Speech-driven Access to the Deep Web on Mobile Devices

Taniya Mishra and **Srinivas Bangalore**
AT&T Labs - Research
180 Park Avenue
Florham Park, NJ 07932 USA.
{taniya,srini}@research.att.com.

Abstract

The Deep Web is the collection of information repositories that are not indexed by search engines. These repositories are typically accessible through web forms and contain dynamically changing information. In this paper, we present a system that allows users to access such rich repositories of information on mobile devices using spoken language.

1 Introduction

The World Wide Web (WWW) is the largest repository of information known to mankind. It is generally agreed that the WWW continues to significantly enrich and transform our lives in unprecedent ways. Be that as it may, the WWW that we encounter is limited by the information that is accessible through search engines. Search engines, however, do not index a large portion of WWW that is variously termed as the *Deep Web*, *Hidden Web*, or *Invisible Web*.

Deep Web is the information that is in proprietory databases. Information in such databases is usually more structured and changes at higher frequency than textual web pages. It is conjectured that the Deep Web is 500 times the size of the surface web. Search engines are unable to index this information and hence, unable to retrieve it for the user who may be searching for such information. So, the only way for users to access this information is to find the appropriate web-form, fill in the necessary search parameters, and use it to query the database that contains the information that is being searched for. Examples of such web forms include, movie, train and bus times, and airline/hotel/restaurant reservations.

Contemporaneously, the devices to access information have moved out of the office and home environment into the open world. The ubiquity of mobile devices has made information access an any time, any place activity. However, information access using text input on mobile devices is tedious and unnatural because of the limited screen space and the small (or soft) keyboards. 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. Filling web-forms using the small screens and tiny keyboards of mobile devices is neither easy nor quick.

In this paper, we present a system, Qme!, designed towards providing a spoken language interface to the Deep Web. In its current form, Qme! provides a unifed interface onn iPhone (shown in Figure 1) that can be used by users to search for *static* and *dynamic* questions. *Static* questions are questions whose answers to these questions remain the same irrespective of when and where the questions are asked. Examples of such questions are *What is the speed of light?*, *When is George Washington's birthday?*. For *static* questions, the system retrieves the answers from an archive of human generated answers to questions. This ensures higher 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.

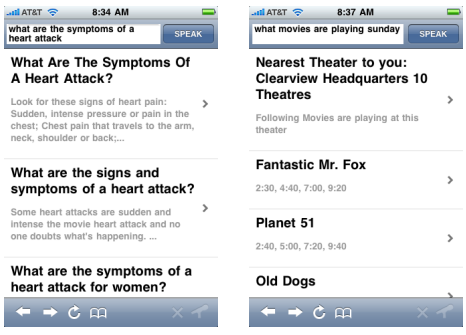

Figure 1: Retrieval results for static and dynamic questions using Qme!

Dynamic questions are questions whose answers depend on when and where they are asked. Examples of such questions are *What is the stock price of General Motors?*, *Who won the game last night?*, *What is playing at the theaters near me?*.

Proceedings of the ACL 2010 System Demonstrations, pages 60–65,
Uppsala, Sweden, 13 July 2010.

The answers to dynamic questions are often part of the Deep Web. Our system retrieves the answers to such dynamic questions by parsing the questions to retrieve pertinent search keywords, which are in turn used to query information databases accessible over the Internet using web forms. However, the internal distinction between dynamic and static questions, and the subsequent differential treatment within the system is seamless to the user. The user simply uses a single unified interface to ask a question and receive a collection of *answers* that potentially address her question directly.

The layout of the paper is as follows. In Section 2, we present the system architecture. In Section 3, we present bootstrap techniques to distinguish dynamic questions from static questions, and evaluate the efficacy of these techniques on a test corpus. In Section 4, we show how our system retrieves answers to dynamic questions. In Section 5, we show how our system retrieves answers to static questions. We conclude in Section 6.

2 Speech-driven Question Answer System

Speech-driven access to information has been a popular application deployed by many companies on a variety of information resources (Microsoft, 2009; Google, 2009; YellowPages, 2009; vlingo.com, 2009). In this prototype demonstration, we describe a speech-driven question-answer application. The system architecture is shown in Figure 2.

