Comparison of Methods for Language-Dependent and Language-Independent Query-by-Example Spoken Term Detection

JAVIER TEJEDOR, Universidad Autónoma de Madrid
MICHAL FAPŠO, IGOR SZÖKE, JAN "HONZA" ČERNOCKÝ,
and FRANTIŠEK GRÉZL, Brno University of Technology

This article investigates query-by-example (QbE) spoken term detection (STD), in which the query is not entered as text, but selected in speech data or spoken. Two feature extractors based on neural networks (NN) are introduced: the first producing phone-state posteriors and the second making use of a compressive NN layer. They are combined with three different QbE detectors: while the Gaussian mixture model/hidden Markov model (GMM/HMM) and dynamic time warping (DTW) both work on continuous feature vectors, the third one, based on weighted finite-state transducers (WFST), processes phone lattices. QbE STD is compared to two standard STD systems with text queries: acoustic keyword spotting and WFST-based search of phone strings in phone lattices. The results are reported on four languages (Czech, English, Hungarian, and Levantine Arabic) using standard metrics: equal error rate (EER) and two versions of popular figure-of-merit (FOM). Language-dependent and language-independent cases are investigated; the latter being particularly interesting for scenarios lacking standard resources to train speech recognition systems. While the DTW and GMM/HMM approaches produce the best results for a language-dependent setup depending on the target language, the GMM/HMM approach performs the best dealing with a language-independent setup. As far as WFSTs are concerned, they are promising as they allow for indexing and fast search.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

General Terms: Experimentation

Additional Key Words and Phrases: Query-by-example, DTW-based query-by-example, GMM/HMM-based query-by-example, WFST-based query-by-example, bottleneck features, keyword spotting

ACM Reference Format:
Tejedor, J., Fapšo, M., Szöke, I., Černocký, J. H., and Grézl, F. 2012. Comparison of methods for language-dependent and language-independent query-by-example spoken term detection. ACM Trans. Inf. Syst. 30, 3, Article 18 (August 2012), 34 pages.
DOI = 10.1145/2328967.2328971 http://doi.acm.org/10.1145/2328967.2328971

1. INTRODUCTION

The ever-growing volume of heterogeneous speech data stored in audio repositories increases the need for efficient methods for retrieving the stored information. Much work

This work was partly supported by Czech Ministry of Trade and Commerce project FR-TI1/034, the Technology Agency of the Czech Republic project TA01011328, the Czech Ministry of Education project MSM0021630528, and by the IT4 Innovations Centre of Excellence CZ.1.05/1.1.00/02.0070. Igor Szöke was supported by the Grant Agency of the Czech Republic post-doctoral project GP202/12/P567.
This work was mainly carried out while Javier Tejedor was visiting BUT.
Authors' addresses: J. Tejedor (contact author), Universidad Autónoma de Madrid, Spain; email: javier.tejedor@uam.es; M. Fapšo, I. Szöke, J. H. Černocký, and F. Grézl, Brno University of Technology, Czech Republic.

DOI 10.1145/2328967.2328971 http://doi.acm.org/10.1145/2328967.2328971

has addressed this issue by means of spoken document retrieval (SDR), spoken term detection (STD), keyword spotting, query-by-example or spoken query approaches. Part of this research was driven by evaluations, such as the 2006 NIST STD evaluation [NIST 2006] and the Spoken Web Search task [Metze et al. 2012] at MediaEval 2011. STD can also be an important component for integration within some other technologies such as SDR or spoken content summarization. Due to the tight relationship between STD and the scope of this work, we briefly outline the STD paradigm in the following.

1.1. Spoken Term Detection

STD has received much interest for years from several institutes/companies such as IBM [Mamou et al. 2007; Mamou and Ramabhadran 2008; Can et al. 2009], BBN [Fiscus et al. 2007], SRI & OGI [Vergyri et al. 2006; Vergyri et al. 2007; Akbacak et al. 2008], BUT [Szöke et al. 2008a, 2008b, 2008c]. The goal of STD is to find a list of terms (a single word or a sequence of words) fast and accurately in audio data, where the list of terms is represented in a textual form. We denote this as text-based STD, which is still the predominant approach nowadays. This way, we differentiate it from query-by-example (QbE) STD, which represents the scope of this work, which we will introduce shortly.

In text-based STD, it is assumed that we have enough resources and an exact knowledge of the target language, and therefore we can employ a reasonable amount of transcribed data, phone sets, and pronunciation dictionaries. That is why building a STD system for languages with low resources is a challenging issue.

On the other hand, there is also a steady demand (especially in the security field) for rapid development of STD systems for languages with low or completely missing resources. In these cases, it is not possible to train target language-specific acoustic models, and hence the system needs to rely on a language-independent modeling approach. In addition, the user often has no knowledge of the textual interpretation of the search term. Paradigm-named QbE STD offers a solution for these challenging cases.

1.2. Query-by-Example Spoken Term Detection

Generally speaking, query-by-example can be defined as a method of searching for an example of an object or a part of it in other objects. Query-by-example has been widely used in audio applications like sound classification [Zhang and Kuo 1999; Helén and Virtanen 2007, 2010], music retrieval [Tzanetakis et al. 2002; Tsai and Wang 2004], and spoken document retrieval [Chia et al. 2008].

Under this paradigm, we consider the scenario where the user has found some interesting data within a data pool (for example, by random browsing or some other method). His/her purpose is to find similar data within the data pool. Therefore, the user selects one or several speech cuts containing the *term* of interest (henceforth, *query*) and the system returns him/her other putative hits from the whole data pool (henceforth, *utterances*). Another scenario for QbE STD considers one or several user speech recordings of the term of interest. Therefore, both QbE STD methods rely on spoken term examples as input, contrary to a text-based STD that relies on a textual form of the input terms.

It must be noted that the unavailability of enough training resources and limited or no knowledge of the target language make it impossible to apply a large vocabulary continuous speech recognition (LVCSR) system to produce word/phone lattices and to conduct text-based STD. This is mainly due to high language-dependency inherent to word lattices, especially if a powerful language model is applied. Phone lattices are also quite language-dependent, since they also depend on the set of phones and accurate phone-based language models. Therefore, using LVCSR is almost impossible for a language-independent QbE STD.

1.3. Related Work

In contrast to text-based STD, QbE STD has received minimal attention in the speech community. So far, it has been addressed from two points of view.

(1) Methods based on phone transcription of the speech signal corresponding to the query [Lin et al. 2008; Parada et al. 2009; Shen et al. 2009; Lin et al. 2009; Barnard et al. 2011], for which the text-based STD technology is suitable. These methods decode the query with a speech recognizer to derive its word/subword representation, which can be employed next to hypothesize detections in a text-based STD-like fashion.
(2) Methods based on template matching of features extracted directly from the speech signal representing the query/utterance [Hazen et al. 2009; Zhang and Glass 2009; Chan and Lee 2010; Anguera et al. 2010; Anguera 2011; Muscariello and Gravier 2011; Szöke et al. 2011; Muscariello et al. 2011]. Some of these methods can be considered language-independent, since they do not make any assumptions about the target language, and hence are also appropriate for low-resource languages. They usually borrow the idea from dynamic time warping (DTW)-based speech recognition, and have been found to outperform phone transcription-based techniques when applied on QbE STD in a language-dependent scenario [Hazen et al. 2009].

Articulary units, aiming at covering a broad set of languages, plus DTW-based search over the segments corresponding to the matched trigrams (to obtain the appropriate timing) have also recently been proposed [Mantena et al. 2011].

Methods that deal with pattern/word/spoken term discovery were also described in the literature [Jansen et al. 2010; Muscariello et al. 2009; Park and Glass 2008]. These methods can also be employed for QbE STD, since they are all based on template matching of features extracted from the speech signal. The discovery task is even more challenging because it has to discover the units (words) by itself. In QbE STD we already have examples of the units, so QbE STD is actually a subtask of pattern discovery.

QbE STD, in a completely unsupervised fashion, which allows for building language-independent systems and is suitable for low-resource scenarios, has been addressed: [Zhang and Glass 2009; Chan and Lee 2010; Anguera et al. 2010], with promising results. The methods mentioned earlier dealing with pattern/spoken term discovery [Jansen et al. 2010; Muscariello et al. 2009; Park and Glass 2008] can also be employed for building language-independent QbE STD systems. However [Anguera et al. 2010; Muscariello et al. 2009; Park and Glass 2008] can only be employed for speaker-dependent QbE as standard Mel frequency cepstral Coefficients (MFCCs) used in these works are speaker-dependent. In the same line, Jansen et al. [2010] employs features derived from an MLP (i.e., phone posterior features) as query and utterance representations. The work in Zhang and Glass [2009] derives Gaussian posteriorgrams as features and then conducts a variant of DTW-like query detector, called segmental DTW (SDTW). Our work differs from this because we propose additional query detectors, and employ phone state posterior features for the DTW-based approach. Finally, the unsupervised approach proposed by Chan and Lee [2010] employs spectrogram-based features segmented by a hierarchical agglomerative clustering to identify speech segments and a DTW-like search as query detector.

Efforts to build universal/language-independent phone recognizers have also been presented in the literature. However, their high phone error rates on a target language [Schultz and Waibel 2001; Kempton et al. 2011; Walker et al. 2003] or the requirement to know the target language [Byrne et al. 2000; Sinischalchi et al. 2008] represent two important drawbacks. When these are mitigated, perhaps they may also be used

to build language-independent QbE STD systems due to their intrinsically language-independent nature.

1.4. Motivation and Organization of this Article

From the literature review, it is clear that previous work was based on standard features such as MFCCs, phone posteriors, and Gaussian posteriors for query/utterance representation and on variants of a DTW-like search as term detector. This leaves a lot of room for improvement and exploration of new QbE STD-related approaches. In addition, no comparison between the language-dependent and the language-independent setups has been carried out so far, given a QbE STD system.

Therefore, the purpose of this work is twofold: (1) to evaluate and compare the performance of several QbE STD systems across different languages, different sets of features (3-state phone posteriors and bottle-neck (see Section 5)) and different speech signal conditions; and (2) to give a deeper insight into the language-independence issue in QbE STD to derive meaningful conclusions about the feasibility of language-independent QbE STD systems.

First, for comparison purposes, we present a language-dependent acoustic keyword spotting (AKWS) system, which is considered an upper-bound of the QbE STD performance exhibited by the other approaches, and will be used in this work to justify the feasibility of language-independent QbE STD systems.[1]

Second, we further investigate our DTW-based QbE STD system using phone posteriors. This method was previously presented by other authors in a language-dependent environment or in an unsupervised fashion [Hazen et al. 2009; Zhang and Glass 2009; Chan and Lee 2010] and in a limited language-independent setup [Tejedor et al. 2010]. Therefore, we consider the DTW approach as our baseline technique. Work that deals with the multi-sequence alignment in biology (DNA, RNA, aminoacids, and so on) [Edgar and Batzoglou 2006] based on genetic algorithms and neural networks [Wu et al. 2008], evolutionary computation [Cai et al. 2000] already exists. However, since the DTW-based search has been applied before for QbE STD, and for the sake of comparison with a widespread baseline, we have preferred to use this sequence alignment technique.

Our previous work [Tejedor et al. 2010] has, however, indicated significant problems with DTW-based QbE STD when dealing with the language-independent environment. These are mainly due to the "blur" of phone posteriors, that become less sharp once the language is switched. A modeling technique that takes into account these *uncertainties* seems therefore more appropriate than template matching. Therefore, we introduce in this article as a third method, a QbE STD approach that relies on Gaussian mixture model/hidden Markov model (GMM/HMM), and is inspired by AKWS. While GMM/HMM has been employed for years to build general speech recognition systems and AKWS systems, and for some other tasks such as audio classification [Helén and Virtanen 2010], to our best knowledge, this is the first work that uses a GMM/HMM-based QbE STD system with a single-step QbE detector.

Finally, we propose another technique which has not been applied to language-independent QbE STD so far: Weighted finite state transducer (WFST)-based QbE STD. The authors note that the concept has been presented previously by Parada et al. [2009], but this work was limited to searching for out-of-vocabulary words (OOVs) in N-best word and phone lattices from a combined hybrid word-subword LVCSR system

[1]All our QbE STD detectors work only with example pronunciations of query terms, which may contain various defects. The AKWS system has an advantage in knowing the correct pronunciation of all terms. Since our phone posterior feature extractor was also trained on the correct pronunciation of training data, AKWS system should perform the best, and hence it is considered to be an upper bound.

Table I.
Amount of data used as feature training set, query training set and evaluation, and numbers of queries, examples and query occurrences in evaluation data for each language.

Language	Data (hours)			Queries		
	Feature training	Query training	Evaluation	Unique	Examples	Occurrences
Czech	100.9	2.2	2.0	58	290	1019
English	277.7	12.5	2.2	168	840	2007
Hungarian	8.5	0.9	2.2	8	40	337
Levantine	19.9	2.8	1.4	51	255	609

under a language-dependent setup. For comparison with a text-based STD system, we also present a WFST system based on pronunciations of the search terms, and hence capable of text-based STD.

Experiments are conducted on four different languages and various speech-signal conditions that examine each of the QbE STD techniques/set of features proposed in this article. A set of queries is chosen for each language and next is searched by each of these techniques. It allows contrastive comparisons across the diverse set of queries/languages/features/speech signal conditions.

