Qme! : A Speech-based Question-Answering system on Mobile Devices

Taniya Mishra
AT&T Labs-Research
180 Park Ave
Florham Park, NJ
taniya@research.att.com

Srinivas Bangalore
AT&T Labs-Research
180 Park Ave
Florham Park, NJ
srini@research.att.com

Abstract

Mobile devices are becoming the dominant mode of information access despite being cumbersome to input text using small keyboards and browsing web pages on small screens. We present `Qme!`, a speech-based question-answering system that allows for spoken queries and retrieves *answers* to the questions instead of web pages. We present bootstrap methods to distinguish dynamic questions from static questions and we show the benefits of tight coupling of speech recognition and retrieval components of the system.

1 Introduction

Access to information has moved from desktop and laptop computers in office and home environments to be an any place, any time activity due to mobile devices. Although mobile devices have small keyboards that make typing text input cumbersome compared to conventional desktop and laptops, the ability to access unlimited amount of information, almost everywhere, through the Internet, using these devices have made them pervasive.

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. We address this issue by allowing the user to query an information repository using speech. We expect that spoken language queries to be a more natural and less cumbersome way of information access using mobile devices.

A second issue we address is related to directly and precisely answering the user's query beyond serving web pages. This is in contrast to the current approach where a user types in a query using keywords to a search engine, browses the returned results on the small screen to select a potentially relevant document, suitably magnifies the screen to view the document and searches 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.

We present a speech-driven question answering system, `Qme!`, as a solution toward addressing these two issues. 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 address the information need directly. For a subclass of questions that we term *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. For a second subclass of questions that we term *dynamic* questions, the system retrieves the answer from information databases accessible over the Internet using web forms.

The layout of the paper is as follows. In Section 2, we review the related literature. In Section 3, we illustrate the system for speech-driven question answering. We present the retrieval methods we used to implement the system in Section 4. In Section 5, we discuss and evaluate our approach to tight coupling of speech recognition and search components. In Section 6, we present bootstrap techniques to distinguish dynamic questions from static questions, and evaluate the efficacy of these techniques on a test corpus. We conclude in Section 7.

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 et al., 2006). In contrast to these approaches, our method matches 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 et al., 2008; Jeon et al., 2005), where the authors match the user's query to a community collected QA archive such as (Yahoo!, 2009; 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 to address dynamic questions as a seamless extension to answering static questions.

Also related is the literature on voice-search applications (Microsoft, 2009; Google, 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 in this section. The user of this application provides a spoken language query to a mobile device intending to find an answer to the question. Some example users' inputs are[1] *what is the fastest animal in water*, *how do I fix a leaky dishwasher*, *why is the sky blue*. The result of the speech recognizer is used to search a large corpus of question-answer pairs to retrieve the answers pertinent to the user's static questions. For the dynamic questions, the answers are retrieved by querying a web form from the appropriate web site (e.g www.fandango.com for movie information). The result from the speech recognizer can be a single-best string or a weighted word lattice.[2] The retrieved results are ranked using different metrics discussed in the next section. In Figure 2, we illustrate the answers that `Qme!` returns for static and dynamic quesitons.

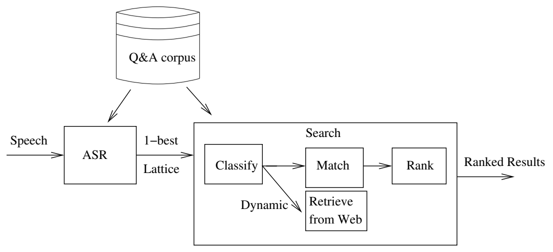


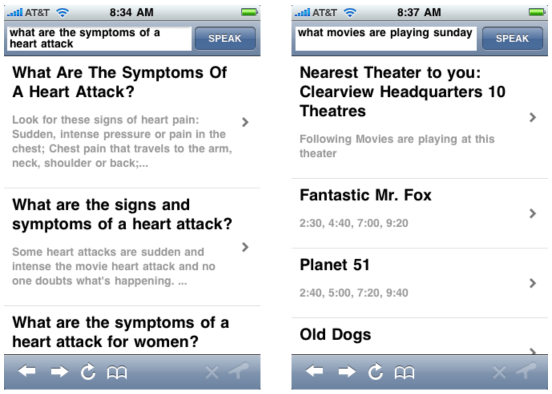
Figure 1: The architecture of the speech-driven question-answering system

