PHAST: Spoken Document Retrieval Based on Sequence Alignment

Pere R. Comas
TALP Reseach Centre
Technical University of Catalonia (UPC)
pcomas@lsi.upc.edu

Jordi Turmo
TALP Reseach Centre
Technical University of Catalonia (UPC)
turmo@lsi.upc.edu

ABSTRACT

This paper presents a new approach to spoken document information retrieval for spontaneous speech corpora. Classical approach to this problem is the use of an automatic speech recognizer (ASR) combined with standard information retrieval techniques, based on terms or n-grams.

However, state-of-the-art large vocabulary continuous ASRs produce transcripts of spontaneous speech with a word error rate of 25% or higher, which is a drawback for retrieval techniques based on terms or n-grams. In order to overcome such a limitation, our method is based on a sequence alignment algorithm drawn from the field of bioinformatics to search "sounds like" sequences in the document collection. These matching sequences are potentially misrecognized words from the ASR and can be used to retrieve relevant passages and documents from the collection. Our approach doesn't depend on extra information provided by the ASR.

We have evaluated and compared our approach to others in the state of the art in both spoken document retrieval and spoken passage retrieval tasks. The evaluation has been performed in the context of Question Answering using a corpus of automatic transcripts from the Spanish and European parliaments. The results show that our method outperforms by 10 points traditional term based search and n-gram search on automatic transcripts.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Search process

General Terms

Algorithms

Keywords

Information retrieval, Spoken document retrieval, Spontaneous speech retrieval, Sequence alignment, Approximate matching, Phonetic distance

1. INTRODUCTION

Since affordable technology allows the storage of large masses of audio media, more and more spoken document sources become available to public access. This great body of spoken audio recordings is mainly unaccessible without accurate techniques of retrieval. Spoken document retrieval (SDR) is the task of retrieving passages from collections of spoken documents according to a user's request or query.

Classically, the approach to SDR problem is the integration of an automatic speech recognizer (ASR) with information retrieval (IR) technologies. The ASR produces a transcript of the spoken documents and these new text documents are processed with standard IR algorithms.

These techniques assume the correctness of the words in the documents and the existence of structural information such as punctuation marks, paragraph boundaries, and capitalization. However, the speech recognition stage introduces errors that challenge traditional IR algorithms.

Any ASR is limited in the size of the vocabulary it can recognize. This vocabulary depends on the amount of audio data used in its training. Therefore the ASR can not recognize all possible words, these are the so called Out of Vocabulary (OOV) words. The ASR transcribes the audio corresponding to these words as other sequence words in its closed vocabulary. Words such as proper names tend to be OOV and this is an additional difficulty since this words typically occur in user queries. OOV words may be very important for some applications as we will show lately.

Recent results show that a reasonable approach to SDR consists in taking the one-best output of ASR (i.e., the most probable sequence of words that generates the input audio) and performing IR on this transcript. It works reasonably well when recognition is mostly correct and documents are long enough to contain correctly recognized query terms.

TREC conference had a spoken document retrieval task using a corpus composed of 550 hour of Broadcast News. In this scenario, documents are long enough and speech clearness allows accurate transcription. TREC 2000 edition concluded that spoken news retrieval systems achieved almost the same performance as traditional IR systems [4].

The Spoken Document Retrieval track in CLEF evaluation campaigns uses a corpus of spontaneous speech for cross-lingual speech retrieval (CL-SR) [22, 13]. CL-SR task challenges the participants to work with noisy automatic transcripts and a more general scenario than the former TREC tracks. CL-SR corpus is composed of nearly 600 hours of spontaneous speech from interviews with Holocaust survivors. The queries were topic descriptions consisting of a title, a short description and a narrative passage. Top ranked participants in CL-SR, see [2, 8, 6, 20], used a wide range of traditional text based IR techniques. Good results were achieved with term-based ranking schemes such Okapi BM25 [15], Divergence From Randomness [3] and Vector Space Models [16]. Most of the work done by the participants was focused on investigating the effects of meta-data, hand-assigned topics, query expansion, thesauri, side collections and translation issues. Some participants used n-gram based search instead of term search. For n-gram search, text collection and topics are transformed into a phonetic transcription, then consecutive phones are grouped into overlapping n-gram sequences, and finally they are indexed. The search consists in finding n-grams of query terms in the collection. Some experiments show how phonetic forms helps to overcome recognition errors. Some results using phonetic n-grams are reported in [7] showing only slightly improvements.

Experimental results show that the traditional approach consisting of ASR and IR is not useful if the task requires the retrieval of short speech segments in a domain with higher word error rate. In this cases, other approaches to SDR have been proposed. Most try to improve retrieval performance using additional information specific to the ASR. For example, Srinivasan and Petkovic [19] use an explicit model of the ASR error typology to address the OOV problem. First, they use two ASRs to generate a word transcript and a phonetic transcript of the input audio. Then they build a phone confusion matrix that models the probability of ASR mistaking any phone for a different one. Finally, the retrieval step uses a Bayesian model to estimate the probability that the phonetic transcript of a speech segment is relevant to the query term.

Another common approach is the use of ASR lattices to make the system more robust to recognition errors. The lattice contains all possible outputs given the audio input. For example, experiments in [18] report an improvement of 3.4% in F_1 measure in Switchboard corpus using a combination of word-lattices and phone-lattices as search space. The use of word-lattices alone cannot overcome the problem of OOV words.

In this paper we present a different method for speech retrieval. With this approach, we try to overcome drawbacks such as OOV words and specific ASR dependency with an algorithm that requires no training data. Briefly, after transforming the transcribed text into a phonetic transcription using a text-to-speech tool, we use a measure of phonetic similarity to find regions in the data of high phonetic resemblance with the query terms. These regions are candidates to contain terms even if the words are misrecognized, and standard IR techniques can be applied to rank the spoken documents. This search can be done in a reasonable efficient time using an approach similar to the search of patterns in DNA sequences. We call this method PHonetic Alignment Search Tool (PHAST).

