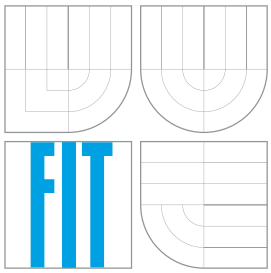
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

VYHLEDÁVÁNÍ VÝRAZŮ V ŘEČI POMOCÍ MLUVENÝCH PŘÍKLADŮ

QUERY-BY-EXAMPLE SPOKEN TERM DETECTION

DISERTAČNÍ PRÁCE
PHD THESIS

AUTOR PRÁCE
AUTHOR

Ing. MICHAL FAPŠO

VEDOUCÍ PRÁCE
SUPERVISOR

Doc. Dr. Ing. JAN ČERNOCKÝ

BRNO 2014

Abstract

This thesis investigates query-by-example (QbE) spoken term detection (STD). Queries are entered in their spoken form and searched for in a pool of recorded spoken utterances, providing a list of detections with their scores and timing. We describe, analyze and compare three different approaches to QbE STD, in various language-dependent and language-independent setups with diverse audio conditions, searching for a single example and five examples per query.

For our experiments we used Czech, Hungarian, English and Levantine data and for each of the languages we trained a 3-state phone posterior estimator. This gave us 16 possible combinations of the evaluation language and the language of the posterior estimator, out of which 4 combinations were language-dependent and 12 were language-independent. All QbE systems were evaluated on the same data and the same features, using the metrics: non-pooled Figure-of-Merit and our proposed utterrance-normalized non-pooled Figure-of-Merit, which provided us with relevant data for the comparison of these QbE approaches and for gaining a better insight into their behavior.

QbE approaches presented in this work are: sequential statistical modeling (GMM/HMM), template matching of features (DTW) and matching of phone lattices (WFST). To compare the performance of QbE approaches with the common query-by-text STD systems, for language-dependent setups we also evaluated an acoustic keyword spotting system (AKWS) and a system searching for phone strings in lattices (WFSTlat). The core of this thesis is the development, analysis and improvement of the WFST QbE STD system, which after the improvements, achieved similar performance to the DTW system in language-dependent setups.

Keywords

Query-by-Example, Spoken Term Detection, Finite State Transducers, System comparison, Language dependency, Low-resource languages

Bibliographic citation

Michal Fapšo: Query-by-Example Spoken Term Detection, Ph.D. Thesis, Brno, Brno University of Technology, Faculty of Information Technology, 2014

Abstrakt

Tato práce se zabývá vyhledáváním výrazů v řeči pomocí mluvených příkladů (QbE STD). Výrazy jsou zadávány v mluvené podobě a jsou vyhledány v množině řečových nahrávek, výstupem vyhledávání je seznam detekcí s jejich skóre a časováním. V práci popisujeme, analyzujeme a srovnáváme tři různé přístupy ke QbE STD v jazykově závislých a jazykově nezávislých podmínkách, s jedním a pěti příklady na dotaz.

Pro naše experimenty jsme použili česká, maďarská, anglická a arabská (levantská) data, a pro každý z těchto jazyků jsme natrénovali 3-stavový fonémový rozpoznávač. To nám dalo 16 možných kombinací jazyka pro vyhodnocení a jazyka na kterém byl natrénovaný rozpoznávač. Čtyři kombinace byly tedy závislé na jazyce (language-dependent) a 12 bylo jazykově nezávislých (language-independent). Všechny QbE systémy byly vyhodnoceny na stejných datech a stejných fonémových posteriorních příznacích, pomocí metrik: nesdružené Figure-of-Merit (non pooled FOM) a námi navrhnuté nesdružené Figure-of-Merit se simulací normalizace přes promluvy (utterrance-normalized non-pooled Figure-of-Merit). Ty nám poskytly relevantní údaje pro porovnání těchto QbE přístupů a pro získání lepšího vhledu do jejich chování.

QbE přístupy použité v této práci jsou: sekvenční statistické modelování (GMM/HMM), srovnávání vzorů v příznacích (DTW) a srovnávání grafů hypotéz (WFST). Abychom porovnali výsledky QbE přístupů s běžnými STD systémy vyhledávajícími textové výrazy, vyhodnotili jsme jazykově závislé konfigurace také s akustickým detektorem klíčových slov (AKWS) a systémem pro vyhledávání fonémových řetězců v grafech hypotéz (WFSTlat). Jádrem této práce je vývoj, analýza a zlepšení systému WFST QbE STD, který po zlepšení dosahuje podobných výsledků jako DTW systém v jazykově závislých podmínkách.

Klíčová slova

vyhledávání podle vzorů, detekce mluvených výrazů, konečné stavové automaty, srovnání systémů, závislost na jazyku, jazyky s malým množstvím dostupných dat

Bibliografická citace

Michal Fapšo: Query-by-Example Spoken Term Detection, disertační práce, Brno, FIT VUT v Brně, 2014

Declaration of Originality

I hereby declare that this thesis and the work reported herein was composed by and originated entirely from me. The work has been supervised by Doc. Dr. Ing. Černocký. Information derived from the published and unpublished work as of others has been acknowledged in the text and references are given in the list of sources. Some of the reported systems and texts were written by Igor Szőke, František Grézl and Javier Tejedor and are included in this thesis for comparison purposes. These parts are always marked explicitly.

Prohlášení

Prohlašuji, že jsem tuto disertační práci vypracoval samostatně pod vedením Doc. Dr. Ing. Jana Černockého. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal. Některé prezentované systémy a texty byly napsány Igorem Szőkem, Františkem Grézlem a Javierem Tejedorem a jsou v této práci zahrnuty z důvodu porovnání různých systémů. Tyto části jsou vždy explicitně označeny.

Acknowledgments

I would like to thank Honza Černocký for his guidance, support, friendship and endless patience not only during my work on this thesis, but since we first met. At the same time I have to thank all members and my friends of the Speech@FIT and Graph@FIT research groups for the positive and inspirational atmosphere they all create. I would especially like to thank Igor Szőke, Franta Grézl, Lukáš Burget, Petr Schwarz, Ondra Glembek, Martin Karafiát, Pavel Matějka, Mirko Hanneman, Karel Veselý, Olda Plchot, Pepa Žižka, Tom Cipr, Kamil Chalupníček and Miro Skácel for all the inspiring conversations, ideas and simply for all the time we spent together.

I would also like to thank Javier Tejedor for visiting our research group, sharing ideas and working together, Tom Kašpárek for keeping our computing clusters in an excellent shape, and our secretaries Sylva Otáhalová and Jana Slámová, for their service and patience.

I have to thank also authors of papers referenced in this thesis, for sharing their knowledge, ideas and inspiration.

Very special thanks to my wife Hanka and my son Maťko, my parents and my parents-in-law for their patience, support, service and for cheering me up especially while I was finishing this thesis.

Contents

Glossary

AKWS	Acoustic Keyword Spotting
CTS	Conversational Telephone Speech
DTW	Dynamic Time Warping
FA	False Alarm
FOM	Figure-Of-Merit
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
HTK	Hidden Markov Model Toolkit
LVCSR	Large Vocabulary Continuous Speech Recognizer
MFCC	Mel-Frequency Cepstral Coefficients
NIST	National Institute of Standards and Technology (United States of America)
npFOM	non-pooled Figure-Of-Merit
OOV	Out-Of-Vocabulary (word/term)
oracleFOM	oracle Figure-Of-Merit
SDR	Spoken Document Retrieval
STD	Spoken Term Detection
STK	Toolkit for speech processing developed at FIT BUT
unnpFOM	utterance-normalized non-pooled Figure-Of-Merit
WFST	Weighted Finite State Transducer

Chapter 1

Introduction

With the growing amount of spoken data, which is recorded, stored and also shared nowadays, the need for its effective indexing and retrieval increases as well.

The most common approach to speech search systems is the "spoken term detection" (STD), which aims at searching a phrase of one or more words in spoken data, outputting a list of detections with score and timing information. Users can enter the search query either in form of text (henceforth, query-by-text or QbT STD[1]) or in form of speech (query-by-example or QbE STD).

Best performing QbT STD systems search in an output of large vocabulary continuous speech recognizers (LVCSR), which are available only for several most widespread languages. Building such recognizer for a new language requires a lot of training data and linguistic resources, so it is not a viable option for most of the world's languages and dialects.

On the other hand, QbE STD systems where queries are entered in their spoken form, can be language-independent and thus they are usually the only option for searching in new or low-resource languages. These systems search in phone posteriorgrams or other, usually unsupervisedly trained, appropriate features. Other applications of QbE STD systems are: searching in multi-language or multi-dialect spoken data, voice-based information systems, searching for out-of-vocabulary words of LVCSR-based QbT STD systems, or for relevance feedback in QbT STD systems.

In this work, we concentrate on the query-by-example spoken term detection. We describe and analyze three different QbE STD systems and compare their performance across several language-dependent and language-independent setups with various audio conditions, using a single example or five examples per query. A major part of this work describes our development, implementation and analysis of a QbE STD system which searches in phone lattices or confusion networks derived from phone posteriorgrams.

[1]STD is commonly interpreted as query-by-text STD, but in this work we use the abbreviation QbT STD to differentiate it from generic STD and QbE STD

1.1 Motivation

The main motivation for this work was to gain a better insight into the behavior of most common types of query-by-example STD systems across various language-dependent and language-independent setups with diverse audio conditions. Since there was no such thorough comparison available, we started to work on it. Part of this work has been already published in [Tejedor et al., 2012].

Since we started our effort, MediaEval evaluations targeting language-independent setups of QbE STD were held several times. Participants were also provided with data to evaluate language-dependent setups, but only few participants compared their own systems for both language-dependent and language-independent setups. A global comparison of participating QbE systems was shown in [Metze et al., 2014], but only for language-independent setups. Also, the MediaEval metric is ATWV and MTWV, where results are influenced by calibration of scores across queries, not only by the differences between QbE STD systems themselves. Nevertheless, there was only a single participating QbE STD system searching in lattices [Barnard et al., 2011], where authors tried to use phone lattices to refine search results, but they were unsuccessful in implementing it correctly and did not report any results with the lattice-based system.

Three main types of QbE STD systems are described in literature, according to the matching technique they deploy: template matching of features (DTW), sequential statistical modeling of features (GMM/HMM) and lattice matching (WFST). Out of these, DTW is the most widespread approach. Sequential statistical modeling and lattice matching approaches are used much less.

Actually there have been only few published QbE STD systems searching in lattices or confusion networks. In [Shen et al., 2009], authors compared several techniques for matching confusion networks of query and utterance, in [Lin et al., 2008, 2009], authors used graphical models for multi-lattice alignment, and in [Parada et al., 2009], authors used transducers to search for out-of-vocabulary words in phone lattices generated by an LVCSR system. In all cases, there was no comparison with other QbE approaches or other languages.

Therefore, we decided to implement the lattice-based QbE STD system and to thoroughly compare all three main types of QbE STD systems. We use a two-blocks architecture in all our STD systems: the first block is the feature extractor which encodes the speech into low dimensional feature vectors, and the second block is the query detector (or spoken-term detector) which hypothesizes putative query detections from the features. To make the comparison of all STD systems more relevant, the feature extractor block remains the same in all our STD systems, so the only difference in their performance can be caused by query detector blocks. In this work, we evaluate the QbE STD systems in four language-dependent setups (Czech, English, Hungarian and Levantine), and twelve language-independent setups (all other combinations of the four target languages with four language-specific phone posterior feature extractors), using a single example or five examples per query.

1.2 Scope of Chapters

Chapter 2 describes STD and QbE STD systems in general with references to related works, and also describes commonly used evaluation metrics.

Chapter 3 introduces our experimental setup including used data, their preparation and the

process of extracting features for our QbE STD systems.

Chapter 4 defines our baseline AKWS system, its architecture and shows its results.

Chapters 5 and 6 present our GMM/HMM and DTW QbE STD systems and their results.

Chapter 7 deals in detail with our WFSTdict baseline system, WFST QbE STD systems, analyzes their various aspects, their performance, and outlines several improvements.

Chapter 8 summarizes and discusses results of all presented systems.

Chapter 9 concludes the work.