4 Methods of Retrieval

We formulate the problem of answering static questions as follows. Given a question-answer archive $\mathbf{QA} = \{(q_1, a_1), (q_2, a_2), \ldots, (q_N, a_N)\}$

[1]The query is not constrained to be of any specific question type (for example, *what, where, when, how*).

[2]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.


AT&T 8:34 AM

what are the symptoms of a heart attack

SPEAK

What Are The Symptoms Of A Heart Attack?

Look for these signs of heart pain: Sudden, intense pressure or pain in the chest; Chest pain that travels to the arm, neck, shoulder or back;...

What are the signs and symptoms of a heart attack?

Some heart attacks are sudden and intense the movie heart attack and no one doubts what's happening. ...

What are the symptoms of a heart attack for women?

AT&T 8:37 AM

what movies are playing sunday

SPEAK

Nearest Theater to you: Clearview Headquarters 10 Theatres

Following Movies are playing at this theater

Fantastic Mr. Fox

2:30, 4:40, 7:00, 9:20

Planet 51

2:40, 5:00, 7:20, 9:40

Old Dogs

Figure 2: Retrieval results for static and dynamic questions using `Qme!`

of N question-answer pairs, and a user's question q_u, the task is to retrieve a subset $\mathbf{QA^r} = \{(q_1^r, a_1^r), (q_2^r, a_2^r), \ldots, (q_M^r, a_M^r)\}$ $M << N$ 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
$Score(q_u, (q_i^r, a_i^r)) = Score(q_u, q_i^r)$.

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.

4.1 Question Retrieval Metrics

We retrieve QA pairs from the data repository based on the similarity of match between the user's query and each of the set of questions (d) in the repository. To measure the similarity, we have experimented 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 Equation 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 used up to 4-grams as terms to retrieve and rank documents.

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

2. **String Comparison Metrics:** Since the length of the user query and the query to be retrieved are similar in length, we use string comparison methods such as Levenshtein edit distance (Levenshtein, 1966) and n-gram overlap (BLEU-score) (Papineni et al., 2002) as similarity metrics.

We compare the search effectiveness of these similarity metrics in Section 5.3.

5 Tightly coupling ASR and Search

Most of the speech-driven search systems use the 1-best output from the 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. 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. An example of a WCN is shown in Figure 3. 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 (0) to the final state (4). Note that the 1-best path is *how old is mama*, while the input speech was *how old is obama* which also is in the WCN, but at a higher cost.

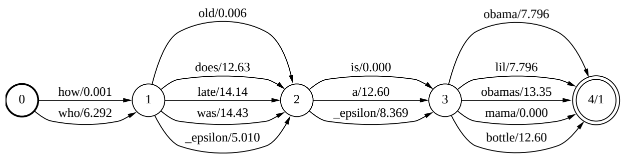


Figure 3: A sample word confusion network with arc costs as negative logarithm of the posterior probabilities.

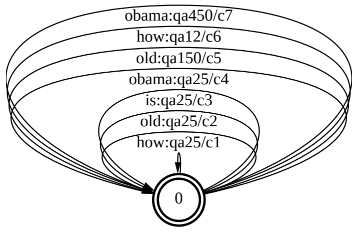


Figure 4: Example of an FST representing the search index.