The rest of this paper is organized as follows. Section 2 introduces some background concepts about algorithms relevant to our work. Section 3 describes PHAST and Section 4 discusses experimental setting and compares the results achieved by PHAST and state-of-the-art approaches. Finally, Section 5 concludes the paper.

2. BACKGROUND

We are interested in building an IR engine for a factual Question Answering system over spoken documents. Question Answering (QA) is the task of finding exact answers to questions formulated in natural language in a document collection. In factual QA, the answer is a named entity (e.g. a proper noun, a date, a measure). Factual QA is specially concerned with OOV words since most of named entities tend to be out of the vocabulary.

In the traditional approach to QA, a first step involves the retrieval of documents or passages potentially containing the answer to a question. In further steps the exact answer is extracted using a wide range of linguistically motivated algorithms. These steps are computationally expensive. Therefore the IR step acts as a filter reducing the document collection to a smaller search space. The very same task can be performed over a collection of spoken documents.

For simplicity, the following sections briefly describe some concepts relevant to our word and related to QA and ASR technologies.

2.1 Automatic Speech Recognition

We access to spoken documents using an ASR. Given an acoustic signal the ASR searches for the most likely word sequence that could produce the signal. The ASR uses two statistical models for this task: an *acoustic model*, which relates signal and phones, and a *language model*, which estimates the probability of a certaint sequence of words.

Generally, ASR performance may vary depending on conditions. For open-domain spontaneous-speech speaker-independent recognition, Word Error Rate (WER) greater than 25% is expected. ASR may produce three kinds of transcription errors: some spoken words are missing in the transcript, some spurious words are added to the transcript, or some spoken words are transcribed as another words. These problems often occur together: because ASR systems try to map sounds into words, ASR recognition errors generally lead to mapping sounds into incorrect words with phonetic similarity to correct words.

Figure 1 shows manual and automatic transcript for three spoken utterances. A first problem is that correctly transcribing a word becomes more difficult when the word is unfrequent (i.e., it has small probability). A second problem is OOV words, all those words not belonging to the language model will never appear in the transcript. Proper names suffer from this problem as exemplifies the third sentence in Figure 1. In this sentence, the word “UNIX” is misrecognized as “unique set”. This is due to the fact that “UNIX” is pronounced as [juniks][1] while “unique set” is [juniksɛt]. The only difference is that the automatic transcript contains two extra phones. Experience shows that most of the missing or misrecognized words are content-bearing words such as “UNIX”, which may be crucial for accurate retrieval. Spurious words added by the ASR tend to be common words with little influence in retrieval [20].

Our approach for IR relies on this phonetic likeliness between misrecognitions and the actual spoken words to set an approximated matching between them. This likeliness is due to the ASR architecture and not to a specific ASR model, and is also valid for OOV words.

2.2 Retrieval in QA

Information retrieval module is one of the main steps in QA. The set of retrieved documents/passages must contain the answer to the question. This means that retrieval needs a recall high enough to guarantee the presence of the answer.

[1]In this paper we have used the international phonetic alphabet (IPA) notation for phonetic transcriptions within brackets.
http://www.arts.gla.ac.uk/IPA/

1M: *"The pattern frequency relevance rate indicates the ratio of relevant documents..."*
1A: *"the putt and frequency illustrating the case the ratio of relevant documents..."*
2M: *"Documents must be separated into relevant documents and irrelevant documents by a manual process, which is very time consuming."*
2A: *"documents must be separated into relevant documents and in relevant document by a manual process witches' of very time consuming"*
3M: *"The host system it is a UNIX Sun workstation"*
3A: *"that of system it is a unique set some workstation"*

Figure 1: Examples of manual (M) and automatic (A) transcripts

It is also desirable to achieve precision enough to reduce computational overhead on further steps.

This section describes the baseline retrieval algorithm we have used for our experiments in document and passage retrieval (cf., Section 4). The reason by which we have selected this algorithm among others is that empirical studies show how this algorithm achieves better results for QA than traditional document ranking schemes [14].

Given a sequence of keywords $S = [k_1, k_2, \ldots, k_n]$ sorted according to priority, the algorithm transforms the sequence into boolean queries to be sent to the IR engine in an iterative process. The first m keywords in the sequence define the query $(k_1 \wedge k_2 \wedge \ldots \wedge k_m)$ and are sent to the IR engine. Also, a word distance threshold t is set in order to produce passages of high keyword density. If the number of returned passages or documents is above or below some fixed thresholds $rMax$ and $rMin$, new keywords may be added/dropped from the query and it is sent to the IR engine again.

Figure 2 is an example of passage retrieval. It shows sentence 2M from Figure 1 and a query containing the keywords "relevant", "document", and "process".

The distance between consecutive keywords within a passage must be smaller than t words. A passage is formed in a segment with high keyword density (i.e., a segment containing all keywords where two consecutive keyword occurrences are separated by at most t words). Note that first occurrence of "documents" is too far from "relevant" to belong to the passage.

Parameters $rMax$, $rMin$ and t may be tuned to the document collection and system characteristics (see [14] for further details). With this dynamic query adjustment, an unordered set of relevant passages or documents is generated. The complete set can be subsequently processed to extract any possible answer, task which is not of the scope of this paper. In section 4, this algorithm is referred to as DQ.

3. OUR RETRIEVAL APPROACH

This section shows how an approximated search based on phonetic similarity can be efficiently implemented using a sequence alignment algorithm drawn from the field of bioinformatics.

The recent sequencing of a large number of genomes has greatly stimulated the development of computational methods for the identification of patterns in biological sequences. Hence a family of pattern-matching algorithms for sequence comparison (sequence homology of proteins) in large databases has recently been developed. The most succeeding algorithm in the field is called BLAST (Basic Local Alignment Search Tool) [1]. BLAST employs a measure based on well-defined mutation scores to find regions of local similarity in protein sequences. BLAST is a simple and robust algorithm that can be applied in a variety of contexts[2].

Following the approach of BLAST, we have implemented PHAST, an IR-engine over large phonetic sequences based on the same principles. Our hypothesis is that it is possible to find the best matchings of the keywords by searching for small contiguous substrings of phones (hooks) in the transcript, extending them and computing its relevancy.

