A Phonetic-Based Approach to Query-by-Example Spoken Term Detection

Lluís-F. Hurtado, Marcos Calvo, Jon Ander Gómez, Fernando García, and Emilio Sanchis

Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València
{lhurtado,mcalvo,jon,fgarcia,esanchis}@dsic.upv.es

Abstract. Query-by-Example Spoken Term Detection (QbE-STD) tasks are usually addressed by representing speech signals as a sequence of feature vectors by means of a parametrization step, and then using a pattern matching technique to find the candidate detections. In this paper, we propose a phoneme-based approach in which the acoustic frames are first converted into vectors representing the *a posteriori* probabilities for every phoneme. This strategy is specially useful when the language of the task is a priori known. Then, we show how this representation can be used for QbE-STD using both a Segmental Dynamic Time Warping algorithm and a graph-based method. The proposed approach has been evaluated with a QbE-STD task in Spanish, and the results show that it can be an adequate strategy for tackling this kind of problems.

Keywords: Spoken Term Detection, Query-by-Example, Automatic Speech Recognition.

1 Introduction

In the last few years both the amount and the availability of digital data have rapidly and substantially increased. These facts have led to the need of interacting in a multimodal way with a variety of information repositories in order to find useful information in them, opening this way new and important challenges in the field of Language Technologies. The Spoken Term Detection (STD) task is among these challenges. It consists on finding all the occurrences of a *search term*, which ortographic transcription is provided and can be composed of more than one word, in the contents of an audio repository. However, the input may also be an utterance representing the search term. In this case, the problem is known as Query-by-Example Spoken Term Detection (QbE-STD) and both the input query and the collection of documents are acoustic signals. Both of these tasks have been studied lately [1–4], and some examples of their interest and importance are the evaluation campaigns carried out in this line, such as the one organized by NIST in 2006 [5] and the MediaEval evaluations [6]. In this paper, we will focus on the Query-by-Example Spoken Term Detection task.

Most of the methods that have been proposed for the QbE-STD task are based on classical pattern matching algorithms. Specifically, the search is performed by

J. Ruiz-Shulcloper and G. Sanniti di Baja (Eds.): CIARP 2013, Part I, LNCS 8258, pp. 504–511, 2013.

means of an algorithm that matches the feature vectors corresponding to both the queries and the documents in the audio repository, looking for occurrences of the queries in the documents. The feature vectors are usually a standard parametrization of the acoustic signal, for example based on cepstrals. Also, in the recent literature one of the most usual algorithms to perform this search is Segmental Dynamic Time Warping (SDTW) [1–4].

In this work, we perform a step after the parametrization, in which the posterior probabilities of the phonemes given the acoustic frames are calculated. This implies that the set of phonemes that are going to be used must be *a priori* known, but it is not a problem if the language of the task is fixed. These phonetic probabilities are computed by means of a process of acoustic clustering and classification in terms of acoustic classes, as explained in Section 2. Then, the phonetic probabilities worked out in this step will be the base for two QbE-STD algorithms. First, Section 3 shows a SDTW algorithm that uses the Kullback-Leibler divergence and a specific set of transitions. Section 4 shows a method to build graphs of phonemes from the phonetic probabilities and an algorithm to perform the QbE-STD task. This algorithm is based on searching common paths in the graphs corresponding to the document and the query, allowing edit operations to gain flexibility. A description of the experiments we have performed and a discussion of their results is shown in Section 5, and finally in Section 6 some conclusions are drawn.

2 Computation of the *a Posteriori* Probabilities of the Phonemes

After a standard parametrization of the acoustic signal using cepstrals, we will carry out a procedure to compute the *a posteriori* probability of every phoneme u in a pre-defined set of phonetic units U given each acoustic frame x_t, it is, $p(u|x_t)$. For this computation, a set of acoustic classes A is obtained using a clustering procedure on the acoustic feature vector space using the unsupervised version of the Maximum Likelihood Estimation (MLE) algorithm. Assuming that the acoustic classes can be modelled as Gaussian distributions, the output of this procedure is a Gaussian Mixture Model (GMM). The use of conditional probabilities allows us to compute the phonetic-conditional probability density $p(x_t|u)$ as follows [7]:

$$p(x_t|u) = \sum_{a \in A} p(x_t|a) \cdot p(a|u) \tag{1}$$

for each $u \in U$, where $p(x_t|a)$ is the acoustic class-conditional probability provided by the GMM, and $p(a|u)$ is the conditional probability that acoustic class a was manifested when phonetic unit u was uttered.