The user of this application provides a spoken language query to a mobile device intending to find an answer to the question. The speech recognition module of the system recognizes the spoken query. The result from the speech recognizer can be either a single-best string or a weighted word lattice.[1] This textual output of recognition is then used to classify the user query either as a dynamic query or a static query. If the user query is static, the result of the speech recognizer is used to search a large corpus of question-answer pairs to retrieve the relevant answers. The retrieved results are ranked using $tf.idf$ based metric discussed in Section 5. If the user query is dynamic, the answers are retrieved by querying a web form from the appropriate web site (e.g *www.fandango.com* for movie information). In Figure 1, we illustrate the answers that `Qme!` returns for static and dynamic questions.

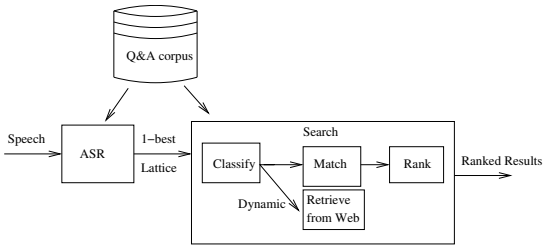

Figure 2: The architecture of the speech-driven question-answering system

2.1 Demonstration

In the demonstration, we plan to show the users static and dynamic query handling on an iPhone using spoken language queries. Users can use the iphone and speak their queries using an interface provided by `Qme!`. A Wi-Fi access spot will make this demonstation more compelling.

3 Dynamic and Static Questions

As mentioned in the introduction, dynamic questions require accessing the hidden web through a web form with the appropriate parameters. Answers to dynamic questions cannot be preindexed as can be done for static questions. They depend on the time and geographical location of the question. In dynamic questions, there may be no explicit reference to time, unlike the questions in the TERQAS corpus (Radev and Sundheim., 2002) which explicitly refer to the temporal properties of the entities being questioned or the relative ordering of past and future events.

The time-dependency of a dynamic question lies in the temporal nature of its answer. For example, consider the question, *What is the address of the theater White Christmas is playing at in New York?*. White Christmas is a seasonal play that plays in New York every year for a few weeks in December and January, but not necessarily at the same theater every year. So, depending when this question is asked, the answer will be different. If the question is asked in the summer, the answer will be "This play is not currently playing anywhere in NYC." If the question is asked during December, 2009, the answer might be different than the answer given in December 2010, because the theater at which White Christmas is playing differs from 2009 to 2010.

There has been a growing interest in temporal analysis for question-answering since the late 1990's. Early work on temporal expressions iden-

[1]For this paper, the ASR 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.

tification using a tagger culminated in the development of TimeML (Pustejovsky et al., 2001), a markup language for annotating temporal expressions and events in text. Other examples include, QA-by-Dossier with Constraints (Prager et al., 2004), a method of improving QA accuracy by asking auxiliary questions related to the original question in order to temporally verify and restrict the original answer. (Moldovan et al., 2005) detect and represent temporally related events in natural language using logical form representation. (Saquete et al., 2009) use the temporal relations in a question to decompose it into simpler questions, the answers of which are recomposed to produce the answers to the original question.

3.1 Question Classification: Dynamic and Static Questions

We automatically classify questions as dynamic and static questions. The answers to static questions can be retrieved from the QA archive. To answer dynamic questions, we query the database(s) associated with the topic of the question through web forms on the Internet. We first use a topic classifier to detect the topic of a question followed by a dynamic/static classifier trained on questions related to a topic, as shown in Figure 3. For the question *what movies are playing around me?*, we detect it is a movie related dynamic question and query a movie information web site (e.g. www.fandango.com) to retrieve the results based on the user's GPS information.

Dynamic questions often contain temporal indexicals, i.e., expressions of the form *today, now, this week, two summers ago, currently, recently*, etc. Our initial approach was to use such signal words and phrases to automatically identify dynamic questions. The chosen signals were based on annotations in TimeML. We also included spatial indexicals, such as *here* and other clauses that were observed to be contained in dynamic questions such as *cost of*, and *how much is* in the list of signal phrases. These signals words and phrases were encoded into a regular-expression-based recognizer.