PHAST algorithm has some advantageous properties for dealing with SDR: finds approximated matchings with independence of sub-word length, it can easily split/merge sequences, and no training data is required. In PHAST, this process is language independent given an appropriated set of phones.

Algorithm 1 shows a general view of PHAST scheme. It is a two-step process: First, term frequency is calculated using phonetic similarity, and second, a standard document ranking process takes place.

Algorithm 1
PHAST algorithm
Parameter: $\mathcal{D}$, collection of phonetically transcribed documents
Parameter: $\mathcal{KW}$, set of phonetically transcribed keywords

```
1: for all $d \in \mathcal{D}, w \in \mathcal{KW}$ do
2:   while $h = detection_\phi(w, d)$ do
3:     $s = extension_\varphi(w, h)$
4:     if $relevant(s, h)$ then
5:       update $tf(w, d)$
6:     end if
7:   end while
8: end for
9: Rank collection $\mathcal{D}$
```

The input data is a collection of documents transcribed into phonetic sequences $\mathcal{D}$, and a set of keywords phonetically transcribed $\mathcal{KW}$. There are three important functions to describe in this algorithm.

- $detection_\phi(w, d)$ is a function that detects hooks within document d considering keyword w. Different functions ϕ can be used to detect hooks as will be discussed in Section 3.1.1.
- $extension_\varphi(w, d, h)$ is a function that extends hook h and computes an accurate similarity score s between keyword w and document d around h. Different functions φ can be used as will be discussed in Section 3.1.2.
- $relevant(s, h)$ judges how this occurrence of w at h with score s is relevant enough for term frequency. It triggers an update procedure relative to the document ranking process to be used. This will be discussed in Section 3.1.3.

[2] http://www.ncbi.nlm.nih.gov/BLAST/

Keywords: relevant, documents, process

$$\dots\ \textbf{\textit{documents}}\ \underbrace{\textit{must be separated into}}_{\text{distance}>t}\ \overbrace{\textbf{\textit{relevant documents}}\ \textit{and irrelevant}\ \textbf{\textit{documents}}\ \underbrace{\textit{by a manual}}_{\text{distance}<t}\ \textbf{\textit{process}}}^{\text{Passage}}\textit{, which is a very}\ \dots$$

Figure 2: Example of passage building

Reference transcript:
"The host system it is a UNIX Sun workstation"
Automatic transcript:
"that of system it is a unique set some workstation"

	junik	← *detection*$_\phi$	
...ðæt ʌβ sɪstəm ɪt ɪz ə	junik	sɛt sʌm	wəʊrksteɪʃən...
	junik	s sʌn	← *extension*$_\varphi$

Figure 3: Search of term "UNIX-Sun"

Finally documents in $\mathcal{D}$ are ordered according to a ranking measure.

Figure 3 shows an example of how functions *detection*$_\phi$ and *extension*$_\varphi$ are used. Document d is the sentence 3A from Figure 1, which has been transcribed to a sequence of phones. The query word w is the term *"UNIX-Sun"*, which is transcribed as [juniks sʌn]. Term w exists in the manual transcript 3M but not in the automatic transcript 3A. In the first step, *detection*$_\phi$ finds hook [junik] related to [juniks sʌn]. In the second step, *extension*$_\varphi$ extends the hook by matching the rest of [juniks sʌn] with the phones surrounding [junik] in the sentence.

The following sections discuss in detail first and second steps.

3.1 Keyword search

The following sections describe the three functions *detection*$_\phi$, *extension*$_\varphi$ and *relevant*.

3.1.1 Detection of Hooks

In order to efficiently detect the occurrences of a phonetic sequence a in a phonetic sequence b, *detection*$_\phi(a, b)$ is sequentially computed over the corpora. Function ϕ has been implemented following an approach similar to the one presented by Altschul [1].

Given a set of phonetically transcribed keywords, a deterministic finite automaton [5] DFA$_k$ is automatically built for each keyword k in order to recognize all its possible substrings of n phones.

For instance, given $n = 3$ and the keyword "alignment", which is phonetically transcribed as [əlaɪnmɪnt], there are seven phonetic substrings of length three (3-grams): əla, laɪ, aɪn, ɪnm, nmɪ, mɪn and ɪnt. One DFA is automatically built to recognize all seven 3-grams at once. Table 1 shows the transition table of this DFA.

Using these DFAs, the collection is scanned once to search for all the hooks. When a hook is found, a process for extending it is executed. This process is described in the following section.

3.1.2 Extension of Hooks

After a hook is found, PHAST uses φ to extend each hook

State	ə	a	ɪ	l	m	n	t	*
0	15	6	10	4	3	1	0	0
1	15	6	10	4	2	1	0	0
2	15	6	13	4	3	1	0	0
3	15	6	19	4	3	1	0	0
4	15	5	10	4	3	1	0	0
5	15	6	18	4	3	1	0	0
6	15	6	7	4	3	1	0	0
7	15	6	10	4	3	8	0	0
+8	15	6	10	4	12	1	9	0
+9	15	6	10	4	3	1	0	0
10	15	6	10	4	3	11	0	0
11	15	6	10	4	12	1	9	0
+12	15	6	13	4	3	1	0	0
+13	15	6	10	4	3	14	0	0
+14	15	6	10	4	12	1	9	0
15	15	6	10	16	3	1	0	0
16	15	17	10	4	3	1	0	0
+17	15	6	18	4	3	1	0	0
+18	15	6	10	4	3	8	0	0
19	15	6	10	4	3	14	0	0

Table 1: Transition table of a DFA recognizing all the 3-grams in [əlaɪnmɪnt]. Final states are marked with '+'. Initial state is 0

h and to compute its score value s. Following the approach of Altschul we have taken the edit distance (Levenshtein distance [12]) as φ function. Recent works have successfully used edit distance as a measure of phonetic similarity.

