QUERY PARSING FOR VOICE-ENABLED MOBILE LOCAL SEARCH

Junlan Feng and Srinivas Bangalore

AT&T Labs-Research, 180 Park Avenue, Florham Park, NJ 07932, USA

ABSTRACT

With the exponential growth in the number of mobile devices, voice enabled local search is emerging as one of the most popular applications. Although the application is essentially an integration of automatic speech recognition (ASR) and text or database search, the potential usefulness of this application has been widely acknowledged. In this paper, we present a data-driven approach to voice query parsing, that segments the input query into several fields that are necessary for high-precision local search. We also demonstrate the robustness of our approach to ASR errors. We present an approach for exploiting multiple hypotheses from ASR, in the form of word confusion networks, in order to achieve tighter coupling between ASR and query parsing. A confusion-network based query parsing outperforms ASR 1-best based query-parsing by 2.6% absolute.

Index Terms— Voice Search, Robustness to ASR errors

1. INTRODUCTION

Local search specializes in serving geographically constrained search queries on a structured database of local business listings. Most text-based local search engines expect a user to submit a search query in two text fields: the "SearchTerm" (e.g. *Best Chinese Restaurant*) and the "LocationTerm" (e.g. a city, state, street address, neighborhood etc.) fields. Most voice-enabled local search systems employ a two-turn dialog strategy. In the first turn, the system solicits from the user a LocationTerm followed by a SearchTerm [1].

Although the two-field interface has been widely accepted, it has several limitations for voice-enabled mobile search. First, most mobile devices are location-aware. When users search for businesses nearby, it is encumbering for them to specify the LocationTerm. Second, it's not always straightforward for users to be aware of the distinction between these two fields. For many local search text queries, we have observed the SearchTerm field contains location information as well. For example, "restaurants near Manhattan" was input for SearchTerm and "NY NY" for LocationTerm. For voice-based local search, it is more natural for users to specify queries in a single utterance. Finally, many queries contain constraints beyond location such as *restaurants that deliver* and *night clubs open 24 hours*. In these two examples, *restaurants* and *night clubs* are the main query subjects while *that deliver* and *open 24 hours* are constraints. It would be very cumbersome to enumerate each constraint as a different text field or a dialog turn. An interface that allows for specifying constraints in a natural language utterance would be most convenient.

In this paper, we introduce a voice search system that allows users to specify search requests in a single natural language utterance. The output of ASR is then parsed by a query parser into three fields: LocationTerm, SearchTerm, and Filler. We use a local search engine, *http://www.yellowpages.com/*, which accepts the SearchTerm and LocationTerm as two query fields and returns the search results. We present a method for parsing the voice query into different fields with particular emphasis on exploiting the ASR output, beyond 1-best hypothesis. We demonstrate that by parsing word confusion networks, the accuracy of the query parser can be improved.

The paper outline is as follows. In Section 2, we discuss some of the related threads of research relevant for our task. In Section 3, we motivate the need for a query parsing module in voice search systems. We present the details of the query parsing model in Section 4 and discuss experimental results in Section 5. We summarize our results in Section 6.

2. RELATED WORK

The role of query parsing in one aspect can be considered as similar to spoken language understanding(SLU) in dialog applications. However, voice local search systems currently do not have SLU as a separate module , instead the words in the 1-best ASR output are directly used for search. In previous work, BBN proposed a two-state HMM approach to do query understanding and search in the same step. One state is called General English (GE) state for capturing carried phrases and ASR errors [2]. The other state is for Listing Name. Parameters for this model are learned from transcribed utterances. Most other voice search applications applied the traditional Information Retrieval approach, vector space model (VSM), for search. In [3], the authors enhanced VSM by deemphasizing term frequence in Listing Names and using character level uni/bi-gram terms instead of word level. The aim was to be more tolerant to ASR errors. This approach improves recall but not precision.

There are two other threads of relevant research literature. Named entity (NE) extraction attempts to identify entities of interest in speech or text. Typical entities include locations, persons, organizations, dates, times monetary amounts and percentages [4]. Most approaches for NE tasks rely on machine learning approaches using annotated data. These algorithms include a hidden markov model, support vector machines, Maximum entropy, and conditional random fields . With the goal of improving robustness to ASR errors, [5] described a Finite State Machine based approach to take as input ASR n-best strings. Although our task of query segmentation has similarity with NE tasks, it is arguable whether the SearchTerm is a well-defined entity, since a user can provide varied expressions as they would for general web search. More importantly the discriminative approaches used in NE literature have not been applied to weighted lattices produced by ASR.

