INTERSPEECH 2009 BRIGHTON

Role of Natural Language Understanding in Voice Local Search

Junlan Feng, Srinivas Banglore, Mazin Gilbert

AT&T Labs Research, Florham Park, NJ, USA

junlan@research.att.com, srini@research.att.com, mazin@research.att.com

Abstract

Speak4it is a voice-enabled local search system currently available for iPhone devices. The natural language understanding (NLU) component is one of the key technology modules in this system. The role of NLU in voice-enabled local search is two-fold: (a) parse the automatic speech recognition (ASR) output (1-best and word lattices) into meaningful segments that contribute to high-precision local search, and (b) understand user's intent. This paper is concerned with the first task of NLU. In previous work, we had presented a scalable approach to parsing, which is built upon text indexing and search framework, and can also parse ASR lattices. In this paper, we propose an algorithm to improve the baseline by extracting the "subjects" of the query. Experimental results indicate that lattice-based query parsing outperforms ASR 1-best based parsing by 2.1% absolute and extracting subjects in the query improves the robustness of search.

Index Terms: Voice Search, Robustness to ASR errors

10.21437/Interspeech.2009-540

1. Introduction

Local search specializes in serving geographically constrained search queries on a structured database of local business listings. A recent study reported that local Internet search grew 58% reaching 15.7 billion searches in 2008 [1]. Location-aware mobile devices make this need even larger by obviating the need to specify a location when users search for listings nearby. Speaking a query is much easier than typing on small devices. One of the common complaints among iPhone users is the difficulty of typing text using the soft keyboard. In this context, voice-enabled mobile local search is naturally emerging as a popular application. In this paper, we present Speak4it, a system developed by yellowpages.com and AT&T Labs-Research, which allows users to speak local search queries in a single utterance and returns information of relevant businesses.

Figure 1 illustrates a schematic diagram of the three major technology components in Speak4it including Automatic Speech Recognizer (ASR), Natural Language Understanding (NLU), and Search. As expected the ASR and Search components perform speech recognition and search tasks. In Speak4it, the speech recognizer is the AT&T Watson ASR engine [2] with a trigram-based language model trained on text query logs. We use a text local search engine, http://www.yellowpages.com/, which accepts SearchTerm (e.g. Best Chinese Restaurant) and LocationTerm (e.g. a city, state, street address, neighborhood etc.) as two query fields and returns the search results from a business listings database. In addition to ASR and Search, we integrate a natural language understanding module, the main focus of this paper, between ASR and Search for a number of reasons explained below.

First, there is a gap between the query format that users favor and the query syntax that text-based local search supports. Typical local search engines including http://www.yellowpages.com accepts queries with two-fields: "SearchTerm" and "LocationTerm". This is due to the special status of LocationTerm, where similarity of two locations is based on physical distance instead of word similarity. Most voice-enabled local search systems mimic this two-field approach and employ a two-turn dialog strategy. The system solicits from the user a LocationTerm in the first turn followed by a SearchTerm in the second turn [3]. This two-turn design adds a burden on the user to split his query into two fields. As our data showed, it's not always straightforward for users to make a clear distinction between the two fields. For instance, a query may include "restaurants near manhattan" for SearchTerm and "new york city" for LocationTerm. Furthermore many queries often contain other constraints (assuming LocationTerm is a constraint) such as *that deliver* in *restaurants that deliver* or *open Christmas Day* in *night clubs open Christmas Day*. It would be very cumbersome to enumerate each constraint as a different text field or a dialog turn. Hence, Speak4it supports an interface that allows for specifying constraints in a natural language utterance. It then relies on the NLU module to parse the query into fields needed by search.

Second, as can be expected the ASR 1-best output is typically error-prone especially when a user query originates from a noisy environment. However, ASR word confusion networks which compactly encode multiple word hypotheses with their posterior weights have the potential to alleviate the errors in a 1-best output. Our motivation to introduce the understanding module is to rescore the 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, we prefer to reuse an existing 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 ASR errors and the complexity of a commercial search engine, we need a module to optimize the coupling between ASR and search.