The process is as follows: when a hook h is found, the edit distance is calculated between the query term w and a subsequence of the document d around the position where h was found. This yields an score for the similarity of this occurrence of w in d.

Given two sequences a and b of lengths n and m, the edit distance algorithm can be computed by means of a dynamic programming algorithm as follows. In a first step, a distance matrix D is computed. $D[i, j]$ holds the minimal distance between the prefixes of lengths i and j of sequences a and b, respectively. Initially, the first row and first column of the matrix are filled with a multiple of the *indel* cost. Then each new element in D is calculated recursively for longer substrings

$$\begin{aligned} D[i,j] = min(\ & D[i-1,j-1] + \delta(a_i, b_j), \\ & D[i-1,j] + \delta(a_i, -), \\ & D[i,j-1] + \delta(-, b_j)\ ) \end{aligned}$$

where $\delta(x, y)$ is the cost of substitute symbol x for symbol y, $\delta(a_i, -)$ states the cost of deleting symbol a_i. The complexity of building this matrix is $\mathcal{O}(m \cdot n)$.

In a second step, the optimal alignment is retrieved from D. It starts at $D[n, m]$ and tracks back the high-scoring path until $D[0, 0]$ is reached. The next entry of D after the current one depends on the choice made in the first step. As a result, $D[n, m]$ is the total similarity score $\Delta(a, b)$. The complexity of the reconstruction step is $\mathcal{O}(m + n)$.

The similarity function Δ

PHAST finds optimal alignment of sequences of phonemes using the edit distance with a phonetically motivated cost function.

Phonetic similarity functions have been used in other domains of research with success (e.g., identification of confusable drug names, dialectometry, spelling correction). A survey on phonetic similarity functions can be found in [9]. Kondrak [10, 11] proposes a flexible and mathematically sound approach. This approach uses the edit distance with two straightforward modifications of the original algorithm. The first one is the use of two new operations: compression and expansion. The second one is the use of a modified reconstruction step that allows global and semi-local alignment. These modifications are described below.

- Compression/expansion: Two contiguous symbols of one string may correspond to a single symbol of the other string. Compression and expansion are the same operation from a computational point of view. Compression allows better detection of phenomenons like consonant merging (e.g. [c] sounds like the pair [tʃ] rather than [t] or [ʃ] alone).

- Semi-local alignment: As descrived before, we have seen that the reconstruction step of the edit distance starts in $D[n, m]$ and builds the best *global* alignment of a and b. If we try to align a short sequence (a keyword) with a longer sequence (a paragraph or sentence), we don't want to scatter the keyword across multiple words even if it involves no substitutions (we will have always a big amount of *indels* in this scenario) since it won't make sense from an empirical point of view. The effect of the alignment is depicted in Figure 4. The "semi-local" alignment has a better scoring around [gwɪst] than the traditional "global" alignment around [lɪŋgwɪst], but it is prefearable to keep the keyword phones together having the most of *indels* after or before the keyword. Although similarity between [f] and [l] is greater than [f] and [g], semi-local alignment gets a better score since it penalizes the four *indels* at [ɪŋgw] because they are within the match of [fɪst]. Therefore semi-local alignment is biased towards keeping together keyword phones. This behavior may be achieved modifying the initialization and reconstruction steps as described in [11].

Kondrak also proposes a metric for measure inter-phoneme similarity based on multi-valuated features with salience coefficients. Each phoneme is described from a physical point of view with multi-valuated features (e.g. articulatory point, roundness, etc.).

Tables 2 and 3 show the phones and features used by our system. The number enclosed in parenthesis is the numerical value of the feature. These features and values are based on

Global alignment

f	-	-	-	-	ɪ	s	t	-	-	-
l	ɪ	ŋ	g	w	ɪ	s	t	ɪ	k	s

Semi-local alignment

-	-	-	f	-	ɪ	s	t	-	-	-
l	ɪ	ŋ	g	w	ɪ	s	t	ɪ	k	s

Figure 4: How global and semi-local affects the alignment of the phonetic transcription of words "fist" and "linguistics"

Ph.	SRL	Place	High	Back
a	+ - -	velar (0.6)	low (0.0)	front (1.0)
ɑ	+ - +	velar (0.6)	low (0.0)	back (0.0)
ɒ	+ + -	velar (0.6)	low (0.0)	back (0.0)
æ	+ - -	velar (0.6)	low (0.0)	front (1.0)
e	+ - -	palatal (0.7)	mid (0.5)	front (1.0)
ɛ	+ - -	palatal (0.7)	mid (0.5)	front (1.0)
i	+ - +	palatal (0.7)	high (1.0)	front (1.0)
ɪ	+ - -	palatal (0.7)	high (1.0)	front (1.0)
o	+ + -	velar (0.6)	mid (0.5)	back (0.0)
ɔ	+ + +	velar (0.6)	mid (0.5)	back (0.0)
ʌ	+ - -	velar (0.6)	mid (0.5)	back (0.0)
u	+ + +	velar (0.6)	high (1.0)	back (0.0)
ʊ	+ + -	velar (0.6)	high (1.0)	back (0.0)
ə	+ - -	velar (0.6)	mid (0.5)	mid (0.5)
j	- - -	velar (0.6)	high (1.0)	front (1.0)
w	- + -	velar (0.6)	high (1.0)	back (0.0)

Table 2: Features for vowels. S, R and L stands for *Syllabic*, *Round* and *Long*, respectively

those used in [11] and enhanced to deal with extra sounds from Spanish and Catalan languages. Table 4 shows the salience of the features used in our system, which are the same as in [11].

The $\delta(x, y)$ function used by Kondrak is the following[3]:

$$
\begin{aligned}
\delta(a_i, -) &= k_1 \\
\delta(a_i, b_j) &= k_2 - \delta'(a_i, b_j) - V(a_i) - V(b_j) \\
\delta(a_i a_{i+i}, b_j) &= k_3 - \delta'(a_i, b_j) - \delta'(a_{i+1}, b_j) - \\
&\quad V(b_j) - max(V(a_i), V(a_{i+1})) \\
V(a_i) &= \begin{cases} 0 & \text{if } a_i \text{ is a consonant} \\ k_4 & \text{otherwise} \end{cases} \\
\delta'(a_i, b_j) &= \sum_{f \in features} diff(a_i, b_j, f) \cdot salience(f)
\end{aligned}
$$

where k_1 is the cost of deleting a symbol, k_2 is the base score when matching two equal symbols, k_3 is the base score when compressing two symbols into one, and k_4 is a penalty for matching vowels with consonants.