The other related literature is natural langague interface to database(NLIDBs), which had been well-studied between 1960s and 1980s [6]. The aim was to map a natural language query into semantic components of a database, such as table, field, Value, Relation, Aggregate (COUNT,SUM,MIN, MAX), Logical operations (AND, OR, NOT). The task was significantly more ambitious than

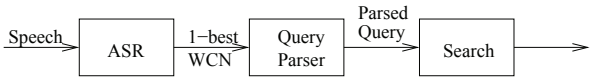

Fig. 1. Architecture of a voice search system

our current task. According to their methodology, these systems were categorized as pattern matching, syntax-based, and grammar-based systems. Issues such as modifier attachment still remain as a challenging problem in our task. We incorporate some of the techniques from that literature in our query parser.

3. VOICE SEARCH SYSTEM ARCHITECTURE

Figure 1 illustrates the architecture of our voice search system[1]. As expected the ASR and Search components perform speech recognition and search tasks. In addition to ASR and Search, we also integrate a query parsing module between ASR and Search for a number of reasons.

First, we prefer to reuse the current local search engine *http://www.yellowpages.com/*, in which many text normalization, task specific tuning, business rules, and scalability issues have been well addressed. Given that, we need a module to translate ASR output to the query syntax that the local search engine supports.

Second, current ASR 1-best output is not accurate enough when users input speech from a mobile device, particularly from a noisy environment. However, ASR word confusion networks which compactly encode multiple word hypotheses with their probabilities have much higher word accuracy compared to the 1-best output. We believe a dedicated understanding module is necessary to reevaluate ASR output for the purpose of maximizing search performance. In this paper, we show promising results using richer ASR output beyond 1-best hypothesis.

Lastly, as mentioned above the query parser we are designing not only provides the search engine "what" and "where" information, it also further segments the query to phrases of certain concepts. For the example we used above *night club open 24 hours*, we segment it to *night club* and *open 24 hours*. Query segmentation has been considered as a key step to achieving higher retrieval accuracy [7].

In the next section, we present our proposed approaches of how we parse ASR output including ASR 1-best string and lattices in a scalable framework.

4. A SCALABLE QUERY PARSING APPROACH

As we discussed above, there are many potential approaches such as those for NE extraction we can explore for parsing a query. In the context of voice local search, users expect overall system response time to be similar to that of web search. Due to the relatively long ASR latency, it leaves no room for a slow parser. On the other hand, the parser needs to be tightly synchronized with changes in the listing database, which is updated at least once a day. Hence, the training process also needs to be fast to meet the requirements. We propose in this section a probablistic query parsing approach using text index and search. We start by presenting a model for parsing ASR 1-best and extend the approach to consider ASR lattices.

[1]We do not address the issue of presentation of results in this paper.

4.1. Query Parsing Using ASR 1-best output

4.1.1. The Problem

We formulate the query parsing task as follows. A 1-best ASR out is a sequence of words: $Q = q_1, q_2, \ldots, q_n$. The parsing task is to segment Q into a sequency of concepts. Each concept can possibly span multiple words. Let $S = s_1, s_2, \ldots, s_k, \ldots, s_m$ be one of the possible segmentations comprising of m segments, where $s_k = q_j^i = q_i, \ldots q_j, 1 \leq i \leq j \leq n+1$. The corresponding concept sequence is represented as $C = c_1, c_2, \ldots, c_k, \ldots, c_m$.

For a given Q, we are interested in searching for the best segmentation and concept sequence (S^*, C^*) as defined by Equation 1, which is rewritten using Bayes rule as Equation 2. The prior probability $P(C)$ is approximated as using a h-gram model on the concept sequence as shown in Equation 3. We model the segment sequence generation probability $P(S|C)$ as shown in Equation 4, using indepdence assumptions. Finally, the query terms corresponding to a segment and concept are generated using Equation 5. The conditional probablity $P(q_j^i|c_k)$ can be considered as Relevancy defined in (6).