5.1 Representing Search Index as an FST

Lucene (Hatcher and Gospodnetic., 2004) is an off-the-shelf search engine that implements the TF-IDF metric. But, we have implemented our own search engine using finite-state transducers (FST) for this reason. The oracle word/phrase accuracy using n-best hypotheses of an ASR is usually far greater than the 1-best output. However, using each of the n-best ($n > 1$) hypothesis as a separate query to the search component is computationally sub-optimal since the strings in the n-best hypotheses usually share large subsequences with each other. The FST representation of the search index allows us to efficiently consider lattices/WCNs 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 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 an FSA (either 1-best or WCN), a *QueryFSA*. The *QueryFSA* (denoted as q) is then transformed into another FSA (*NgramFSA(q)*) that represents the set of n-grams of the *QueryFSA*. Due to the arc costs from WCNs, the *NgramFSA* for a WCN is a weighted FSA. 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 2 to 5.[3]

$$D1 = \pi_2(NgramFSA(q) \circ SearchFST) \quad (2)$$

$$R1 = fsmbestpath(D1, 1) \quad (3)$$

$$D2 = \pi_2(NgramFSA(R1) \circ SearchFST) \quad (4)$$

$$TopN = fsmbestpath(fsmdeterminize(D2), N) \quad (5)$$

The process of retrieving documents using the Levenshtein-based string similarity metric can also be encoded as a composition of FSTs.

5.3 Experiments and Results

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 question in the database exactly. [4] 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 twenty 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.[5]

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

[4]There may however be semantically matching questions.

[5]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 homogenity of the questions retrieved, with the assumption that the user might want to see similar questions as the returned results.

5.3.1 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.

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.

5.3.2 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 1, 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, 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.

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

In Table 2, we present the retrieval results using different methods on the *Unseen* queries. For 240 of the 645 unseen queries, the human expert found that 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.

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

Table 2: Retrieval results for the Unseen queries

5.3.3 Speech-driven query retrieval

In Equation 6, we show the tight integration of WCNs and *SearchFST* using the FST composition operation ($\circ$). λ is used to scale the weights[6] from the acoustic/language models on the WCNs against the weights on the *SearchFST*. As before, we use Equation 3 to retrieve the top N QA pairs. The tight integration is expected to improve both the ASR and Search accuracies by co-constraining both components.

$$D = \pi_2(Unigrams(WCN)^{\lambda} \circ SearchFST) \quad (6)$$

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

[6]fixed using the development set

development set. In Table 3, 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.

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)

Table 3: ASR accuracies of the best path before and after (in parenthesis) the composition of SearchFST

Since we have the speech utterances of the *Unseen* set, we were also able to compute the search results obtained by integrating the ASR WCNs with the *SearchFST*, as shown in line 5 of Table 2. These results show that the the integration of the ASR WCNs with the *SearchFST* produces higher search accuracy compared to ASR 1-best.

6 Dynamic and Static Questions

Storing previously answered questions and their answers allows `Qme!` to retrieve the answers to a subclass of questions quickly and accurately. We term this subclass as *static* questions since the 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?*. In contrast, there is a subclass of questions, which we term *dynamic* questions, for which the answers 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 game last night?*, *What is playing at the theaters near me?*.

We define *dynamic* questions as questions whose answers change more frequently than once a year. 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 dynamic 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 it does not necessarily at the same theater every year. So, depending when this question is asked, the answer will be different.

Interest in temporal analysis for question-answering has been growing since the late 1990's. Early work on temporal expressions identification using a tagger led to 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.

6.1 Dynamic/Static Classification

We automatically classify questions as dynamic and static questions. 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 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 5. Given 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.

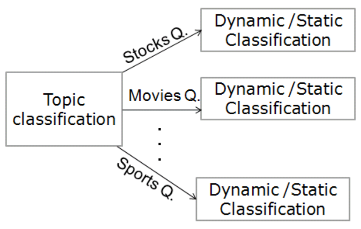

Topic classification
Stocks Q.
Movies Q.
Sports Q.
Dynamic /Static Classification
Dynamic /Static Classification
Dynamic /Static Classification

Figure 5: Chaining two classifiers

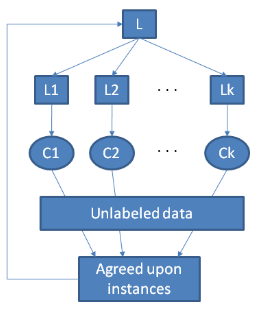
We used supervised learning to train the topic classifier, since our entire dataset is annotated by human experts with topic labels. In contrast, to train a dynamic/static classifier, we experimented with the following three different techniques.

