SPOKEN QUERY PROCESSING FOR INFORMATION RETRIEVAL

A. Moreno-Daniel[†]*, S. Parthasarathy*[‡]*, B. H. Juang*[†]*, J. G. Wilpon*[‡]

[†]Center for Signal and Image Processing
Georgia Institute of Technology, Atlanta, GA, USA

[‡]AT&T Shannon Labs, Florham Park, NJ, USA

ABSTRACT

This work proposes a way to integrate an information retrieval (IR) system with an automatic speech recognition (ASR) engine to support natural spoken queries. A broader interaction between the two modules is achieved by transmitting a lattice of terms to the IR system. This is in contrast with conventional systems where only the *best-path* recognition output is transmitted. Acoustic scores associated with the term-lattice are used to weigh the terms. A latent semantic indexing (LSI) scheme in which documents and terms are mapped to a single reduced *feature-space* with 400 semantic components is used. The conventional LSI method is nevertheless modified to allow the aforementioned broader interaction between acoustic hypothesis and semantic determination. The results show that the proposed method moderately outperforms the traditional approach for spoken queries formulated as casual phrases.

Index Terms— Speech recognition, information retrieval, spoken query processing

1. INTRODUCTION

Over the past two decades we have witnessed an information explosion, in part driven by modern computer networks, storage technology and the digitalization of media. Library and academic archives, hypertext, images, audio (music or speech), video (TV, movies), among others, are instances of such information. Ironically, a large portion of the information available is often unstructured and is difficult to categorize or to find the data of interest. Unless there is an effective mechanism for retrieving the information needed, it will remain in obscurity. The popularity of text search engines bespeaks the significance of this problem.

Under certain scenarios, users are expected to be in an office desk environment, where text input can be comfortably acquired. In the case of retrieving entertainment media, however, users are likely to be in a media room in front of a high-definition display and home theater equipment. With the advent of IPTV and on-line video clips, the need for an effective yet natural voice-enabled access to the vast availability of video information has become crucial. User queries for information in such entertainment environment are not expected to be sophisticated nor complex, but rather casual.

Some of the traditional approaches used for such entertainment scenario rely on the structure (if any) present in the data. Others rely on the availability of a keyboard to capture the user's input in text format. A number of speech-enabled methods process the spoken queries using an ASR engine, driven by a class-based language model, to produce a text sequence that is ultimately subject to a linguistic parsing (as it is commonly done for text input). However, the error level in the current ASR technology can easily break the fragile parsing scheme, not to mention that such parsing technology is not mature enough to deal with casual natural queries.

This leads us to the need of redefining and limiting the pursued degree of understanding to one that can be represented statistically as a semantic retrieval. Since in the past, retrieval has evolved from text-based data, under this framework ASR has been used simply as a speech-to-text transducer, with recognition and retrieval treated as two independent steps. The design of a broader bridge connecting these two steps can allow a more sensible integration, and subsequently lead to an enhanced and more robust abstraction of understanding than the existing practice.

Spoken query processing (SQP) is the task of retrieving the set of relevant documents from a collection to meet a user's need expressed as a spoken query. Unlike spoken document retrieval or SDR (its reverse problem), where there has been significant progress reported in the TREC evaluation [1], SQP is vulnerable to ASR errors because the length of the spoken queries is very short and lacks semantic redundancy [2, 3].

In this work, we propose a SQP method for retrieving information (media) in an entertainment scenario by statistically finding the semantics of the natural language under the IR framework; and by integrating the indexing scheme with a richer set of ASR output.

Section 2 briefly explains some background concepts and terminology in IR, as well as related techniques for the processing of natural spoken language for different applications. Section 3 explains the details for the proposed methodology, and then section 4 presents a set of experiments that analyze the different components of the algorithm. Finally section 5 discusses the conclusions as well as future work.

2. BACKGROUND

Depending on the target application, different strategies have been proposed for dealing with natural spoken language. Since performing full linguistic parsing on the recognized word sequence can be a difficult task, especially in the presence of ASR errors, some of them avoid this step by concentrating the analysis on only certain fragments of speech, from which a level of understanding can be inferred.