The rest of the article is organized as follows: Section 2 describes the data used for experimentation, and Section 3 presents the evaluation metrics. Section 4 gives an overview of our QbE STD system and presents the audio preprocessing steps. The remaining sections present each system component individually: in Section 5, we present the feature extractor that serves as front-end. Section 6 deals with our acoustic keyword spotter that serves as a back-end upper-bound reference. The sections thereafter cover three QbE detectors that serve as back-ends: Section 7 presents the DTW-based QbE STD, the GMM/HMM-based approach for QbE STD is covered in Section 8, and WFST-based QbE STD is presented in Section 9. Section 10 contains the results and some discussion, and finally, Section 11 concludes the article.

2. DATA SETUP

In order to inspect the language-independent setup across the different techniques, we trained and evaluated proposed approaches on several languages across different groups: Czech (Slavic), English (Germanic), Hungarian (Uralic), and Levantine (Arabic).

We used conversational telephone speech (CTS) for training and evaluation of all languages except Hungarian and Czech. Whereas Hungarian was trained and evaluated only on prompted read telephone speech, Czech training data contained partly prompted and read speech. Therefore, we would expected that the Hungarian language would present the best performance.

Since all our QbE STD approaches are composed of two steps (i.e., feature extraction and query detection), three different sets of data, which correspond to the training part of both steps and the evaluation data, are necessary. They are called the feature training set, query training set, and evaluation set. The feature training set is used to train the feature extractors corresponding to each language in Table I. The query training set was used for extracting query examples and, if necessary, for parameter estimation and model estimation. Finally, the evaluation set was used for testing the approaches.

For Czech, data created for the project VD20072010B16, supported by the Czech Ministry of Interior, was used. In contrast to the rest of the languages, the feature training set is a mixture of several audio conditions (45.6 hours of real CTS, 18.9 hours of radio telephone speech (people calling into broadcasts), and 36.4 hours of read or prompted speech recorded via telephone). The expansion of training CTS data with read speech was not found harmful according to our previous experiments on acoustic

keyword spotting [Szöke et al. 2010]. In addition, it made the system more robust without any significant degradation of accuracy on CTS data. For English, the CTS comes from the Switchboard I, Switchboard Cellular, and Call Home English corpora as a feature training set, whereas the Fisher corpus was used as a query training set and the NIST STD 2006 development set was used for evaluation. For Hungarian, the telephone prompted read speech comes from the Hungarian part of the SpeechDat East corpus, which was divided into a feature training set, query training set, and an evaluation set.[2] Finally, the Levantine Arabic CTS Corpus, also was divided into a feature training set, a query training set, and an evaluation set was used for the Levantine, where nondiacritized forms of transcripts were employed. The different acoustic conditions across the languages and the differences inherent to each group of languages make the data setup appealing enough for our QbE STD task. Table I shows the data statistics for these languages.

For the approaches that work with the phone transcription for each query term (i.e., AKWS system and WFST from the pronunciation of the search terms), the transcription is obtained from a reference dictionary, and hence the pronunciation derived from each query is the correct one. However, for nondiacritized Levantine, the query transcription is derived directly from the set of graphemes that compose the query term. Therefore, both the feature extractor training and the phone transcription employed in the AKWS and WFST from dictionary pronunciation systems make use of this set of graphemes. In Levantine, a grapheme can represent several phones, which may vary depending on the context. This typically leads to lower recognition accuracy, and hence we expected worse overall performance on this language, no matter the method employed to hypothesize detections.

2.1. Query Selection

Similarly to previous work [Hazen et al. 2009; Tejedor et al. 2010], we have randomly selected queries with at least five examples in the query training set. In so doing, five examples per query were used through all the experiments. In addition, the queries fulfill the following requirements: they are longer than four phones, queries which are substrings of longer queries are discarded and queries contain only a single word. Czech, English, and Levantine query training sets provide a sufficient number of queries. We were able to extract only eight queries fulfilling the above-mentioned conditions for Hungarian data. The numbers of queries, query examples along with the number of occurrences in the evaluation set for all languages are summarized in Table I. A complete list of the queries along with the number of phones, the average time length per query, and the number of occurrences of each query in the evaluation set is in the online Appendix see the ACM Digital Library.

3. EVALUATION

The purpose of this work is to identify regions in utterances that match the spoken query. The word/orthographic transcription was used to derive the corresponding reference phone transcription for each language. Forced alignment was carried out to get the time information for each phone. Decision on the hit/false alarm (FA) was done on the basis of this phone alignment. So if a sequence of phones representing the detected query also appears in the underlying phone alignment, it is classified as a hit (regardless of the orthographic transcription, i.e., no matter if it actually represents a whole word or not). Let Q be a search query, Δ the set of queries, thr a certain threshold, $N_{target}(Q)$ the number of all occurrences of the query Q in the evaluation data, $N_{HIT}(Q, thr)$ the number of detections of the query Q whose score remains above the

[2]http://www.fee.vutbr.cz/SPEECHDAT-E/.

threshold thr and are therefore considered hits, and $N_{FA}(Q, thr)$ the number of false detections (i.e., FAs) of the query Q with a score larger than thr.

Three different metrics have been used for QbE STD evaluation. The first one is the nonpooled *figure-of-merit* (denoted *npFOM* in this article to differentiate it from another figure-of-merit we will present shortly), defined by NIST [NIST 1991]. It is an upper-bound estimate of the keyword-spotting accuracy averaged over 1 to 10 false alarms per hour. The npFOM estimation assumes that the total duration of the evaluation speech is T hours. For each query Q, all detections are sorted by score. The nonpooled hit percentage $npP_{HIT}(i)$ of queries found before the ith false alarm is calculated for $i = 1 \ldots N + 1$ where N is the first integer $\geq 10T - 0.5$. The nonpooled figure-of-merit is then defined as

$$npFOM = \frac{1}{10T}(npP_{HIT}(1) + npP_{HIT}(2) + \cdots + npP_{HIT}(N) + anpP_{HIT}(N+1)), \quad (1)$$

where $a = 10T - N$ is a factor that interpolates to 10 false alarms per hour and $npP_{HIT}(i)$ is defined as

$$npP_{HIT}(i) = \sum_{Q \in \Delta} \frac{N_{HIT}(Q, npthrFA(Q, i))}{N_{target}(Q)} \times 100\%. \quad (2)$$

The auxiliary function $npthrFA(Q, i)$ finds the proper threshold thr for given query Q and the ith false alarm per hour.

A putative detection is considered to be a hit in case the midpoint of a reference occurrence is between the start and end times where the given detection resides, according to the npFOM definition in Young et al. [2006]. A higher npFOM value means better performance.

We also present the system performance with the pooled *FOM*. Here, all terms are considered together, which makes it impossible to tune a term-dependent threshold. The pooled figure-of-merit is defined as

$$FOM = \frac{1}{10T}(P_{HIT}(1) + P_{HIT}(2) + \cdots + P_{HIT}(N) + aP_{HIT}(N+1)), \quad (3)$$

where $a = 10T - N$ is a factor that interpolates to 10 false alarms per hour and $P_{HIT}(i)$ is a pooled hit percentage of queries found before the ith false alarm:

$$P_{HIT}(i) = \sum_{Q \in \Delta} \frac{N_{HIT}(Q, thrFA(i))}{N_{target}(Q)} \times 100\%, \quad (4)$$

where the auxiliary function $thrFA(i)$ finds the proper threshold thr for the ith false alarm in all (pooled) queries.

Finally, the equal-error-rate (EER) is used. It reflects the QbE STD system "accuracy" for threshold thr_{EER}. It represents the percentage of missed detections for the threshold where the system achieves the same number of missed detections and false alarms. The EER is a pooled metric and is defined as follows:

$$EER = \frac{\sum_{Q \in \Delta} N_{target}(Q) - N_{HIT}(Q, thr_{EER})}{\sum_{Q \in \Delta} N_{target}(Q)}, \quad (5)$$

where the following equation is satisfied:

$$\sum_{Q \in \Delta} N_{target}(Q) - N_{HIT}(Q, thr_{EER}) = \sum_{Q \in \Delta} N_{FA}(Q, thr_{EER}). \quad (6)$$

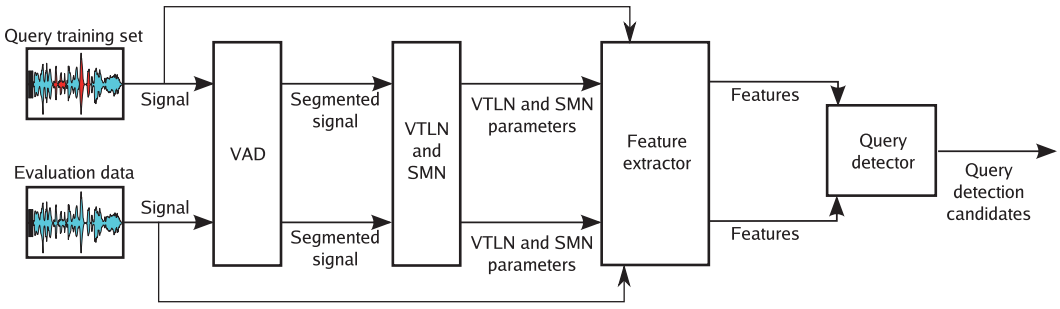

Fig. 1. High-level schema of our query-by-example STD system.

As in the pooled FOM metric, a detection is considered to be a hit in case its start and end times are within a 100ms shift of those of the reference. This high restriction level we impose on both pooled metrics may lead to a low FOM value and a high EER value when evaluating the approaches.

FOM and EER metrics are computed from the whole set of ranked detections—the list of detections is first ordered from the best to the worst score, and FOM and ERR values are computed according to Eqs. (3) and (5). For FOM, a higher value is better, while for EER a lower value is better. It must be noted that our EER definition, as well as that of FOM, does not take into account term labels, and therefore the same threshold thr_{EER} is used for all the terms.

"Pooled" and "nonpooled" metrics reflect the system behavior in different ways. A "pooled" metric considers all detections of all terms gathered to one set and scored independently of the term (query) label. A "nonpooled" metric considers that an individual value of the metric (FOM in our case for npFOM) is calculated for each term separately. Then, npFOM is a weighted average of FOMs for the whole set of terms, where each term's contribution depends on the number of its reference occurrences. Therefore, the smaller the difference between pooled and nonpooled FOM (i.e., FOM and npFOM) is, the better the system is calibrated.

4. OVERVIEW OF QUERY-BY-EXAMPLE SYSTEM: AUDIO PREPROCESSING, FEATURE EXTRACTION AND QUERY-BY-EXAMPLE DETECTION

Our QbE STD system is represented in Figure 1. It contains four different blocks: voice activity detection (VAD); vocal tract length normalization (VTLN); and speaker mean normalization (SMN) blocks represent standard preprocessing steps. First, a VAD is employed to filter out nonspeech parts of the audio representing the queries and the utterances for proper estimation of the speaker normalization/adaptation parameters in the subsequent step. In the second step, we apply both VTLN and SMN to the remaining audio. The feature extractor and the query detector blocks represent the core of our QbE STD system. Since one of our goals is to evaluate the different techniques referred to earlier within a language-independent QbE STD setup, the core of the full QbE STD system is split into two blocks: *Feature extractor* encodes the speech in low dimensional feature vectors, and the *query detector* (or spoken-term detector) hypothesizes putative query detections from the features.

We experimented with two sets of features: 3-*state phone posteriors* derived from the output of an artificial neural net (NN) classifier and *bottle-neck features*, which are also based on a NN classifier, derived as output of a hidden compression layer of the NN.

We experimented with three configurations of the QbE detector: (1) a DTW-based approach where a DTW-based search over a phonetic posteriorgram matrix hypothesizes detections; (2) a GMM/HMM-based approach where an AKWS-based search is

Table II. Combination of Feature Extraction and QbE STD Approaches in this Article
The WFSTdict system is derived from our WFST approach where "language-dependent" lattices were generated and reference query pronunciations (i.e., as in text-based STD) were used for searching.

	Query-by-Example detector			Upper-bound	
Feature extraction	DTW	GMM/HMM	WFST	AKWS	WFSTdict
3-state phone posteriors	X	X	X	X	X
Bottle-neck features		X			

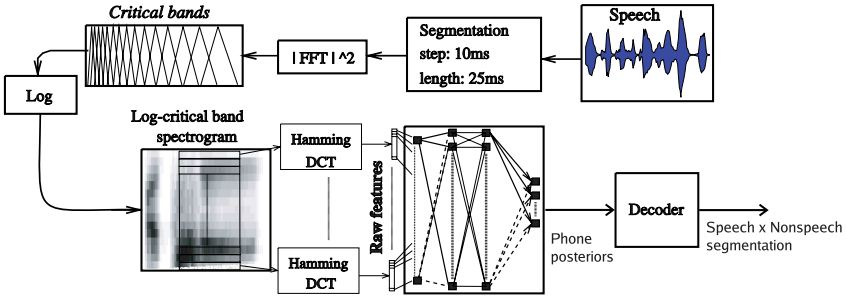

Fig. 2. Voice activity detector.

employed from a GMM/HMM representing the query and the background models; (3) a WFST-based approach, where phone lattices are employed to represent both the query examples and the evaluation data and a WFST-like framework hypothesizes detections. Table II presents the features used within each query detector in this article.