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.

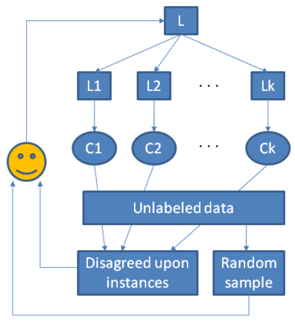
Self-training with bagging: The general self-training with bagging algorithm (Banko and Brill, 2001) is presented in Table 6 and illustrated in Figure 7(a). The benefit of self-training is that we can build a better classifier than that built from the small seed corpus by simply adding in the large unlabeled corpus without requiring hand-labeling.

1. Create k bags of data, each of size $|L|$, by sampling with replacement from labeled set L.
2. Train k classifiers; one classifier on each of k bags.
3. Each classifier predicts labels of the unlabeled set.
4. The N labeled instances that j of k classifiers agree on with the highest average confidence is added to the labeled set L, to produce a new labeled set L'.
5. Repeat all 5 steps until stopping criteria is reached.

Figure 6: Self-training with bagging



(a)



(b)

Figure 7: (a) Self-training with bagging (b) Committee-based active-learning

In order to prevent a bias towards the majority class, in step 4, we ensure that the distribution of the static and dynamic questions remains the same as in the annotated seed corpus. The benefit of bagging (Breiman, 1996) is to present different views of the same training set, and thus have a way to assess the certainty with which a potential training instance can be labeled.

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. Figure 7(b) illustrates the algorithm and Figure 8 describes the algorithm, also known as committee-based active-learning (Banko and Brill, 2001).

1. Create k bags of data, each of size $|L|$, by sampling with replacement from the labeled set L.
2. Train k classifiers, one on each bag of the k bags.
3. Each classifier predicts the labels of the unlabeled set.
4. Choose N instances from the unlabeled set for human labeling. $N/2$ of the instances are those whose labels the committee of classifiers have highest vote entropy (uncertainity). The other $N/2$ of the instances are selected randomly from the unlabeled set.
5. Repeat all 5 steps until stopping criteria is reached.

Figure 8: Active Learning algorithm

We used the maximum entropy classifier in Llama (Haffner, 2006) for all of the above classification tasks.

6.2 Experiments and Results

6.2.1 Topic Classification

The topic classifier was trained using a training set consisted 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 4. 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 4: Results of topic classification

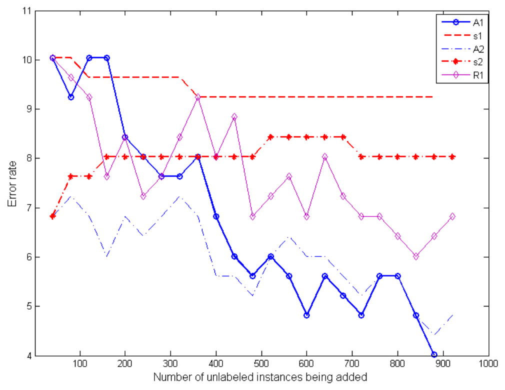


Figure 9: Change in classification results

6.2.2 Dynamic/static Classification

As mentioned before, we experimented with three different approaches to bootstrapping a dynamic/static question classifier. We evaluate these methods on a 250 question test set drawn from the broad topic of *Movies*. For the baseline model, we used the words/phrases discussed earlier based on temporal and spatial indexicals. For the "supervised" model, we use the baseline model to tag 500K examples and use the machine-annotated corpus to train a MaxEnt binary classifier with word trigrams as features. The error rate in Table 5 shows that it performs better than the baseline model mostly due to better lexical coverage contributed by the 500K examples.