One example of such techniques, for the particular case of call routing, is HMIHY (how may I help you?®) [4]. From a transcribed customer dialog database, HMIHY learns a set of the salient fragments and their probabilistic inter-dependency to build a belief network (or Bayesian network) that ultimately leads to the destination

The first author did part of this work while attending the AT&T summer internship program.

call route. On the one hand, salient fragments help to discard parts of the recognized sequence that contribute very little towards understanding; on the other hand, these fragments automatically perform word-chunking by connecting a sequence of words into a single entity, therefore defining its own semantic.

Other techniques for natural spoken language processing are based on the information retrieval (IR) framework, originally developed for text-based input. Text, as a format for capturing the user's need, offers a clean channel to transport the message from the user to the query processing stage; therefore it is reasonable to rely on the string captured to determine what the user needs. The IR paradigm in general first builds an *index file* (off-line) by extracting the terms (or keywords) from the collection of documents (see figure 1), and listing these terms and the corresponding documents that contain such terms. When a text query is to be processed (on-line), it is represented by a set of terms extracted from it. With the *index file* in one hand, and the query terms in the other, the IR system then performs a match and produces a ranked list of documents. However speech, as a format for capturing the user's need, is not necessarily a clean channel; recognition errors introduce noise that propagates to subsequent steps. Traditional methods based on IR simply attach a speech-to-text ASR engine to the front end of a text-based information retrieval system.

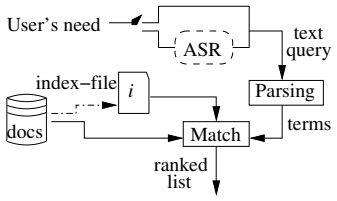

Fig. 1. Traditional text-based IR paradigm for written or spoken queries.

The performance of IR systems is typically measured by their *recall* (nr. relevant documents retrieved / tot nr. relevant documents in the database) and *precision* (nr. relevant document retrieved / tot nr. documents retrieved). Depending on the application, *recall* and *precision* need to be balanced. In some cases, *recall* is not feasible to obtain because the actual number or relevant documents in the database is unknown.

3. METHODOLOGY

The design strategy for the proposed SQP method is to 1) represent the semantics of the queries using a statistical model so that natural language parsing is unnecessary 2) exploit a richer representation from the ASR output and integrate this with the retrieval process; and 3) collapse groups of words into single compound terms.

This SQP method is based on the IR framework with latent semantic indexing [5] (LSI) as the indexing scheme. The stochastic nature of the spoken queries, along with the statistical abstraction of semantics by LSI, suggest that an integrated framework can be devised for processing queries that have a degree of uncertainty. The proposed SQP method is a step in this direction.

The mechanics of the algorithm can be broken into two parts: pre-query and post-query (off-line and on-line respectively). The pre-query step is done once. It processes the static data (documents) and prepares the elements needed for the post-query step. The post-query step processes the spoken query and it is repeated for every spoken query instance. A summary of the steps is presented below followed by a detailed explanation.

Pre-query steps:

A1. Collect the set of text documents: $\mathcal{D}$ (let $\mathcal{V}_1$ be the vocabulary).
A2. Build a word n-gram language model from $\mathcal{D}$.
A3. Represent the j-th document in $\mathcal{D}$ as the column vector:
$$\mathbf{d_j} = \begin{bmatrix} d_{1,j} \\ \vdots \\ d_{I_1,j} \end{bmatrix}; \; d_{i,j} = \text{occurrence of } w_i \text{ in } d_j, \text{ and } I_1 = |\mathcal{V}_1|.$$
A4. Assemble the co-occurrence matrix as: $C_1 = [\mathbf{d_1}, \ldots, \mathbf{d_J}]$.
A5. Map $\mathcal{V}_1 \rightarrow \mathcal{V}_2$, forming:
$C_2 = \mathrm{T}_{2\mathrm{grm}}\{\mathrm{T}_{\mathrm{stem}}[\mathrm{T}_{\mathrm{stop}}(C_1)]\}$; let $I_2 = |\mathcal{V}_2|$.
A6. Normalize C_2 intra- and inter-document wise to obtain W.
A7. Decompose W using SVD and reduce its rank to ρ:
$W \approx USV'$.