In this paper, we present a scalable approach for parsing the voice query into different fields with particular emphasis on exploiting the ASR output beyond the 1-best hypothesis. We demonstrate that by parsing word confusion networks, the accuracy of the query parser can be improved. We further propose an algorithm to improve the baseline by extracting the main subject in the query and investigate the effect of this improvement on the search task.

The paper outline is as follows. In Section 2, we discuss some of the related threads of research. In Section 3, we present a baseline parsing approach and the algorithm for improvement. We discuss experimental results in Section 4. We summarize our results in Section 5.

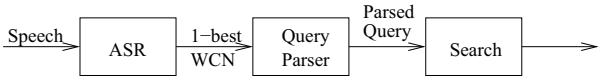

Figure 1: A schematic diagram of major technology components in Speak4it

2. Related Work

There are several threads of relevant research literature including spoken language understanding (SLU) in dialog applications, query segmentation in information retrieval, named entity (NE) extraction, as well as natural language interface to database (NLIDBs).

The role of query parsing in one aspect can be considered as similar to SLU. However, most voice driven local search systems as of today do not have an SLU as a separate module. However, some capability of SLU are either integrated into ASR or Search components. For instance, BBN proposed a two-state HMM approach to do query understanding and search in the same step [6]. Most other voice search applications have applied the traditional Information Retrieval approach, vector space model (VSM), for search. In [4], the authors enhanced VSM by deemphasizing term frequency in listing names and using character level instead of word level uni/bi-gram terms. The aim was to be more tolerant to ASR errors. This approach improves recall but not precision.

Query segmentation is an important task in web search. The goal is to partition a text query into segments of text. Each text segment is believed to be an instantiation of a certain concept. Query segmentation could help a retrieval system to improve its accuracy, as segments carry implicit word proximity / ordering constraints, which may be used to filter documents [5]. An example of query segmentation is Named Entity (NE) extraction [6], which attempts to identify a predefined set of entities of interest in speech or text. However, query segments are typically more general than NEs. Typical approaches for query segmentation are based on unsupervised learning using query logs [5]. 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. Our task of voice query parsing confronts a combination of challenges in both NE extraction and query segmentation. Concepts such as *LocationTerm* and *SearchTerm* are similar to entities, though 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. We also see a need to further segment the query for each field to achieve higher search performance. For instance, *night clubs open 24 hours nyc* is segmented to *SearchTerm: night clubs, open 24 hours* and *LocationTerm: nyc*. Furthermore, the learning approaches used in NE literature and Query segmentation literature have not been applied to weighted lattices produced by ASR.

The other related literature is natural language interface to databases (NLIDBs), which had been well-studied between 1960s and 1980s [7]. The aim is to map a natural language query into a structured query that could be used to access a database. However, most of the literature pertains to textual queries, not spoken queries. Although in its full generality the task of NLIDB is significantly more ambitious than our current task, some of the challenging problems (e.g. modifier attachment in queries) can also be seen in our task as well.

3. Query Parsing

3.1. Baseline Approach : Query Parsing Using ASR 1-best output

As we discussed above, there are many potential approaches we can explore for parsing a query. In the context of voice-driven 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 NL parser. Furthermore, 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 accommodate the updates. In this section, we describe a probabilistic query parsing approach using text indexing and search.

We formulate the query parsing task as follows. A 1-best ASR output is a sequence of words: $Q = q_1, q_2, \ldots, q_n$. The parsing task is to segment Q into a sequence 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 $s_k = q_i^j = q_i, \ldots q_j, 1 \leq i \leq j \leq n+1$. 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 using an 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 independence assumptions. Finally, the query terms corresponding to a segment and concept are generated using Equation 5. The conditional probability $P(q_i^j|c_k)$ can be considered as *Relevancy* defined in Equation (6).