1.3 Claims of this Thesis

The goal of this work was to implement a lattice-based query-by-example spoken term detection system and to compare it with other state-of-the-art systems across various language-dependent and language-independent setups. Several parts of this thesis were partly contributed by Igor Szőke and František Grézl from the BUT Speech@FIT research group and by Javier Tejedor from the HTCLab at Madrid university. We also worked together on several of the presented experiments. My own claims of this thesis are the following:

- Analysis of STD evaluation metrics, especially the FOM metric and its proposed unnpFOM variant, which simulates an ideal calibration of scores across utterances.
- Implementation, analysis and improvements of two WFST-based QbE systems. Although these systems were inspired by [Parada et al., 2009], our systems are different and original in several aspects.
- Comparison and analysis of DTW, GMM/HMM and WFST QbE systems in several language-dependent and language-independent setups with a single example or five examples per query. I worked on some of the experiments with Igor Szőke from the BUT Speech@FIT research group, and Javier Tejedor from the HTCLab at Madrid university. Our joint work was published in [Tejedor et al., 2012], but many experiments and results are new and original in this thesis.

The source code for our WFST system, scoring tool and other relevant scripts are available at `http://michalfapso.github.io`.

Chapter 2

Query-by-Example Systems Overview

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. Besides spoken term detection, query-by-example approach 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; Salamon et al., 2013], and spoken document retrieval [Chia et al., 2010]. In this work, we concentrate on query-by-example systems searching for a spoken query in speech data, related to spoken term detection.

2.1 Spoken Term Detection (STD)

STD aims at searching textual queries (terms) in already recorded spoken data, where queries consist of a single word or a phrase of several words. We denote this approach as query-by-text STD to differentiate it from the query-by-example STD. In literature, it is also referred to as keyword spotting.

In query-by-text STD, it is usually assumed, that the target language is well known and we have enough resources like transcribed data, phone sets, and pronunciation dictionaries, to train an LVCSR system or at least a phone recognizer for that language. For rare words not included in the LVCSR dictionary (out-of-vocabulary words), or in case of using only a phone recognizer, user queries have to be first automatically transcribed to strings of phones or other sub-word units. The gap between the text form of queries and spoken form of data we search in, make the query-by-text STD difficult to employ for low-resource languages.

Although query-by-text STD has many applications, there are scenarios where it is unsuitable. For example, in voice-based information systems or in the security field where users may not be able to enter text queries, or they have no knowledge of textual representation of queries, or the target language is low-resource making it impossible to train language-specific acoustic models. In all these cases, the query-by-example approach to STD, becomes more appropriate.

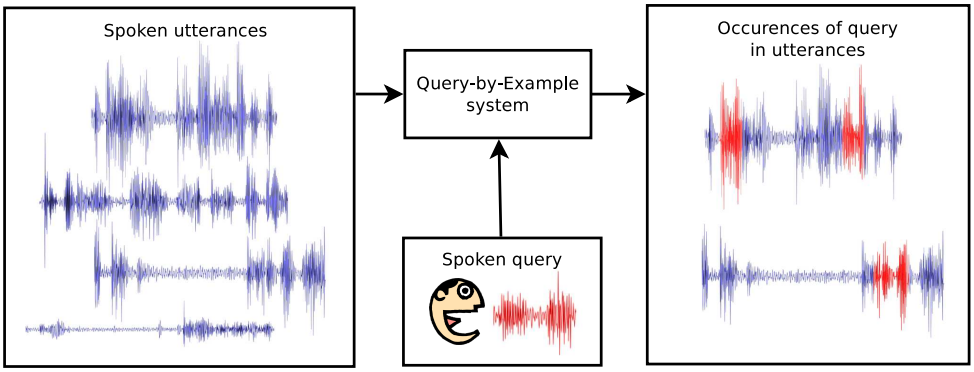


Figure 2.1: General diagram of a query-by-example spoken term detection system. User-defined spoken query is searched in a database of spoken utterances, providing the user with occurrences of the query. The query can be defined by direct input from a microphone or by a region of speech selected in another utterance.

2.2 Query-by-Example Spoken Term Detection (QbE STD)

Query-by-example STD systems search for a spoken query in a pool of spoken utterances, providing a list of detections with their scores and timing. Unlike query-by-text STD systems where queries are entered in their textual representation, queries in QbE STD systems can be entered through a microphone or selected in already recorded spoken data. A general diagram of a QbE STD system is shown in Figure 2.1.

There are several scenarios where QbE STD is more appropriate than the query-by-text STD. There may be no target-language data for training even a basic phone recognizer, the user may not know the textual representation of a query, or he/she may not be able to enter the query through a keyboard.

We can even consider a scenario where the two types of STD systems are combined together. User enters a text query into a query-by-text STD system and while browsing through search results, he/she marks some detections as hits. These selected detections are then used as queries for a QbE STD system, which then provides a new list of detections that can be merged with the previous list, adding new detections and/or re-ranking detections previously obtained from the query-by-text STD system. This can be performed also without the user's interaction, when the query-by-text STD system automatically selects first few hits and uses them as examples for the QbE STD pass. These techniques are also known as relevance-feedback [Chen et al., 2013].

To achieve speaker and channel independence, spoken queries and utterances for QbE STD systems have to be converted to more robust features, usually to posteriorgrams of phones [Hazen et al., 2009; Tejedor et al., 2012] or other unsupervisedly trained units [Zhang and Glass, 2009].

Three main approaches dominate in QbE STD systems:

Template matching of features (DTW), where a distance matrix is computed between query features and utterance features, and then the shortest paths traversing the whole query and any part of utterance, are searched through the distance matrix [Hazen et al., 2009].

Sequential statistical modeling (GMM/HMM), where the query model is trained on query

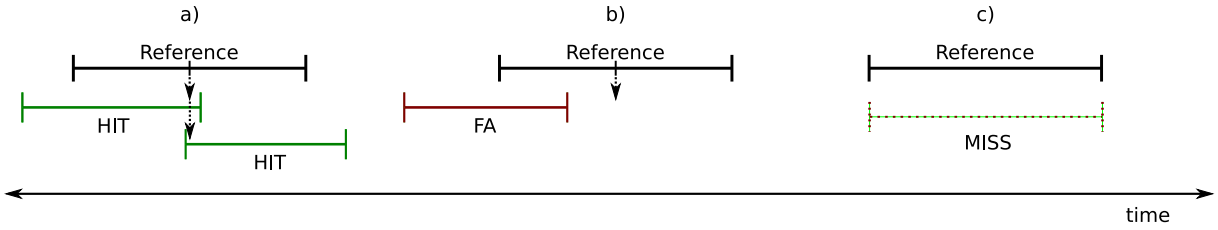


Figure 2.2: Example of a hit, false alarm and miss as implemented in HTK toolkit. This figure was taken from [Szöke, 2010].

features and a background model is trained on features of all utterances in the data pool. Then both these models are matched against utterance features and their likelihood ratio in each frame of the utterance provides the confidence score of query ending in that frame [Velivelli et al., 2003].

Lattice matching (WFST), where both query and utterance features are first converted to phone lattices or phone confusion networks, then they are converted to weighted finite state transducers. The search is performed by a composition operation of the query transducer with the utterance transducer, which results in a new transducer from which actual detections have to be extracted [Parada et al., 2009].

In this work, we analyze all three described types of QbE STD systems in two main setups: language-dependent, where we have enough resources to train a phone recognizer, and language-independent, where we don't have any information about the target language (no target phone recognizer).

2.3 Evaluation

There are several evaluation metrics commonly used in the area of Spoken Term Detection (STD). We will describe them in this section and eventually choose some of them for evaluating our systems.

To compute any of these metrics, an STD system has to provide a list of detections with their start time, end time and confidence score. Then we have to compare detections with a reference transcription and mark each detection either as hit or false alarm. Figure of merit (FOM) metric as implemented by the HTK toolkit considers a detection to be a hit when the mid-point of a reference is between start and end time of the detection (see Figure 2.2). Term weighted value (TWV) metric loosens this requirement and considers a detection to be a hit when the mid-point of the detection is within 0.5s range from the reference time span, but only one hit is allowed for each reference. All other detections which overlap with the same reference are considered to be false alarms (see Figure 2.3). For the evaluation of our systems, we allowed the 0.5s range around a reference, but when more detections overlap with the same reference, we take only the one with the highest confidence score and discard the others.

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, $N_{nontarget}(Q)$ the number of opportunities for incorrect detection of Q in the evaluation data (constrained by some predefined sampling, e.g. "a detection can occur every second"), $N_{HIT}(Q, thr)$ the number of detections

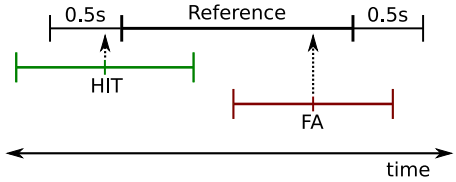


Figure 2.3: Example of a hit and false alarm as defined by NIST for STD evaluation metric TWV. In the LSE toolkit we only keep the best detection and discard all others that point to the same reference. This figure was taken from [Szöke, 2010].

of the query Q which are identified as hits and their score remains above the threshold thr, and $N_{FA}(Q, thr)$ the number of false detections (i.e., FAs) of the query Q with a score larger than thr. Probability of hit is

$$p_{HIT}(Q, thr) = \frac{N_{HIT}(Q, thr)}{N_{target}(Q)} \tag{2.1}$$

probability of miss is

$$p_{MISS}(Q, thr) = 1 - p_{HIT}(Q, thr) = 1 - \frac{N_{HIT}(Q, thr)}{N_{target}(Q)} \tag{2.2}$$

and probability of false alarm is

$$p_{FA}(Q, thr) = \frac{N_{FA}(Q, thr)}{N_{nontarget}(Q)}. \tag{2.3}$$

In all scoring tools we used, a detection was considered to be a hit in case its start– and end–times were within a 500ms margin of those of the reference. We chose such a loose time constraints to cancel incorrect timing errors possibly introduced by converting posteriors to lattices and further to confusion networks. When more detections belong to the same reference, we take only the one with the highest score, and remove the others from scoring.

2.3.1 DET

Detection error trade-off is defined by NIST [Fiscus et al., 2006] as a dependency of miss probability $p_{MISS}(thr)$ and false alarm probability $p_{FA}(thr)$ where

$$p_{MISS}(thr) = \underset{Q}{average}\{p_{MISS}(Q, thr)\}$$

$$p_{FA}(thr) = \underset{Q}{average}\{p_{FA}(Q, thr)\}.$$

p_{MISS} and p_{FA} are averaged over only those terms with a non-zero number of true occurrences in the evaluation data, so that p_{MISS} is defined. Instead of providing a single value, DET curve is a more complex graph showing system's performance for various operating points. DET curves of systems with the best performance approach the lower bottom corner of the graph. For comparing two or more systems, their DET curves can be stacked together into one graph as shown in Figure 2.4.

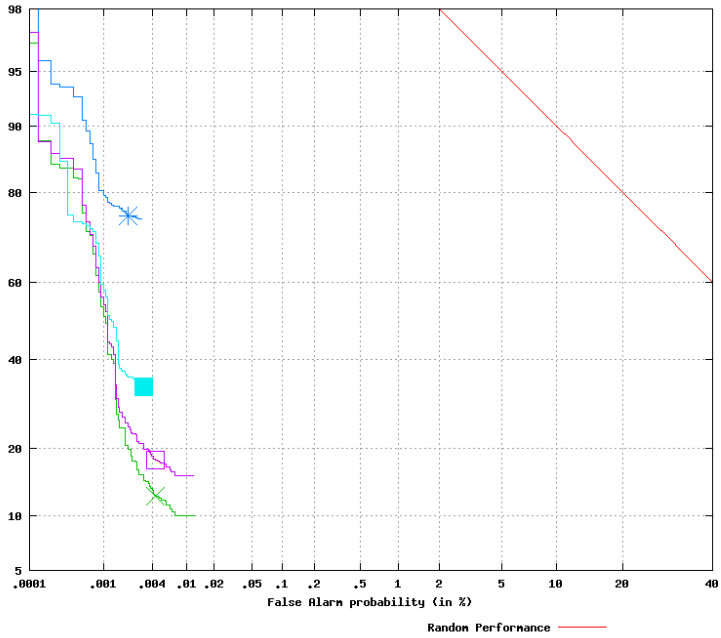


Figure 2.4: Example of DET curves of several STD systems. As STD systems provide limited number of detections, DET curves do not spread across all possible p_{FA} values (x axis). The figure shows performance of our systems submitted for NIST STD 2006 evaluations [Fiscus et al., 2007].