4.1. Voice Activity Detection

Two-step VAD is conducted to properly derive the speaker normalization/adaptation parameters, as depicted in Figure 2. The first step is based on a simple set of heuristics applied on spectrum, energy and signal to produce raw features. This step filters out silence, stationary and/or technical noises (beeps or faxes). Next, this "clean" signal (raw feature) is sent to a 4-layer NN, with 200 neurons in each of its two hidden layers and $N + 1$ outputs in the output layer that represent N phones and one silence. The output of the NN is further passed through a decoder to obtain the phone segmentation. All nonsilence phones are then merged into `speech` segments. It should be noted that to derive the raw features, both VTLN and SMN techniques are omitted. For the VAD NN, the length of the temporal patterns is 310 ms ($L_{TP} = 31$). The patterns are further reduced by the discrete cosine transform (DCT) to 16 coefficients ($L_{DCT} = 16$). The reason we used a NN-like VAD is its better performance than standard GMM/HMM-based approaches for phone recognition [Schwarz 2009].

4.2. Speaker Adaptation

The VAD segmentation is taken "as is" for the VTLN parameter estimation, while for a robust SMN parameter estimation, speech segments are expanded by a 100 ms-length silence. These parameters are applied on the speech representing the queries and the utterances before going into the feature extraction block (see Figure 1). Informed by the results of previous experiments [Szöke et al. 2010], we did not use speaker-based variance normalization.

The speaker adaptation is on an utterance basis. Parameters are estimated on the speech of the whole telephone call in the case of Levantine, Czech, and English data. For Hungarian data (SpeechDat corpus), all utterances belonging to the particular speaker

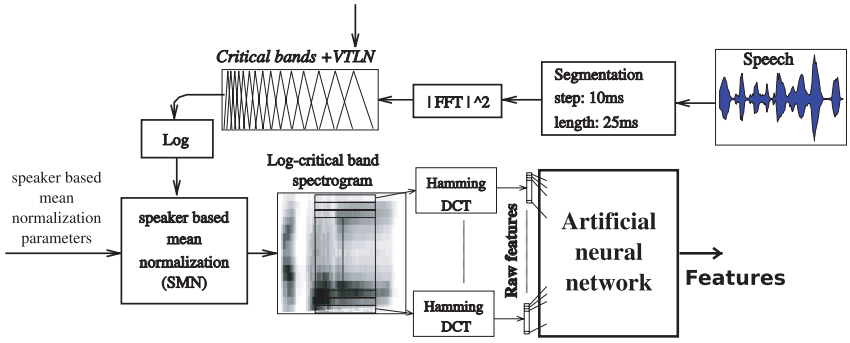

Fig. 3. Feature extraction.

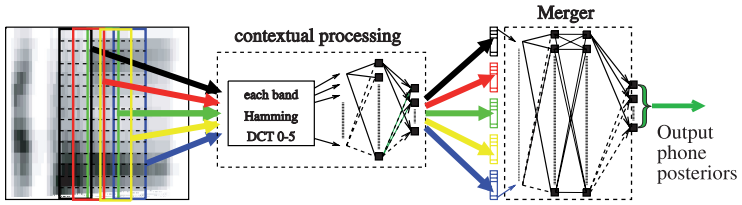

Fig. 4. Universal context (UC) neural net architecture.

are concatenated to one "utterance" and then the speaker adaptation parameters are estimated.

The fast VTLN estimation proposed in Welling et al. [1999] is employed to derive the VTLN parameters properly. This approach uses maximum a posteriori (MAP) adaptation from a universal background model (UBM), with 32 diagonal Gaussians to derive specific models for each warping factor. These models are next retrained using the maximum mutual information (MMI) criterion. The features used to derive these models are 13 perceptual linear prediction (PLP) coefficients, including c_0 with deltas and double deltas (without any normalization).

5. FEATURE EXTRACTOR

This block, depicted in Figure 3, converts the input audio signal to features (3-state phone posteriors or bottleneck features). The input speech is first segmented into 25ms frames with a 10ms frame-shift, and its power spectrum is calculated for each frame. Pre-estimated VTLN is applied, and energies from 15 Mel-scale critical bands, ranging from 64 Hz to 3800 Hz, are extracted and passed through a logarithm. Next, speaker mean normalization is performed. We obtain a log-critical band spectrogram (CRB), from which long temporal patterns of length 15 are extracted. Hamming window and dimensionality reduction by DCT to six coefficients are applied to each long temporal critical band trajectory. Finally, these reduced temporal patterns are concatenated to one feature vector to derive the raw features in Figure 3 which are next fed into the NN. These raw features are the same as those used in the VAD, but here, VTLN and SMN are applied.

The topology of the NN classifier to derive the final set of features in Figure 3 is crucial. Based on our previous experiments in LVCSR [Grézl et al. 2009], we use a hierarchical structure called a *bottleneck universal context network*, depicted in Figure 4, which consists of two different parts: a context network and a merger.

Table III. Numbers of Phones and Sizes of Layers for Different Languages
$size(hid\_UC)$ represents the size of the hidden layer in the universal context NN. $size(hidMer\_3stphn)$ is the size of the hidden layers in the merger with 3-state phone posterior output. $size(hidMer\_BN)$ is the size of the hidden layer in the merger with bottleneck output, and $size(out)$ is the size of the 3-state phone posterior output layer.

Language	Phones	$size(hid\_UC)$	$size(hidMer\_3stphn)$	$size(hidMer\_BN)$	$size(out)$
Czech	37	1373	495	871	114
English	44	1298	488	840	135
Hungarian	64	1128	470	765	193
Levantine	32	1432	500	894	99

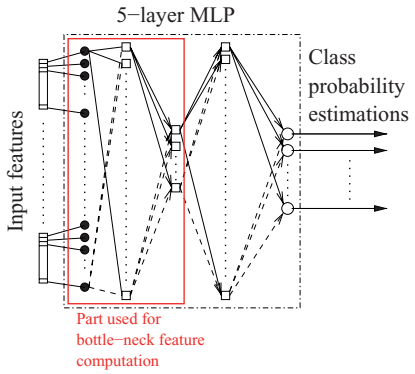

Fig. 5. Training of the context bottleneck neural network. A 5-layer neural network with an 80-neuron bottleneck layer in the middle and 3-state phone posterior classes are trained first. Then the 4^{th} and 5^{th} layers are removed and the output of the network is taken from the bottleneck layer.

The input of the context network is a context of 15 frames around the current one, each represented by six DCT coefficients. The input size is $15 \times 6 = 90$. The context NN is a so-called *bottleneck network*. It is trained as a five-layer network with the 3^{rd} layer as the bottleneck of size 80 neurons. The size of the 2^{nd} and 4^{th} layers are $size(hid\_UC(Lang))$ and the number of outputs (5th layer) $size(out(Lang))$, corresponds to the number of phone states: $size(out(Lang)) = 3 \times (phn(Lang) + 1)$ and $phn(Lang)$ is the number of phones of the language on which the feature extractor is trained. For $size(hid\_UC(Lang))$, it holds that the size of the whole NN (five layers) is fixed to 500k parameters. Since the size of the output layer depends on the number of phones, the size of the hidden layers also depends on the language. Table III shows the sizes of this output layer for each language.

After training the context network as a 5-layer network, the fourth and fifth layers are cut-off so the output size of the context network is 80, as shown in Figure 5.

The merger receives five context net outputs sampled every five frames (for frame t, this is $t - 10, t - 5, t, t + 5, t + 10$), so it actually "sees" a 310ms context in the CRB matrix, and the merger input size is $5 \times 80 = 400$. The merger is one of the following.

—*A standard 4-layer NN.* Its outputs are $size(out(Lang))$ 3-state phone posteriors (including silence). An example of a 3-state phone posteriorgram is depicted in Figure 6.
—*Or A 5-layer bottle-neck NN.* Its outputs are $size(out(Lang))$ and they are used only for training. The size of the bottleneck is fixed to 30 for all languages.

There are several differences between posterior and bottleneck features.

—The most remarkable difference is the size of the feature vector. The posterior feature vector size varies according to the language on which the feature extractor is trained,

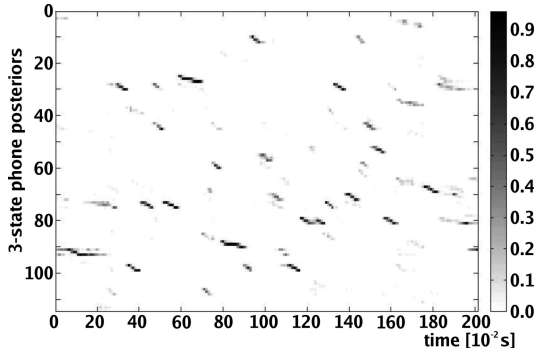

Fig. 6. An example of 114 3-state phone posteriors (38 phones times 3 states per phone) for 2 seconds of speech. The 3 states represent beginning, center, and end of a phone. The x-axis represents the time in a hundredth of a second and the y-axis represents the 3-state phone posteriors.

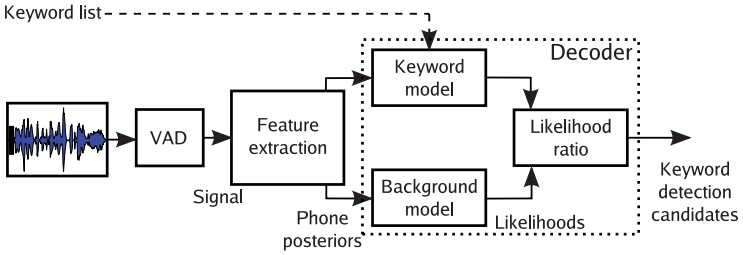

Fig. 7. Schema of acoustic keyword-spotting system.

according to the column *size*(*out*) in Table III. On the other hand, the bottleneck feature vector always has the size fixed to 30.

—Theoretically, the amount of information encoded in the feature vector should be the same. This is because the bottleneck neural network is trained to classify the same number of classes as the "posterior" neural net. In LVCSR, bottleneck features achieved higher accuracy than posterior features (reduced to the same dimensionality) [Grézl et al. 2007].

—An important difference relies on the distribution of the features. While bottleneck features have a normal (Gaussian) distribution, 3-state phone posteriors have a non-Gaussian distribution. Posterior features have a limited range from 0 to 1. According to Grézl and Fousek [2008], the bottleneck features are Gaussian and can be approximated by a GMM more accurately than by phone posteriors.

—The last difference is in computational effectiveness. Both posterior and bottle-neck features are trained with an equal number of parameters in the merger. However, the size of the bottleneck NN merger (3-layer NN) is half the trained size (5-layer NN) (see Figure 5).

6. ACOUSTIC KEYWORD SPOTTING (AKWS) – UPPER-BOUND EXPERIMENT

We took the acoustic keyword spotting as our upper-bound technique and also derived the GMM/HMM approach for QbE STD from it. The schema of the acoustic keyword spotter appears in Figure 7. First, the utterances are passed through a VAD that removes the nonspeech parts. Next, the remaining audio is converted to 3-state phone posterior features, whose logarithm is taken and fed into the decoder. The keyword-spotting network in the decoder is depicted in Figure 10, and is built from the given

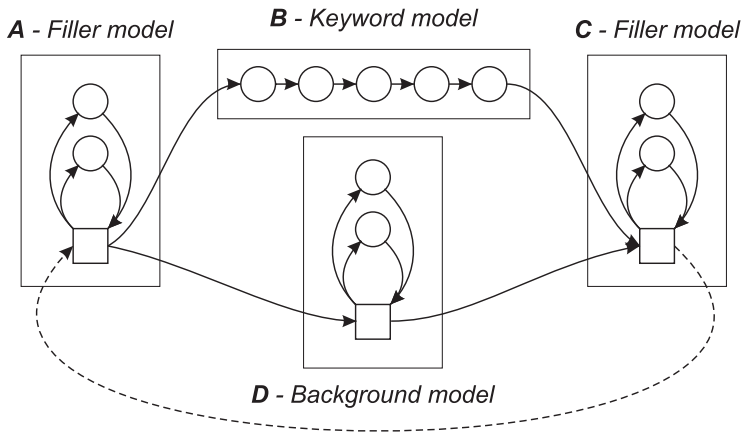

Fig. 8. General acoustic keyword-spotting network.

set of queries (i.e., keywords) and their pronunciations. The output of the decoder is a set of hypothesized detections whose final score is the likelihood ratio divided by the length of the detection to compensate for the different length of the queries. Note that our acoustic keyword spotter contains neither language model nor vocabulary (except the list of searched keywords). The searched keywords do not affect each other. Also note that the AKWS system is not a state-of-the-art spoken term detection system, since the list of query terms is used for speech decoding, and does not employ any language model, but it is the most comparable standard language-dependent approach. A full description of the acoustic keyword-spotting system can be found in Szöke [2010].

The core of our AKWS system is a standard Viterbi-based decoder, modified to calculate the likelihood ratio (see Figure 8). The filler models (`A`) and (`C`) should model all the speech preceding and succeeding the keyword, and are represented by a free phone loop. The keyword model (`B`) is a concatenation of phone models of which the keyword consists. The background model (`D`) is again a single phone loop. It must be noted that this configuration allows for multiple keywords to appear in a single utterance and multiple instances of the same keyword in the same utterance.

