Effects of Word Confusion Networks on Voice Search

Junlan Feng, Srinivas Bangalore
AT&T Labs-Research
Florham Park, NJ, USA
junlan,srini@research.att.com

Abstract

Mobile voice-enabled search is emerging as one of the most popular applications abetted by the exponential growth in the number of mobile devices. The automatic speech recognition (ASR) output of the voice query is parsed into several fields. Search is then performed on a text corpus or a database. In order to improve the robustness of the query parser to noise in the ASR output, in this paper, we investigate two different methods to query parsing. Both methods exploit multiple hypotheses from ASR, in the form of word confusion networks, in order to achieve tighter coupling between ASR and query parsing and improved accuracy of the query parser. We also investigate the results of this improvement on search accuracy. Word confusion-network based query parsing outperforms ASR 1-best based query-parsing by 2.7% absolute and the search performance improves by 1.8% absolute on one of our data sets.

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 provide two text fields: the "SearchTerm" (e.g. *Best Chinese Restaurant*) and the "LocationTerm" (e.g. a city, state, street address, neighborhood etc.). Most voice-enabled local search dialog systems mimic this two-field approach and employ a two-turn dialog strategy. The dialog system solicits from the user a LocationTerm in the first turn followed by a SearchTerm in the second turn (Wang et al., 2008).

Although the two-field interface has been widely accepted, it has several limitations for *mobile* voice search. First, most mobile devices are location-aware which obviates the need to specify the LocationTerm. Second, it's not always straightforward for users to be aware of the distinction between these two fields. It is common for users to specify location information in the SearchTerm field. For example, "restaurants near Manhattan" for SearchTerm and "NY NY" for LocationTerm. For voice-based search, it is more natural for users to specify queries in a single utterance[1]. Finally, many queries often contain other constraints (assuming LocationTerm is a constraint) such as *that deliver* in *restaurants that deliver* or *open 24 hours* in *night clubs open 24 hours*. 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-based 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 from a business listings database. We present two methods 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 investigate the effect of this improvement on the search task and demonstrate the benefit of tighter coupling of ASR and the query parser on search accuracy.

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-based search systems. We present two different query parsing models in Section 4 and Section 5 and discuss experimental results in Section 6. We summarize our results in Section 7.

[1] Based on the returned results, the query may be refined in subsequent turns of a dialog.

Proceedings of the 12th Conference of the European Chapter of the ACL, pages 238–245,
Athens, Greece, 30 March – 3 April 2009.

2 Related Work

The role of query parsing can be considered as similar to spoken language understanding (SLU) in dialog applications. However, voice-based search systems currently do not have SLU as a separate module, instead the words in the ASR 1-best output are directly used for search. Most voice-based search applications apply a conventional vector space model (VSM) used in information retrieval systems for search. In (Yu et al., 2007), the authors enhanced the VSM by deemphasizing term frequency in Listing Names and using character level instead of word level uni/bigram terms to improve robustness to ASR errors. While this approach improves recall it does not improve precision. In other work (Natarajan et al., 2002), the authors proposed a two-state hidden Markov model approach for query understanding and speech recognition in the same step (Natarajan et al., 2002).

There are two other threads of research literature relevant to our work. 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 (Kubala et al., 1998). 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, (Favre et al., 2005) described a finite-state machine based approach to take as input ASR n-best strings and extract the NEs. 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 a general web search. Also, it is not clear how the current best performing NE methods based on maximum entropy or conditional random fields models can be extended to apply on weighted lattices produced by ASR.

The other related literature is natural language interface to databases (NLIDBs), which had been well-studied during 1960s-1980s (Androutsopoulos, 1995). In this research, 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.

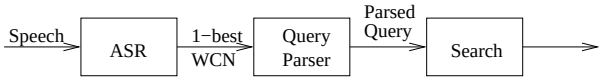

Figure 1: Architecture of a voice-based search system

3 Voice-based Search System Architecture

Figure 1 illustrates the architecture of our voice-based search system. 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, 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 probabilities 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.

Second, as mentioned earlier, the query parser not only provides the search engine "what" and "where" information, but also segments the query to phrases of other concepts. For the example we used earlier, we segment *night club open 24 hours* into *night club* and *open 24 hours*. Query segmentation has been considered as a key step to achieving higher retrieval accuracy (Tan and Peng, 2008).

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 that, we need a module to translate ASR output to the query syntax that the local search engine supports.

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 Text Indexing and Search-based Parser (PARIS)

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. Consequently, the relatively long ASR latency 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 parser's training process also needs to be quick to accomodate these changes. In this section, we propose a probabilistic query parsing approach called PARIS (parsing using indexing and search). We start by presenting a model for parsing ASR 1-best and extend the approach to consider ASR lattices.

