambridge, UK.

Tomuro, N., and Lytinen, S. L. 2004. Retrieval models and Q and A learning with FAQ files. In *New Directions in Question Answering*, pp. 183–202.

vlingo.com (2009). http://www.vlingomobile.com/downloads.html. vlingo.com.

Waldinger, R. J., Appelt, D. E., Dungan, J. L., Fry, J., Hobbs, J. R., Israel, D. J., Jarvis, P., Martin, D. L., Riehemann, S., Stickel, M. E., and Tyson, M. 2004. Deductive question answering from multiple resources. In *New Directions in Question Answering*, pp. 253–62.

Weizenbaum, J. 1966. ELIZA - a computer program for the study of natural language communication between man and machine. *Communications of the ACM* **1**: 36–45.

Winograd, T. 1972. *Understanding Natural Language*. New York, NY, USA: Academic Press.

Woods, W. A. 1973. Progress in natural language understanding - an application to lunar geology. In *Proceedings of American Federation of Information Processing Societies (AFIPS) Conference*, pp. 441–50.

Xue, X., Jeon, J., and Croft, W. B. 2008. Retrieval models for question and answer archives. In *SIGIR '08: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, USA. ACM, pp. 475–82.

Yahoo! (2009). http://answers.yahoo.com/. Yahoo!

Yang, F., Feng, J., and DiFabbrizio, G. 2006. A data driven approach to relevancy recognition for contextual question answering. In *Proceedings of the Interactive Question Answering Workshop at HLT-NAACL 2006*, New York, USA, pp. 33–40.

YellowPages 2009. http://www.speak4it.com. YellowPages.

https://doi.org/10.1017/S1351324911000040 Published online by Cambridge University Press