The conditional probabilities $p(a|u)$ for all $a \in A$ and for all $u \in U$ are computed by a progressive refinement algorithm for phonetic segmentation [8]. It starts from an initial coarse segmentation and continues until no improvements on the segmentation are found. As a labeled corpus for phonetic segmentation

is needed to perform this process, we have used the training subcorpus of the Spanish Albayzin database [9].

Thus, the *a posteriori* probability of each phonetic unit u given an acoustic vector x_t, $p(u|x_t)$, can be rewritten as

$$\Pr(u|x_t) = \frac{\sum\limits_{a \in A} p(x_t|a) \cdot p(a|u)}{\sum\limits_{v \in U} \Big(\sum\limits_{a \in A} p(x_t|a) \cdot p(a|v) \Big)} \tag{2}$$

In the next two sections we will show two different ways of using these phonetic probabilities for a QbE-STD task.

3 Segmental Dynamic Time Warping with *a Posteriori* Phonetic Probabilities

Segmental DTW (SDTW) [10] is a modification of the well-known Dynamic Time Warping algorithm. The goal of SDTW is to find multiple local alignments of two input utterances, represented as a sequence of vectors. The main difference between SDTW and DTW is that, while in DTW there is only one start point for the alignment, SDTW allows the alignment to start at any point along the speech document. This is very convenient for the QbE-STD task, as the goal is to find all the occurrences of the query in each of the documents.

In our case, the vectors corresponding to the utterances will contain the *a posteriori* probabilities for each phoneme, given each frame.

Instead of using the DTW typical transitions, we have used (as in other works like [1]) a different set of transitions, as shown in Equation 3. This set of movements ensures that the paths found will represent alignments where the number of frames taken in the document is between half and twice the length of the query. Therefore, the minimization function at each point is given by:

$$D(i,j) = \begin{cases} 0 & j < 1 \\ \min \begin{pmatrix} D(i-1,j-1) \\ D(i-2,j-1) \\ D(i-1,j-2) \end{pmatrix} + KL(A(i), B(j)) & j \geq 1 \end{cases} \tag{3}$$

where $A(i)$ is the vector of *a posteriori* phonetic probabilities for the frame i of the speech document, $B(j)$ represents the *a posteriori* probabilities of phonemes for the frame j of the query, and KL is the Kullback–Leibler divergence [11].

All the paths in the Dynamic Programming matrix that arrive to the end of the query are considered candidate detections. However, many of these detections are false positives. To filter out these detections, Algorithm 1 is performed. This way, our final set of detections has only at most d elements, and all of them are the ones with best scores. It must be noted that the sorting performed in the first line of Algorithm 1 could be either in ascending or descending order, depending on the objective function of the search procedure.

Algorithm 1. Algorithm to filter a list of detections

Require: A list of candidate detections CD,
a maximum number of filtered detections d
Ensure: A list of filtered detections FD
1: SCD = sort the hypothesis in CD by their score
2: $FD2$ = empty list
3: **while** SCD is not empty **do**
4: h = first element of SCD
5: Move h to $FD2$
6: Delete from SCD all the detections whose timespan overlaps h
7: **end while**
8: Determine a threshold t considering the score of the elements in $FD2$
9: FD = first d elements of $FD2$ whose score fulfills the threshold t
10: **return** FD

Determining the threshold t for this algorithm is a task that can be addressed in a variety of ways. In our case, we have performed a linear combination of some statistics of the scores, like the mean, the median, the maximum and the standard deviation. The weights assigned to each of these statistics provide us a range of thresholds that can be used to tune the performance of the system. Also we have considered as the input list of candidate detections CD all the detections found in all the documents for a specific query. This means that the pruning made by this algorithm is local to the specific query, considering all the documents in the repository as a whole. In consequence, for each query at most d detections among all the documents are considered as confirmed detections.