The utterance is modeled using model `A-B-C` (concatenation of models `A`, `B`, and `C`), and model `A-D-C` (concatenation of models `A`, `D`, and `C`). Models `B` and `D` score exactly the same part of utterance, so the likelihoods of models `A-B-C` (L_{ABC}) and `A-D-C` (L_{ADC}) differ only because of models `B` and `D`. If there is a keyword beneath model `B`, $L_{Ratio} = L_{ADC}/L_{ABC}$ will approach 1 and will be lower for nonkeywords. If a noise appears in the speech, both likelihoods L_{ABC} and L_{ADC} will be lower, but due to the likelihood ratio, the influence of noise should be limited.

For simplification, the filler model `C` is omitted, as depicted in Figure 9, since L_C is a constant on both sides of the term, and hence it is simpler to implement likelihood ratio calculation just after model `B`. In addition, in the real recognition network, there is only one phone loop representing both `A` and `C`, as depicted in Figure 10. This simplification does not affect the ability of the keyword spotter to detect any number of putative hits of a term in the whole utterance. The keyword spotter produces a term likelihood ratio for each frame.

7. DTW-BASED QUERY-BY-EXAMPLE DETECTOR

The DTW-based QbE detector relies on template matching. The features (i.e., 3-state phone posteriors) are used to compute a similarity measure between query and

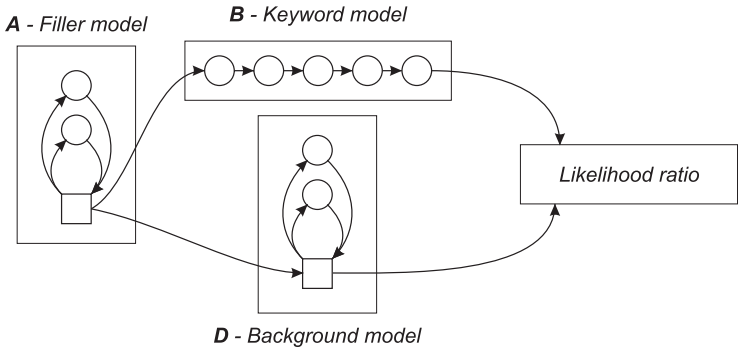

Fig. 9. Likelihood ratio computation in acoustic keyword spotting.

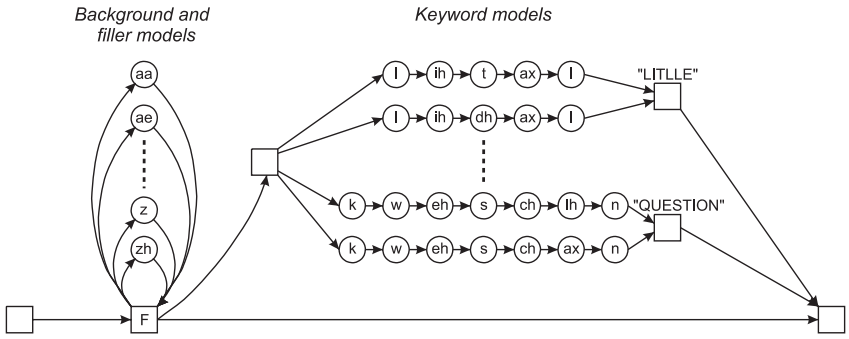

Fig. 10. Example of a real keyword-spotting network in AKWS. Each phone model is represented by a 3-state hidden Markov model tied to 3-state phone posteriors.

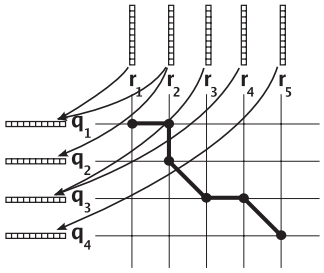

Fig. 11. An example of a combination of posterior vectors of the best example (Q) and the worst example (R) using the DTW path.

utterance and the template matching employs a variant of DTW-like search to hypothesize detections.

To explain our DTW-based approach for QbE, let us first explain how the query is constructed from its five examples and next how the similarity matching of posteriorgrams works. Finally, we explain how the DTW-like search hypothesizes detections.

7.1. Query Construction from Example Combination

As shown in our previous work [Tejedor et al. 2010], a combination of several individual examples into one "average" representative of the query, should lead to a better performance than using a single example as a query. As in Tejedor et al. [2010], the

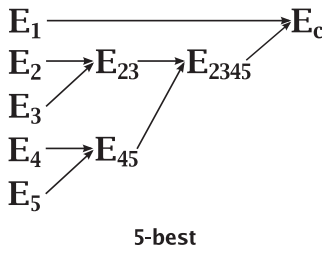

Fig. 12. Combination of five examples. The E_c is the final "average" (combined) example.

combination in this work for two or more query examples relies on a feature level-based combination. In so doing, the new query example is built from multiple single examples. First, let us explain how the combination of two examples works: (1) order the examples by score according to a certain metric; (2) run the DTW search with the best example acting as "query" and the worst acting as "utterance"; and (3) update the phone state posteriors of the best example ("query") with the phone state posteriors of the worst example ("utterance") according to the best path derived from the DTW search.

Let us now explain the process in detail for k examples (five in our case). We order the examples by a DTW-based metric: a DTW search is conducted in the same way as in the search step for every example. Therefore, a $k \times k$ scoring matrix is derived in which the score produced by the DTW search of each query example on the rest of the examples is stored. The individual score assigned to each example $c_{DTW}(E_i)$ is the sum of the ith row in the scoring matrix. As during the search phase, the similarity function that will be explained in Section 7.2 (i.e., cosine distance) was employed to compute the score for each example. The example with the lowest score is considered to be the best.

For the third step, where two examples have to be combined, let us define the best example $Q = \{\vec{q}_1, \ldots, \vec{q}_N\}$ containing N frames and the worst example $R = \{\vec{r}_1, \ldots, \vec{r}_M\}$ containing M frames. Let us define P as the best path found by the DTW search between Q and R, containing pairs of indices pointing to Q and R, respectively.

The combination of the best and worst example posterior matrices consists of updating the phone state posteriors of Q by the frames of R according to path P. The new value of $\vec{q}_i$ is simply computed as an average of $\vec{q}_i$ and all $\vec{r}_j$ assigned to i by path P:

$$\vec{q}_i^{new} = \frac{\vec{q}_i + \sum_{\forall j:\{i,j\}\in P} \vec{r}_j}{1 + N_j}, \tag{7}$$

where N_j is the number of vectors in R belonging to $\vec{q}_i$.

To combine more than two examples, the combination must be split in several example subcombinations. In so doing, we get the 5-example-based combination referred to before, as follows: we combine the fourth and fifth examples into a temporary one E_{45}, and the third and second ones into another one, E_{23}. After that, we combine these two temporal examples into E_{2345}, and finally we merge E_{2345} with the best example to derive the merged query E_c, as depicted in Figure 12. It must be noted that in this combination, the final length of the combined example keeps the length of the first example. The reason is to follow the same length of the first (best) example in the combined example as the final query length.

7.2. Similarity Matching of Posteriorgrams

Inspired by our previous work [Tejedor et al. 2010], we compute the similarity between the query example and regions of the utterance from phonetic posteriorgrams, as illustrated in Figure 6. The phonetic speech classes are 3-state phones, similar to

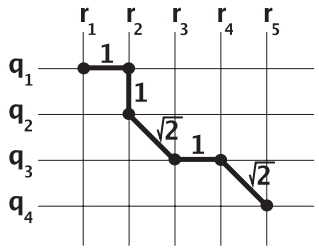

Fig. 13. An example of a path and a path cost normalization for the DTW search.

standard HMM in our case. To hypothesize similar audio segments in the utterance and the query, a similarity function is needed. Based on our previous work [Tejedor et al. 2010], our similarity function is a log-likelihood based on the *cosine distance*:

$$D(\vec{q}, \vec{r}) = -\log\left(\frac{\vec{q} \cdot \vec{r}}{|\vec{q}| \cdot |\vec{r}|}\right), \tag{8}$$

where $\vec{q}$ is a vector drawn from a 3-state phone posteriorgram of the query and $\vec{r}$ comes from the searched utterance.

We compute the similarity between each individual posterior distribution for all N frames representing the query against each individual posterior distribution for all M frames representing the utterance. It results in an $N \times M$ matrix.

7.3. DTW-Based Query Detector

As query detector, a standard DTW search is conducted to hypothesize regions that match the query well with putative segments in the utterance. It is run iteratively starting in every frame in the utterance and ending in a frame of the utterance. The DTW search finds the minimum scoring path via the similarity matrix. After the DTW search, overlapped regions that hypothesize the same term are removed and the utterance region whose score produces a local minimum is sent to the output. The final score for every path computed during the DTW search is normalized by the length of the path. As in our previous work [Tejedor et al. 2010], right or down steps have cost 1, and diagonal steps have cost $\sqrt{2}$ (see Figure 13).

8. GMM/HMM-BASED QUERY-BY-EXAMPLE DETECTOR

Filler model-based acoustic keyword spotters, like the one in Section 6, have been successfully applied when spotting words from speech signals [Manos and Zue 1997; Cuayahuitl and Serridge 2002; Kim et al. 2004; Xin and Wang 2001; Ou et al. 2001; Szöke et al. 2005; Hazen and Bazzi 2001; Tejedor 2009]. However, this approach is still language-dependent, since both the keyword models and the filler models are built from a predefined set of phone models belonging to a target language. To address language-independence in our QbE STD task, this language-dependence should be mitigated. In this direction, our GMM/HMM-based QbE STD system is inspired by acoustic keyword spotting.

The query (keyword) model in AKWS is a linear concatenation of phone models representing the pronunciation of the keyword. We retain an acoustic representation of the query in GMM/HMM-based QbE, as previously discussed in Section 6, but in contrast with concatenation of pretrained phone models in AKWS, the query GMM/HMM is trained on examples. The number of states of each query model is set to three times the number of phones of which the pronunciation consists (in the "query" target language). This is the only knowledge we use from the "query" language in terms of word/phone transcriptions. In our future work, this number of phones will be estimated from the

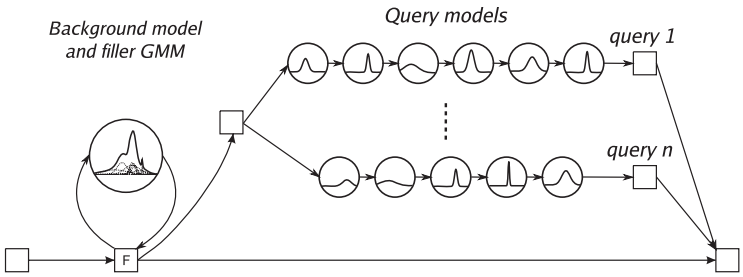

Fig. 14. Recognition network for the GMM/HMM-based QbE detector. The background model is represented by a single state with 40 Gaussian mixtures. The query model is a linear concatenation of states (HMM), each represented by a single Gaussian.

length of the query to completely remove the language-dependency of this approach. The five examples from the training query set (see Section 2) were used to train each query model. One GMM component was found to be optimal to model each state in experiments (by a margin of at least 2.6% relative compared with more GMM components both for language-dependent and language-independent setups).

In contrast to AKWS, where the background model consists of a free loop of phone and silence models (see Section 6), in our GMM/HMM-based QbE detector, we define the background model as a GMM, whose number of components was empirically set to 40. This background model was trained on the query training set, and hence data from the target language is necessary. However, no transcription is needed.

The same decoder as for AKWS is employed to hypothesize detections, with the recognition network illustrated in Figure 14. As in the AKWS system, the likelihood ratio divided by the length of the detection represents each detection score. Both 3-state phone posteriors and bottleneck features have been experimented with for query/background modeling in the GMM/HMM-based QbE detector.

9. WFST-BASED QUERY-BY-EXAMPLE DETECTOR

All the approaches presented earlier used phone posterior features directly when searching for queries. In this section, we work with phone lattices[3] derived from posterior features, representing both queries and utterances. This approach aims at finding all occurrences of the query lattice inside the utterance lattice, while preserving the timing and score of each occurrence. Weighted finite state transducers (WFSTs) offer a well-defined framework for this purpose.

Inspired by the work of Parada et al. [2009] who aimed at searching OOV terms using WFSTs, we focus on the language-independent aspect of QbE. Instead of using LVCSR or hybrid word/phone lattices, we create phone lattices from posterior features by a simple phone loop. For working with transducers, we use the OpenFst toolkit [Allauzen et al. 2007].

Examples of queries are cut from query-training set phone lattices. Both query and evaluation lattices are converted to a WFST representation. We obtain a transducer R, representing all detections of the query in the evaluation data, by a composition of transducers [Allauzen et al. 2007]:

$$R = E \circ Q, \tag{9}$$

where E is a transducer for evaluation data, and Q represents the query. We present some examples of transducers in Figures 15, 16, and 17. Let us now describe the process of converting lattices to E and Q transducers.

[3]Acyclic graphs of phone hypotheses.