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

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

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

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

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

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, stop words, slang phrases and typical filler words in speech. The generation probabilities as defined in (4) can be learned from these three corpora.

We implemented this approach using standard text indexing and search software. 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 approximate $p_{c_k}(q_j^i)$.

$$p_{c_k}(q_i^j) = \frac{tf(q_i^j, d_k) + \sigma}{N} \tag{6}$$

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

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

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

3.2. Query Subject Extraction

In this section, we emphasize on parsing queries containing concepts beyond business names, business categories and business addresses. For instance, the above baseline approach parses *night clubs open christmas day new york city* into *night clubs open christmas day* as the SearchTerm and *new york city* as the LocationTerm. However, this segmentation will lead search to return no result since a high-precision local search tries to match all query words. If the listing database doesn't contain information about *night clubs* that are *open christmas day* for the given region, then search fails. Local search emphasizing on recall might consider the query to be a bag of words and thus may return listings only relevant to *christmas* but not to *night club* . *night club* is the main subject of this query. Without matching *night club*, any match with *open christmas day* results in irrelevant listings. We were motivated by this observation to extract the query subject[1], which is the core concept of query and considered as the "must" match part of the query.

To solve this problem, we first considered a dependency parser [9] that parses a sentence into a dependency tree. However, off-the-shelf syntactic parsing tools do not work well on ungrammatical queries and do not provide dependency relationship on the concept level. We approach this problem by mining query logs, which latently encapsulate the most likely subject phrases. We rarely see "open christmas day" as a complete query. It often co-exists with other keywords or keyphrases. However, the term *night clubs* appears as one of the frequent queries. Based on these heuristics, we propose calculating the probability of a query term to be a subject. "Subject" here specifically means a complete query or a business listing/category name.

We refer to this probability as subject likelihood. Given a candidate query segment $s = w_1, w_2, ..w_m$, we represent the subject likelihood as $P_{sb}(s)$. In our experiments, we estimate P_{sb} using relative frequency normalized by the number of words in s, $nw(s)$.

$$P_{sb}(s) = (\lambda * \frac{Freq(s)}{N} + (1 - \lambda) * \frac{1}{N})^{\gamma} \tag{8}$$

where $Freq(s)$ is the number of times that s appears as a complete query in the corpus. N is the number of entries in the corpus and $\gamma = 1/nw(s)$. We create this corpus by concatenating simple queries in the query log and two copies of listing names in the listing database. Simple queries are defined as queries that don't have explicit constraint indicators such as *that, with, etc.* or contain more than 5 words. Each simple query or listing name is treated as an entry. The query subject is a segment s' from the extracted SearchTerm that returns highest subject likelihood.

3.3. Query Parsing using ASR Word Confusion Networks

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

We applied the subject model described above to prefer the hypotheses in WCN that have meaningful concepts.

For the example shown in Figure 2, we observe the negative log subject likelihood for "Dairy Queen" is 9.3. "Gary Crites" scores 15.3. We refer to this probability as subject likelihood. We use the following formula to combine subject likelihood with posterior probabilities in WCNs $P_{cf}(s)$:

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

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 first parse the ASR 1-best and hold the extracted Location terms unchanged. Only word alternatives corresponding to the search terms are used for reranking. This also 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.

4. Experiments

Our training data consists of 18 million web queries to the two text fields: SearchTerm and LocationTerm and 11 million unique business listing names and their addresses. We use the combined data to train the parameters of the parsing models as discussed in the previous sections. We tested our

[1] This may not correspond to the notion of a syntactic subject of a sentence.