4 A Graph-Based Algorithm for QbE-STD

Taking advantage of the sequentiallity of speech, our graphs of phonemes have a left-to-right topology. Nodes act like time marks, and every node has a timestamp. The arcs have associated the phonetic unit uttered between the timestamps kept by origin and destination nodes, and also its phonetic probability.

The construction algorithm has two steps: phoneme detection, and error correction. In the first step, each vector of phonetic probabilities is analyzed in order to find if there is any probability above a detection threshold. If the probability of a phoneme is above this threshold, we consider that it has been uttered, but the time in which its pronunciation started and finished is still undetermined. In order to fix the starting and ending time of the detected phoneme, a new threshold (called extension threshold), less restrictive than the detection threshold, is used. That is, starting from the frame, or frames, where a phoneme was detected, an extension process is performed considering the previous and following frames that overpass the extension threshold for that phoneme. This extension process finishes as soon as the extension threshold for the phoneme is not exceeded. The reason for using this lower threshold is that the initial or final parts of phonemes are less clearly pronounced and detected than its central part. Both thresholds are empirically determined.

The error correcting step consists on detecting and correcting both spurious aparitions and misses of phonemes. This is the case of very short phonemes, or some gaps in a zone where a phoneme was detected with enough probability.

Finally, the graph of phonemes is built according to these corrections. A node is created whenever the detection of any phonetic unit begins or ends. Arcs are built in a way that all go from a node to the following one. Thus, each arc may represent either a complete detection of a phoneme, or a part of it. The weight of each arc is the accumulated log-probability of the detection between the instants represented by the starting and ending nodes.

4.1 Search Algorithm

Once the documents and the queries are represented as graphs of phonemes, we can take them in pairs to perform the QbE-STD task. The basis of this algorithm is to find, for each node i in the graph corresponding to the document, the common path in both graphs that goes through all the query, finishes at i in the document, and has the maximum combined score, defined as the sum of the weights of both paths individually. To find these common paths, edit operations on the arcs are allowed, in order to make this search more flexible and to correct possible errors made while building the graph of phonemes. Insertion and deletion operations have a constant penalization, while the cost of a substitution may depend on the pair of phonemes being considered. For this work, we have only allowed coincidences, as well as substitutions of vowels by their semivowels, consonants by their semiconsonants and vice versa.

The algorithm that searches for these common paths follows a Dynamic Programming (DP) strategy. Let M be a matrix of dimensions $I \times J$, where I and J are the number of nodes of the graphs representing the document and the query, respectively. Thus, $M(i,j)$ will contain the best score for arriving to node i in the document and j in the query, using both the arcs in the graphs and the edit operations allowed. Also, given an arc a, let $\mathrm{ori}(a)$ and $\mathrm{dest}(a)$ be functions that return respectively the position in the graph of the starting and ending nodes of a, $\mathrm{W}(a)$ a function that returns the weight of the arc, and $\mathrm{S}(a)$ a function that provides the symbol (phoneme) attached to the arc. Thus, the algorithm can be stated as follows:

$$M(i,j) = \begin{cases} 0 & \text{if } j = 0 \\ \max\{arcSub(i,j), arcIns(i,j), arcDel(i,j)\} & \text{otherwise} \end{cases} \tag{4}$$

where:

$$arcSub(i,j) = \max_{\substack{\text{arcs } a,b: \\ \forall\, \mathrm{dest}(a)=i \,\wedge\, \mathrm{dest}(b)=j}} \{M(\mathrm{ori}(a),\mathrm{ori}(b)) + \mathrm{W}(a) + \mathrm{W}(b) + k_{sub}(\mathrm{S}(a),\mathrm{S}(b))\}$$

$$arcIns(i,j) = \max_{\forall \text{ arc } a\text{: } \mathrm{dest}(a)=i} \{M(\mathrm{ori}(a), j) + k_{ins}\}$$

$$arcDel(i,j) = \max_{\forall \text{ arc } b\text{: } \mathrm{dest}(b)=j} \{M(i, \mathrm{ori}(b)) + k_{del}\}$$

$$k_{sub}(x,y) = \begin{cases} 0 & \text{if } x = y \\ 0 & \text{if } x \text{ is semivowel or semiconsonant of } y \text{ or vice versa} \\ -\infty & \text{otherwise} \end{cases}$$

k_{ins} and k_{del} are constants that must be empirically determined.