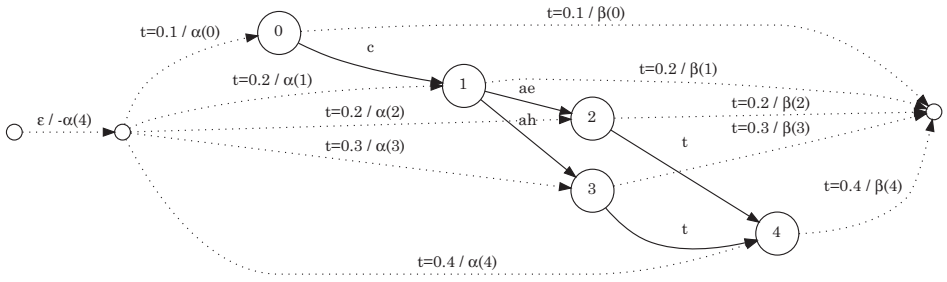

Fig. 15. An example of a WFST E.

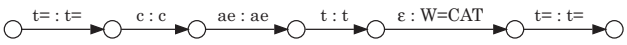

Fig. 16. An example of a term "CAT" encoded into Q transdcuer.

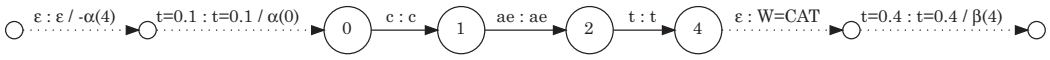

Fig. 17. Result of a composition $R = E \circ Q$.

9.1. Converting Evaluation Lattices to WFSTs

To be able to search in lattices using the WFST framework, we need to properly convert them to a transducer representation. We inject links with timing information to the WFST, so that each possible path through an evaluation WFST E has to start and end with a time link. After a composition with a query lattice Q, each possible path in the resulting WFST R starts and ends with a time link. In this way we preserve the timing of each detection.

When injecting time links, we add links with timing information in labels from the start node of the lattice to all other nodes, and similar links from all nodes to the final node. Then, we can jump in and out of the WFST only through these time links, as depicted in Figure 15. To preserve correct posterior probabilities, we need to set a forward probability in each start-time link and a backward probability in each end-time link. To normalize all paths by the full probability of the whole lattice, another link is prefixed to the WFST. A detailed algorithm for creating the transducer E follows.

ALGORITHM 1:

(1) Each link $L_{ij} : N_i \rightarrow N_j$ is copied from a lattice to the transducer E, containing a phone label and the link's likelihood l. In our case we used only acoustic likelihoods l_a (with an optionally added phone insertion penalty pen_{ins}). Weights in the log semiring are encoded in $-\log$ domain: $weight(L) = -\log(l_a) + pen_{ins}$.

(2) Forward (α) and backward (β) likelihoods are computed for each node in the lattice using the Baum–Welch forward-backward algorithm [Baum et al. 1970].

(3) Two new nodes $N_{first-1}$ and N_{last+1} are added. Then, for each node i, two time links are added: $L_{in} : N_{first-1} \rightarrow N_i$ with $weight(L_{in}) = \alpha(N_i)$ (forward likelihood to node i) and $L_{out} : N_i \rightarrow N_{last+1}$ (N_{last} is the lattice's final node) with $weight(L_{out}) = \beta(N_i)$ (backward likelihood to node i). The label of both links carries the time of the node i (e.g., "t=14.23").

(4) A new node $N_{first-2}$ and a new link $L_{norm} : N_{first-2} \rightarrow N_{first-1}$ with an ϵ label and $weight(L) = -\alpha(N_{last})$ (likelihood of the best path through the lattice) are prefixed to the transducer, where $N_{first-1}$ is the node from which all time links start and $N_{first-2}$ is a new start node.

Figure 15 illustrates such a transducer: any path traversing E must now start with links L_{norm}, L_{in}, and must end with L_{out}. Therefore, the links carrying time information in their labels are always at the beginning and at the end of each path. Also, the weight of each path corresponds to the log posterior probability of the path, as follows:

$$\begin{aligned} weight(path(L_1, L_2, \ldots L_M)) \;=\; & -\alpha(N_{last}) \\ & +\alpha(start(L_1)) \\ & +l(L_1)+l(L_2)+\cdots+l(L_M) \\ & +\beta(end(L_M)), \end{aligned}$$

where $start(L)$ and $end(L)$ represent start and end nodes of link L. It is obvious that in linear domain:

$$p(path) = \frac{\alpha(start(L_1))l(L_1)l(L_2)...l(L_M)\beta(end(L_M))}{\alpha(N_{last})} \tag{10}$$

so that $p(path)$ represents the posterior probability for a certain path.

Another way of converting evaluation lattices to WFSTs is described in Parada et al. [2009], who put the timing information on the output label of each link and then used a weight-pushing algorithm to convert weights (likelihoods) to posterior probabilities, as previously described by Allauzen et al. [2004].

9.2. Preparing Query WFSTs

Similarly to evaluation lattices, we also have to convert the query lattices to transducers, but the process is slightly different. First, we need to cut out the part of a lattice where the query example resides. The example lattice is then converted to a transducer Q suitable for a composition with evaluation transducer E.

We use special WFST symbols ρ or σ (denoted by "t=" in our case) for the first and the last links of Q, which match all start- and end-time links in the utterance WFST E. The time labels of E are then copied to the output labels of R, providing us with the timing information of detections.

In case we know the dictionary pronunciation of a word (text-based WFST STD), we can simply create a chain of phones and add the time-consuming labels and keyword label as shown in Figure 16. In this case, the query is simply a phone string.

In QbE STD, we want to use phone lattice examples of terms as queries. By knowing the start- and end-times of an example in a query training utterance, we want to cut it from a phone lattice, while preserving correct posterior probabilities. All the links overlapping the example's start time will be reconnected to a joint start state, and similarly links overlapping with the example's end time will be reconnected to a joint end state. Weights of these starting and finishing links are set to forward or backward probabilities to those links in the original lattice. The rest of the lattice outside the example's time boundaries is thrown away. More formally, for cutting examples with time boundaries cut_start_time and cut_end_time and creating the transducer Q, we use the following algorithm.

It must be noted that our approach for cutting examples from lattices differs from the one described in Parada et al. [2009], where the authors preprocessed the labels of a WFST and then extracted only the desired part of the lattice from a composition with a filter transducer.

9.3. Composition and Merging of Detections

From the composition in Eq. (9), we obtain a transducer R, in which all paths represent detections of Q in E. All these paths start with a time label (e.g., "t=0.15") and end with a term label (e.g., "W=HELLO") followed by an end-time label (e.g., "t=0.40"). All paths

ALGORITHM 2:

(1) For each link $L_{cut\_start} : N_i \rightarrow N_j$, where $time(N_i) < cut\_start\_time$ and $time(N_j) > cut\_start\_time$, the link's start node is replaced by N_{first} with $weight(L_{cut\_start}) = \alpha(N_j)$. Similarly for links traversing the cut_end_time, the end node is replaced by N_{last} with $weight(L_{cut\_end}) = \beta(start(L_{cut\_end}))$.

(2) A new link $L_{tstart} : N_{first-1} \rightarrow N_{first}$ with $weight = \alpha(N_{last})$ and the ρ label ("t=") is prefixed to the lattice.

(3) A new link $L_{term} : N_{last} \rightarrow N_{last+1}$ with an output label describing the term (e.g., "W=HELLO") is suffixed to the lattice. The input label has to be set to ϵ to be correctly composed with E.

(4) A new link $L_{tend} : N_{last+1} \rightarrow N_{last+2}$ with the ρ label ("t=") is suffixed to the lattice.

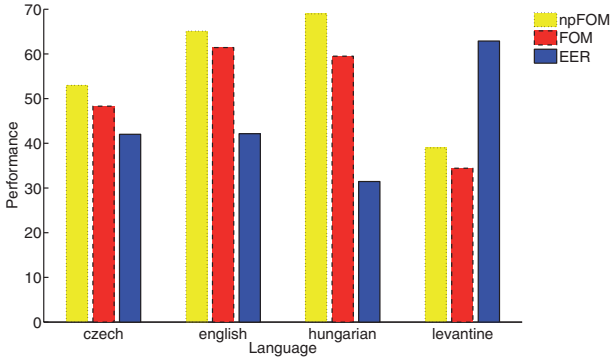

Fig. 18. Results of acoustic keyword-spotting experiments.

of R must then be traversed and overlapping detections of the same term must be merged. In that case, weights of the overlapping detections are summed up (*logadd* in the *log* domain) and the timing of the best detection is taken. For fetching detections from the transducer R, we used dynamic programming. The result of composing E in Figure 15 with Q in Figure 16 is shown in Figure 17.

For our experiments, we composed each evaluation utterance with each example/phone pronunciation of each term separately for both WFST-based QbE STD and text-based WFST STD. Each R transducer was then converted to a list of non-overlapping detections containing the term name, timing, and posterior probability. To merge detections of all five examples of each term in the WFST-based QbE STD approach, we treated the overlapping detections as if they appeared in the same R transducer (weights of the overlapping detections were summed up and the timing of the best detection was taken). Thus the score of each detection is its posterior probability.

10. RESULTS AND DISCUSSION

10.1. Acoustic Keyword Spotting

We built an acoustic keyword-spotting system as an upper-bound reference for each of our evaluation languages. The results are summarized in Figure 18. We can see that the Hungarian system, which is trained and tested on clean read telephone speech provides the best overall performance among all the languages. We can also see (by inspecting the FOM and npFOM results) the differences between both metrics. Pooling brings small deterioration and shows that the likelihood ratio score is slightly dependent on

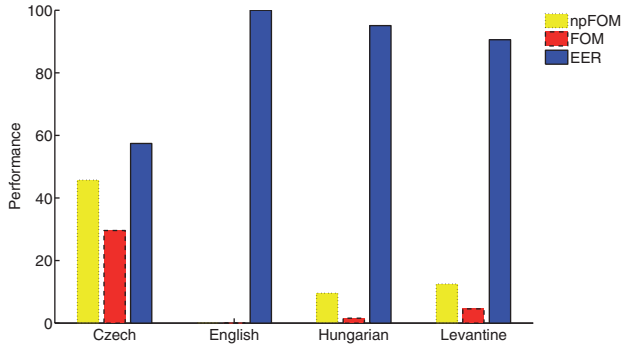

Fig. 19. Results of the DTW-based QbE STD system for the Czech data. The x-axis shows languages used to train the feature extractor.

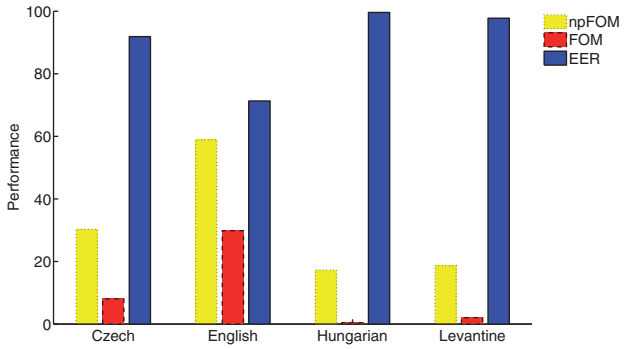

Fig. 20. Results of the DTW-based QbE STD system for the English data.

the query. The worst accuracy was on the Levantine data. This was probably caused by the nondiacritized approach, with a complex structure for acoustic model training and recognition (i.e., we are actually recognizing graphemes, as one grapheme may contain several phones, depending on the context). Czech data exhibits the second worst overall performance, due to its data mismatch (read speech versus CTS).

10.2. DTW-Based Query-by-Example

We evaluate our four DTW QbE STD systems on each of the four target languages. Each system was trained on one of the target languages, and we used 3-state phone posteriors as features. The results are summarized in Figures 19 to 22 for Czech, English, Hungarian, and Levantine data, respectively.

Similar patterns were observed across each language for both the language-dependent and the language-independent feature extractors. As expected, it was observed that the language-dependent setup outperforms the language-independent one, since posterior features are more robust when the feature extractor matches the target language. Paired t-tests show that this improvement is statistically significant ($p < 10^{-13}$) for the English, Czech, and Levantine data. All the figures also show that the phone posterior features in the language-independent setups dramatically decrease the final performance, especially for the pooled metrics. This is due to unreliable scores assigned to each putative hit when unreliable phone posteriors (corresponding to language-independent setups) are fed into the DTW search.

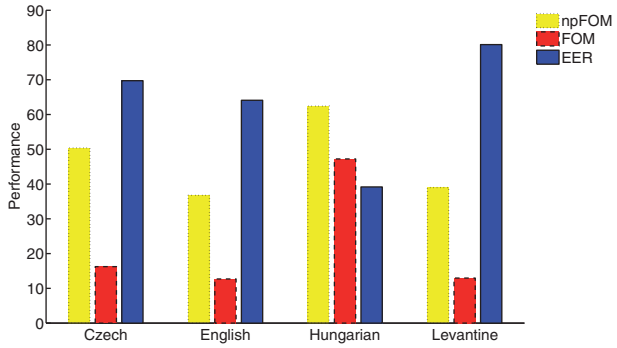

Fig. 21. Results of the DTW-based QbE STD system for the Hungarian data.

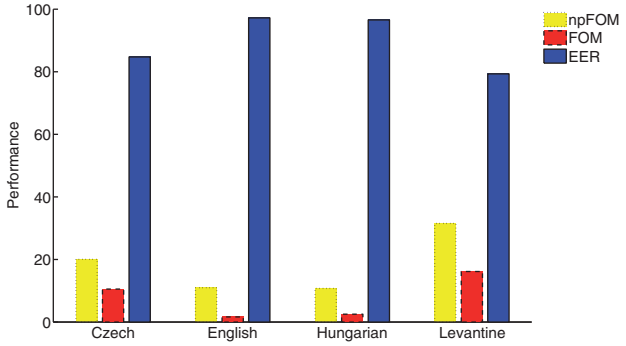

Fig. 22. Results of the DTW-based QbE STD system for the Levantine data.