2.3.2 TWV (ATWV, UBTWV)

Actual term weighted value (ATWV) metric was defined by NIST for 2006 spoken term detection evaluations as

$$ATWV(thr) = 1 - \underset{Q}{average}\{p_{MISS}(Q, thr) + \beta p_{FA}(Q, thr)\} \tag{2.4}$$

where

$$\beta = \frac{C}{V}(Pr_Q^{-1} - 1),$$

C is cost of a false alarm, V is value of a hit, Pr_Q is prior probability of a query. For NIST STD 2006 evaluations, the ratio $\frac{C}{V}$ was set to 0.1 and Pr_Q to 10^{-4}, which made β constant and equal to 999.9. However, for other tasks and datasets, the value of β might be set differently, e.g. for 2012 MediaEval evaluations β was set to 15.32.

When we carefully look at denominators in equations for $p_{MISS}(Q, thr)$ and $p_{FA}(Q, thr)$ (Eqs. 2.2 and 2.3), we can see that $p_{MISS}(Q, thr)$ depends mainly on the number of reference occurrences of Q, while $p_{FA}(Q, thr)$ depends mainly on the length of evaluation data, and almost not at all on Q. Thus the cost of a false alarm is roughly the same for all queries, while the cost of miss is the higher the fewer reference occurrences the particular query has.

As was stated in [Wegmann et al., 2013], ATWV can be an extremely unstable performance measure, e.g., small changes in underlying recognition performance may result in large changes to ATWV.

For computing ATWV, a hard threshold has to be specified. Although it is suitable for production-ready STD systems, *maximum term weighted value* is more appropriate for development. It is computed as the ATWV for the optimal threshold:

$$MTWV = \underset{thr}{max}\, TWV(thr)$$

There is still a drawback that MTWV uses only one global threshold for all queries. This restriction is loosened in *upper bound term weighted value* [Szöke, 2010] which finds an optimal threshold for each query separately:

$$thr_{ideal}(Q) = \underset{thr}{arg\,max}\, TWV(Q, thr)$$

and the metric is then defined as

$$UBTWV = 1 - \underset{Q}{average}\{p_{MISS}(Q, thr_{ideal}(Q)) + \beta p_{FA}(Q, thr_{ideal}(Q))\}. \tag{2.5}$$

The main drawback of all these three TWV metrics might be setting of the β constant which defines a particular operating point of the STD system. Another drawback is the dependence of miss cost on the number of reference occurrences of each query. For these reasons, to compare two different systems, we can not only compare their term weighted values, but we also have to take into account the β constant and also counts of reference occurrences of searched queries. Despite these drawbacks, TWV metrics were used in 2006 NIST STD evaluations, in many

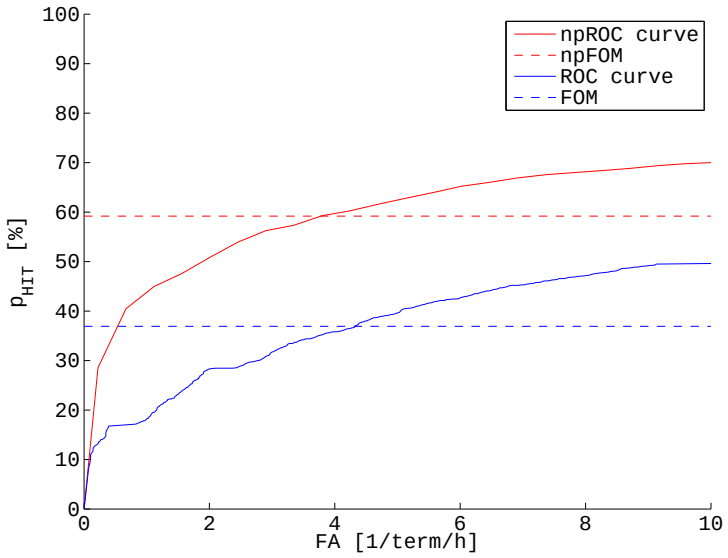


Figure 2.5: Example of a receiver operating characteristics (ROC) and figure of merit (FOM), showing pooled and non-pooled variants of the metrics. The huge difference between pooled and non-pooled variants in this particular example is caused by an inconsistency of scores for different queries. If scores across all queries were well calibrated, pooled results would be much closer to non-pooled ones.

following STD-related papers, in MediaEval Spoken Web Search task, and are preferred also in the current BABEL program[1] of IARPA.

2.3.3 EER

Equal error rate (EER) 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, incorporating only one global threshold for all queries and is defined as follows:

$$EER = \frac{\sum_{Q\in\Delta} N_{target}(Q) - N_{HIT}(Q, thr_{EER})}{\sum_{Q\in\Delta} N_{target}(Q)}, \tag{2.6}$$

where the following equation is satisfied:

$$\sum_{Q\in\Delta} N_{target}(Q) - N_{HIT}(Q, thr_{EER}) = \sum_{Q\in\Delta} N_{FA}(Q, thr_{EER}). \tag{2.7}$$

2.3.4 ROC and FOM

Receiver operating characteristics (ROC) for a particular query Q is a plot of $p_{HIT}(Q, thr)$ as a function of $N_{FA}(Q, thr)$ per hour (see Figure 2.5 for an example). The larger the area under the ROC curve is, the better the system performs.

[1] www.iarpa.gov/index.php/research-programs/babel

Figure of merit is a keyword-spotting accuracy averaged over 1 to 10 false alarms per hour. The FOM calculation assumes that the total duration of the evaluation speech is T hours. All detections are sorted by score and a hit percentage $P_{HIT}(i)$ of queries found before the ith false alarm is calculated for $i = 1 \ \ldots \ N+1$ where N is the first integer $\geq \ 10T - 0.5$. Figure of merit is then defined as

$$FOM \quad = \quad \frac{1}{10T}(P_{HIT}(1) + P_{HIT}(2) + \ldots + P_{HIT}(N) + a\, P_{HIT}(N+1)), \tag{2.8}$$

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

$$P_{HIT}(i) = \sum_{Q \in \Delta} \frac{N_{HIT}(Q, thrFA(i))}{N_{target}(Q)} \times 100\%, \tag{2.9}$$

where the auxiliary function $thrFA(i)$ finds the proper threshold thr for the ith false alarm in all queries. Since this metric does not distinguish individual queries, the $thrFA(i)$ threshold should be ideally similar for all evaluation queries and thus it requires a proper calibration of detection scores across different queries.

According to the FOM definition in [Young et al., 2006], a putative detection is considered to be a hit in case the midpoint of a reference occurrence is between the start and end times of the given detection. However, in our implementation we loosen the time constraints to cancel the effect of incorrect timing in lattices and confusion networks, and we stretch each reference by 0.5s to both directions and when more detections overlap with the same reference, we keep only the one with the highest score and the others are discarded from scoring.

Higher value of FOM means better performance. A drawback of the FOM metric might be the artificial restriction of false alarms per hour from 1 to 10. Even though there are applications which could operate even with a higher number of false alarms, this restriction is valid for most real use cases.

2.3.4.1 Oracle FOM

Similarly to an oracle term weighted value described in [Wegmann et al., 2013], it is possible to compute an oracle FOM value which simulates ideal ordering of hits and false alarms, when all hits have better scores than false alarms across all queries. It is just a theoretical FOM which we could get by modifying scores of the detections we already got from our STD system.

$$FOM_{oracle} \quad = \quad \frac{1}{10T}(N + a) oracle P_{HIT}, \tag{2.10}$$

where a is same as in Eq. 2.8 and $oracleP_{HIT}$ is the overall hit probability

$$oracle P_{HIT} = \sum_{Q \in \Delta} \frac{N_{HIT}(Q)}{N_{target}(Q)} \times 100\%. \tag{2.11}$$

However, if the STD system produces a detection for each second of evaluation data, FOM_{oracle} can get easily to 100%. So the interpretation of the value of FOM_{oracle} should always take into account also the STD system's detection rate

$$detrate = \underset{Q \in \Delta}{average} \left\{ \frac{N_{target}(Q) + N_{nontarget}(Q)}{T} \right\}$$

where T is the total duration of evaluation data in seconds.

2.3.4.2 Unpooling Queries

On the contrary to a global threshold inside the FOM metric, a non-pooled figure of merit (npFOM) calculates the threshold for each query independently and it is therefore more suitable for evaluating STD systems in early stages of their development. The metric was defined by NIST [NIST, 1991] and can be interpreted as an upper-bound estimate of FOM. For each query Q, all detections are first sorted by score. Then the non-pooled figure of merit is defined as

$$npFOM = \frac{1}{10T}(npP_{HIT}(1) + npP_{HIT}(2) + \ldots + npP_{HIT}(N) + a\, npP_{HIT}(N+1)), \quad (2.12)$$

where a is same as in Eq. 2.8 and $npP_{HIT}(i)$ is defined as

$$npP_{HIT}(i) = \sum_{Q \in \Delta} \frac{N_{HIT}(Q, npthrFA(Q,i))}{N_{target}(Q)} \times 100\%. \quad (2.13)$$

The auxiliary function $npthrFA(Q,i)$ finds the proper threshold thr for given query Q and the ith false alarm per hour. The non-pooled FOM metric calculates an individual value of the FOM metric (2.8) for each query separately. Then, the resulting npFOM is a weighted average of FOMs for the whole set of queries, where each term's contribution depends on the number of its reference occurrences. This metric simulates the scenario when a user searches for a single query and then browses through query detections sorted by confidence scores.

The contribution of a given query Q to the overall npFOM can be written as

$$npFOM_{contrib}(Q) = npFOM(Q) \frac{N_{target}(Q)}{\sum_{Q \in \Delta} N_{target}(Q)}$$

where $npFOM(Q)$ is the $npFOM$ computed only for a single query Q.

2.3.4.3 Unpooling Queries and Utterances

In our experiments, we used also another non-pooled FOM metric which simulates an ideal calibration across evaluation utterances, denoted in this work as $unppFOM$, the "utterance-normalized npFOM". Inside this metric, npFOM is "unpooled" even further. The optimal threshold is evaluated separately not only for each query, but also for each evaluation utterance.

$$unnpFOM = \frac{1}{10T}(unnpP_{HIT}(1) + \ldots + unnpP_{HIT}(N) + a\, unnpP_{HIT}(N+1)), \quad (2.14)$$

where a is same as in Eq. 2.8 and $unnpP_{HIT}(i)$ is defined as

$$unnpP_{HIT}(i) = \sum_{Q \in \Delta} \sum_{U \in \Upsilon} \frac{N_{HIT}(Q, U, npthrFA(Q,U,i))}{N_{target}(Q,U)} \times 100\%. \quad (2.15)$$

where Υ is a set of all evaluation utterances, $N_{HIT}(Q,U,thr)$ the number of detections of the query Q in utterance U which are identified as hits and their score remains above the threshold thr, the auxiliary function $npthrFA(Q,U,i)$ finds the proper threshold thr for given query Q, utterance U and the ith false alarm per hour and $N_{target}(Q,U)$ is the number of all occurrences of the query Q in the utterance U.

The value of $unnpFOM$ tells us how well an STD system could perform if we perfectly calibrated scores across both queries and utterances. It can also be interpreted as an average FOM if we searched for only one query in one utterance. It has to be noted that using this metric is appropriate only for longer test utterances with a large enough number of opportunities for incorrect detections. High value of $unnpFOM$ tells us that for each query and utterance pair, hits have higher scores than false alarms, so sorting detections by scores moves true hits to the top of the list.

It should be noted that for each query Q, this metric takes into account only utterances where Q occurs in the reference. All other utterances together with all their detections are discarded. This significantly reduces number of false alarms, especially when the ratio of utterances where Q occurs is low.

In our work, we did not incorporate techniques for calibrating scores across utterances, but they were described e.g. in [Srikanth, 2013].