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

Table 5: Best Results of dynamic/static classification

In the *self-training* approach, we start with a small seed corpus of 250 hand-labeled examples from the *Movies* topic annotated with dynamic or static tags. We used the same set of 500K unlabeled examples as before and word trigrams from the question were used as the features for a MaxEnt classifier. We used 11 bags in the *bagging* phase of this approach and required that all 11 classifiers agree unanimously about the label of a new instance. Of all such instances, we randomly selected N instances to be added to the training set of the next iteration, while maintaining the distribution of the static and dynamic questions to be the same as that in the seed corpus. We experimented with various values of N, the number of newly labeled instances added at each iteration. The error rate at initialization is 10.4% compared to 22.1% of the "supervised" approach which can be directly attributed to the 250 hand-labeled questions. The lowest error rate of the self-training approach, obtained at N=100, is 8.84%, as shown in Table 5. In Figure 9, we show the change in error rate for N=40 (line S1 in the graph) and N=100 (line S2 in the graph).

For the *active learning* approach, we used the same set of 250 questions as the seed corpus, the same set of 500K unlabeled examples, the same test set, and the same set of word trigrams features as in the self-training approach. We used 11 bags for the bagging phase and selected top 20 new unlabeled instances on which the 11 classifiers had the greatest vote entropy to be presented to the human labeler for annotation. We also randomly selected 20 instances from the rest of the unlabeled set to be presented for annotation. The best error rate of this classifier on the test set is 4.02%, as shown in Table 5. The error rate over successive iterations is shown by line A1 in Figure 9.

In order to illustrate the benefits of selecting the examples *actively*, we repeated the experiment described above but with all 40 unlabeled instances selected randomly for annotation. The error rate over successive iterations is shown by line R1 in Figure 9. Comparing A1 to R1, we see that the error decreases faster when we select some of the unlabeled instances for annotation actively at each iteration.

7 Conclusion

In this paper, we have presented a system `Qme!`, a speech-driven question-answering system for mobile devices. We have proposed a query retrieval model for question-answering and demonstrated the mutual benefits of tightly coupling the ASR and Search components of the system. We have presented a novel concept of distinguishing questions that need dynamic information to be answered from those questions whose answers can be retrieved from an archive. We have shown results on bootstrapping such a classifier using semi-supervised learning techniques.

References

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

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.

L. Breiman. 1996. Bagging predictors. *Machine Learning*, 24(2):123–140.

S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng. 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*, pages 291–298, New York, NY, USA. ACM.

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

B.F. Green, A.K. Wolf, C. Chomsky, and K. Laughery. 1961. Baseball, an automatic question answerer. In *Proceedings of the Western Joint Computer Conference*, pages 219–224.

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.

J. Jeon, W. B. Croft, and J. H. Lee. 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*, pages 84–90, New York, NY, USA. ACM.

B. Katz. 1997. Annotating the world wide web using natural language. In *Proceedings of RIAO*.

V.I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertion and reversals. *Soviet Physics Doklady*, 10:707–710.

M. T. Maybury, editor. 2004. *New Directions in Question Answering*. AAAI Press.

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

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.

MSN-QnA, 2009. *http://qna.live.com/*.

K. Papineni, S. Roukos, T. Ward, and W.J. Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of 40^{th} Annual Meeting of the Association of Computational Linguistics*, pages 313–318, Philadelphia, PA, July.

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.

S. Robertson. 2004. Understanding inverse document frequency: On theoretical arguments for idf. *Journal of Documentation*, 60.

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.

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

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

R. J. Waldinger, D. E. Appelt, J. L. Dungan, J. Fry, J. R. Hobbs, D. J. Israel, P. Jarvis, D. L. Martin, S. Riehemann, M. E. Stickel, and M. Tyson. 2004. Deductive question answering from multiple resources. In *New Directions in Question Answering*, pages 253–262.

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

T. Winograd. 1972. *Understanding Natural Language*. Academic Press.

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

X. Xue, J. Jeon, and W. B. Croft. 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*, pages 475–482, New York, NY, USA. ACM.

Yahoo!, 2009. *http://answers.yahoo.com/*.

F. Yang, J. Feng, and G. DiFabbrizio. 2006. A data driven approach to relevancy recognition for contextual question answering. In *HLT-NAACL 2006 Workshop on Interactive Question Answering*, New York, USA, June 8-9.

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