The results for the Hungarian data exhibit a slightly different behavior for the language-independent setup than the rest of the languages. We can observe that Czech, and even Levantine feature extractors under the npFOM metric, achieved a reasonably good performance, although they are worse than the language-dependent feature extractor. A paired t-test shows the lowest significance with these feature extractors ($p < 0.004$ with the Czech feature extractor and $p = 0.001$ with the Levantine feature extractor). This is probably due to the read nature of the Hungarian data, which may lead to a more robust set of phone posteriors, even when we are dealing with a language-independent feature extractor.

The Levantine data results show the worst performance across the language-dependent feature extractors, as in the acoustic keyword spotting system, due to the nondiacritized approach. The Czech data exhibits the same behavior, due to the mismatch issue (read speech versus CTS).

From the DTW-based experiments, we can conclude that DTW achieves good performance for language-dependent QbE STD, and this performance is dramatically decreased when applied within a language-independent setup.

10.3. GMM/HMM-Based Query-by-Example

We evaluate our GMM/HMM QbE STD systems on each of the four languages. Each system was trained on one of the target languages and has two versions: 3-state phone posteriors and bottleneck features. The results are summarized in Figures 23 to 26 for Czech, English, Hungarian, and Levantine data, respectively.

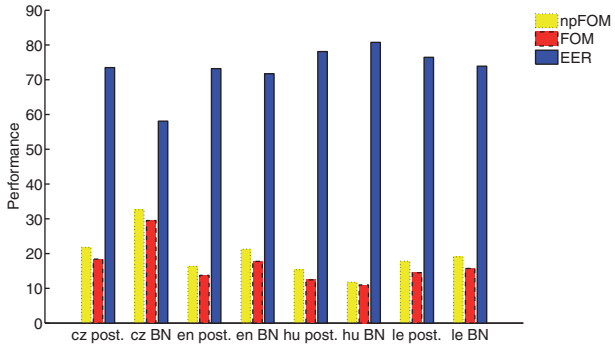

Fig. 23. Results of the GMM/HMM-based QbE STD system for the Czech data. x-axis shows languages used to train the feature extractor. "en" stands for English, "cz" for Czech, "hu" for Hungarian, and "le" for Levantine. "post." refers to the 3-state phone posteriors as features while "BN" refers to the bottleneck features.

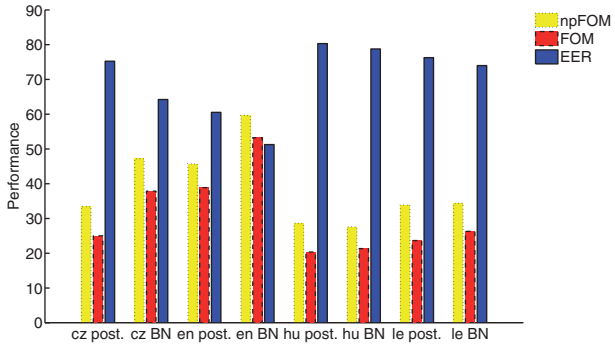

Fig. 24. Results of the GMM/HMM-based QbE STD system for the English data.

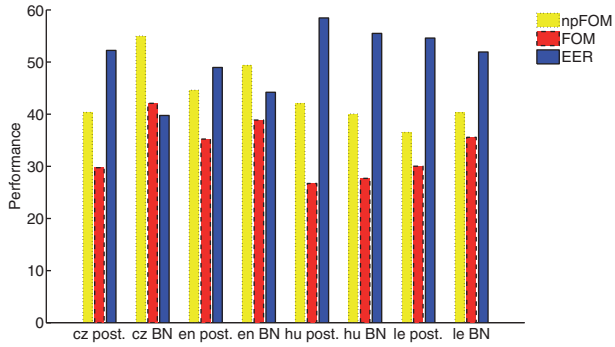

Fig. 25. Results of the GMM/HMM-based QbE STD system for the Hungarian data.

We clearly see that the bottle-neck features outperform the 3-state phone posteriors on a language-dependent scenario except for the Hungarian data and npFOM. Paired t-tests show that this improvement is statistically significant for English and Czech data ($p < 10^{-7}$). However, for Hungarian and Levantine data, there is no significant difference between the two sets of features for each language-dependent feature extractor. We consider that this discrepancy is due to the small amount of data used to train

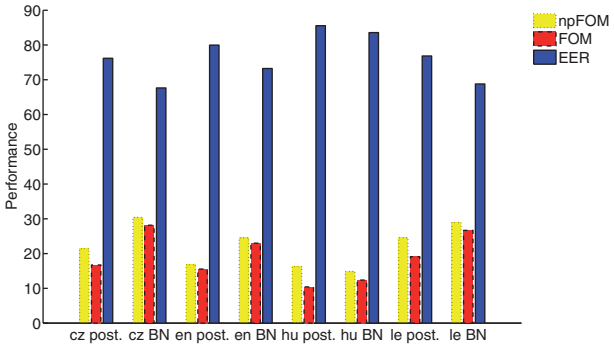

Fig. 26. Results of the GMM/HMM-based QbE STD system for the Levantine data.

the Hungarian feature extractor, which makes a poorer estimation of the bottle-neck features and the nondiacritized approach on the Levantine data, which leads to more complex patterns when estimating the features.

By inspecting the Hungarian data in Figure 25, we can see that the bottleneck Czech feature extractor performs the best, which supports our conjecture that the small amount of data used for training the Hungarian feature extractor results in a worse feature estimation in such a way that a better training of a language-independent feature extractor provides much better performance. This improvement is statistically significant for a paired t-test ($p < 0.01$). In addition, this is also consistent with the results corresponding to the Hungarian feature extractor on the rest of the language-independent datasets.

When the language-dependent and the language-independent feature extractors are compared, we can see the former outperform the latter consistently and significantly ($p < 0.01$), except for the Hungarian data, which is due to the small amount of data employed to train the Hungarian feature extractor and for the Levantine data due to the nondiacritized approach. We also note that the bottleneck features with the Czech feature extractor outperform the language-dependent feature extractor for the Levantine data due to Levantine data complexity, although this difference is not significant ($p \approx 0.2$).

When the performance across the different datasets is compared, we can see that the Levantine and Czech data exhibit the worst overall performance, depending on the feature extractor used. This is because of the data mismatch of the Czech data (read speech versus CTS) and the nondiacritized Levantine data. The Hungarian data, which is read speech, achieved the best performance across each language-independent feature extractor due to the nature of the data.

We can also observe that the degradation from pooled FOM to nonpooled FOM (npFOM) is "stable" across the different data and feature extractors.

We can conclude that the set of features used in the GMM/HMM approach is stable across each target language for both the language-dependent and language-independent setups in such a way that the bottleneck features outperform the 3-state phone posteriors in case the feature extractor is provided with enough training data.

10.4. WFST-Based Query-by-Example

We present two different WFST-based approaches: WFST based on pronunciation dictionaries (WFSTdict) related to text-based STD and WFST based on examples from lattices related to QbE STD. The text-based WFST STD simply searches in a FST-like framework the phone transcription of each query term.

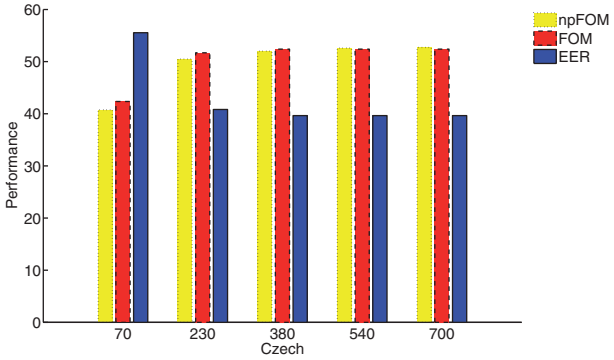

Fig. 27. Results of the text-based WFST STD system for the Czech data: searching for dictionary pronunciations in lattices. The x-axis shows the average number of links per second of speech in the evaluation lattices.

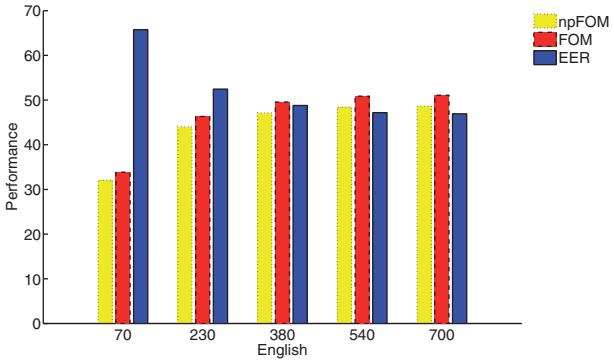

Fig. 28. Results of the text-based WFST STD system for the English data: searching for dictionary pronunciations in lattices.

10.4.1. Dictionary Pronunciations. In contrast to the upper-bound AKWS system described in Section 6 which employs full posterior features, the WFST approach, based on dictionary pronunciation works with lattices, and hence some information is lost due to posterior pruning. The results are summarized in Figures 27 to 30. They show that, as expected, the performance increases with denser lattices since more information is kept. Small differences between FOM and npFOM indicate that scores based on posterior probability are well calibrated in this experiment. The system performance saturates with 700 links per second of speech for all the languages. Therefore, we choose the 700 links per second as the configuration for the following experiments. This configuration allows us to be sure that enough information is kept in the lattices, which is especially needed for the language-independent experiments, where the 3-state phone posteriors become blurred. For a deeper insight into the density of lattices, we present the number of unique parallel links per frame in Figure 31.

The Hungarian data again presents the best overall performance, and again, the Levantine data gives the worst performance, due to the nondiacritized approach, where we model graphemes instead of phones.

10.4.2. Examples from Lattices. We present the language-dependent and language-independent results for searching five examples per query in Figures 32 to 35. Both utterance and query phone lattices were generated with 700 links per second. It can be

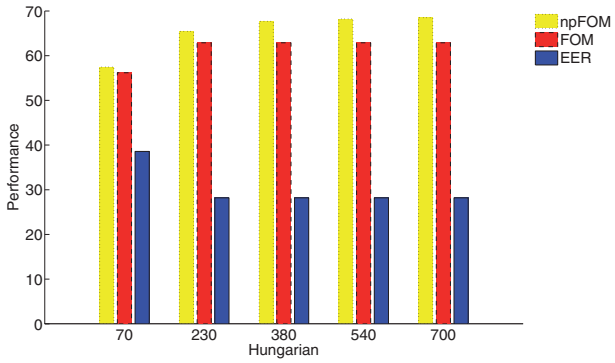

Fig. 29. Results of the text-based WFST STD system for the Hungarian data: searching for dictionary pronunciations in lattices.

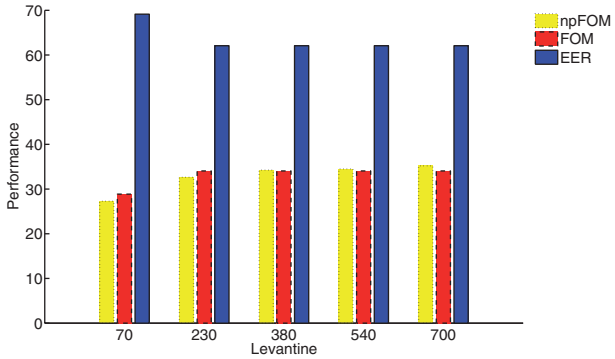

Fig. 30. Results of the text-based WFST STD system for the Levantine data: searching for dictionary pronunciations in lattices.

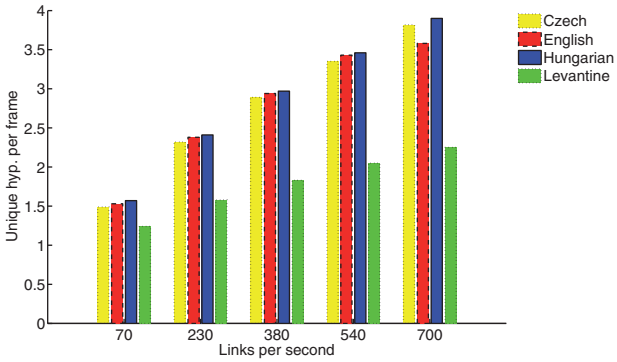

Fig. 31. Number of unique hypotheses per frame of speech kept in lattices with different densities. Many parallel links differ only in time alignment, which makes the number of unique hypotheses per frame low. Only language-dependent setups are shown. For language-independent setups, the numbers of unique hypotheses per frame are only slightly higher.

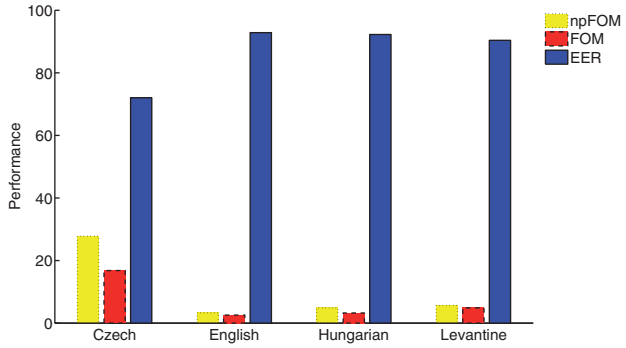

Fig. 32. Results of the WFST-based QbE STD system for the Czech data: searching for five examples per query in lattices with 700 links per second. The x-axis shows languages used to train the feature extractor.