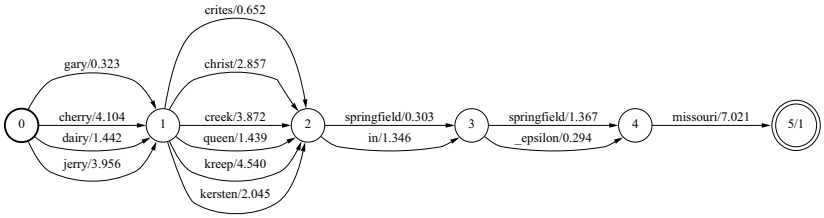

Figure 2: An example confusion network for "Gary crities Springfield missouri"

approaches on 1000 voice queries, which are randomly selected from Speak4it query logs. Labelers transcribed and annotated the test data using SearchTerm, LocationTerm, and Filler tags. For example, *the Walmart on Nicholasville Road* is annotated as *<Filler> the </Filler> <SearchTerm> Walmart </SearchTerm> <Filler> on </Filler> <LocationTerm> Nicholasville Road </LocationTerm>* .

We used an ASR with a trigram-based language model trained on web query logs and achieved 67.2% ASR 1-best word accuracy and 55.6% sentence accuracy. 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.

Slots	1-best	WCN	Transcription
SearchTerm	60.1%	**62.2%**	91.0%
LocationTerm	88.5%	88.5%	94.1%

Table 1: Slot Extraction Accuracy of the Query Parser

Table 1 reports the parsing performance for the two slots. The "Transcription" column presents the parser's performances on human transcriptions (i.e. word accuracy=100%) of the speech. The "1-best" and "WCN" respectively corresponds to ASR 1-best and WCN output. As expected, the parser's performance is much lower for ASR output than on human transcription. The promising aspect is that we improved SearchTerm extraction accuracy when using WCN as input by 2.1%. We pruned the WCN by keeping only those arcs that are within $cthresh$ (=1 in our experiments) of the lowest cost arc between two states. The pruned WCN contains 1.2 words on average for each word position. Hence, it does not add much latency to the parser. Our approach of using WCNs via the subject/constrain model is designed for improving SearchTerm accuracy. We extracted LocationTerm from the top best even when using WCN as input. Hence, Table 1 shows the same LocationTerm extraction accuracy for both 1-best and WCN.

We have not fully evaluated the impact of parsing performance on search accuracy for this data set. Previous experiments with trial data had been published at [10].

We conducted preliminary experiments to evaluate the contribution of the *subject* model to search performance. We observed that 18% of queries with search terms return no search results. For these queries, we applied the subject model to extract the subject of the query. We use the extracted subject terms to resubmit the query, if the subject extracted is not the entire search term. We noticed that now only 8% of the queries did not retrieve a result; though we have not evaluated the overall impact on the precision of search.

5. Summary

This paper describes a scalable baseline approach for parsing ASR 1-best into fields such as SearchTerm and LocationTerm for local search. We implemented this approach using Lucene, a full-featured standard text engine library. We further proposed an algorithm to mine query logs for query subject extraction. Query subject is defined as the core of query and considered as the "must" match part. We applied this subject model to parse ASR WCN output and achieved 2.1% improvement on parsing accuracy. We have also presented results from preliminary experiments that evaluate the impact of this subject model on search performance. We were able to reduce the rate that search returns no results from 18.0% to 8.0%.

Our future work will be to improve the synergy between NLU and search. ASR and NL parser make errors, however these errors are not equally important in terms of their impact on search. The interesting question is how to directly optimize NL for the best search performance.

6. Acknowledgements

We are grateful to Vincent Goffin and Barbara B. Hollister for their support for the work presented in this paper.

7. References

[1] L. Sullivan, "Internet yellow pages become pulse of local search," *MediaPost*, 2009, March 27.

[2] V. Goffin, G. Riccardi C. Allauzen D. Hakkani A. Ljolje S. Parthasarathy M. Rahim, and M. Saraclar., "The at&t watson speech recognizer," in *ICASSP*, 2005.

[3] 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.

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

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

[6] 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.

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

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

[9] B. MacCartney M. Marneffe and C. D. Manning., "Generating typed dependency parses from phrase structure parses," in *LREC*, 2006.

[10] J. Feng and S. Bangalore, "Effects ofword confusion networks on voice search," in *EACL*, 2009.