2.3.5 Choosing the right metric

In this work, we present results of several query-by-example systems, each with 16 combinations of data and recognizer languages. Besides that, we had to score and compare many other results during the development. Therefore we needed to use a metric which produces a single score. Since we did not calibrate our systems, candidates for the primary scoring metric were npFOM and UBTWV. Because the term weighted value has an inherent dependence of miss cost on the reference occurrences of each query, and also it requires to set the β constant to define the system's operating point, we decided for the npFOM metric. It is more coherent and in all papers it is reported for the same operating point, so it is much better for comparison across systems and data sets. For some experiments we will also present npROC curves to show systems' performance over a wider range of operating points and also the pooled FOM metric to show systems' inherent calibration across queries. The smaller the difference between pooled and non-pooled FOM (i.e., FOM and npFOM) is, the better the system is calibrated.

During the development, we often needed to compare results of our systems. It turned out that comparing only overall npFOM values may be a bit misleading, because one system could perform better on some set of queries, while the other performs better on another set of queries. Thus we compute also the best $npFOM$ for each query and calculate the overall value:

$$npFOM_{max}(A,B) = \sum_{Q \in \Delta} max\{npFOM_{contrib}(Q,A),\, npFOM_{contrib}(Q,B)\}$$

where A and B are sets of detections generated by two STD systems and $npFOM_{contrib}(Q,A)$ is the contribution of query Q to the overall $npFOM$ for system A. When the value of $npFOM_{max}(A,B)$ is considerably higher than the best overall $npFOM$ score of the two individual systems, it means that the systems are complementary to some extent. Each of the systems outperforms the other one for some set of queries, so it could be possible to create a fusion which combines them to produce results closer to $npFOM_{max}(A,B)$.

2.3.6 Reading Result Figures

Throughout this work, we present results in form of figures which we find more readable than tables. An example with description of important figure elements is shown in Figure 2.6. For comparing multiple results, we omit npROC curves and show only npFOM part of these figures.

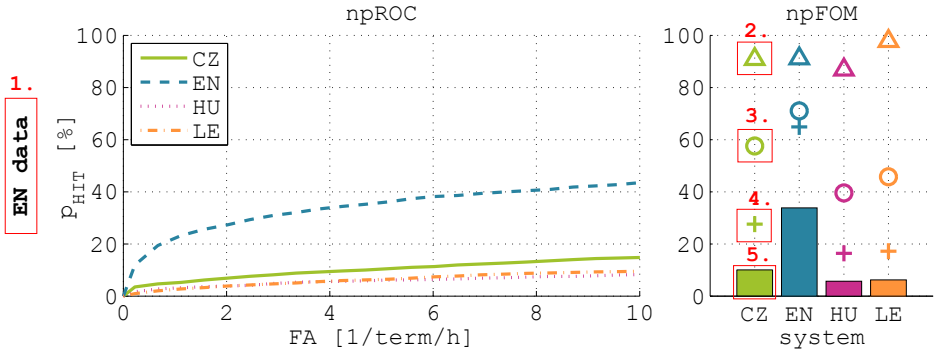


Figure 2.6: Example of results figure as used for presenting results in this work. Left part contains npROC curves, right part npFOM metric and its derivatives, unnpFOM and oracleFOM. "1." denotes the evaluation dataset, either one of the four languages (CZ, EN, HU, LE) or a language dependent setup, where the dataset language corresponds to each system. "2." shows the oracleFOM, "3." unnpFOM, "4." is the maximum npFOM of all 5 examples of queries and "5." is npFOM.

Performance of systems with a single example per query is influenced by the choice of examples. To show the potential of the systems, we show the maximum npFOM of all five examples for each query by a "+" marker in the figures. As it denotes npFOM of the single best example, it is shown only for systems with a single example per query.

2.4 Related Work

In this section, we present an overview of published works, related to query-by-example spoken term detection, which helped us during the research and development of our own work.

2.4.1 Spoken Term Detection

The main boom in STD research was launched by the 2006 NIST STD evaluations, where we participated with 9 other institutions and companies [Fiscus et al., 2007]. The main objective of these evaluations was to help develop technology for rapidly and accurately searching very large quantities of audio data. Therefore, systems had to be implemented in two phases: indexing and search. Also, a new metric was defined for evaluating STD systems, the term-weighted value (TWV) [Fiscus et al., 2006]. This metric, although being very popular since then, has few specific drawbacks already described in Section 2.3.2.

Various types of input data were investigated for STD systems. Using 1-best string output of a recognizer is usable only for very low word or phone error rates and was investigated in [Pinto et al., 2008], where authors used a probabilistic pronunciation model for queries to compensate for errors in the 1-best phone output of a recognizer. Results were comparable with using phone lattices only for longer queries of at least 14 phones. The lower was the number of phones, the larger was the degradation of performance of 1-best strings. The main advantage of using 1-best phone strings was the index size and search speed.

The predominant STD approach seems to be using word, phone or hybrid lattices, or more compact confusion networks [Mangu et al., 2014], which contain also less probable hypotheses

[Szöke et al., 2008; Wallace et al., 2007; Norouzian and Rose, 2014]. Phone or hybrid lattices are needed to address the out-of-vocabulary (OOV) issue of LVCSR. In [Parada et al., 2010], authors converted word lattices to phone lattices using a pronunciation dictionary to be able to search for OOVs. To get more detections, each word query converted to a phone string was enriched by a pre-trained phone confusion matrix which lead to a considerable performance gain.

To speed up the search phase of STD systems, lattices can also be indexed, usually by creating an inverted index similar to the one used in text search systems. The index contains individual words, phones [Mamou et al., 2007] or their n-grams [Iwami et al., 2010]. A drawback of this indexing approach may be that even though two words or phones may appear close to each other in the inverted index, they may not share a common path in the original lattice. As an extension to this basic inverted index scheme, in [Burget et al., 2006] we proposed a method for fast verification of word, phone and hybrid sequences in indexed lattices. However, a more elegant indexing approach was proposed in [Can and Saraclar, 2011], where authors use WFSTs to create an inverted index where all paths from original lattices are inherently preserved and directly traversed during the search.

2.4.2 Query-by-Example STD

As we mentioned earlier in Section 2.2, in contrast to query-by-text STD systems where queries are entered in a textual form, queries in QbE STD systems are in spoken form, which has to be converted to appropriate features, lattices or confusion networks.

As the interest in query-by-text STD grew up after the 2006 NIST STD evaluations, similarly the interest in QbE STD grew up rapidly after 2011 MediaEval evaluations, particularly its Spoken Web Search task, which is held annually since then. It aims at QbE for low-resource languages [Metze et al., 2014].

Majority of QbE STD systems are based on *template matching* (DTW) of phone posteriorgrams or other, usually unsupervisedly trained, appropriate features [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; Mantena and Prahallad, 2013]. Similar template matching techniques are also applied in motif/word discovery task [Jansen et al., 2010; Muscariello et al., 2009; Park and Glass, 2008], which can be considered an extension to the QbE STD. Depending on the used set of features, some of these approaches are language-independent, having no knowledge of the target language, and are therefore suitable for low-resource languages. Several works also investigated ways of speeding up the DTW approach [Zhang and Glass, 2011; Schmalenstroeer et al., 2011; Mantena and Anguera, 2013; Anguera, 2013] which is otherwise too slow for most applications.

Sequential statistical models (GMM/HMM) are used in much fewer works. In this approach, GMM/HMM (or similar) model is trained on query features, a background model is trained on features of all available utterances and they are both matched against each utterance, providing likelihood ratios of the two models as detection scores [Velivelli et al., 2003; Chan and Lee, 2011; Szöke et al., 2012; Abad et al., 2012].

There are also several QbE STD systems working with *lattices or confusion networks* [Lin et al., 2008; Parada et al., 2009; Shen et al., 2009; Lin et al., 2009; Barnard et al., 2011], where the matching of a query lattice against utterance lattices is done usually either by dynamic programming, discrete HMMs (DHMMs), graphical models or by composition of transducers (WFST). In [Hazen et al., 2009], authors compared QbE performance of DTW on phone poste-

riorgrams and DHMMs on confusion networks, finding out that DTW outperformed DHMMs probably because of the larger granularity of DHMM (time-aligned segments) than that of DTW (frames).

All these three approaches to QbE STD seem to be complementary to some extent and their fusion can lead to improved performance [Abad et al., 2013; Szöke et al., 2014]. For example in [Parada et al., 2009], authors use lattice representation of both queries and utterances, convert them to WFSTs, and then perform search in two steps. First they compose the query with an inverted index providing a list of matching utterances, and afterwards the query is composed with each utterance transducer returned in the first step. They experimented also with the relevance feedback technique, where for each query, the best detection returned by a query-by-text STD was searched also by the QbE STD system, enriching the list of results, leading to better performance, without any fusion of scores. In [Szöke et al., 2014], phone posteriorgrams of 13 different languages were employed for both DTW and GMM/HMM QbE systems, then fused together leading to considerable improvement in term weighted value metrics.

There were also efforts to build language-independent phone recognizers [Schultz and Waibel, 2001; Walker et al., 2003; Kempton et al., 2011; Knill et al., 2014], which could be directly used for language-independent QbE STD, but they still have relatively high phone error rates.

In this work, we describe, compare and analyze the three main approaches to QbE STD: template matching (DTW), sequential statistical modeling (GMM/HMM) and composition of lattice transducers (WFST), while all three systems will be evaluated in four language-dependent setups (Czech, English, Hungarian and Levantine), and twelve language-independent setups (all other combinations of the four target languages with four language-specific phone posterior feature extractors).

Chapter 3

Experimental Setup

A significant portion of this chapter was originally written by Igor Szőke and František Grézl from the BUT Speech@FIT research group and it was published with few differences in [Tejedor et al., 2012]. Igor and František also trained the feature extractor used for our experiments.

Description of our experimental setup is important for interpretation of our results, for comparing the systems and for reproducibility of our experiments. In this work, we evaluate DTW, GMM/HMM and WFST query-by-example systems on Czech, English, Hungarian and Levantine evaluation datasets. A phone recognizer, trained on different datasets of the same set of languages, produces input data for all the three query-by-example systems.

We used two main setups for our experiments. Language-dependent setup where the phone recognizer was trained on the evaluation data language and language-independent setup where the evaluation language and the language on which the phone recognizer was trained, do not match.

There is an important fact to note here, about our experimental setup: all our evaluated systems ran on the same data (features, queries, ...) which makes their comparison more relevant. The only exception is the GMM/HMM system which we evaluated, besides the common posterior features, also with bottleneck features.

3.1 Data

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 English and Levantine. Hungarian was trained and evaluated only on prompted read telephone speech, Czech training data contained partly prompted 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 3.1. The *query training set* was used for extracting query examples and, if necessary, for parameter and model estimation. Finally, the *evaluation*

	Data (hours)			Queries			Data
Language	Feature training	Query training	Eval.	Unique	Examples	occurrences	type
English	277.7	12.5	2.3	168	840	2007	CTS
Hungarian	8.5	0.9	2.4	8	40	337	PRTS
Levantine	19.9	2.8	2.5	51	255	609	CTS
Czech	100.9	2.2	1.4	58	290	1019	CTS RTS PTS

Table 3.1: Overview of data. The table consists of three parts. The first one shows amounts of data used as feature training set, query training set, and evaluation set. The second one shows numbers of queries, examples and query occurrences in evaluation data for each language. The third part shows the type of data: CTS - continuous telephone speech, PRTS - prompted read telephone speech, RTS - radio telephone speech, PTS - prompted telephone speech.

set was used for testing the approaches.

English - as a feature training set, CTS data from the Switchboard I, Switchboard Cellular, and Call Home English corpora, were used, whereas the Fisher corpus was used as a query training set and the NIST STD 2006 development set was used for evaluation.

Hungarian - telephone prompted read speech from the Hungarian part of the SpeechDat East corpus, was divided into a feature training set, query training set, and an evaluation set[1].

Levantine - Levantine Arabic CTS Corpus, was also divided into a feature training set, a query training set and an evaluation set was used for the Levantine. We employed only nondiacritized forms of transcripts.

Czech - data created for project No. VD20072010B16 (supported by the Czech Ministry of Interior), were 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.