4.1 Query Parsing on ASR 1-best output

4.1.1 The Problem

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 $s_k = q_j^i = q_i, \ldots q_j, 1 \leq i \leq j \leq n+1$. 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 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 Equations 5 and 6.

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

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

$$P(C) = P(c_1) * \prod_{i}^{m} P(c_i | c_{i-1}^{i-h+1}) \quad (3)$$

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

$$P(s_k|c_k) = P(q_j^i|c_k) \quad (5)$$

$$P(q_j^i|c_k) = P_{c_k}(q_i) * \prod_{l=i+1}^{j} P_{c_k}(q_l \mid q_{l-1}^{l-k+1}) \quad (6)$$

To train this model, we only have access to text query logs from two distinct fields (SearchTerm, LocationTerm) and the business listing database. We built a SearchTerm corpus by including valid queries that users typed to the SearchTerm field and all the unique business listing names in the listing database. Valid queries are those queries for which 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 in the listing database. We also built a small corpus for Filler, which contains common carrier phrases and stop words. The generation probabilities as defined in 6 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 use Apache-Lucene (Hatcher and Gospodnetic, 2004), a standard text indexing and search engines for query parsing. 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 $P(q_j^i|c_k)$ using $Relevancy$ as defined below.

$$P(q_j^i|c_k) = Relevancy(q_j^i, d_k) = \frac{tf(q_j^i, d_k) + \sigma}{N} \quad (7)$$

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.

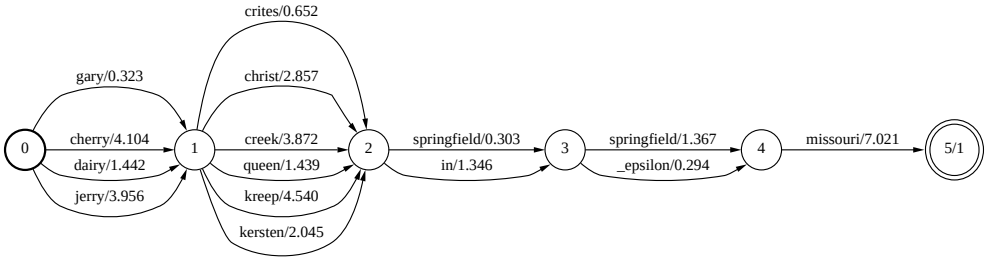

Figure 2: An example confusion network for "Gary crities Springfield Missouri"

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 $P(q_j^i|c_k))$ for each pair of c_k and q_j^i using Lucene Search
- Boost $P(q_j^i|c_k))$ based on the position of q_j^i in the query $P(q_j^i|c_k) = P(q_j^i|c_k) * boost_{c_k}(i, j, n)$
- Search for the best segment sequence and concept sequence using Viterbi search

Fig.3. Parsing procedure using Text Indexer and Searcher

$$p_{c_k}(q_j^i) = \frac{tf(q_i^i \sim dis(i,j), d_k) + \sigma}{N * shift} \quad (8)$$

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. For instance, "burlington west virginia" $\sim$ 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. The discounting formula is given in 8.

Figure 3 shows the procedure we use for parsing. It enumerates possible segments q_j^i of a given Q. It then obtains $P(q_j^i|c_k)$ 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 on ASR Lattices

Word confusion networks (WCNs) is a compact lattice format (Mangu et al., 2000). 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 present our use of WCNs for query parsing task.

Figure 2 shows a pruned WCN example. 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 Crities". However, the correct words "Dairy Queen" do appear in the lattice, though with lower probability. The challenge is to select the correct words from the lattice by considering both ASR posterior probabilities and parser probabilities.

The hypotheses in WCNs 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 2, 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 $P_{cf}(s)$:

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

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 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 experiment section.

Another capability that the parser does for both ASR 1-best and lattices is spelling correction. It corrects words such as *restaurants* to *restaurants*. ASR produces spelling errors because the language model is trained on query logs. We need to make more efforts to clean up the query log database, though progresses had been made.

5 Finite-state Transducer-based Parser

In this section, we present an alternate method for parsing which can transparently scale to take as input word lattices from ASR. We encode the problem of parsing as a weighted finite-state transducer (FST). This encoding allows us to apply the parser on ASR 1-best as well as ASR WCNs using the composition operation of FSTs.