Once the DP matrix has been filled, all the cells corresponding to the last node of the query represent candidate detections. Thus, they must be filtered in order to reject as many false positives as possible. In this case, Algorithm 1 is also used for finding the confirmed detections.

5 Experiments and Results

To evaluate these approaches, we have performed several experiments using the MAVIR database [12]. This is the Query-by-Example Spoken Term Detection corpus that was used in the Search on Speech track of the 2012 Albayzin Evaluation. A feature of this task is that the language of both the queries and the collection of documents is Spanish, so it is *a priori* known.

In this task we can distinguish two kinds of files. First, there are 10 files corresponding to recordings of conferences and academic acts carried out in Madrid between 2006 and 2008. The speech in these files is spontaneous and was acquired in a variety of conditions using different microphones. Also different accents of the Spanish language are represented. In addition, these files are very long, with a duration between 19 and 75 minutes. These facts make this task very hard. Second, the other kind of files is the set of queries, which is composed of 120 terms. The whole set of files is divided this way: 60 queries and 7 documents for development and 60 queries and 3 documents for test.

As it is usual in Information Retrieval (IR) tasks, we have considered the standard Precision and Recall, and its combination by means of the F1-Measure. Figures 1 and 2 show the evolution of these measures for the development set using a variety of thresholds, considering as the maximum number of confirmed detections for the filtering algorithm the one that provided the best results in our experiments. In the case of the graph-based approach, a large amount of combinations of insertion and deletion constants have been tried, and Figure 2 shows the evolution for the configuration that achieved the best results.

Figure 1 shows that for the development set the SDTW approach reaches a Precision of more than 30%, while the best Recall is around 14%. However, in some IR applications it is more important to find some detections with a relatively large precision, than finding them all. Another interesting fact is that there is a point where, even varying the threshold, the results do not change. This happens when too many candidate detections surpass the threshold of the filtering algorithm, and the pruning is just done by considering the maximum number of hypotheses specified beforehand. The results shown in Figure 2 are not as good as the obtained with the SDTW algorithm. This is due to the fact that the difficulty of the task and the noisy conditions of the audio recordings make the graph builder algorithm generate many errors that can not be recovered when the graphs of phonemes are processed.

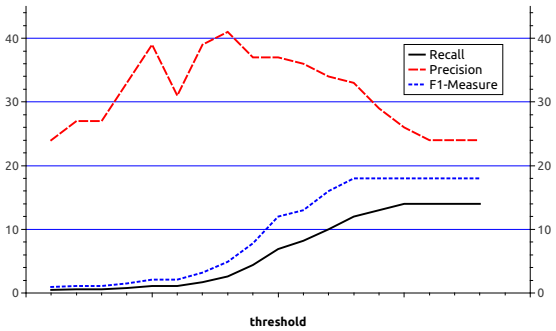

Fig. 1. Precision, Recall and F1 for the development data for the SDTW approach

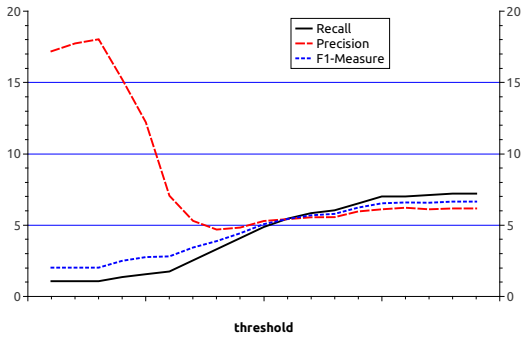

Fig. 2. Precision, Recall and F1 for the development data for the graph-based approach

Table 1 shows the results obtained for the test set using the parameters that optimized the F1-Measure in the development phase.