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 3.1 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

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

Language	Duration [h]		Average duration per utterance [h]		Utterances count
	Total	Speech	Total	Speech	
English	5.9	2.3	0.082	0.032	72
Hungarian	3.5	2.4	0.017	0.012	200
Levantine	5.0	2.5	0.100	0.050	50
Czech	1.9	1.4	0.026	0.019	76

Table 3.2: Evaluation data summary.

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.

3.1.1 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 the 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 numbers of occurrences in the evaluation set for all languages are summarized in Table 3.1. 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 Appendix B.

3.1.2 Reference Transcriptions

The purpose of this work is to identify regions in utterances that match the spoken query. To get closer to the spoken query nature, we transformed word transcriptions for each evaluation set to phone ones. Word/orthographic transcriptions together with pronunciation dictionaries were used to derive corresponding reference phone transcriptions for our set of languages. Forced alignment was carried out to get the time information for each phone. Then each query was transcribed to phones by a pronunciation dictionary and these phone sequences for each query were looked up in the phonetic forced-alignment producing a new phone based reference. 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). This way we generated references suitable for query-by-example systems working at the phone level instead of words.

For the interpretation of the unnpFOM metric described in Section 2.3.4.3, lengths of individual evaluation utterances are important. Table 3.2 summarizes them.

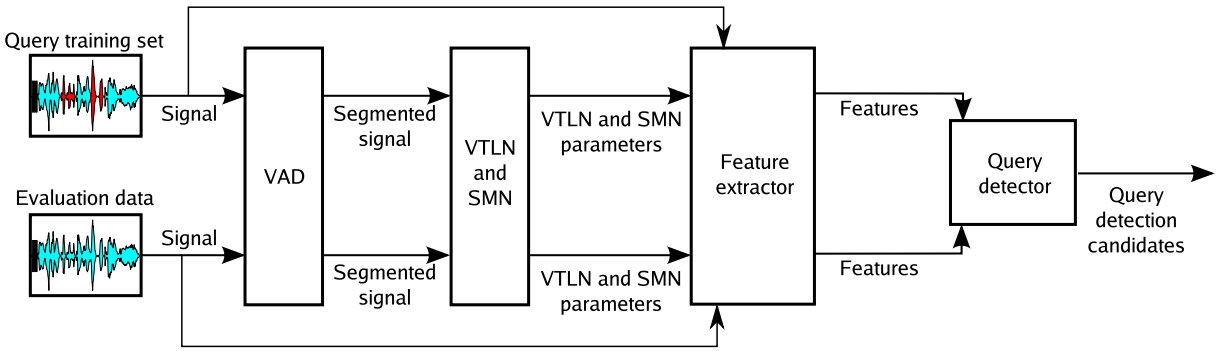


Figure 3.1: High-level schema of our query-by-example STD system.

3.2 Audio Preprocessing and Feature Extraction

Our QbE STD system is shown in Figure 3.1. 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 core of this work is the query detector block. All other blocks were developed by the BUT Speech@FIT research group and are briefly described in following sections.

Since one of our goals is to evaluate various QbE STD systems across language-dependent and language-independent setups, the core of our QbE STD systems 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. To localize the differences of our query-by-example systems only into the query detector block, we kept the feature extractor block the same for all our systems.

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 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 3.3 presents the features used within each query detector in this thesis.

3.2.1 Voice Activity Detection

VAD is conducted to properly derive the speaker normalization/adaptation parameters, as depicted in Figure 3.2. Raw features of an input audio signal are 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

	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			

Table 3.3: Combination of Feature Extraction and QbE STD Approaches in this thesis. The WFSTdict system is derived from our WFST approach where "language-dependent" lattices were generated and reference query pronunciations (i.e., as in query-by-text STD) were used for searching.

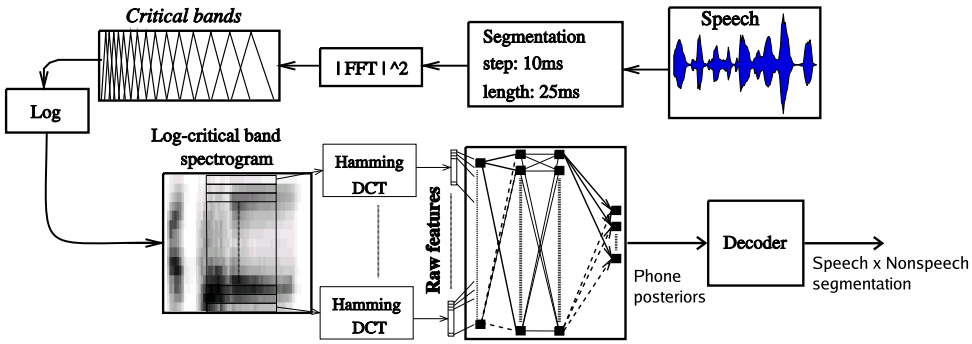


Figure 3.2: Voice activity detector.

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].

3.2.2 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 3.1). 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 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).

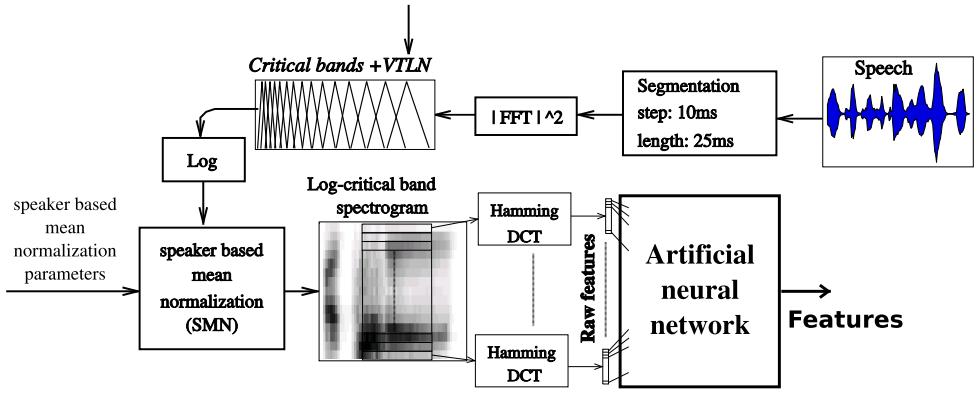


Figure 3.3: Feature extraction.

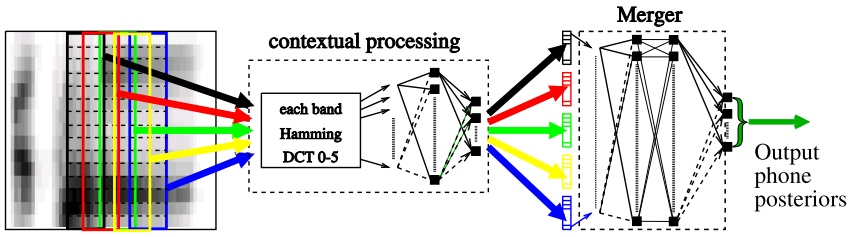


Figure 3.4: Universal context (UC) neural net architecture.

3.2.3 Feature extractor

This block, depicted in Figure 3.3, 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 3.3 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 3.3 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 3.4, which consists of two different parts: a context network and a merger.

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

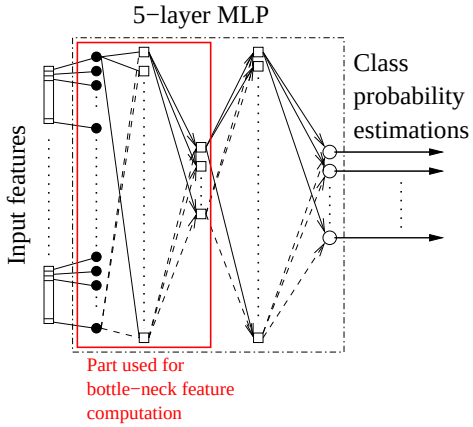


Figure 3.5: Training of the context bottleneck neural network. A 5-layer neural network with an 80-neuron bottleneck layer in the middle and 3-state phone posterior classes are trained first. Then the 4^{th} and 5^{th} layers are removed and the output of the network is taken from the bottleneck layer.

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 3.4 shows the sizes of this output layer for each language and Table 3.5 shows the phone recognition performance on the four evaluation languages.

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 3.5.

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 3.6.
- 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, according to the column $size(out)$ in Table 3.4. 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 (Figure 3.7f), 3-state phone posteriors have a non-Gaussian distribution (Figures 3.7b and 3.7d). Posterior features have a limited range

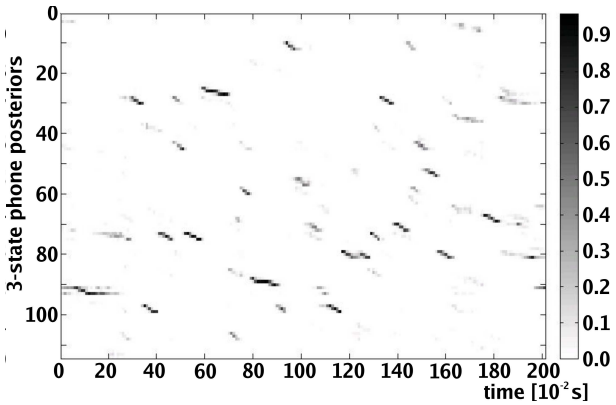


Figure 3.6: An example of 114 3-state phone posteriors (38 phones times 3 states per phone) for 2 seconds of speech. The 3 states represent beginning, center, and end of a phone. The x-axis represents the time in a hundredth of a second and the y-axis represents the indices of 3-state phone posteriors.

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

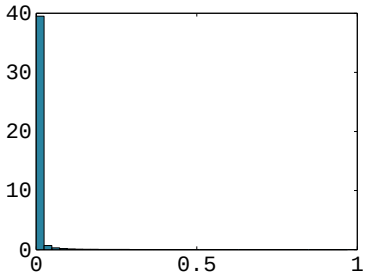
Table 3.4: Numbers of Phones and Sizes of Layers for Different Languages. $size(hid\_UC)$ represents the size of the hidden layer in the universal context NN. $size(hidMer\_3stphn)$ is the size of the hidden layers in the merger with 3-state phone posterior output. $size(hidMer\_BN)$ is the size of the hidden layer in the merger with bottleneck output, and $size(out)$ is the size of the 3-state phone posterior output layer.

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 phone posteriors.

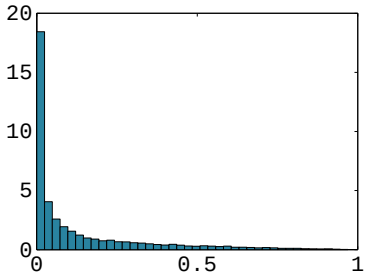
- The last difference is in computational effectiveness. Both posterior and bottle-neck features are trained with an equal number of parameters in the merger. However, the size of the bottleneck NN merger (3-layer NN) is half the trained size (5-layer NN) (see Figure 3.5).

Language	Corr. [%]	Acc. [%]	H	D	S	I	N
English	56.39	47.36	82198	35701	27865	13162	145764
Hungarian	69.38	66.18	65442	11061	17823	3014	94326
Levantine	64.00	43.76	40595	9818	13018	12839	63431
Czech	66.93	63.24	41129	10036	10283	2269	61448

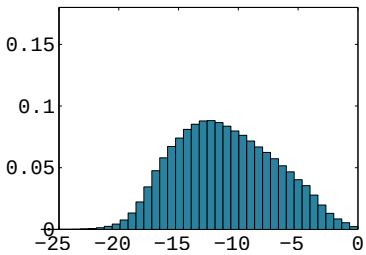
Table 3.5: Phone recognition performance on evaluation data. H is the number of correct phone labels, D is the number of deletions, S is the number of substitutions, I is the number of insertions and N is the total number of phone labels in transcriptions.



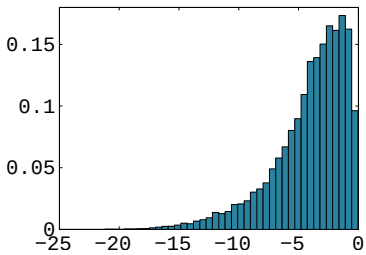
(a) Posteriors of the middle state of phone "aa" across all utterances.