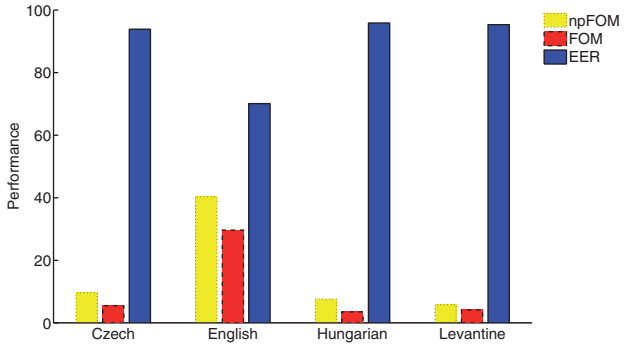

Fig. 33. Results of the WFST-based QbE STD system for the English data: searching for five examples per query in lattices with 700 links per second.

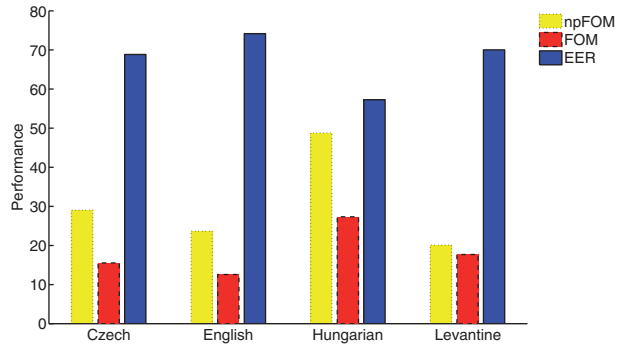

Fig. 34. Results of the WFST-based QbE STD system for the Hungarian data: searching for five examples per query in lattices with 700 links per second.

seen that the language-dependent results are worse when the examples are extracted from the lattices than from dictionary pronunciations. This is due to the inherent advantage of the pronunciation dictionary, since it is actually composed of the set of phones which had been trained previously to get the posterior features. Paired t-tests show that the improvement in the dictionary pronunciation-based method is statistically significant ($p < 0.01$) compared with the examples from lattices method for Czech, Hungarian, and Levantine data. The English data exhibits the closest performance

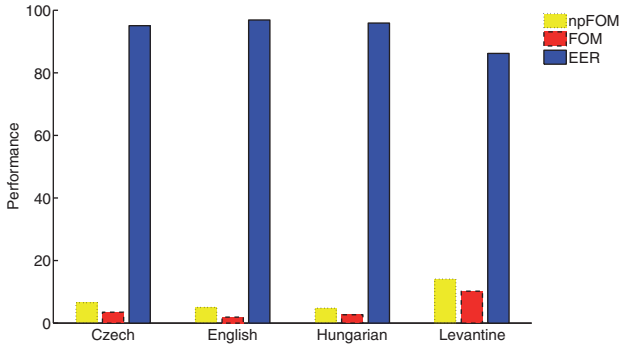

Fig. 35. Results of the WFST-based QbE STD system for the Levantine data: searching for five examples per query in lattices with 700 links per second.

between searching with dictionary pronunciations and searching with examples from lattices when the feature extractor matches the target language and the performance gap is less statistically significant ($p < 0.03$). We consider that this is due to the large amount of data used to train the English feature extractor, which leads to more robust lattice generation, and hence to improved performance.

When the language-independent and the language-dependent setups are compared, we can see a considerable degradation in the former (except for the clean speech in Hungarian data, for which the language-independent setups achieved a performance closer to the language-dependent setup than for the other languages). Paired t-tests show that the improvement in the language-dependent feature extractor over the language-independent feature extractors is statistically significant ($p < 10^{-8}$) for English and Czech data. For Levantine data, the improvement is less significant ($p < 10^{-3}$) compared with the English and Hungarian feature extractors, and ($p < 0.02$) compared with the Czech feature extractor due to the nondiacritized approach described earlier, which results in more errors, even when the feature extractor matches the target language. For Hungarian data, the Hungarian feature extractor significantly outperforms the English and Czech feature extractors ($p < 0.02$), and is significantly weaker than the Levantine feature extractor ($p < 0.05$). This is due to the read speech acoustic condition of the Hungarian data.

For a deeper insight in order to examine the bad performance exhibited by the language-independent setups, we analyzed the hit/FA ratio computed under the npFOM metric. It shows that, although obtaining a fair amount of hits (at about 84% of hit rate), we get about 274× more FAs than the reference occurrences. In addition, the scores of these hits are so low that many FAs are retained with better scores, which dramatically degrades the npFOM metric.

10.5. Comparison of QbE Detectors

To gain a deeper insight into the best QbE detector, we present the main results for all detectors and languages in Table IV as nonpooled FOM (npFOM), to exclude the calibration power of the other approaches from the comparison.

When QbE STD systems are compared, in language-dependent experiments, the DTW achieves the best overall performance. On the other hand, DTW loses about 6% absolute and statistically significant precision ($p < 0.01$) to the upper-bound AKWS. We can also see that the speech data condition is important. DTW achieves higher accuracy on Hungarian data, which is composed of prompted read speech. However, a possible drawback of the DTW approach is its difficulty in finding an efficient method

Table IV. Comparison of Proposed Approaches for QbE STD
The metric is npFOM; language-dependent experiments are shown in bold; best result for each language and detector is underlined.

Query detector	Feature extractor	Evaluation data: Czech npFOM	English npFOM	Hungarian npFOM	Levantine npFOM
AKWS	–	**52.96**	**65.11**	**68.97**	**38.99**
WFSTdict	–	**52.71**	**48.60**	**68.51**	**35.20**
DTW	Czech	**45.74**	30.30	50.41	20.03
	English	0.00	**58.89**	36.75	10.98
	Hungarian	9.56	17.30	**62.43**	10.73
	Levantine	12.53	18.67	38.99	**31.51**
GMM/HMM Bottle-neck features	Czech	**32.57**	47.19	55.02	30.40
	English	21.29	**59.65**	49.36	24.52
	Hungarian	11.78	27.60	**40.06**	14.85
	Levantine	19.07	34.38	40.30	**28.94**
WFST	Czech	**27.77**	9.66	29.01	6.52
	English	3.29	**40.29**	23.66	5.00
	Hungarian	4.89	7.65	**48.82**	4.78
	Levantine	5.63	5.86	20.11	**14.08**

for combining a high number of examples (e.g., how to combine 50 examples into one query representation).

The GMM/HMM approach is the most accurate in the language-independent domain. The best results are obtained by Czech and English feature extractors in almost all the cases that are trained on much larger amounts of data than those for the Levantine and Hungarian. The Levantine data with the English feature extractor is the only exception, due to the language-dependent condition of the Levantine feature extractor. It must be noted that the best performance of the Czech feature extractor over the language-dependent one is not statistically significant ($p \approx 0.2$) in the Levantine data. It means that in case of a more complex QbE STD system (i.e., the nondiacritized data), a language-independent feature extractor may achieve comparable results to those obtained with a language-dependent one. In addition, inspecting the Hungarian data, we can claim that a small amount of data used to train the feature extractor can result in an unreliable feature estimation, and hence a language-independent feature extractor may outperform a language-dependent one.

The DTW-based QbE STD system achieves about 50% to 75% of precision (statistically significant ($p < 10^{-3}$)) of the GMM/HMM system in a language-independent setup, except for the Hungarian data, where the small amount of data used to train the feature extractor results in a worse GMM/HMM-based query term modeling. This confirms our conjecture that a model-based approach is able to deal with the phone posterior uncertainty in a language-independent setup where enough training data is available.

The WFST approach shows a significant ($p < 0.03$) degradation when it goes from text-based STD (WFSTdict system) to QbE STD mode. This WFST-based approach for QbE STD also dramatically degrades the performance exhibited by the two other QbE STD approaches. This shows a lot of potential for improvement. A more robust way of combining example lattices and an investigation of the language-independent task need more research; but this approach is promising, as it is the only one that can lead to indexing and hence a fast QbE STD.

10.6. One Example Selection versus Five Example Selection

According to previous work [Hazen et al. 2009; Tejedor et al. 2010], we selected five examples to conduct the query search. However, there could be scenarios for which just one example per query term is available. When our DTW, GMM/HMM and WFST-based

approaches for QbE STD are compared, we observe that the number of examples can play a critical role in the final performance.

In Tejedor et al. [2010] we showed that the use of one example for our DTW-based approach does not dramatically reduce the QbE STD performance for the language-dependent scenario where this example is chosen under a selection criterion (DTW-based score for example matching). We showed that a random selection of the query example can dramatically reduce performance, even though it sounds to the human ear like an acceptable example. Therefore, we can conclude that five examples are needed to achieve a good QbE STD performance under the DTW-based approach.

With our GMM/HMM-based approach, preliminary experiments showed that the use of a single example to build the query model achieves such bad results for both language-dependent and language-independent scenarios that any comparison is meaningless. This is due to the small amount of data used to train the query model, which leads to a poor estimation of Gaussian parameters.

The WFST-based system's performance is also significantly improved when more examples of a query are used. With five examples, the npFOM increased by 40% and FOM by 80% relative to the average for all languages over using only a single randomly selected example. This is because a phone lattice of one example may contain errors, but combining more examples leads to a better and more general model of the query.

11. CONCLUSIONS AND FUTURE WORK

This article presented several methods that are capable of QbE STD from two different sets of features: 3-state phone posteriors and a bottleneck. Our DTW-based search on posteriorgrams achieves good performance on the language-dependent setup and our GMM/HMM-based QbE detector performs best for the language-independent one. The DTW-based QbE detector degrades dramatically when dealing with the language-independent setup, as also shown in related work on QbE STD [Tejedor et al. 2010; Muscariello et al. 2011]. On the other hand, the WFST-based QbE detector performs the worst. However, due to its computational attractiveness (ability for indexing and fast search), the WFST approach needs more work (e.g., example combination and term-based score normalization) to match the performance of the acoustic counterparts. We have also shown that bottleneck features outperform 3-state phone posterior features (as discussed for LVCSR experiments in Grézl et al. [2007]) for our GMM/HMM-based QbE detector where enough training data is available. This means that bottleneck features are more sensitive than 3-state phone posterior features to the amount of training data. Future work will examine the performance of some other features such as GMM-based posteriorgrams built from standard MFCC, PLP, and phone posterior features, and articulatory features that can be derived automatically from the speech signal with no language knowledge.

Based on the results, we can conclude that replacing a classical keyword spotter with text entry of the query by a language-independent QbE STD system is still a difficult task. This is clear when comparing the best results obtained by the GMM/HMM approach on a language-independent setup and those achieved with AKWS. This is due to several causes: (1) AKWS is inherently language-dependent, since both the feature extractor and each query pronunciation match the target language; (2) AKWS employs the correct phone transcription to model each query; and (3) AKWS employs language-dependent filler models to deal with the nonkeyword part of speech of the utterances. We can observe that the GMM/HMM results are quite promising, since for English data, the language-independent setup performs very close (2% absolute degradation, statistically insignificant ($p \approx 0.3$)) to the text-based WFST, and for Levantine data, the results are still quite reasonable (about 5% absolute degradation, statistically insignificant ($p \approx 0.3$)).

In summary, we can claim that the performance of a language-independent QbE system is not very far from that of a text-based STD system, especially if the phone transcriptions are unknown and have to be derived by a trained grapheme-to-phoneme (G2P) converter. Therefore, we consider that building language-independent QbE STD systems to be a viable alternative to classical STD systems—this is confirmed by the GMM/HMM-based approach presented in this work.

From the various approaches we have presented in this work, we can suggest several interesting future research lines, especially those addressing language-independence. First, for the DTW-based QbE STD approach, we can study the combination of a larger number of examples than five, although we have shown in this article that this is sufficiently accurate for a language-dependent setup and also for "easy" data as read speech.

We see a large potential in our GMM/HMM-based QbE, and propose to focus on the following.

—What happens if the number of examples is increased? Our experiments have shown that bottleneck features generally outperform phone posteriors for five examples. In this direction, an analysis of the set of features that are used for term modeling with more than five examples should be done, since taking more examples for term modeling may compensate for the differences in feature performance.
—We propose a concatenation (combination) of the features from different languages. In this way, decorrelation and dimensionality reduction techniques may enhance the final performance, especially for language-independent QbE STD.
—We propose a deeper study of the number of Gaussian components in term modeling for more than five examples under different languages and acoustic speech conditions.
—In this article, the number of states for query modeling has been estimated from its number of phones in the target language. To completely remove any information corresponding to the target language, a way to estimate it is needed. In this direction, we propose to estimate this quantity from the average length of the query examples.
—The background model could also be improved. In this way, a loop of automatically derived units will be examined.

We will also study the fusion of several language-independent QbE STD systems. Here, it must be noted that detection of several language-independent systems cannot be merged directly, since their scores are not directly comparable. Our future work will also include a study of rescoring methods to enhance the performance of the approaches presented in this article.