Post-query steps:

B1. Perform ASR on the spoken query and obtain recognition lattice: R.
B2. Normalize R by removing the residual cost.
B3. Map R from $\mathcal{V}_1 \rightarrow \mathcal{V}_2$, and extract the terms ($t_i, i = 1, \ldots, I_2$) of interest along with their expected path cost in $\mathcal{G}$.
$\mathcal{G} = \mathrm{Det}_{\log}\left\{\Pi_1\left[\Pi_1\left(R^1 \circ T_P\right) \circ T_S\right] \circ T_F\right\}$.
B4. Build a WBOT.
B5. Create the term-vector:
$$\mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_{I_2} \end{bmatrix}; \text{ where } x_i = epc(i)$$
B6. Find the closest documents in the feature space.

In steps **A1**-**A2**, $\mathcal{D}$ is simply the collection of documents from which an initial statistical language model (word n-gram) is created. The steps **A3**-**A4** represent the entire $\mathcal{D}$ as the co-occurrence matrix C_1, which is a sparse matrix of size I_1xJ.

In **A5**, the vocabulary $\mathcal{V}_1$ suffers three transformations: 1) stop-word removal ($\mathrm{T}_{\mathrm{stop}}$), 2) word-stemming ($\mathrm{T}_{\mathrm{stem}}$), and 3) bi-gram chunking ($\mathrm{T}_{2\mathrm{grm}}$). The transformation $\mathrm{T}_{\mathrm{stop}}$ discards words from a predefined list of stop-words such as articles or prepositions, $\mathrm{T}_{\mathrm{stem}}$ maps words with similar root into a single term, and $\mathrm{T}_{2\mathrm{grm}}$ adds the most common bi-grams to the vocabulary as compound terms. As a result, rows in C_1 are removed, merged and combined forming C_2 with a new vocabulary: $\mathcal{V}_2$ with terms: $t_1, \ldots, t_{I_2}$.

Step **A6** builds a normalized matrix W, where $\omega_{i,j}$ (i-th row and j-th column of W) is the product of two normalized factors: $\omega_{i,j} = \omega_a(i,j) \cdot \omega_e(i)$, corresponding to the intra- and inter-document normalization.

One scheme is TF-ϵ (term-frequency normalized-entropy) where the intra-document normalization factor (ω_a) simply consists of the term frequency (tf) within the document:

$$\omega_a(i,j) = tf = \frac{c_{i,j}}{\sum_i c_{i,j}}, \tag{1}$$

(c_{ij} is an element of the matrix C_2); and the inter-document normalization factor (ω_e) is defined as one minus the normalized term entropy (ϵ) across all the documents:

$$\omega_e(i) = 1 - \epsilon = 1 + \frac{1}{\ln J} \sum_{j=1}^{J} \frac{c_{i,j}}{\sum_j c_{i,j}} \ln \frac{c_{i,j}}{\sum_j c_{i,j}}. \tag{2}$$

Another popular normalization scheme is TF-IDF (term-frequency inverse-document-frequency) simply defined as:

$$\omega_{i,j} = tf \cdot idf = \frac{c_{i,j}}{\sum_i c_{i,j}} \cdot \log \frac{J}{n}, \tag{3}$$

where n is the number of documents where the term t_i is found.

For both schemes, the resulting normalized value of $\omega_{i,j}$ is a measure of the degree in which the term t_i characterizes the document $\mathbf{d}_j$.

Step **A7** uses SVD (singular value decomposition) to decompose W and reduce its rank to ρ, yielding three matrices such that $W \approx USV'$. The columns of U and V are a set of orthonormal vectors, on the *term-space* and on the *document-space* respectively, pointing in the direction of ρ basis vectors that span a ρ-dimensional *feature-space*. The matrix S is a diagonal matrix with the largest ρ eigenvalues representing the variance of each component on the *feature-space*.