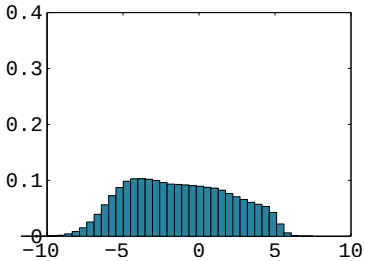
(b) Posteriors of the middle state of phone "aa" only for reference occurrences of "aa".



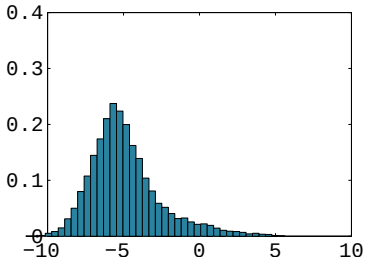
(c) Log posteriors of the middle state of phone "aa" across all utterances.



(d) Log posteriors of the middle state of phone "aa" only for reference occurrences of "aa".



(e) Single bottleneck feature "bn01" across all utterances.



(f) Single bottleneck feature "bn01" only for reference occurrences of the phone "aa".

Figure 3.7: Example histograms of posterior and bottleneck features, showing the inherent Gaussian-like distribution of bottleneck features.

Chapter 4

Acoustic Keyword Spotting (AKWS) – Upper-bound

The acoustic keyword spotting system's setup and experiments were done by Igor Szőke from the BUT Speech@FIT research group, where the AKWS system was developed. Igor also wrote a major part of this chapter, which was published with the same results in our common paper [Tejedor et al., 2012].

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 4.1. First, the utterances are passed through a VAD that removes the non-speech 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 4.4, and is built from the given 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 in frames to compensate for different lengths 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 4.2). 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}

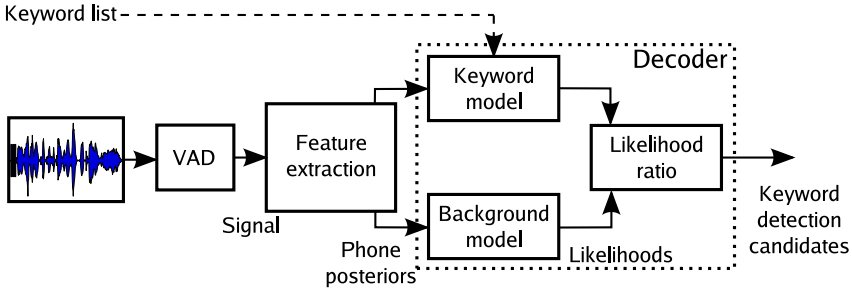


Figure 4.1: Schema of acoustic keyword-spotting system.

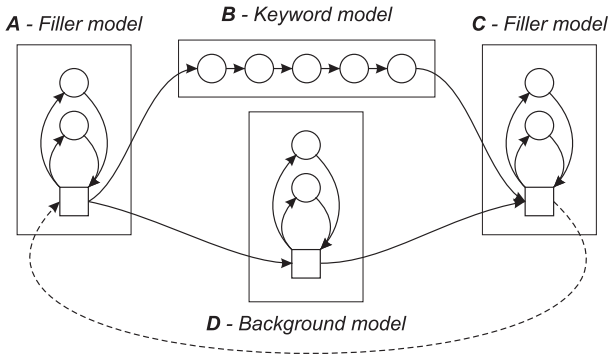


Figure 4.2: General acoustic keyword-spotting network.

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 4.3, 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 4.4. 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.

4.1 Results

We built the acoustic keyword-spotting system as an upper-bound reference for each of our evaluation languages. Results are summarized in Figure 4.5. We can see that the Hungarian system, which is trained and tested on clean read telephone speech provides the best overall performance among all four languages. The worst accuracy was on 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 exhibit the second worst overall performance, due to its data mismatch (read speech versus CTS). Gaps between npFOM and utterance-normalized npFOM values show that all four AKWS systems could benefit from an utterance normalization technique. By comparing unnpFOM and oracleFOM results, we can see that for Levantine data, the gap between the two metrics is larger than for the other languages. It means that even for a single evaluation utterance, the Levantine system produces slightly

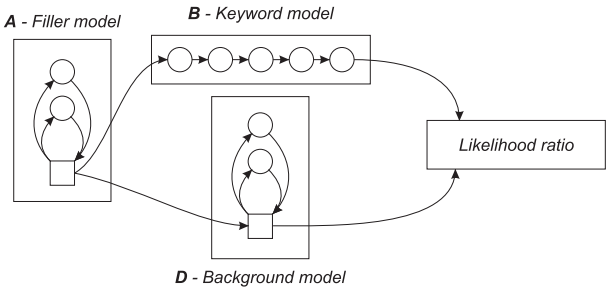


Figure 4.3: Likelihood ratio computation in acoustic keyword-spotting.

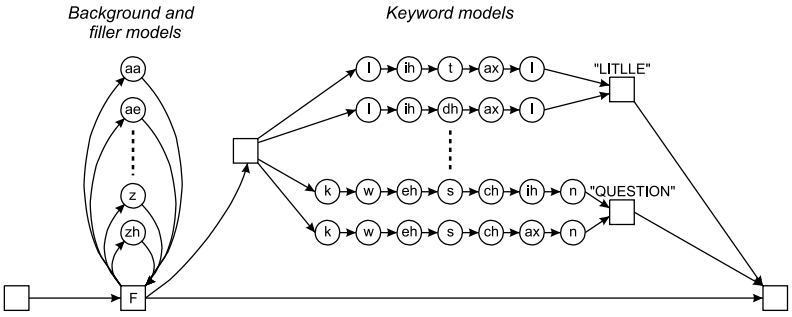


Figure 4.4: Example of a real keyword-spotting network in AKWS. Each phone model is represented by a 3-state hidden Markov model tied to 3-state phone posteriors.

larger amount of false alarms with higher confidence scores than hits. The Levantine system has also significantly higher detection rate, which may cause higher value of oracleFOM. Levantine system produces 0.27 detections per second, while the other systems range from 0.028 to 0.038.

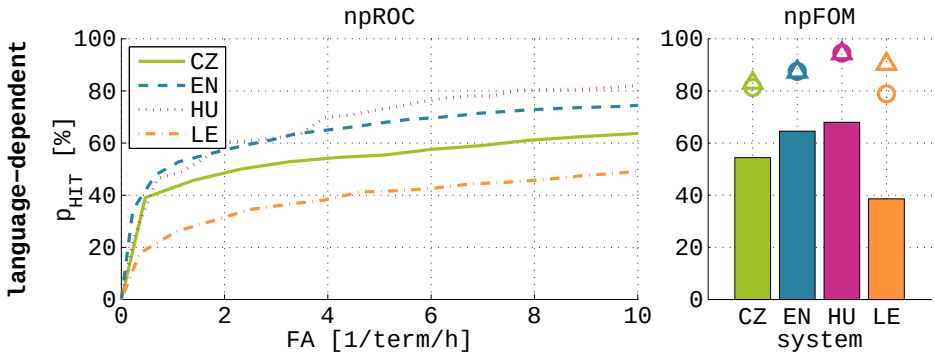


Figure 4.5: Results of acoustic keyword-spotting experiments.

Chapter 5

GMM/HMM-based Query-by-Example Detector

The GMM/HMM QbE system setup was done by Igor Szőke from the BUT Speech@FIT research group. Some of the presented experiments were done by Javier Tejedor from the HTCLab at Madrid university. Major part of this chapter was written by Igor and Javier and was published with similar results and minor differences in our common paper [Tejedor et al., 2012].

Filler model-based acoustic keyword spotters, the one like in Section 4, 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 should be mitigated. In this direction, our GMM/HMM-based QbE STD system is inspired by acoustic keyword spotting, similarly to [Velivelli et al., 2003].

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 4, 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 length of the query to completely remove the language-dependency of this approach. The five examples from the training query set (see Section 3) 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 4), 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 recogni-

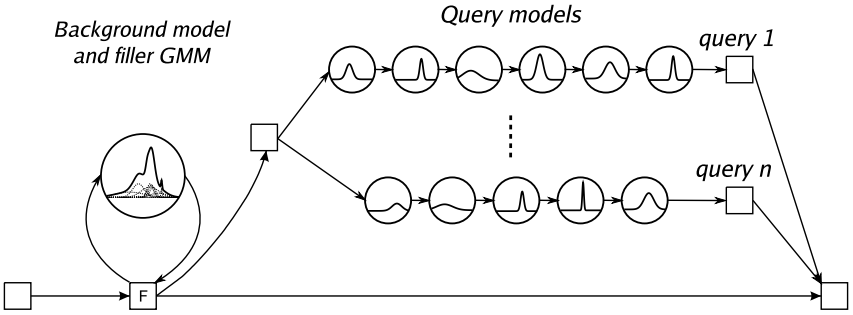


Figure 5.1: Recognition network for the GMM/HMM-based QbE detector. The background model is represented by a single state with 40 Gaussian mixtures. The query model is a linear concatenation of states (HMM), each represented by a single Gaussian.

tion network illustrated in Figure 5.1. 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.

5.1 Results

Similarly to all our QbE STD systems, we evaluate the GMM/HMM system in all language-dependent and language-independent setups of Czech, English, Hungarian and Levantine languages with 3-state phone posterior features, using a single example and five examples per query. Besides that, the GMM/HMM system is evaluated also with bottleneck features.

5.1.1 Language-dependent

Results of language-dependent setups with single example and five examples per query are summarized in Figure 5.2. We can see that in the case of a single example per query, the performance is quite low due to low amount of training data for GMM/HMM models of queries. However, for five examples per query, performance increases considerably. We clearly see that bottle-neck features outperform 3-state phone posteriors in all language-dependent setups except for the Hungarian data where the difference is not significant. We consider that this discrepancy is due to the small amount of data used to train the Hungarian feature extractor, which makes a poorer estimation of the bottle-neck features.

5.1.2 Language-independent

Results for all language-dependent and language-independent setups are shown in Figure 5.3. By inspecting Hungarian data in the figure, we can see that Czech bottleneck 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. In addition, this is also consistent with the results corresponding to the Hungarian feature extractor on the rest of the language-independent setups.

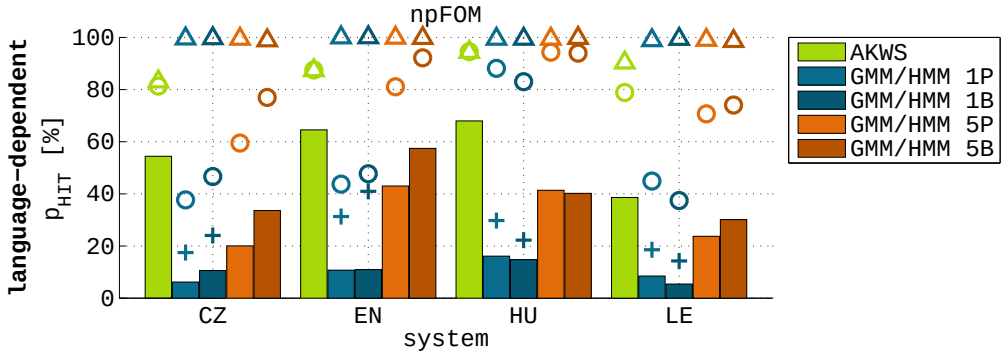


Figure 5.2: Comparison of results of GMM/HMM-based QbE STD systems for language-dependent setups. "1" and "5" denote the number of examples used for training each query, "P" and "B" denote 3-state phone posterior and bottleneck features.

When the language-dependent and the language-independent feature extractors are compared, we can see the former outperform the latter consistently, 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 bottleneck features from the Czech feature extractor slightly outperform the language-dependent feature extractor for Levantine data due to Levantine data complexity.

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 mixed audio conditions of the Czech data (read speech versus CTS) and the nondiacritized approach of the Levantine data. 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 conclude that the set of features used in the GMM/HMM approach is stable across each target language for both language-dependent and language-independent setups in such a way that bottleneck features outperform the 3-state phone posteriors in case the feature extractor is provided with enough training data.