The scoring function

The edit distance described above provides a scoring procedure for extension of hooks. The greater the edit distance between a keyword and an extension of a hook, the lower the score value.

[3] $\delta(x, y)$ function is symmetric. For the sake of simplicity, just one direction is presented.

Ph.	VNRLT	Manner	Place
b	+ - - - -	stop (1.0)	bilabial (1.0)
β	+ - - - -	fricative (0.8)	bilabial (1.0)
c	- - - - -	stop (1.0)	palatal (0.7)
d	+ - - - -	stop (1.0)	alveolar (0.85)
ð	+ - - - -	fricative (0.8)	dental (0.9)
f	- - - - -	fricative (0.8)	labiodental (0.95)
g	+ - - - -	stop (1.0)	velar (0.6)
h	- - - - -	fricative (0.8)	glottal (0.1)
k	- - - - -	stop (1.0)	velar (0.6)
l	+ - - + -	approx. (0.6)	alveolar (0.85)
ʎ	+ - - - -	approx. (0.6)	palatal (0.7)
m	+ + - - -	stop (1.0)	bilabial (1.0)
n	+ + - - -	stop (1.0)	alveolar (0.85)
ŋ	+ + - - -	stop (1.0)	velar (0.6)
ɲ	+ + - - -	approx. (0.6)	palatal (0.7)
p	- - - - -	stop (1.0)	bilabial (1.0)
r	+ - + - +	fricative (0.8)	alveloar (0.85)
ɹ	+ - + - -	approx. (0.6)	retroflex (0.8)
s	- - - - -	fricative (0.8)	alveolar (0.85)
ʃ	- - - - -	affricate (0.9)	alveolar (0.85)
t	- - - - -	stop (1.0)	alveolar (0.85)
θ	- - - - -	fricative (0.8)	dental (0.9)
x	- - - - -	fricative (0.8)	velar (0.6)
z	+ - - - -	fricative (0.8)	alveolar (0.85)
ʒ	+ - - - -	affricate (0.9)	alveolar (0.85)

Table 3: Features for consonants. V, N, R, L and T stands for *Voice*, *Nasal*, *Retroflex*, *Lateral* and *Trill*, respectively

Feature	Salience	Feature	Salience
Syllabic	5	Nasal	10
Round	5	Retroflex	10
Long	1	Lateral	10
High	5	Trill	10
Back	5	Place	40
Voice	10	Manner	50

Table 4: Features and their salience.

This score is a bounded integer value. Its boundaries depend on the length of the sequences. In order to know when an extension of a hook exactly matches the keyword, it is necessary to normalize the score values. Given two sequences a and b of lengths n and m, with m longer than n, the following normalization rule has been used:

$$|\Delta(a,b)| = \frac{\Delta(a,b)}{\frac{\Delta(a,a)}{n} \cdot length(a,b)}$$

where $length(a,b)$ is the length of the best matching between a and b. For example, the lengths of both best matches in examples from Figure 4 are 8 and 5. The normalized values are between 0 and 1. The final score value s for the extension of a hook h and a keyword w is $|\Delta(w,h)|$.

3.1.3 Updating Term Frequency

In traditional term-based IR the score is binary. Each term occurrence scores 1. Therefore, the measure of *term frequency* (tf) is a particular case of our setting where all the matchings are perfect. We have devised three methods to compute term frequency with non-integer scores. For a given matching score s, tf can be updated with:

1. $tf = tf + s$
2. $tf = \begin{cases} tf + 1 & \text{if } s > t \\ tf + 0 & \text{if } s \leq t \end{cases}$
3. $tf = \begin{cases} tf + s & \text{if } s > t \\ tf + 0 & \text{if } s \leq t \end{cases}$

given a fixed threshold t.

For example, if two occurrences of a certain word w are found in document d with scores 0.55 and 0.8 respectively, and t is 0.7, term frequency $tf_{w,D}$ is 1.3, 1 and 0.8 respectively for the three methods. The motivation for setting a threshold to filter out some of the matchings is that in some cases the process of similarity detection for a word w will output far more hooks than occurrences of w in the original speech. This specially occurs with words containing very common syllables that will produce a lot of noisy matchings with low similarity score. Initial experiments have shown that the third approach achieves better results. This is why this approach has been used in our experiments (cf. Section 4).

3.2 Document Ranking

The first step of PHAST finds all the occurrences of the keywords in the document collection. Each occurrence has a score s between 0 and 1 rating its confidence, and a term frequency is calculated for each keyword. The second step of PHAST uses these term frequencies to create the output of the algorithm.

Most of the state-of-the-art ranking functions can be used to build the document list in this step. The only condition is that these functions can deal with non-integer values as term frequency.

We have tested several different ranking functions in the next section, including the DQ algorithm for passage retrieval described in Section 2.2.

4. EXPERIMENTAL RESULTS

This section presents the results of our experiments. We have conducted experiments using a set of 76 queries over more than 50.000 words of automatically transcribed speeches from the European and Spanish parliaments. All the data is in Spanish. Further details about the data can be found in Section 4.1.

The experiments have been done in the general framework of a Question Answering task as described in Section 4.2.

4.1 Evaluation Data Set

For a proper evaluation of IR for QA we need a corpus of spontaneous speech documents with both manual and automatic transcripts. The CLEF CL-SR corpus lacks of manual transcripts and it can be used only within the CLEF CL-SR task evaluation.