An arbitrary *term-vector*, $\mathbf{x}$, can be projected into the *feature-space* as follows:

$$\tilde{\mathbf{x}} = U' \cdot \mathbf{x}. \tag{4}$$

Similarly, the rows of V are the document vectors in the *feature-space*,

$$V' = \left[\tilde{\mathbf{d}}_1, \tilde{\mathbf{d}}_2, \ldots, \tilde{\mathbf{d}}_J\right]. \tag{5}$$

Once a spoken query is available (**O**), the step **B1** performs ASR, using the language model built in **A2**, to generate the recognition lattice: R, a data structure that in addition to representing a set of possible recognized paths: π^k (word sequences), it also stores their acoustic score:

$$\ell(\pi^k|\mathbf{O}) = p(\mathbf{O}|\pi^k) \cdot p(\pi^k)^\gamma, \quad \pi^k \in R, \tag{6}$$

$$\mathcal{C}(\pi^k) = -\log \ell(\pi^k|\mathbf{O}), \tag{7}$$

where γ is the grammar scale factor and $\mathcal{C}(\pi^k)$ is the path cost. The *best-path* is the π^k with the lowest cost:

$$\pi^\beta = \arg\min_{\pi^k} \mathcal{C}(\pi^k). \tag{8}$$

Step **B2** normalizes the paths in R by removing the residual cost from the lattice (i.e. $\mathcal{C}(\pi^\beta) = 0$), resulting in the normalized lattice R^1.

The goal of step **B3** is to extract terms of interest (listed in $\mathcal{V}_2$) from R^1 with the associated expected path counts given by:

$$epc(i) = \mathcal{E}[c(t_i|\mathbf{O})] = \sum_{\pi^k \in Q_i} \ell(\pi^k|\mathbf{O}) c(t_i|\pi^k), \tag{9}$$

where Q_i is the set of all paths with term t_i and $c(t_i|\pi^k)$ is the count of t_i in path π^k. Stop-word removal, word-stemming, bi-gram chunking, and the expected count calculation can be represented using WFSM (weighted finite state machine) operations [6] as follows:

$$\mathcal{G} = \mathrm{Det}_{\log}\left\{\Pi_1\left[\Pi_1\left(R^1 \circ T_P\right) \circ T_S\right] \circ T_F\right\}, \tag{10}$$

where T_P, T_S, and T_F are transducers which perform the functions of stop-words removal, mapping words to their root term, and extracting terms of interest, respectively. The result is an acceptor ($\mathcal{G}$) in which each path contains a relevant term t_i with its associated expected cost ($-\log epc(i)$).

The terms with lowest cost in $\mathcal{G}$ correspond to segments where the paths converge within the lattice, therefore $\mathcal{G}$ can be pruned by keeping only the terms with the lowest cost. The pruning criterion selected was three times the number of terms found in π^β.

The step **B4** builds a "weighted bag of terms" (WBOT), which is the list of the terms found in $\mathcal{G}$ along with their expected path counts ($epc(i)$), serving as a confidence measure.

Next, step **B5** represents the spoken query as the *term-vector*: $\mathbf{x}$, where the components are determined by the WBOT, resulting in a vector that points more in the direction of the terms with the largest recognition confidence. Moreover, this vector is then projected into the *feature-space* using U as shown in equation 4, representing the spoken query as $\tilde{\mathbf{x}}$.

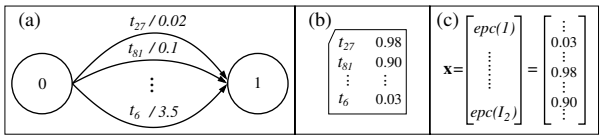

Fig. 2. Sub-figure (a) shows a sample acceptor $\mathcal{G}$, (b) shows the corresponding WBOT and (c) shows the query *term-vector*: $\mathbf{x}$.