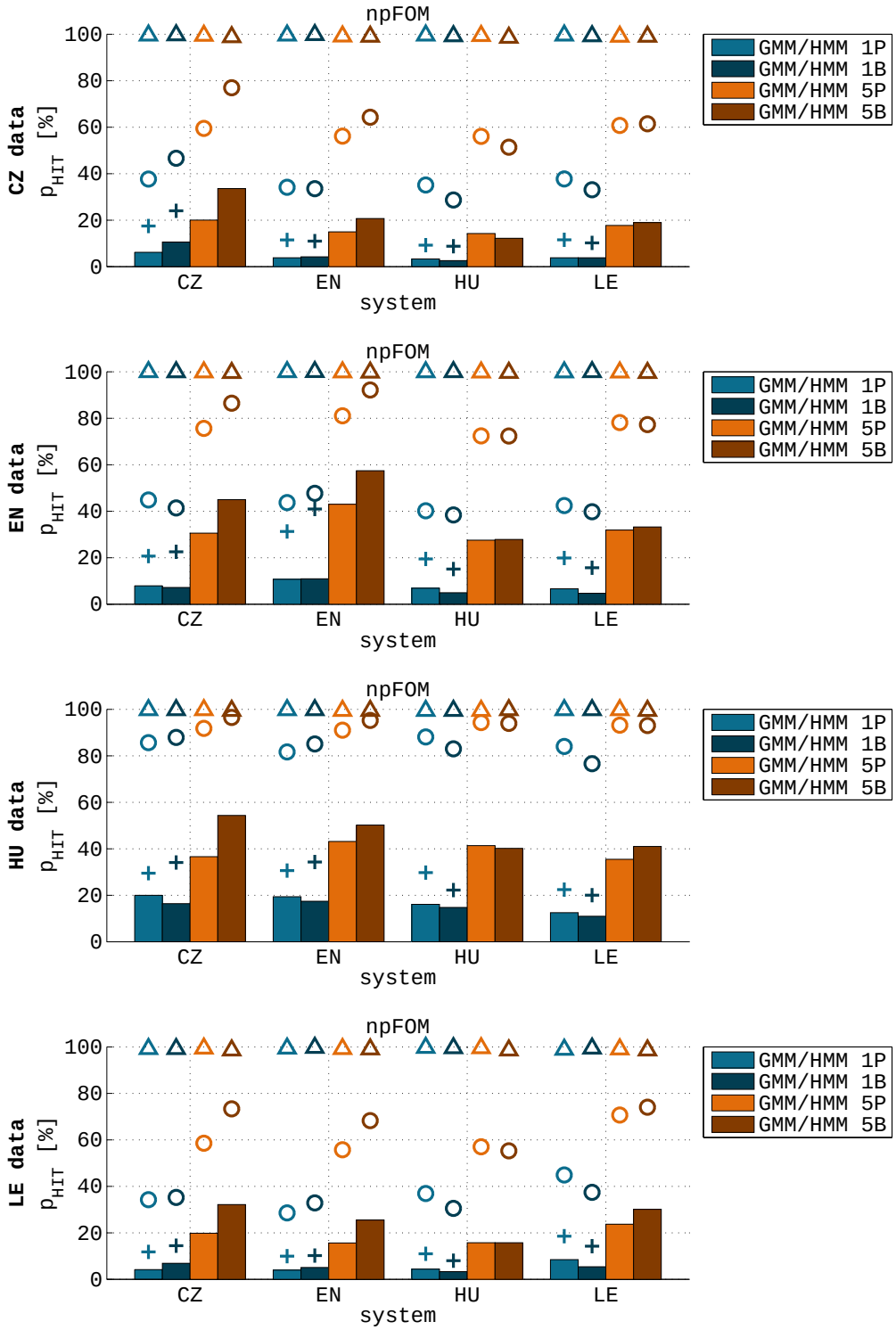


Figure 5.3: Results of the GMM/HMM-based QbE STD systems for all language-dependent and language-independent setups. "1" and "5" denote the number of examples used for training each query, "P" and "B" denote 3-state phone posterior and bottleneck features.

Chapter 6

DTW-based Query-by-Example detector

The DTW-based system was developed by Miroslav Skácel from the BUT Speech@FIT research group, the system for combining query examples was developed by Javier Tejedor from the HTCLab at Madrid university, who also helped us with experiments and wrote a substantial part of this chapter. This chapter with similar results was published in our common paper [Tejedor et al., 2012].

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 utterance and the template matching employs a 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.

6.1 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 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 6.2 (i.e., cosine

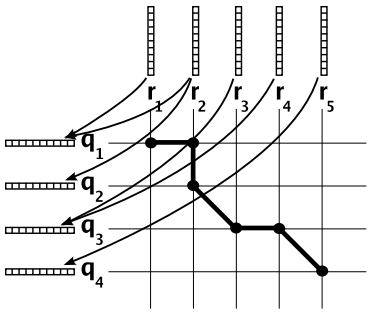


Figure 6.1: An example of a combination of posterior vectors of the best example (Q) and the worst example (R) using the DTW path.

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 $\mathbf{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_N\}$ containing N frames and the worst example $\mathbf{R} = \{\mathbf{r}_1, \ldots, \mathbf{r}_M\}$ containing M frames. Let us define $\mathbf{P}$ as the best path found by the DTW search between $\mathbf{Q}$ and $\mathbf{R}$, containing pairs of indices pointing to $\mathbf{Q}$ and $\mathbf{R}$, 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 $\mathbf{q}_i$ is simply computed as an average of $\mathbf{q}_i$ and all $\mathbf{r}_j$ assigned to i by path $\mathbf{P}$:

$$\mathbf{q}_i^{new} = \frac{\mathbf{q}_i + \sum_{\forall j:\{i,j\}\in\mathbf{P}} \mathbf{r}_j}{1 + N_j}, \tag{6.1}$$

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

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 6.2. Thus 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. It must also be noted that weights for averaging examples by this approach, are $\frac{1}{2}$ for the first example and $\frac{1}{8}$ for all other four examples. Finding the optimal weights of query examples has yet to be investigated more deeply.

6.2 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 3.6. The phonetic speech classes are 3-state phones, similar to 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*:

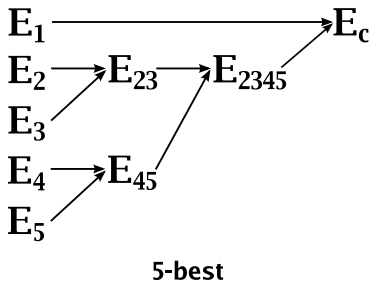


Figure 6.2: Combination of five examples. The E_c is the final "average" (combined) example.

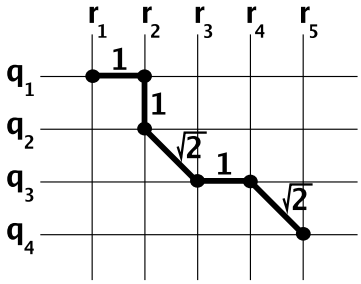


Figure 6.3: An example of a path and a path cost normalization for the DTW search.

$$D(\mathbf{q}, \mathbf{r}) = -\log\left(\frac{\mathbf{q}^T\mathbf{r}}{|\mathbf{q}||\mathbf{r}|}\right), \tag{6.2}$$

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

6.3 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 for every 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 6.3).

6.4 Results

Similarly to all our QbE STD systems, we evaluate the DTW system in all language-dependent and language-independent setups of Czech, English, Hungarian and Levantine languages with

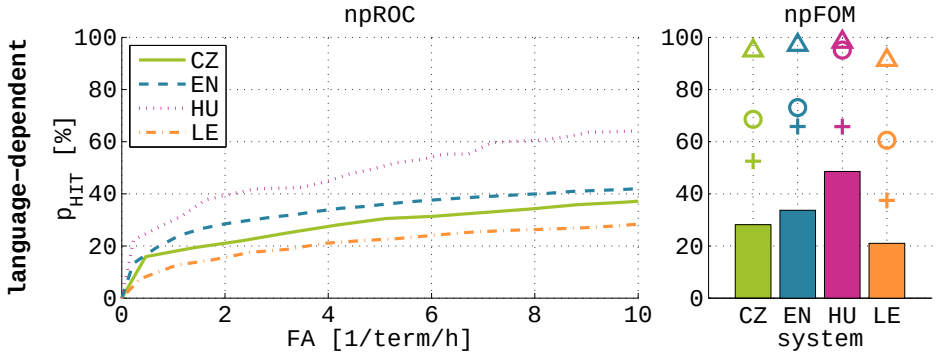


Figure 6.4: Results of language-dependent setups for DTW-based QbE STD systems with *single example per query* for Czech, English, Hungarian and Levantine datasets.

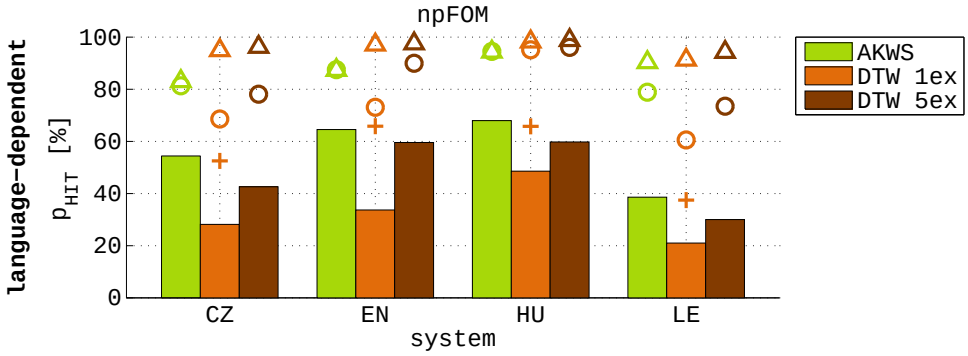


Figure 6.5: Comparison of results of language-dependent setups for AKWS baseline and DTW-based QbE STD systems with single example and five examples per query for Czech, English, Hungarian and Levantine datasets.

3-state phone posterior features, using a single example and five examples per query.

6.4.1 Language-dependent

Figure 6.4 shows the performance of our basic DTW-based QbE STD system for language-dependent setups where each query is represented by a single example. Comparing it to the AKWS baseline results in Figure 6.5, we can see that with only a single example per query, DTW performs significantly worse. Ranking of languages is similar with the lowest performance for Levantine and the highest for Hungarian. One thing should be pointed out in these results: Hungarian has a very high value of unnpFOM and also for other languages, unnpFOM is much higher than npFOM, which means that detection scores are not well calibrated across utterances.

Results for combining five examples of each query are shown in Figure 6.5. We can see that all systems performed better than for the single example per query case. The most significant improvement was achieved for English which performed similarly to Hungarian, while having even a steeper npROC curve in the range of very low false alarm rates (npROC curves are shown in Figure 6.6). Levantine results show the worst performance, similarly to the baseline AKWS results, due to the nondiacritized approach.

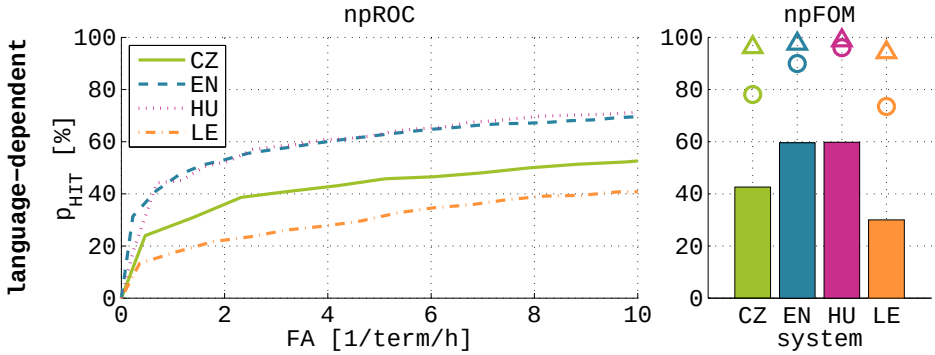


Figure 6.6: Results of language-dependent setups for DTW-based QbE STD systems with *five examples per query* for Czech, English, Hungarian and Levantine datasets.