We have used a new corpus provided by TALP Research Center within the framework of TC-STAR project[4]. It consists of transcripts from European Parliamentary Plenary Sessions (EPPS) mixed with Spanish Parliamentary Session (PARL). Manual reference transcript is over 50.000 words long, corresponding to nearly four hours of speech. Automatic transcript has been done by a large vocabulary continuous speech recognizer with an average word error rate of 26.6%. The automatic transcript contains about 53.000

[4]http://www.talp.upc.edu
http://www.tc-star.org

words. Both manual and automatic data have been manually segmented into 224 documents, each one containing a complete intervention from one person. The average document length is 239 words. Lately, the transcripts have been transcribed into phones using an in-house rule-based system (this task is fairly easy for the Spanish language, even proper nouns). All the phonetic information was encoded using SAMPA [21], a computer readable version of the IPA alphabet that uses one readable ascii character for each sound.

For the query set, we have built 76 natural language questions answerable within our collection. The questions are factoid (the answer is a proper noun) although this is not relevant for retrieval step. The following samples exemplify the questions we have:

Question 031:
"¿Cuando se empezaron a abrir embajadas españolas en los países de la ampliación al Centro y Este de Europa?"
[*"When where opened the first Spanish embassies in countries within the East an Center enlargement area?"*]
Question 057:
"¿Quien es el presidente de la comisión de relaciones externas del senado mexicano?"
[*"Who is the president of the foreign affairs committee of the Mexican senate?"*]

To access a document collection via a keyword-based engine, we need to extract keywords from this question. This process is a crucial step, since a bad selection may leave the answers out of reach.

Extraction of keywords from question is done by using linguistic tools such as a part of speech tagger, a chunk parser or a named entity recognizer, and a set of heuristic rules (see [14]). A priority value is assigned to every non-stop word from the question. The higher the priority value, the more important the word is considered (more discriminant). For example, proper names, words within quotations and adjectives are usually more discriminant than verbs or single nouns. This selection is useful for both passage and document retrieval.

The keywords ranked according priority for questions 031 and 057 are the following:

Question 031: Europa (8), embajadas (7), españolas (7), Centro (7), Este (7), países (4), ampliación (4), empezaron (3),abrir (3).
Question 057: Mexicano (8), relaciones (7), externas (7), senado (7), presidente (4), comisión (4).

Keywords were phonetically transcribed using the same method than the corpus.

4.2 Experimental Setting

We expect the correct answer to be contained in one or more of the documents or passages returned by the IR engine (we will call them *gold* documents/passages).

This is why the evaluation procedure consists in checking the cases in which the IR engine is able to return at least one gold document/passage for each query among the top ranked ones. In this setting we are not judging the relevance of the documents to a certain topic described by a query but just that the answer can be found in the document. It is not crucial to have the gold document in the first place since all will be automatically processed by other modules with semantic awareness and the ranking itself becomes irrelevant. We measure the performance with the number of queries returning gold documents/passages over the total number of queries. This may be interpreted as both precision and recall measures. No operations of query expansion or relevance feedback are carried over the data. Note that the transcript may not actually contain the answer since it can be misrecognized. The evaluation checks if the original audio recording contains the answer in the same segment corresponding to IR's output.

The setting for both document retrieval and passage retrieval tasks are described as follows.

4.2.1 Document Retrieval

We have used the dynamic query adjustment described in Section 2.2 (DQ) as a baseline system for document retrieval. DQ can be used in both reference and automatic transcripts. Using the manual reference corpus, it fixes a theoretical upper bound limit to what could you get with automatic transcripts. Using the automatic transcripts it shows the performance fall-out due to ASR action. DQ algorithm is implemented using Lucene search engine[5].

We have set up four systems for term detection in automatic transcripts: word-based index (WORD), 3-grams of characters (3GCH), 3-grams of phones (3GPH) and PHAST. The four systems have been tested with three families of rank functions: Okapi BM25 (BM25), divergence from randomness (DFR) and vector space models (VSM).

A 5-fold cross-validation evaluation has been conducted. For each fold the full question set has been randomly splitted into two subsets: a development set of 25 questions and a test set of 51 questions. For each fold the best set of parameters has been selected and the same setting has been applied to the test set. The best results for all the systems with each ranking function are reported in Section 4.3.

The rest of this section describes the parameters selected for each ranking function.

BM25 For terms t_i and document d, the BM25 weighting formula used is the following:

$$weight(t_i, d) = \log \frac{N - df_i + 0.5}{df_i + 0.5} \cdot \frac{tf_{i,d} \cdot (k+1)}{tf_{i,d} + k \cdot \left((1-b) + b \cdot \frac{length(d)}{avgl}\right)}$$

where N is the number of documents in the collection and df_i is the number of documents in which term t_i occurs, $length(d)$ is the word length of document d and $avgl$ is the average document length in the whole collection. k and b are two constants to be tuned for a particular collection: k modifies the effect of term frequency and b modifies the effect of document length. The best values results are achieved with k in the range $[0, 1]$ and very small values of b in $[0, 0.1]$.

Vector Space Model: We have tested several combinations to find the best weighting scheme for VSM. Employing the notation used in SMART [17] to describe

[5]http://lucene.apache.org

System	Okapi BM25			Vector Space Models			Divergence from Rand.		
	P1	P3	P5	P1	P3	P5	P1	P3	P5
DQ$_{ref}$	**84.21%**								
WORD$_{ref}$	43.92%	57.25%	65.10%	36.86%	52.15%	60.39%	45.88%	59.60%	**67.45%**
PHAST	48.62%	71.37%	**75.29%**	31.37%	56.47%	65.47%	46.67%	67.06%	72.15%
3GCH	16.47%	52.94%	**65.10%**	8.84%	34.50%	50.19%	10.98%	46.67%	59.29%
3GPH	23.53%	47.45%	**58.82%**	8.62%	30.58%	44.31%	13.72%	41.96%	56.07%
DQ$_{auto}$	**57.89%**								
WORD$_{auto}$	38.03%	51.37%	54.50%	31.37%	49.02%	54.90%	36.46%	52.94%	**56.07%**