The final step (**B6**) finds the "closest" documents in the *feature-space* ($\tilde{\mathbf{d}}_j$) as a ranked list. The distance measure used was defined as:

$$\mathrm{d}(\tilde{\mathbf{x}}, j) = \cos^{-1}\left(\frac{\tilde{\mathbf{x}}' \cdot \tilde{\mathbf{d}}_j}{\|\tilde{\mathbf{x}}\| \, \|\tilde{\mathbf{d}}_j\|}\right). \tag{11}$$

4. EXPERIMENTS

4.1. Data set: movies

The case study selected for testing the proposed algorithm is a movie retrieval system, where the user input is a spoken query and the outcome is a set of relevant movies. The data set collected consists of documents (text) made of the movie synopses, and spoken queries (speech) that search for a particular attribute of interest.

For the purpose of analysis and development, two thousand movies were randomly chosen from an Internet archive. Each of these synopses has an average of 94 words (a couple of paragraphs) that act as meta-data descriptors of the content of the movies. Additionally, the source database provides a set of "keywords" for each movie; these are appended twice to their corresponding synopsis paragraphs to emphasize their relevance.

A total of 28 speakers volunteered to record two types of spoken utterances: *q1* and *q2*. A set of 120 attributes were selected from the list of bi-grams and mono-grams taken from the documents (synopses). The *q1* and *q2* sets are made of 180 utterances each. The *q1* queries simply say the attributes as keywords in isolation. The *q2* queries embed these keywords in a suitable casual phrase.

type	description	example
q1	prompted isolated keywords	*car crash*
q2	prompted casual phrase	*show me a movie with car crash*

The goal of these experiments is to conduct an analysis and diagnosis of the proposed algorithm by drawing relative comparisons of results under several configurations; rather than a comprehensive evaluation with absolute results.

4.2. Evaluation criteria

For many ASR applications, the performance of the system is measured by comparing the hypothesized outcome against a ground truth reference created manually in advance by an expert. Another way for performing such evaluation is to wait for the system's outcome, then let the expert "grade" it.

In our work, we used the latter scheme under a consumer-oriented evaluation strategy, where the worth of merit of a given outcome

is determined from the user perspective. The evaluators were instructed to inspect the eight top retrieved entries obtained for each of the spoken queries. The criterion was built under the assumption that his/her job was to grade the outcome of a machine-generated list of suggested movies given a customer's query, in terms of how useful these movies are in relation to the user's original need. The evaluators chose between one of four choices: *relevant*, *somewhat-relevant*, *irrelevant*, *don't-know*.

In IR terms, this evaluation relates to *precision* because it is limited to only inspect the retrieved set of documents. A measure of *recall*, in our data set, would require an exhaustive (expensive) search in the entire database for the relevance of each of the 120 attributes; which quickly becomes an unfeasible task as the database expands.

4.3. Results

In the first set of experiments, we examine the effect some elements of the pre-query step (co-occurrence matrix normalization and *feature-space* dimension: ρ) have in the evaluation results.

q1						
$\rho =$	**200**		**300**		**400**	
	tf-idf	tf-ϵ	tf-idf	tf-ϵ	tf-idf	tf-ϵ
relevant [%]	53	54	55	56	58	58
some-rel [%]	22	23	23	23	23	23
irrelevant [%]	24	23	21	21	19	19
don't-know [%]	1	1	1	0	0	0

q2						
$\rho =$	**200**		**300**		**400**	
	tf-idf	tf-ϵ	tf-idf	tf-ϵ	tf-idf	tf-ϵ
relevant [%]	45	46	49	50	52	52
some-rel [%]	19	19	20	19	19	19
irrelevant [%]	36	35	30	30	29	29
don't-know [%]	1	0	1	1	0	1

Table 1. Pre-query step analysis. Evaluation results for the two types of queries: *q1* (isolated keywords) and *q2* (keywords in casual phrase), two types of normalization schemes for the occurrence matrix: TF-IDF and TF-ϵ, and a *feature-space* with $\rho = \{200,300,400\}$.

From the results shown in Table 1, it is observed that both normalization schemes (tf-idf and tf-ϵ) are comparable under all the scenarios studied. In general, queries type *q1* consistently outperform *q2*. This is an intuitive result since *q2* queries have to deal with filler words that may not be well recognized. The number of *relevant* entries gradually increased as the dimensionality of the *feature-space* was set to 200, 300 and 400 dimensions; however the number of *somewhat-relevant* entries does not change significantly.