We formulate the parsing problem as associating with each token of the input a label indicating whether that token belongs to one of a business listing (*bl*), city/state (*cs*) or neither (*null*). Thus, given a word sequence ($W = w_1, \ldots, w_n$) output from ASR, we search of the most likely label sequence ($T = t_1, \ldots, t_n$), as shown in Equation 9. We use the joint probability $P(W, T)$ and approximate it using an k-gram model as shown in Equations 10,11.

$$T^* = \underset{T}{argmax}\ P(T|W) \quad (9)$$
$$= \underset{T}{argmax}\ P(W, T) \quad (10)$$
$$= \underset{T}{argmax} \prod_{i}^{n} P(w_i, t_i \mid w_{i-1}^{i-k+1}, t_{i-1}^{i-k+1}) \quad (11)$$

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

1. pizza_bl hut_bl new_cs york_cs new_cs york_cs
2. home_bl depot_bl around_null san_cs francisco_cs
3. please_null show_null me_null indian_bl restaurants_bl in_null chicago_cs
4. pediatricians_bl open_null on_null sundays_null
5. hyatt_bl regency_bl in_null honolulu_cs hawaii_cs

Table 1: A Sample set of annotated sentences

The FSA on the joint alphabet is converted into an FST. The paired symbols (w_i, t_i) are reinterpreted as consisting of an input symbol w_i and output symbol t_i. The resulting FST (M) is used to parse the 1-best ASR (represented as FSTs (I)), using composition of FSTs and a search for the lowest weight path as shown in Equation 12. The output symbol sequence (π_2) from the lowest weight path is T^*.

$$T^* = \pi_2(Bestpath(I \circ M)) \quad (12)$$

Equation 12 shows a method for parsing the 1-best ASR output using the FST. However, a similar method can be applied for parsing WCNs. The WCN arcs are associated with a posterior weight that needs to be scaled suitably to be comparable to the weights encoded in M. We represent the result of scaling the weights in WCN by a factor of λ as WCN^{λ}. The value of the scaling factor is determined empirically. Thus the process of parsing a WCN is represented by Equation 13.

$$T^* = \pi_2(Bestpath(WCN^{\lambda} \circ M)) \quad (13)$$

6 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 the two parsing models as discussed in the previous sections. We tested our approaches on three data sets, which in total include 2686 speech queries. These queries were collected from users using mobile devices from different time periods. Labelers transcribed and annotated the test data using SearchTerm and LocationTerm tags.

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 an ASR with a trigram-based language model trained on the query logs. Table 2 shows the ASR word accuracies on the three data sets. The accuracy is the lowest on Test1, in which many users were non-native English speakers and a large percentage of queries are not intended for local search.

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

Table 3 reports parsing performance using the PARIS approach for the two slots. The "Transcription" columns present the parser's performances on human transcriptions (i.e. word accuracy=100%) of the speech. As expected, the parser's performance heavily relies on ASR word accuracy. We achieved lower parsing performance on Test1 compared to other test sets due to lower ASR accuracy on this test set. The promising aspect is that we consistently improved SearchTerm extraction accuracy when using WCN as input. The performance under "Oracle path" column shows the upper bound for the parser using the oracle path[2] from the WCN. We pruned the 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 7.6% (82.5%-74.9%) absolute. Our proposed approach using pruned WCN achieved 2.7% improvement, which is 35% of the maximum potential gain. We observed smaller improvements on Test1 and Test3. Our approach did not take advantage of WCN for LocationTerm extraction, hence we obtained the same performance with WCNs as using ASR 1-best.

In Table 4, we report the parsing performance for the FST-based approach. We note that the FST-based parser on a WCN also improves the SearchTerm and LocationTerm extraction accuracy over ASR 1-best, an improvement of about 1.5%. The accuracies on the oracle path and the transcription are slightly lower with the FST-based parser than with the PARIS approach. The performance gap, however, is bigger on ASR 1-best. The main reason is PARIS has embedded a module for spelling correction that is not included in the FST approach. For instance, it corrects *nieman* to *neiman*. These improvements from spelling correction don't contribute much to search perfor-

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

Data Sets	SearchTerm Extraction Accuracy				LocationTerm Extraction Accuracy			
Input	ASR 1-best	WCN	Oracle Path 4	Transcription	ASR 1best	WCN	Oracle Path 4	Transcription
Test1	60.0%	**60.7%**	67.9%	94.1%	80.6%	80.6%	85.2%	97.5%
Test2	74.9%	**77.6%**	82.5%	98.6%	89.0%	89.0%	92.8%	98.7%
Test3	64.7%	**65.7%**	71.5%	96.7%	88.8%	88.8%	90.5%	97.4%

Table 3: Parsing performance using the PARIS approach

Data Sets	SearchTerm Extraction Accuracy				LocationTerm Extraction Accuracy			
Input	ASR 1-best	WCN	Oracle Path 4	Transcription	ASR 1best	WCN	Oracle Path 4	Transcription
Test1	56.9%	**57.4%**	65.6%	92.2%	79.8%	79.8%	83.8%	95.1%
Test2	69.5%	**71.0%**	81.9%	98.0%	89.4%	89.4%	92.7%	98.5%
Test3	59.2%	**60.6%**	69.3%	96.1%	87.1%	87.1%	89.3%	97.3%

Table 4: Parsing performance using the FST approach

mance as we will see below, since the search engine is quite robust to spelling errors. ASR generates spelling errors because the language model is trained using query logs, where misspellings are frequent.