This regular-expression based recognizer identified 3.5% of our dataset – which consisted of several million questions – as dynamic. The type of questions identified were *What is playing in the movie theaters tonight?*, *What is tomorrow's weather forecast for LA?*, *Where can I go to get Thai food near here?* However, random samplings of the same dataset, annotated by four independent human labelers, indicated that on average 13.5% of the dataset is considered dynamic. This shows that the temporal and spatial indexicals encoded as a regular-expression based recognizer is unable to identify a large percentage of the dynamic questions.

This approach leaves out dynamic questions that do not contain temporal or spatial indexicals. For example, *What is playing at AMC Loew's?*, or *What is the score of the Chargers and Dolphines game?*. For such examples, considering the tense of the verb in question may help. The last two examples are both in the present continuous tense. But verb tense does not help for a question such as *Who got voted off Survivor?*. This question is certainly dynamic. The information that is most likely being sought by this question is what is the name of the person who got voted off the TV show Survivor *most recently*, and not what is the name of the person (or persons) who have gotten voted off the Survivor at some point in the past.

Knowing the broad topic (such as movies, current affairs, and music) of the question may be very useful. It is likely that there may be many dynamic questions about movies, sports, and finance, while history and geography may have few or none. This idea is bolstered by the following analysis. The questions in our dataset are annotated with a broad topic tag. Binning the 3.5% of our dataset identified as dynamic questions by their broad topic produced a long-tailed distribution. Of the 104 broad topics, the top-5 topics contained over 50% of the dynamic questions. These top five topics were sports, TV and radio, events, movies, and finance.

Considering the issues laid out in the previous section, our classification approach is to chain two machine-learning-based classifiers: a topic classifier chained to a dynamic/static classifier, as shown in Figure 3. In this architecture, we build one topic classifier, but several dynamic/static classifiers, each trained on data pertaining to one broad topic.

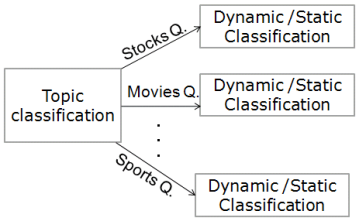

Figure 3: Chaining two classifiers

We used supervised learning to train the topic

classifier, since our entire dataset is annotated by human experts with topic labels. In contrast, to train a dynamic/static classifier, we experimented with the following three different techniques.

Baseline: We treat questions as dynamic if they contain temporal indexicals, e.g. *today*, *now*, *this week*, *two summers ago*, *currently*, *recently*, which were based on the TimeML corpus. We also included spatial indexicals such as *here*, and other substrings such as *cost of* and *how much is*. A question is considered static if it does not contain any such words/phrases.

Self-training with bagging: The general self-training with bagging algorithm (Banko and Brill, 2001). The benefit of self-training is that we can build a better classifier than that built from the small seed corpus by simply adding in the large unlabeled corpus without requiring hand-labeling.

Active-learning: This is another popular method for training classifiers when not much annotated data is available. The key idea in active learning is to annotate only those instances of the dataset that are most difficult for the classifier to learn to classify. It is expected that training classifiers using this method shows better performance than if samples were chosen randomly for the same human annotation effort.

We used the maximum entropy classifier in LLAMA (Haffner, 2006) for all of the above classification tasks. We have chosen the active learning classifier due to its superior performance and integrated it into the `Qme!` system. We provide further details about the learning methods in (Mishra and Bangalore, 2010).

3.2 Experiments and Results

3.2.1 Topic Classification

The topic classifier was trained using a training set consisting of over one million questions downloaded from the web which were manually labeled by human experts as part of answering the questions. The test set consisted of 15,000 randomly selected questions. Word trigrams of the question are used as features for a MaxEnt classifier which outputs a score distribution on all of the 104 possible topic labels. The error rate results for models selecting the top topic and the top two topics according to the score distribution are shown in Table 1. As can be seen these error rates are far lower than the baseline model of selecting the most frequent topic.

Model	Error Rate
Baseline	98.79%
Top topic	23.9%
Top-two topics	12.23%

Table 1: Results of topic classification

3.2.2 Dynamic/static Classification

As mentioned before, we experimented with three different approaches to bootstrapping a dynamic/static question classifier. We evaluated these methods on a 250 question test set drawn from the broad topic of *Movies*. The error rates are summarized in Table 2. We provide further details of this experiment in (Mishra and Bangalore, 2010).