Now, if we let $\rho = 400$ and TF-ϵ be the base configuration for the pre-query step, the post-query step can be analyzed for different configurations of the WBOTs (weighted bag of terms).

The simplest configuration is *WBOT-0*, where the bag of terms is the list of equally weighted terms found in π^{β} (the *best-path*). *WBOT-0* corresponds to the traditional plain text interface between ASR and the IR scheme. On the other hand, the proposed method defines *WBOT-1* as a list terms found in the entire lattice with the largest expected path counts (as defined in Eq. 9) and along with their corresponding weight as it is done in step **B4**.

The results shown in Table 2 suggest that the performance is comparable for *WBOT-0* and *WBOT-1* in queries type *q1* (isolated keywords). Since the utterance in *q1* is relatively short (1-3 words),

	q1		***q2***	
	WBOT-0	*WBOT-1*	*WBOT-0*	*WBOT-1*
relevant [%]	57	58	48	52
some-rel [%]	23	23	20	19
irrelevant [%]	19	19	31	29
don't-know [%]	1	0	1	1

Table 2. Post-query step analysis. Evaluation results for the two types of queries: *q1* and *q2*, and two definitions of *WBOTs*.

there are not enough alternative paths (π^k, $k \neq \beta$) from which words, with a high expected path count, can be considered.

For queries type *q2* (casual phrases), the proposed SQP algorithm (*WBOT-1*) obtained an increase in the number of retrieved entries labeled as *relevant* (from 48% to 52%) and a slight drop in the number of entries labeled as *somewhat-relevant*. *WBOT-1*, in contrast to *WBOT-0*, considers terms out the best-path, and discards or gives low weight to words with low expected path count (epc) as defined in Eq. 9.

5. CONCLUSION AND FUTURE WORK

This work proposed an algorithm for SQP based on IR and LSI as the indexing scheme. This method exploits a richer representation of the ASR output (recognition lattice) to integrate the ASR with the retrieval process. The algorithm was tested on a movie retrieval task and the outcome was subject to a customer-oriented evaluation.

The results show that, for spoken queries formulated as casual phrases, this technique was able to obtain a moderate improvement (4%) in the number of *relevant* entries with respect to the traditional method; while dropping slightly (1%) in the number of *somewhat-relevant* entries. In the case of spoken queries formulated as isolated keywords, the performance of the traditional approach was matched. This work represents a step in the design of an integrated framework for processing queries of stochastic nature (such as spoken queries).

Future work includes establishing a formal evaluation method and an integrative theoretical framework that will lead to an even closer interaction between the ASR engine and the retrieval system.

6. ACKNOWLEDGMENTS

The first author thanks M. Johnston, B. Renger, M. Levine and B. Hollister for collaborating in the data collection.

7. REFERENCES

[1] J. S. Garofolo, C. G. P. Auzanne, and E. M. Voorhees, "The TREC spoken document retrieval track: A success story," *Proceedings of TREC-8*, pp. 107–130, Apr. 2000.

[2] J. Barnett, S. Anderson, J. Broglio, M. Singh, R. Hudson, and S. W. Kuo, "Experiments in spoken queries for document retrieval," *Eurospeech*, pp. 1323–1326, 1997.

[3] F. Crestani and H. Du, "Written versus spoken queries: A qualitative and quantitative comparative analysis," *J. Am. Soc. Inf. Sci. Technol.*, vol. 57, no. 7, pp. 881–890, 2006.

[4] A. L. Gorin, G. Riccardi, and J. H. Wright, "How may I help you?," *Speech Communication*, vol. 23, no. 1/2, pp. 113–127, 1997.

[5] J. R. Bellegarda, "Latent semantic mapping," *IEEE Signal Processing Magazine*, vol. 22, no. 5, pp. 70–80, Sept. 2005.

[6] M. Mohri, "Finite-state transducers in language and speech processing," *Computational Linguistics*, vol. 23, no. 2, pp. 269–311, June 1997.