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

In order to evaluate our voice search system, we submitted the two fields resulting from the query parser on the ASR output (1-best/WCN) to the search engine. We extracted the top 5 results from the returned pages and we computed the Precision, Recall and F1 scores between this set of results and the reference search set. Precision is the ratio of relevant results among the top 5 results the voice search system returns. Recall refers to the ratio of relevant results to the reference search result set. F1 combines precision and recall as: (2 * Recall * Precision) / (Recall + Precision) (van Rijsbergen, 1979).

In Table 5 and Table 6, we report the search performance using PARIS and FST approaches. The overall improvement in search performance is not as large as the improvement in the slot accuracies between using ASR 1-best and WCNs. On Test1, we obtained higher recall but lower precision with WCN resulting in a slight decrease in F1 score. For both approaches, we observed that using WCNs consistently improves recall but not precision. Although this might be counterintuitive, given that WCNs improve the slot accuracy overall. One possible explanation is that we have observed errors made by the parser using WCNs are more "severe" in terms of their relationship to the original queries. For example, in one particular

Data Sets		Precision	Recall	F1
ASR 1-best	Test1	71.8%	66.4%	68.8%
	Test2	80.7%	76.5%	78.5%
	Test3	72.9%	68.8%	70.8%
WCN	Test1	70.8%	67.2%	69.0%
	Test2	81.6%	79.0%	80.3%
	Test3	73.0%	69.1%	71.0%

Table 5: Search performances using the PARIS approach

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

Table 6: Search performances using the FST approach

case, the annotated SearchTerm is "book stores", for which the ASR 1-best-based parser returned "books" (due to ASR error) as the SearchTerm, while the WCN-based parser identified "banks" as the SearchTerm. As a result, the returned results from the search engine using the 1-best-based parser were more relevant compared to the results returned by the WCN-based parser.

There are few directions that this observation suggests. First, the weights on WCNs may need to be scaled suitably to optimize the search performance as opposed to the slot accuracy performance. Second, there is a need for tighter coupling between the parsing and search components as the eventual goal for models of voice search is to improve search accuracy and not just the slot accuracy. We plan to investigate such questions in future work.

7 Summary

This paper describes two methods for query parsing. The task is to parse ASR output including 1-best and lattices into database or search fields. In our experiments, these fields are SearchTerm and LocationTerm for local search. Our first method, referred to as PARIS, takes advantage of a generic search engine (for text indexing and search) for parsing. All probabilities needed are retrieved on-the-fly. We used keyword search, phrase search and proximity search. The second approach, referred to as FST-based parser, which encodes the problem of parsing as a weighted finite-state transduction (FST). Both PARIS and FST successfully exploit multiple hypotheses and posterior probabilities from ASR encoded as word confusion networks and demonstrate improved accuracy. These results show the benefits of tightly coupling ASR and the query parser. Furthermore, we evaluated the effects of this improvement on search performance. We observed that the search accuracy improves using word confusion networks. However, the improvement on search is less than the improvement we obtained on parsing performance. Some improvements the parser achieves do not contribute to search. This suggests the need of coupling the search module and the query parser as well.

The two methods, namely PARIS and FST, achieved comparable performances on search. One advantage with PARIS is the fast training process, which takes minutes to index millions of query logs and listing entries. For the same amount of data, FST needs a number of hours to train. The other advantage is PARIS can easily use proximity search to loosen the constrain of N-gram models, which is hard to be implemented using FST. FST, on the other hand, does better smoothing on learning probabilities. It can also more directly exploit ASR lattices, which essentially are represented as FST too. For future work, we are interested in ways of harnessing the benefits of the both these approaches.

References

C. Allauzen, M. Mohri, M. Riley, and B. Roark. 2004. A generalized construction of speech recognition transducers. In *ICASSP*, pages 761–764.

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

B. Favre, F. Bechet, and P. Nocera. 2005. Robust named entity extraction from large spoken archives. In *Proceeding of HLT 2005*.

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

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

L. Mangu, E. Brill, and A. Stolcke. 2000. Finding consensus in speech recognition: Word error minimization and other applications of confusion networks. *Computation and Language*, 14(4):273–400, October.

P. Natarajan, R. Prasad, R.M. Schwartz, and J. Makhoul. 2002. A scalable architecture for directory assistance automation. In *ICASSP 2002*.

B. Tan and F. Peng. 2008. Unsupervised query segmentation using generative language models and wikipedia. In *Proceedings of WWW-2008*.

C.V. van Rijsbergen. 1979. *Information Retrieval*. Boston. Butterworth, London.

Y. Wang, D. Yu, Y. Ju, and A. Alex. 2008. An introduction to voice search. *Signal Processing Magzine*, 25(3):29–38.

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