$$(S^*, C^*) = \underset{C}{argmax}\ P(S, C) \quad (1)$$

$$= \underset{S,C}{argmax}\ P(C) * P(S|C) \quad (2)$$

$$P(C) = P(c_1) * \prod_{k} = 1^m\ P(c_k|c_{k-1}, ...c_{k-h+1}) \quad (3)$$

$$P(S|C) = \prod_{k=1}^{m} P(s_k \mid c_k) \quad (4)$$

$$P(s_k|c_k) = P(q_j^i|c_k) = P_{c_k}(q_i) * \prod_{l=i+1}^{j} P_{c_k}(q_l \mid q_{l-1}^{l-k+1}) \quad (5)$$

To train this model, we only have access to text query logs from two distinct fields (SearchTerm, LocationTerm) and the listing database. We built a SearchTerm corpus by including valid queries users typed to the SearchTerm field and all the unique business listing names in the listing database. Valid queries are those queries that the search engine returns at least one business listing result or a business category. Similarly, we built a corpus for LocationTerm by concatenating valid LocationTerm queries and unique addresses including street address, city, state, and zip-code. We also built a small corpus for Filler, which contains common carrier phrases and stop words. The generation probabilities as defined in (4) can be learned from these three corpora.

In the following section, we describe a scalable way of implementation using standard text indexer and searcher.

4.1.2. Probabilistic Parsing using Text Search

We explored using standard text indexing and search engines for query parsing. More specifically, we use Apache-Lucene in our experiments [8]. Lucene is an open-source full-featured text search engine library. Both lucene indexing and search are efficient enough for our tasks. It takes a few milliseconds to return results for a common query. Indexing millions of search logs and listings can be done in minutes. Reusing text search engines allows a seamless integration between query parsing and search.

We changed the `tf.idf` based document-term relevancy metric in Lucene to reflect $p_{c_k}(q_j^i)$.

$$Relevancy(q_j^i, d_k) = p_{c_k}(q_j^i) = \frac{tf(q_j^i, d_k) + \sigma}{N} \quad (6)$$

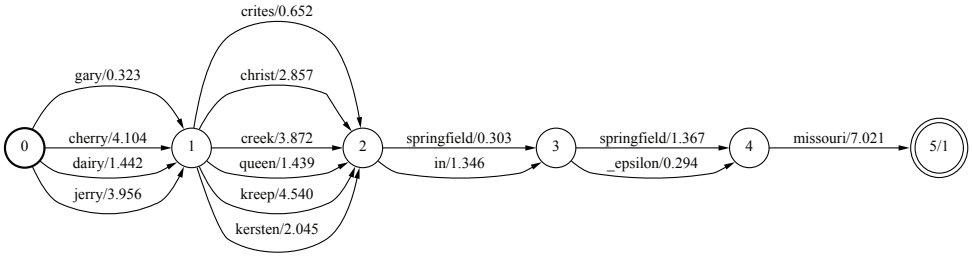

Fig. 2. An example confusion network for "Gary crities Springfield missouri"

where d_k is a corpus of examples we collected for the concept c_k; $tf(q_j^i, d_k)$ is referred as the term frequency, the frequency of q_j^i in d_k; N is the number of entries in d_k; σ is an empirically determined smoothing factor.

When $tf(q_j^i, d_k)$ is zero for all concepts, we loosen the phrase search to be proximity search, which searches words in q_j^i within a specific distance away. For instance, "burlington west virginia" ~ 5 will find entries that include these three words within 5 words of each other. $tf(q_j^i, d_k)$ is discounted for proximity search. For a given q_j^i, we allow a distance of $dis(i,j) = (j - i + shift)$ words. *shift* is a parameter that is set empirically.

$$p_{c_k}(q_j^i) = \frac{tf(q_i^i \sim dis(i,j), d_k) + \sigma}{N * shift} \quad (7)$$

Inputs:

- A set of K concepts:$C = c_1, c_2, \ldots, c_K$, in this paper, $K = 3$, $c_1 = SearchTerm$, $c_2 = LocationTerm$, $c_3 = Filler$
- Each concept c_k associates with a text corpus: d_k. Corpora are indexed using Lucene Indexing.
- A given query: $Q = q_1, q_2, \ldots, q_n$
- A given maximum number of words in a query segment: Ng

Parsing:

- Enumerate possible segments in Q up to Ng words long: $q_j^i = q_i, q_{i+1}, \ldots, q_j, j >= i, |j - i| < Ng$
- Obtain $p_{c_k}(q_j^i))$ for each pair of c_k and q_j^i using Lucene Search
- Boost $p_{c_k}(q_j^i))$ based on the position of q_j^i in the query $p_{c_k}(q_j^i) = p_{c_k}(q_j^i) * boost_{c_k}(i,j,n)$
- Search for the best segment sequence and concept sequence using Viterbi search

Fig.3. Parsing procedule using Text Indexer and Searcher

Figure 3 shows the procedure we use for parsing. It enumerates possible segments q_j^i of a given Q. It then obtains $p_{c_k}(q_j^i)$ using Lucene Search. We boost $p_{c_k}(q_j^i))$ based on the position of q_j^i in Q. In our case, we simply set: $boost_{c_k}(i,j,n) = 3$ if $j = n$ and $c_k = LocationTerm$. Otherwise, $boost_{c_k}(i,j,n) = 1$. The algorithm searches for the best segmentation using the Viterbi algorithm. Out-of-vocabulary words are assigned to c_3 (Filler).

4.2. Query Parsing using ASR lattices

Word confusion networks (WCNs) [6] is a compact lattice format. It aligns a speech lattice with its top-1 hypothesis, yielding a "sausagez"-like approximation of lattices. It has been used in applications such as word spotting and spoken document retrieval. In the following, we examine how we use WCNs for our query parsing task.

Figure 2 shows an example of a pruned WCN . For each word position, there are multiple alternatives and their associated negative log posterior probabilities. The 1-best path is "Gary Crites Springfield Missouri". The reference is "Dairy Queen in Springfield Missouri". ASR misrecognized "Dairy Queen" as "Gary Cities". But the correct words "Dairy Queen" do appear in the lattice though with lower probability. The challenge is to select the correct ones from the lattice by considering both ASR posterior probabilities and parser scores.

The hypotheses in WCN have to be reranked by the Query Parser to prefer those that have meaningful concepts. Clearly, each business name in the listing database corresponds to a single concept. However, the long queries from query logs tend to contain multiple concepts. For example, a frequent query is "night club for 18 and up". We know "night club" is the main subject. And "18 and up" is a constraint. Without matching "night club", any match with "18 and up" is meaningless. The data fortunately can tell us which words are more likely to be a subject. We rarely see "18 and up" as a complete query. Given these observations, we propose calculating the probability of a query term to be a subject. "Subject" here specifically means a complete query or a listing name. For the example shown in Figure 1, we observe the negative log probability for "Dairy Queen" to be a subject is 9.3. "Gary Crites" gets 15.3. We refer to this probability as subject likelihood. Given a candidate query term $s = w_1, w_2, ..w_m$, we represent the subject likelihood as $P_{sb}(s)$. In our experiments, we estimate P_{sb} using relative frequency normorlized by the length of s. We use the following formula to combine it with posterior probabilities in WCNs $P_{cf}(s)$:

$$P(s) = P_{cf}(s) * P_{sb}(s)^{\lambda}$$
$$P_{cf}(s) = \prod_{j=1,\ldots,nw} P_{cf}(w_i)$$

where λ is used to flatten ASR posterior probabilities and nw is the number of words in s. In our experiments, λ is set to 0.5. We then re-rank ASR outputs based on $P(s)$. We will report experimental results with this approach. "Subject" is only related to SearchTerm. Considering this, we parse the ASR 1-best out first and keep the Location terms extracted as they are. Only word alternatives corresponding to the search terms are used for reranking. This also

Data Sets	SearchTerm Extraction Accuracy				LocationTerm Extraction Accuracy			
Input	ASR-1best	WCN	Oracle Path	Transcription	ASR 1best	WCN	Oracle Path	Transcription
Test1	57.7%	**59.6%**	65.0%	93.0%	79.4%	79.4%	83.5%	96.1%
Test2	74.1%	**76.7%**	81.6%	97.8%	88.8%	88.8%	93.0%	98.5%
Test3	62.2%	**63.8%**	68.8%	96.0%	86.5%	86.5%	88.7%	96.7%

Table 1. Parsing performance on three data Sets

improves speed, since we make the confusion network lattice much smaller. In our initial investigations, such an approach yields promising results as illustrated in the next section.