Noise robustness is also a critical issue in realistic speech recognition systems, and henceforth in QbE STD. We propose to make a deeper study of this aspect in future work, in which noise robust features will play a very important role.

ELECTRONIC APPENDIX

The electronic appendix to this article is available in the ACM Digital Library.

REFERENCES

Akbacak, M., Vergyri, D., and Stolcke, A. 2008. Open-vocabulary spoken term detection using graphone-based hybrid recognition systems. In *Proceedings of the ICASSP'08*. 5240–5243.

Allauzen, C., Mohri, M., and Saraclar, M. 2004. General indexation of weighted automata: application to spoken utterance retrieval. In *Proceedings of the Workshop on Interdisciplinary Approaches to Speech Indexing and Retrieval (HLT-NAACL'04)*. 33–40.

ALLAUZEN, C., RILEY, M., SCHALKWYK, J., SKUT, W., AND MOHRI, M. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Proceedings of the International Conference on Implementation and Application of Automata*. Vol. 4783, 11–23.

ANGUERA, X. 2011. Telefonica system for the spoken web search task at MediaEval 2011. In *Proceedings of MediaEval'11*. 3–4.

ANGUERA, X., MACRAE, R., AND OLIVER, N. 2010. Partial sequence matching using an unbounded dynamic time warping algorithm. In *Proceedings of ICASSP'10*. 3582–3585.

BARNARD, E., DAVEL, M., VAN HEERDEN, C., KLEYNHANS, N., AND BALI, K. 2011. Phone recognition for spoken web search. In *Proceedings of MediaEval'11*. 5–6.

BAUM, L. E., PETRIE, T., SOULES, G., AND WEISS, N. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *Ann. Math. Stat. 41,* 1, 164–171.

BYRNE, W., BEYERLEIN, P., HUERTA, J. M., KHUDANPUR, S., MARTHI, B., MORGAN, J., PETEREK, N., PICONE, J., AND WANG, W. 2000. Towards language independent acoustic modeling. In *Proceedings of ICASSP'00*. 1029–1032.

CAI, L., JUEDES, D., AND LIAKHOVITCH, E. 2000. Evolutionary computation techniques for multiple sequence alignment. In *Proceedings of the Congress on Evolutionary Computation*. 829–835.

CAN, D., COOPER, E., SETHY, A., WHITE, C., RAMABHADRAN, B., AND SARACLAR, M. 2009. Effect of pronunciations on OOV queries in spoken term detection. In *Proceedings of the ICASSP'09*. 3957–3960.

CHAN, C. AND LEE, L. 2010. Unsupervised spoken-term detection with spoken queries using segment-based dynamic time warping. In *Proceedings of the Interspeech'10*. 693–696.

CHIA, T. K., SIM, K. C., LI, H., AND NG, H. T. 2008. A lattice-based approach to query-by-example spoken document retrieval. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 363–370.

CUAYAHUITL, H. AND SERRIDGE, B. 2002. Out-of-vocabulary word modeling and rejection for spanish keyword spotting systems. In *Proceedings of Mexican International Conference on Artificial Intelligence*. 156–165.

EDGAR, R. C. AND BATZOGLOU, S. 2006. Multiple sequence alignment. *Current Opinion in Structural Biology 16*, 368–373.

FISCUS, J. G., AJOT, J., GAROFOLO, J. S., AND DODDINGTON, G. 2007. Results of the 2006 spoken term detection evaluation. In *Proceedings of the Workshop on Searching Spontaneous Conversational Speech (SIGIR-SSCS'07)*.

GREZL, F. AND FOUSEK, P. 2008. Optimizing bottle-neck features for LVCSR. In *Proceedings of the ICASSP'08*. 4729–4732.

GREZL, F., KARAFIAT, M., AND BURGET, L. 2009. Investigation into bottle-neck features for meeting speech recognition. In *Proceedings of Interspeech'09*, 2947–2950.

GREZL, F., KARAFIAT, M., KONTAR, S., AND ČERNOCKY, J. 2007. Probabilistic and bottle-neck features for LVCSR of meetings. In *Proceedings of the ICASSP'07*, 757–760.

HAZEN, T. AND BAZZI, I. 2001. A comparison and combination of methods for OOV word detection and word confidence scoring. In *Proceedings of the ICASSP'01*. 397–400.

HAZEN, T. J., SHEN, W., AND WHITE, C. M. 2009. Query-by-example spoken term detection using phonetic posteriorgram templates. In *Proceedings of ASRU'09*. 421–426.

HELEN, M. AND VIRTANEN, T. 2007. Query by example of audio signals using Euclidean distance between Gaussian mixture models. In *Proceedings of the ICASSP'07*. 225–228.

HELEN, M. AND VIRTANEN, T. 2010. Audio query by example using similarity measures between probability density functions of features. *EURASIP, J. Audio, Speech Music Process. 2010*, 2, 1–2, 12.

JANSEN, A., CHURCH, K., AND HERMANSKY, H. 2010. Towards spoken term discovery at scale with zero resources. In *Proceedings of Interspeech'10*, 1676–1679.

KEMPTON, T., MOORE, R. K., AND HAIN, T. 2011. Cross-language phone recognition when the target language is not known. In *Proceedings of Interspeech'11*. 3165–3168.

KIM, J., JUNG, H., AND CHUNG, H. 2004. A keyword spotting approach based on pseudo N-gram language model. In *Proceedings of the Conference on Speech and Computer*. 156–159.

LIN, H., STUPAKOV, A., AND BILMES, J. 2008. Spoken keyword spotting via multi-lattice alignment. In *Proceedings of Interspeech'08*, 2191–2194.

LIN, H., STUPAKOV, A., AND BILMES, J. 2009. Improving multi-lattice alignment based spoken keyword spotting. In *Proceedings of ICASSP'09*, 4877–4880.

MAMOU, J. AND RAMABHADRAN, B. 2008. Phonetic query expansion for spoken document retrieval. In *Proceedings of Interspeech'08*, 2106–2109.

MAMOU, J., RAMABHADRAN, B., AND SIOHAN, O. 2007. Vocabulary independent spoken term detection. In *Proceedings of ACM-SIGIR'07*, 615–622.

MANOS, A. AND ZUE, V. 1997. A segment-based wordspotter using phonetic filler models. In *Proceedings of ICASSP'97*, vol. 2, 899–902.

MANTENA, G. V., BOLLEPALLI, B., AND PRAHALLAD, K. 2011. SWS task: Articulary phonetic units and sliding DTW. In *Proceedings of MediaEval'11*. 7–8.

METZE, F., RAJPUT, N., ANGUERA, X., DAVEL, M., GRAVIER, G., VAN HEERDEN, C., MANTENA, G. V., MUSCARIELLO, A., PRAHALLAD, K., SZOKE, I., AND TEJEDOR, J. 2012. The spoken web search task at MediaEval 2011. In *Proceedings of ICASSP'12*. 5165–5168.

MUSCARIELLO, A. AND GRAVIER, G. 2011. Irisa MediaEval 2011 spoken Web search system. In *Proceedings of MediaEval'11*. 9–10.

MUSCARIELLO, A., GRAVIER, G., AND BIMBOT, F. 2009. Audio keyword extraction by unsupervised word discovery. In *Proceedings of Interspeech'09*. 2843–2846.

MUSCARIELLO, A., GRAVIER, G., AND BIMBOT, F. 2011. Zero-resource audio-only spoken term detection based on a combination of template matching techniques. In *Proceedings of Interspeech'11*. 921–924.

NIST. 1991. The road rally word-spotting corpora (RDRALLY1). NIST Speech Disc 6-1.1.

NIST. 2006. The spoken term detection (STD) 2006 evaluation plan. http://www.nist.gov/speech/tests/std.

OU, J., CHEN, C., AND LI, Z. 2001. Hybrid neural-network/HMM approach for out-of-vocabulary words rejection in Mandarin place name recognition. In *Proceedings of the International Conference On Neural Information Processing*.

PARADA, C., SETHY, A., AND RAMABHADRAN, B. 2009. Query-by-example spoken term detection for OOV terms. In *Proceedings of ASRU'09*. 404–409.

PARK, A. S. AND GLASS, J. R. 2008. Unsupervised pattern discovery in speech. *IEEE Trans. Audio, Speech Lang. Process. 16,* 1, 186–197.

SCHULTZ, T. AND WAIBEL, A. 2001. Experiments on cross-language acoustic modeling. In *Proceedings of Interspeech'01*. 2721–2724.

SCHWARZ, P. 2009. Phoneme recognition based on long temporal context. Ph.D. dissertation, Brno University of Technology, Brno, Czech Republic.

SHEN, W., WHITE, C. M., AND HAZEN, T. J. 2009. A comparison of query-by-example methods for spoken term detection. In *Proceedings of Interspeech'09*. 2143–2146.

SINISCHALCHI, S. M., SVENDSEN, T., AND LEE, C. H. 2008. Toward a detector-based universal phone recognizer. In *Proceedings of ICASSP'08*. 4261–4264.

SZOKE, I. 2010. Hybrid word-subword spoken term detection. Ph.D. dissertation, Brno University of Technology, Brno, Czech Republic.

SZOKE, I., BURGET, L., ČERNOCKY, J., AND FAPSO, M. 2008a. Sub-word modeling of out of vocabulary words in spoken term detection. In *Proceedings of SLT'08*. 273–276.

SZOKE, I., FAPSO, M., BURGET, L., AND ČERNOCKY, J. 2008b. Hybrid word-subword decoding for spoken term detection. In *Proceedings of the Speech Search Workshop at SIGIR (SSCS'08)*. 42–49.

SZOKE, I., FAPSO, M., KARAFIAT, M., BURGET, L., GREZL, F., SCHWARZ, P., GLEMBEK, O., MATFJKA, P., KOPECKY, J., AND ČERNOCKY, J. 2008c. Spoken term detection system based on combination of LVCSR and phonetic search. In *Machine Learning for Multimodal Interaction*, Lecture Notes in Computer Science, vol. 4892, Springer, Berlin, 237–247.

SZOKE, I., GREZL, F., ČERNOCKY, J., AND FAPSO, M. 2010. Acoustic keyword spotter—Optimization from end-user perspective. In *Proceedings of SLT'10*. 177–181.

SZOKE, I., SCHWARZ, P., MATEJKA, P., BURGET, L., KARAFIAT, M., FAPSO, M., AND CERNOCKY, J. 2005. Comparison of keyword spotting approaches for informal continuous speech. In *Proceedings of Interspeech'05*. 633–636.

SZOKE, I., TEJEDOR, J., FAPSO, M., AND COLAS, J. 2011. BUT-HCT Lab approaches for spoken web search: MediaEval 2011. In *Proceedings of MediaEval'11*. 11–12.

TEJEDOR, J. 2009. Contributions to keyword spotting and spoken term detection for information retrieval in audio mining. Ph.D. dissertation, Universidad Autonoma de Madrid.

TEJEDOR, J., SZOKE, I., AND FAPSO, M. 2010. Novel methods for query selection and query combination in query-by-example spoken term detection. In *Proceedings of Searching Spontaneous Conversational Speech (SSCS'10)*. 15–20.

TSAI, W.-H. AND WANG, H.-M. 2004. A query-by-example framework to retrievemusic documents by singer. In *Proceedings of the IEEE International Conference on Multimedia and Expo*. 1863–1866.

TZANETAKIS, G., ERMOLINSKYI, A., AND COOK, P. 2002. Pitch histograms in audio and symbolic music information retrieval. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR)*. 31–38.

VERGYRI, D., SHAFRAN, I., STOLCKE, A., GADDE, R. R., AKBACAK, M., ROARK, B., AND WANG, W. 2007. The SRI/OGI 2006 spoken term detection system. In *Proceedings of Interspeech'07*. 2393–2396.

VERGYRI, D., STOLCKE, A., GADDE, R. R., AND WANG, W. 2006. The SRI 2006 spoken term detection system. In *Proceedings of NIST Spoken Term Detection Workshop (STD'06)*.

WALKER, B. D., LACKEY, B. C., MULLER, J. S., AND SCHONE, P. J. 2003. Language-reconfigurable universal phone recognition. In *Proceedings of Interspeech'03*. 153–156.

WELLING, L., KANTHAK, S., AND NEY, H. 1999. Improved methods for vocal tract normalization. In *Proceedings of ICASSP'99*. 761–764.

WU, S., LEE, M., LEE, Y. S., AND GATTON, T. M. 2008. Multiple sequence alignment using GA and NN. *Int. J. Signal Process. Image Process. Pattern Recogn. 1*, 21–31.

XIN, L. AND WANG, B. 2001. Utterance verification for spontaneous Mandarin speech keyword spotting. In *Proceedings of the International Conference on Infotec and Infonet*. Vol. 3, 397–401.

YOUNG, S. J., KERSHAW, D., ODELL, J., OLLASON, D., VALTCHEV, V., AND WOODLAND, P. 2006. *The HTK Book* Version 3.4. Cambridge University Press.

ZHANG, T. AND KUO, C.-C. J. 1999. Hierarchical classification of audio data for archiving and retrieving. In *Proceedings of ICASSP'99*. 3001–3004.

ZHANG, Y. AND GLASS, J. R. 2009. Unsupervised spoken keyword spotting via segmental DTW on Gaussian posteriorgrams. In *Proceedings of ASRU'09*. 398–403.

Received March 2011; revised September 2011; accepted May 2012