Training approach	Lowest Error rate
Baseline	27.70%
"Supervised" learning	22.09%
Self-training	8.84%
Active-learning	4.02%

Table 2: Best Results of dynamic/static classification

4 Retrieving answers to dynamic questions

Following the classification step outlined in Section 3.1, we know whether a user query is static or dynamic, and the broad category of the question. If the question is dynamic, then our system performs a vertical search based on the broad topic of the question. In our system, so far, we have incorporated vertical searches on three broad topics: Movies, Mass Transit, and Yellow Pages.

For each broad topic, we have identified a few trusted content aggregator websites. For example, for dynamic questions related to Movies-related dynamic user queries, `www.fandango.com` is a trusted content aggregator website. Other such trusted content aggregator websites have been identified for Mass Transit related and for Yellow-pages related dynamic user queries. We have also identified the web-forms that can be used to search these aggregator sites and the search parameters that these web-forms need for searching. So, given a user query, whose broad category has been determined and which has been classified as a dynamic query by the system, the next step is to parse the query to obtain pertinent search parameters.

The search parameters are dependent on the broad category of the question, the trusted content aggregator website(s), the web-forms associated with this category, and of course, the content

of the user query. From the search parameters, a search query to the associated web-form is issued to search the related aggregator site. For example, for a movie-related query, *What time is Twilight playing in Madison, New Jersey?*, the pertinent search parameters that are parsed out are *movie-name: Twilight*, *city: Madison*, and *state: New Jersey*, which are used to build a search string that Fandango's web-form can use to search the Fandango site. For a yellow-pages type of query, *Where is the Saigon Kitchen in Austin, Texas?*, the pertinent search parameters that are parsed out are *business-name: Saigon Kitchen*, *city: Austin*, and *state: Texas*, which are used to construct a search string to search the Yellowpages website. These are just two examples of the kinds of dynamic user queries that we encounter. Within each broad category, there is a wide variety of the sub-types of user queries, and for each sub-type, we have to parse out different search parameters and use different web-forms. Details of this extraction are presented in (Feng and Bangalore, 2009).

It is quite likely that many of the dynamic queries may not have all the pertinent search parameters explicitly outlined. For example, a mass transit query may be *When is the next train to Princeton?*. The bare minimum search parameters needed to answer this query are a *from-location*, and a *to-location*. However, the *from-location* is not explicitly present in this query. In this case, the *from-location* is inferred using the GPS sensor present on the iPhone (on which our system is built to run). Depending on the web-form that we are querying, it is possible that we may be able to simply use the latitude-longitude obtained from the GPS sensor as the value for the *from-location* parameter. At other times, we may have to perform an intermediate latitude-longitude to city/state (or zip-code) conversion in order to obtain the appropriate search parameter value.

Other examples of dynamic queries in which search parameters are not explicit in the query, and hence, have to be deduced by the system, include queries such as *Where is XMen playing?* and *How long is Ace Hardware open?*. In each of these examples, the user has not specified a location. Based on our understanding of natural language, in such a scenario, our system is built to assume that the user wants to find a movie theatre (or, is referring to a hardware store) *near* where he is currently located. So, the system obtains the user's location from the GPS sensor and uses it to search for a theatre (or locate the hardware store) within a five-mile radius of her location.

In the last few paragraphs, we have discussed how we search for answers to dynamic user queries from the hidden web by using web-forms. However, the search results returned by these web-forms usually cannot be displayed as is in our Qme! interface. The reason is that the results are often HTML pages that are designed to be displayed on a desktop or a laptop screen, not a small mobile phone screen. Displaying the results as they are returned from search would make readability difficult. So, we parse the HTML-encoded result pages to get just the answers to the user query and reformat it, to fit the Qme! interface, which is designed to be easily readable on the iPhone (as seen in Figure 1).[2]

5 Retrieving answers to static questions

Answers to static user queries – questions whose answers do not change over time – are retrieved in a different way than answers to dynamic questions. A description of how our system retrieves the answers to static questions is presented in this section.