Table 5: Results of document retrieval. In bold the highest precision for each system

the combined schemes, the best scheme for term weighting has been *nsn* (see section 4.3) for almost all folds and all systems. That means that each term has been weighted according squared inverse of document frequency

$$weight_{nsn}(t_i) = tf_i \cdot \left(\log \frac{N}{df_i}\right)^2.$$

Divergence From Randomness: The best weighting models for PHAST and some of the other systems for DFR has been $I(n)LH2$ and $I(n)LH1$ in all folds. That means that term frequency in a document is modelled with the Bayes' Rule, all the terms are considered as independent, uses the Laplace model of aftereffect and it is normalized by document length assuming a decreasing density function for the terms for $H2$. In $H1$ a constant density function is assumed

$$weight_{I(n)L}(t_i, d) = \frac{tf_{ne}}{1 + tf_{ne}} \cdot log_2 \frac{N+1}{df_i + 0.5}$$

$$\begin{aligned} H1) \quad tf_{ne} = &\quad tf_{i,d} \cdot \frac{avgl}{length(d)} \\ H2) \quad tf_{ne} = &\quad tf_{i,d} \cdot \log_2\left(1 + \frac{avgl}{length(d)}\right). \end{aligned}$$

Models $I(F)LH1$ and $I(F)LH2$ have been the best in some of the other systems. See [3] for further explanations on DFR models.

For PHAST there are also two tunable parameters. A threshold r described in Section 3.2 and a substring length n described in Section 3.1.1. We have fixed $r = 0.80$ and $n = 4$ for both passage and document retrieval experiments.

4.2.2 *Passage Retrieval*

For passage retrieval we have the same baseline system DQ as in document retrieval. In this scenario the system output a set of passages with high keyword density rather than full documents. Two or more passages may belong to the same document.

We have set up PHAST for term detection in automatic transcriptions using the same dynamic query adjustment than in DQ. In this scenario we consider the scores as described in Section 3.2. Just the ones above a fixed threshold are taken as a hit for the purpose of building passages. The process of adding/dropping keywords in DQ is exactly the same.

4.3 Results

4.3.1 *Document Retrieval*

Table 5 shows the results of the crossvalidation. The baseline system DQ has been used with reference manual transcripts (DQ$_{ref}$) and with automatic transcripts (DQ$_{auto}$). Also traditional word-based retrieval has been tested over reference and automatic transcripts as WORD$_{ref}$ and WORD$_{auto}$ respectively.

We have used precision at x as evaluation measure. It is defined as the number of queries returning a gold document within the top x results of the ranking. As we have noted in Section 2.2, the baseline systems doesn't return a ranked list of documents but an unordered set of documents judged relevant. DQ$_{ref}$ returned an average of 3.78 documents per query and DQ$_{auto}$ returned an average of 5.71. Therefore we have chosen precision at 3 (P3) and precision at 5 (P5) as our main evaluation measure. We also provide P1 for the sake of completeness. In this setting precision and recall measures are equivalent since we are interested in how many times the engine is able to return a gold document in the top 3 or 5 results.

For each system we include the average cross-validation P1, P3 and P5 for the three weighting schemes and five systems. The results are discussed in terms of P5 for an easier comparison with DQ. Similar conclusions may be achieved with P3.

Precision loss between DQ$_{ref}$ and DQ$_{auto}$ is 26.3%, this is due solely to the effect of ASR transcript. For WORD$_{ref}$, the best result is 67.45%, 16.5 points behind DQ$_{ref}$. With automatic transcripts WORD$_{auto}$ loses 21.3% with respect to WORD$_{ref}$, this loss is comparable to the 26.3% for DQ. The best result of WORD$_{ref}$ (at P5) is still worse than DQ$_{auto}$, these results support what stated in Section 2.2: better results in QA-oriented retrieval would be achieved with DQ rather than traditional ranking techniques.

The family of n-gram systems outperforms WORD$_{auto}$ and DQ$_{auto}$ by almost 10 points, but they are still 2 points behind WORD$_{ref}$ and 19 behind DQ$_{ref}$. In terms of P1 and P3, n-gram scores are behind WORD$_{auto}$ ones.

PHAST outperforms DQ$_{auto}$ in 18.7 points and it is behind DQ$_{ref}$ by 10.5. In P3, PHAST has still the best performance overall, 15.5 points behind DQ$_{ref}$. PHAST also outperforms 3GCH by 10 points, 3GPH by 17 and WORD$_{ref}$ by 7.8.

From our point of view, PHAST better than to WORD and 3Gx approaches in two aspects. When the ASR misrecognizes one of the keywords (e.g., a proper name) it is impossible for WORD to find this term, and this information is lost. Thus, PHAST outperforms WORD in term matching capabilities allowing an approximate matching of terms; it can be seen as a raising in coverage. The n-gram approach also improves the coverage and allows approximate matching but it has no control of the n-grams distribution over the text, so it lacks of a higher precision (3Gx only outperforms WORD at P5). PHAST provides a more precise and meaningful term detection.

4.3.2 Passage Retrieval

We have also experimented with passage retrieval using the dynamic query algorithm described in Section 2.2 in combination with PHAST.

Table 6 shows the results of our experiments. Three systems have been examined: DQ_{ref} is the baseline algorithm over manual reference transcripts, DQ_{auto} is over automatic transcripts, and DQ_{PHAST} is the same baseline using PHAST algorithm for term detection. In this scenario we have tested if at least one of the returned passages contains the answer.

Minimum number of passages threshold, $rMin$, and maximum threshold, $rMax$, have been fixed to 1 and 100 respectively. This setting has been used in all systems. Recall measures the number of queries with correct answer in the passages returned. Precision measures the number of times that the correct answer is present in the passages if any is returned.

There is a loss of 40 points between automatic and manual transcripts in precision and recall. However, in average, DQ_{ref} has returned 3.78 passages per query while DQ_{auto} has returned 5.71. In automatic transcripts DQ_{auto} obtains worse results even returning more passages than in reference transcripts. We think that this is due to the fact that DQ_{auto} is dropping more keywords (it uses an average of 2.2 per query) to build the passages than DQ_{ref} (which uses an average of 2.9). Since a substantial number of content words are illtranscribed, it is easier to find a passage containing n keywords than containing $n + 1$. In fact, DQ_{auto} outputs a passage built around just one keyword (i.e., all occurrences of the keyword) in 24 queries, while DQ_{ref} does it in 10 queries.