Table 1. Results obtained for the test set

System	Precision (%)	Recall (%)	F1-Measure (%)
SDTW	31.2	18.3	23.1
Graph-based	9.0	10.2	9.6

In the test set the experiments using Segmental DTW also outperform the experiments with the graphs of phonemes. Thus, the codification of the frames in terms of the posterior probabilities of phonemes seems to be a good representation, and the Segmental DTW algorithm using this representation gives good enough results. However, SDTW has a higher time complexity than the algorithm based on graphs of phonemes, as the number of nodes of the graphs is usually much lower than the number of frames. In consequence, our graph-based approach seems promising, and we will continue working on how to improve it.

6 Conclusions

In this work, we have presented two algorithms for Query-by-Example Spoken Term Detection based on the computation of *a posteriori* phonetic probabilities of the phonemes given the speech signals. One of these algorithms performs a Segmental DTW search, while the other represents the query and the document as graphs of phonemes and searches for common paths in both graphs using edit operations. The experimental results show that our codification of the frames in terms of *a posteriori* probabilities of the phonemes and the proposed algorithms are a good approach to QbE-STD. As future work, we want to improve the performance of the graph-based method presented in this paper, for example trying to make the phoneme detection process more robust.

Acknowledgements. Work partially supported by the Spanish Ministerio de Economía y Competitividad under contract TIN2011-28169-C05-01 and FPU Grant AP2010-4193, and by the Vic. d'Investigació of the UPV (PAID-06-10).

References

1. Anguera, X., Macrae, R., Oliver, N.: Partial sequence matching using an unbounded dynamic time warping algorithm. In: ICASSP, pp. 3582–3585 (2010)
2. Hazen, T., Shen, W., White, C.: Query-by-example spoken term detection using phonetic posteriorgram templates. In: ASRU, pp. 421–426 (2009)
3. Zhang, Y., Glass, J.: Unsupervised spoken keyword spotting via segmental DTW on gaussian posteriorgrams. In: ASRU, pp. 398–403 (2009)
4. Akbacak, M., Vergyri, D., Stolcke, A.: Open-vocabulary spoken term detection using graphone-based hybrid recognition systems. In: ICASSP, pp. 5240–5243 (2008)
5. Fiscus, J.G., Ajot, J., Garofolo, J.S., Doddingtion, G.: Results of the 2006 spoken term detection evaluation. In: Proceedings of ACM SIGIR Workshop on Searching Spontaneous Conversational, pp. 51–55 (2007)
6. Metze, F., Barnard, E., Davel, M., Van Heerden, C., Anguera, X., Gravier, G., Rajput, N., et al.: The spoken web search task. In: Working Notes Proceedings of the MediaEval 2012 Workshop (2012)
7. Gómez, J.A., Castro, M.J.: Automatic segmentation of speech at the phonetic level. In: Caelli, T.M., Amin, A., Duin, R.P.W., Kamel, M.S., de Ridder, D. (eds.) SSPR & SPR 2002. LNCS, vol. 2396, pp. 672–680. Springer, Heidelberg (2002)
8. Gómez, J.A., Sanchis, E., Castro-Bleda, M.J.: Automatic speech segmentation based on acoustical clustering. In: Hancock, E.R., Wilson, R.C., Windeatt, T., Ulusoy, I., Escolano, F. (eds.) SSPR & SPR 2010. LNCS, vol. 6218, pp. 540–548. Springer, Heidelberg (2010)
9. Moreno, A., Poch, D., Bonafonte, A., Lleida, E., Llisterri, J., Marino, J., Nadeu, C.: Albayzin speech database: Design of the phonetic corpus. In: Third European Conference on Speech Communication and Technology (1993)
10. Park, A., Glass, J.: Towards unsupervised pattern discovery in speech. In: ASRU, pp. 53–58 (2005)
11. Kullback, S.: Information theory and statistics. Courier Dover Publications (1997)
12. MAVIR corpus, `http://www.lllf.uam.es/ESP/CorpusMavir.html`