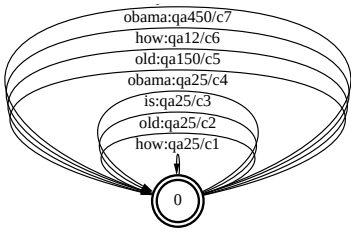

Figure 4: An example of an FST representing the search index.

5.1 Representing Search Index as an FST

To obtain results for static user queries, we have implemented our own search engine using finite-state transducers (FST), in contrast to using Lucene (Hatcher and Gospodnetic., 2004) as it is a more efficient representation of the search index that allows us to consider word lattices output by ASR as input queries.

The FST search index 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 4. 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

[2]We are aware that we could use SOAP (Simple Object Access Protocol) encoding to do the search, however not all aggregator sites use SOAP yet.

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 4, 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.

5.2 Search Process using FSTs

A user's speech query, after speech recognition, is represented as a finite state automaton (FSA, either 1-best or WCN), *QueryFSA*. The *QueryFSA* is then transformed into another FSA (*NgramFSA*) that represents the set of n-grams of the *QueryFSA*. In contrast to most text search engines, where stop words are removed from the query, we weight the query terms with their *idf* values which results in a weighted *NgramFSA*. The *NgramFSA* is composed with the *SearchFST* and we obtain all the arcs $(w_q, qa_{w_q}, c_{(w_q,qa_{w_q})})$ where w_q is a query term, qa_{w_q} is a QA index with the query term and, $c_{(w_q,qa_{w_q})}$ is the weight associated with that pair. Using this information, we aggregate the weight for a QA pair (qa_q) across all query words and rank the retrieved QAs in the descending order of this aggregated weight. We select the top N QA pairs from this ranked list. The query composition, QA weight aggregation and selection of top N QA pairs are computed with finite-state transducer operations as shown in Equations 1 and 2[3]. An evaluation of this search methodology on word lattices is presented in (Mishra and Bangalore, 2010).

$$D = \pi_2(NgramFSA \circ SearchFST) \quad (1)$$

$$TopN = fsmbestpath(fsmdeterminize(D), N) \quad (2)$$

6 Summary

In this demonstration paper, we have presented `Qme!`, a speech-driven question answering system for use on mobile devices. The novelty of this system is that it provides users with a single unified interface for searching both the visible and the hidden web using the most natural input modality for use on mobile phones – spoken language.

7 Acknowledgments

We would like to thank Junlan Feng, Michael Johnston and Mazin Gilbert for the help we received in putting this system together. We would also like to thank ChaCha for providing us the data included in this system.

[3]We have dropped the need to convert the weights into the *real* semiring for aggregation, to simplify the discussion.

References

M. Banko and E. Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th annual meeting of the association for computational linguistics: ACL 2001*, pages 26–33.

J. Feng and S. Bangalore. 2009. Effects of word confusion networks on voice search. In *Proceedings of EACL-2009*, Athens, Greece.

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

P. Haffner. 2006. Scaling large margin classifiers for spoken language understanding. *Speech Communication*, 48(iv):239–261.

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

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

T. Mishra and S. Bangalore. 2010. Qme!: A speech-based question-answering system on mobile devices. In *Proceedings of NAACL-HLT*.

D. Moldovan, C. Clark, and S. Harabagiu. 2005. Temporal context representation and reasoning. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 1009–1104.

J. Prager, J. Chu-Carroll, and K. Czuba. 2004. Question answering using constraint satisfaction: Qa-by-dossier-with-contraints. In *Proceedings of the 42nd annual meeting of the association for computational linguistics: ACL 2004*, pages 574–581.

J. Pustejovsky, R. Ingria, R. Saurí, J. Casta no, J. Littman, and R. Gaizauskas., 2001. *The language of time: A reader*, chapter The specification languae – TimeML. Oxford University Press.

D. Radev and B. Sundheim. 2002. Using timeml in question answering. Technical report, Brandies University.

E. Saquete, J. L. Vicedo, P. Martínez-Barco, R. Mu noz, and H. Llorens. 2009. Enhancing qa systems with complex temporal question processing capabilities. *Journal of Artificial Intelligence Research*, 35:775–811.

vlingo.com, 2009. *http://www.vlingomobile.com/downloads.html*.

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