This results show how term detection is decisive for passage building. The difference between DQ_{auto} and DQ_{ref} in passage retrieval is 40% while it is "only" 29% in document retrieval. Passage retrieval is adding a new constraint to the task of document retrieval: now the keywords must be close together to be retrieved. Therefore, any transcript error changing a keyword in the transcript may prevent the formation of a passage. Because of its lack of redundancy, passage retrieval is less robust than document retrieval.

DQ_{PHAST} returns an average of 3.80 passages, almost the same than DQ_{ref}, using 2.69 keywords. It surpasses DQ_{auto} by 18% in precision and 17% in recall, taking an intermediate place between DQ_{auto} and DQ_{ref}. The difference among DQ_{PHAST}, DQ_{auto} and DQ_{ref} in passage retrieval is similar to the difference among PHAST, DQ_{auto} and DQ_{ref} in document retrieval.

5. CONCLUSIONS

In this paper we have presented a novel approach to spoken document retrieval. We can overcome part of automatic speech recognition errors using a sound measure of phonetic similarity and a fast search algorithm based on phonetic sequence alignment. This algorithm can be used in combination with traditional document ranking models. We have tested Okapi BM25, divergence from randomness and vector space models for spoken document retrieval, and an algorithm for spoken document passage retrieval. The results show similar improvement in passage retrieval and in document retrieval. Our approach significantly outperforms other systems in 18 and 10 points respectively.

We think that both results in passage retrieval and in document retrieval show that SDR based on phonetic sequence alignment is a very promising technique and may be easily applied to other languages.

System	Precision	Recall	Passages
DQ_{ref}	86.56%	76.31%	3.78
DQ_{PHAST}	64.61%	55.26%	3.80
DQ_{auto}	46.77%	38.15%	5.71

Table 6: Results of passage retrieval. Showing precision, recall and average number of passages returned per query

Acknowledgements

This work has been partially funded by European Commission and the Spanish Ministry of Science and Technology.

6. REFERENCES

[1] S. Altschul, W. Gish, W. Miller, E. W. Meyers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[2] M. Alzghool and D. Inkpen. University of Ottawa's participation in the CL-SR task at CLEF 2006. *In Proceedings of the CLEF 2006 Workshop on Cross-Language Information Retrieval and Evaluation*, 2006.

[3] G. Amati and C. V. Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems (TOIS)*, 20(4):357–389, 2002.

[4] J. Garofolo, G. Auzanne, and E. Voorhees. The TREC spoken document retrieval track: A success story. *Proceedings of the Recherche d'Informations Assiste par Ordinateur: ContentBased Multimedia Information Access Conference*, 2000.

[5] J. Hopcroft and J. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, Reading, Massachusetts, 1979.

[6] D. Inkpen, M. Alzghool, and A. Islam. Using various indexing schemes and multiple translations in the CL-SR task at CLEF 2005. *In Proceedings of CLEF 2005, Lecture Notes in Computer Science 4022, Springer-Verlag*, 2006.

[7] D. Inkpen, M. Alzghool, G. Jones, and D. Oard. Investigating cross-language speech retrieval for a spontaneous conversational speech collection. In *HLT-NAACL*, 2006.

[8] G. Jones, K. Zhang, and A. Lam-Adesina. Dublin city university at CLEF 2006: Cross-language speech retrieval (CL-SR) experiments. *In Proceedings of the CLEF 2006 Workshop on Cross-Language Information Retrieval and Evaluation*, 2006.

[9] B. Kessler. Phonetic comparison algorithms. *Transactions of the Philological Society*, 103:243–260, 2005.

[10] G. Kondrak. A new algorithm for the alignment of phonetic sequences. *Proceedings of the First Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 288–295, 2000.

[11] G. Kondrak. *Algorithms for Language Reconstruction.* PhD thesis, University of Toronto, 2002.

[12] V. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics-Docklandy*, 10:707–710, 1966.

[13] D. Oard, J. Wang, G. Jones, R. White, P. Pecina, D. Soergel, X. Huang, and I. Shafran. Overview of the CLEF-2006 cross-language speech retrieval track. *Proceedings of the CLEF 2006 Workshop on Cross-Language Information Retrieval and Evaluation*, 2006.

[14] M. Paşca. *High-performance, open-domain question answering from large text collections.* PhD thesis, Southern Methodist University, Dallas, TX, 2001.

[15] S. Robertson, S. Walker, K. Spärck-Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In D. Harman, editor, *Overview of the Third Text REtrieval Conference (TREC-3).* Gaithersburd, MD: NIST, 1995.

[16] G. Salton, editor. *Automatic text processing.* Addison-Wesley Longman Publishing Co., Inc., 1988.

[17] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. Technical report, Cornell University, 1987.

[18] M. Saraclar and R. Sproat. Lattice-based search for spoken utterance retrieval. In *HLT-NAACL*, 2004.

[19] S. Srinivasan and D. Petkovic. Phonetic confusion matrix based spoken document retrieval. In *SIGIR*, 2000.

[20] J. Wang and D. Oard. CLEF-2005 CL-SR at maryland: Document and query expansion using side collections and thesauri. *In Proceedings of the CLEF 2005 Workshop on Cross-Language Information Retrieval and Evaluation*, 2005.

[21] J. Wells. SAMPA computer readable phonetic alphabet. In D. Gibbon, R. Moore, and R. Winski, editors, *Handbook of Standards and Resources for Spoken Language Systems.* Berlin and New York: Mouton de Gruyter, 1997.

[22] R. White, D. Oard, G. Jones, D. Soergel, and X. Huang. Overview of the CLEF-2005 cross-language speech retrieval track. *Proceedings of the CLEF 2005 Workshop on Cross-Language Information Retrieval and Evaluation*, 2005.