5. EXPERIMENTS

We have access to text query logs consisting of 18 million queries to the two text fields: SearchTerm and LocationTerm. 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 our model as discussed in the previous section. We tested our approaches on three data sets, which in total include 2686 speech queries. These queries were collected from users using mobile devices for different time periods. Labelers transcribed and annotated the test data using LocationTerm and SearchTerm tags.

Data Sets	Number of Speech Queries	WACC
Test1	1484	70.1%
Test2	544	82.9%
Test3	658	77.3%

Table 2. ASR Performance on three Data Sets

We use a trigram-based ASR trained on the query logs and listing data sets. Table 2 shows the ASR performances on three data sets. We achieved 70.1% word accuracy on Test1. Test1 is a hard dataset, where many speakers are non-native and a big percentage of queries are not intended for local search. We observed 82.9% word accuracy on Test2 and 77.3% on Test3.

Table 1 reports parsing performance using the proposed approach. We measure parsing performance in terms of extraction accuracy. In our task, we have two non-filler slots, namely, LocationTerm and SearchTerm. Table 1 reports extraction accuracy for both of them. The "Transcription" columns present the parser's performances on transcriptions. As expected, the parser's performance heavily relies on ASR word accuracy. We achieved lower parsing performance on Test1 than other test sets due to lower ASR accuracy on this test set. The promising aspect is we consistently improved SearchTerm extraction accuracy when using pruned WCN as input. The performance under "Oracle path" shows the upper bound for the parser using the oracle path[2] from the pruned WCN. We pruned WCN by keeping only those arcs that are within $cthresh$ of the lowest cost arc between two states. $Cthresh = 4$ is used in our experiments. For Test2, the upper bound improvement is 7.5% (81.6%-74.1%) absolute. Our proposed approach using pruned WCN achieved 2.6% improvement, which is 35% of the maximum potential gain. We reached 26% of the upper bound improvement on Test1 and 24% on Test3. Our approach didn't take advantage of WCN for LocationTerm extraction, hence we obtained the same performance with WCNs as using ASR 1-best.

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

6. SUMMARY

This paper describes a probabilistic approach for parsing ASR 1-best and lattice output into fields such as SearchTerm and LocationTerm for local search. We implemented this approach using Lucene, a full-featured standard text engine library. Our data resources include millions of query logs and business listing entries. We evaluated our approach on three test sets of voice queries and achieved SearchTerm extraction accuracy of 59.6%, 76.7% and 63.8% and LocationTerm extraction accuracy of 79.4%, 88.8%, and 86.5%. We observed consistent improvement using ASR WCNs.

Our near-term future work will be evaluating Search performance. Both ASR and Query parser make errors, however these errors are not equally important in terms of their impact on search.

7. ACKNOWLEDGEMENTS

We are grateful to Benson Tang, Remi Zajac, Premkumar Mani, Mazin Gilbert, Vincent Goffin, and Barbara B. Hollister for their help in discussing and improving the ideas presented in this paper.

8. REFERENCES

[1] Y.Wang, D.Yu, Y. Ju, and A. Alex, "An introduction to voice search," *Signal Processing Magzine*, vol. 25, no. 3, pp. 29–38, 2008.

[2] P. Natarajan, R. Prasad, R.M. Schwartz, and J. Makhoul, "A scalable architecture for directory assistance automation," in *ICASSP 2002*, 2002.

[3] G.Zweig D.Yu Y.-C.Ju Y.-Y.Wang and A.Acero, "Automated directory assistance system - from theory to practice," in *Interspeech*, 2007.

[4] F. Kubala, R. Schwartz, R. Stone, and R. Weischedel, "Named entity extraction from speech," in *in Proceedings of DARPA Broadcast News Transcription and Understanding Workshop*, 1998, pp. 287–292.

[5] P. Nocera B. Favre, F. Bechet, "Robust named entity extraction from large spoken archives," in *Proceeding of HLT 2005*, 2005.

[6] I. Androutsopoulos, "Natural language interfaces to databases - an introduction," *Journal of Natural Language Engineering*, vol. 1, pp. 29–81, 1995.

[7] B. Tan F. Peng, "Unsupervised query segmentation using generative language models and wikipedia," in *Proceedings of WWW-2008*, 2008.

[8] Erik Hatcher and Otis Gospodnetic, *Lucene in Action (In Action series)*, Manning Publications Co., Greenwich, CT, USA, 2004.