6.4.2 Language-independent

DTW results for all combinations of data and system languages, where DTW uses only a single example per query, are shown in Figure 6.7. The figure clearly shows, that language-independent setups perform significantly worse than language-dependent setups, the only exception being Hungarian data where English and Czech systems performed closer to the language-dependent Hungarian system. This is probably due to the read nature of the Hungarian data, which may lead to better phone posterior estimates, even when we are dealing with a language-independent feature extractor. Czech system worked also relatively well as language-independent for Levantine data, compared to the Levantine system. Language-independent performance is also low for Czech data, due to the mismatch issue (read speech versus CTS, see Table 3.1).

The performance increased, when we combined five examples of each query, as is shown in Figure 6.8.

Similar patterns were observed across each language for both language-dependent and 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. The figures also show that for language-independent setups, performance dramatically decreases. 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.

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 in language-independent setups.

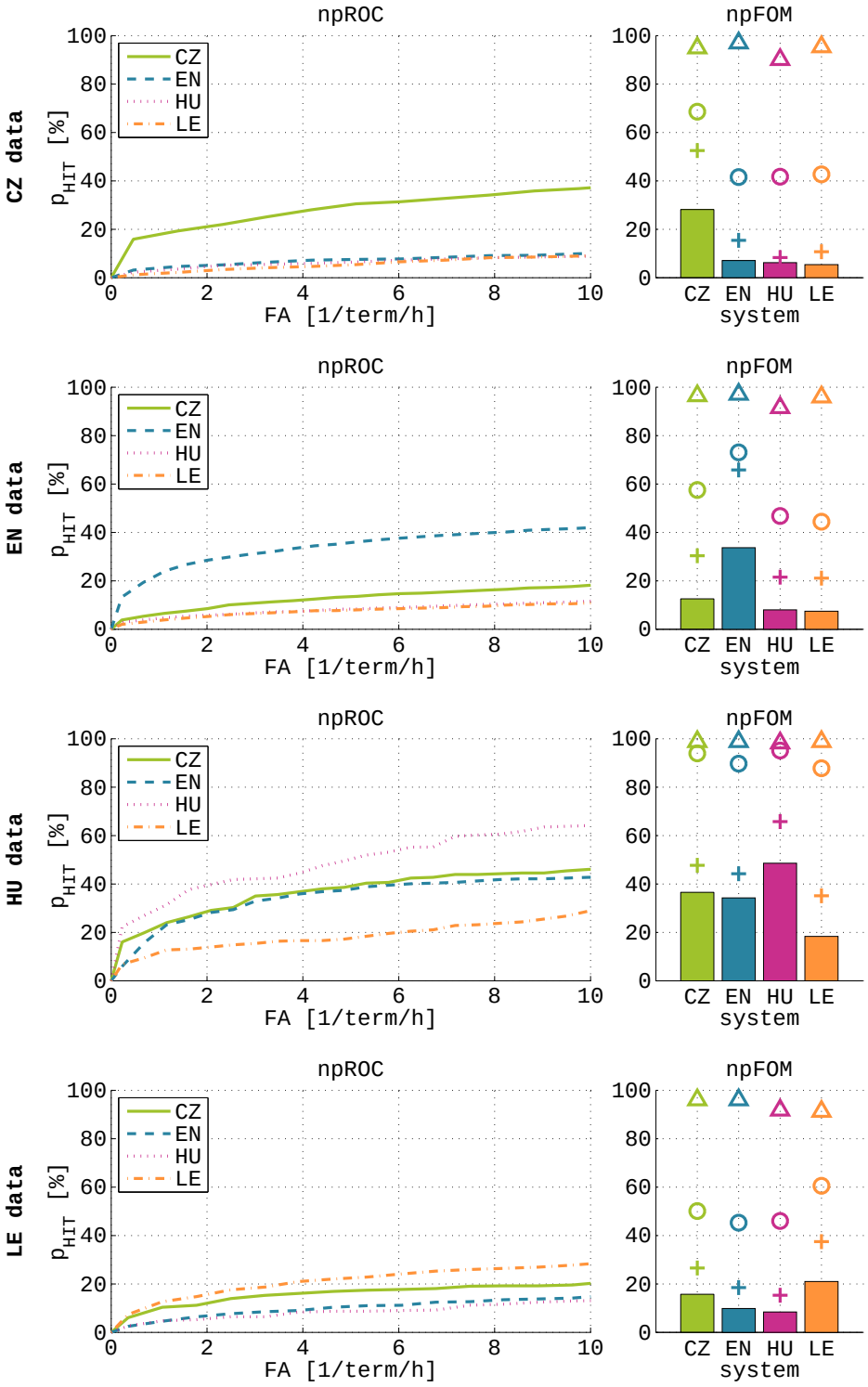


Figure 6.7: Results of the DTW-based QbE STD systems with single example per query for Czech, English, Hungarian and Levantine datasets. For each dataset, results of Czech, English, Hungarian and Levantine query-by-example STD systems are shown.

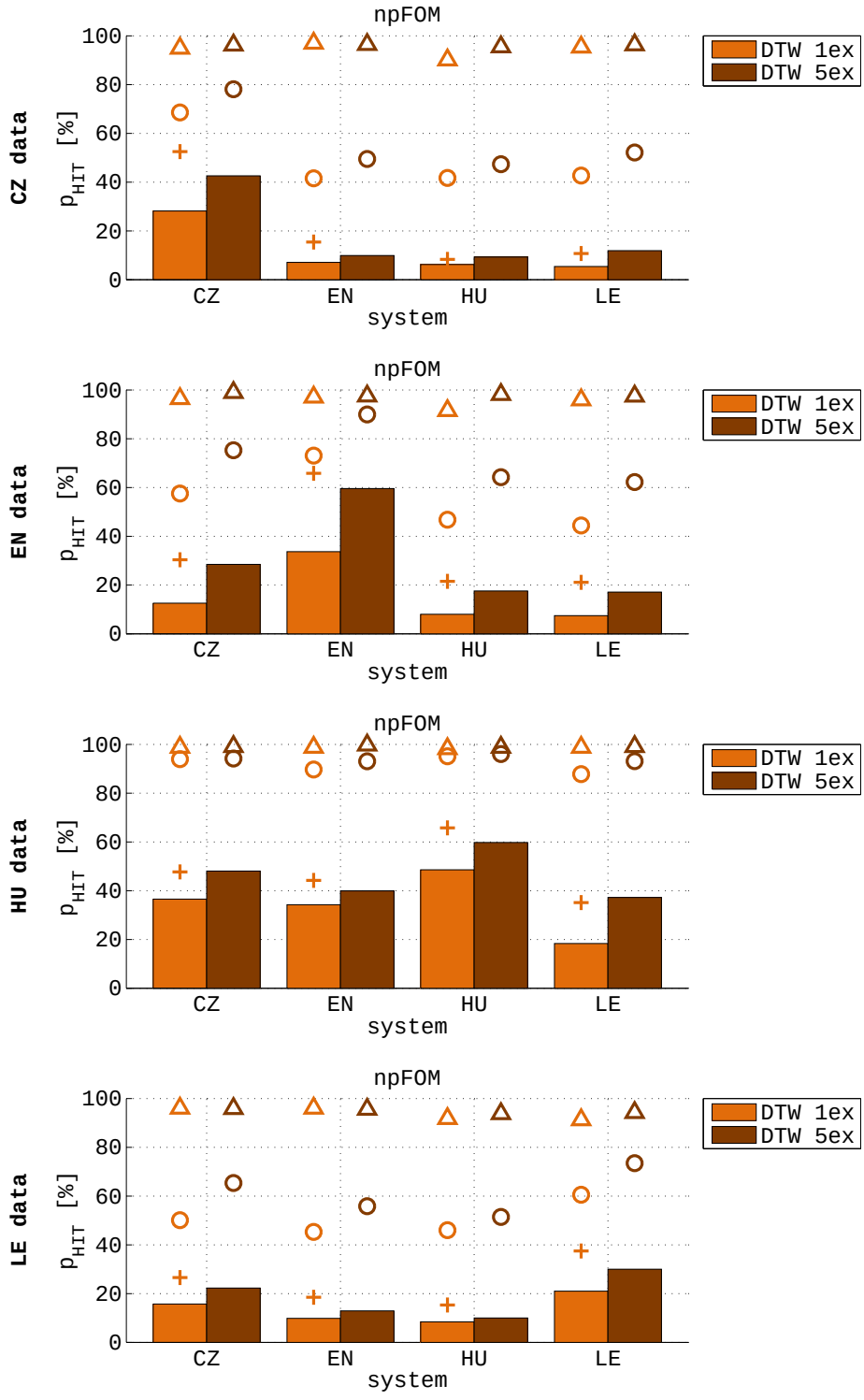


Figure 6.8: Comparison of results for DTW-based QbE STD systems searching for a single example and five examples per query.

Chapter 7

WFST-based Query-by-Example detector

The WFST-based QbE detector forms the core of this thesis. Part of this chapter with early experiments and results was published in [Tejedor et al., 2012]. In this thesis, we developed the system further and achieved substantial improvements.

All the approaches presented earlier used phone posterior features directly when searching for queries. In this chapter, we work with phone lattices[1] or confusion networks[2], derived from phone posterior features, representing both queries and utterances. This approach aims at finding all occurrences of a query lattice inside an 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 and instead of using LVCSR or hybrid word/phone lattices, we create phone lattices from posteriors generated by the phone recognizer described in Section 3.2.

We start this chapter by a brief introduction to transducers and by description of our two WFST systems, one that works with lattices and the other one that works with confusion networks. Then we present initial results of our WFST system as they were published in [Tejedor et al., 2012]. Afterwards, we analyze the system's performance and describe several improvements.

7.1 Introduction to Finite State Transducers

For working with lattices, weighted finite state transducers (WFST) offer a very flexible framework commonly used in LVCSR systems [Mohri et al., 2008; Stoimenov and Schultz, 2009], spoken document retrieval [Allauzen et al., 2004], spoken term detection [Parada et al., 2009; Mangu et al., 2014], etc. All these research areas use WFSTs mainly for combining sequences of noisy probabilistic data in various parts of their systems.

[1] Acyclic graphs of weighted phone hypotheses produced by speech recognizers. Links in these graphs usually include label, likelihood and time information.

[2] Sausage-like structures derived from lattices, consisting of a single sequence of hypotheses groups, where hypotheses of each group overlap in time.

The following brief introduction to transducers is taken with minor modifications from [Pereira et al., 1994]. We do not describe all aspects of WFSTs here, only those which are required for understanding this thesis.

In language processing tasks, we are usually given an observation sequence o, and we have to find out which intended message w is most likely to generate that observation sequence by maximizing

$$P(w, o) = P(o|w)P(w),$$

where $P(o|w)$ characterizes the transduction between intended messages and observations, and $P(w)$ is a prior probability of messages which characterizes the message generator. More generally, the transduction between messages and observations may involve several intermediate stages

$$P(s_o, s_k) = P(s_k|s_o)P(s_o) \quad (7.1)$$
$$P(s_k|s_o) = \sum_{s_1,\ldots,s_{k-1}} P(s_k|s_{k-1}) \cdots P(s_1|s_0) \quad (7.2)$$

where $P(s_k|s_0)$ is the probability of transducing s_0 to s_k through the intermediate stages, assuming that each step in the cascade is conditionally independent on the previous ones. Each s_j is a sequence of units in an appropriate representation (phones, syllables, words, ...). A straightforward but useful observation is that any such a cascade can be factored at any intermediate stage

$$P(s_i|s_j) = \sum_{s_k} P(s_i|s_k)P(s_k|s_j) \quad (7.3)$$

In the transduction cascade (7.2), each step corresponds to a mapping from input-output pairs (r, s) to probabilities $P(s|r)$. More formally, steps in the cascade will be **weighted transductions** $T : \Sigma^* \times \Gamma^* \to K$ where Σ^* and Γ^* are sets of strings over the alphabets Σ and Γ, and K is an appropriate set of weights, for instance the real numbers between 0 and 1 in the case of probabilities.

The equation (7.3) is represented by a **composition** operation of transducers. Given two transductions $S : \Sigma^* \times \Gamma^* \to K$ and $T : \Gamma^* \times \Delta^* \to K$, we can define their composition $S \circ T$ by

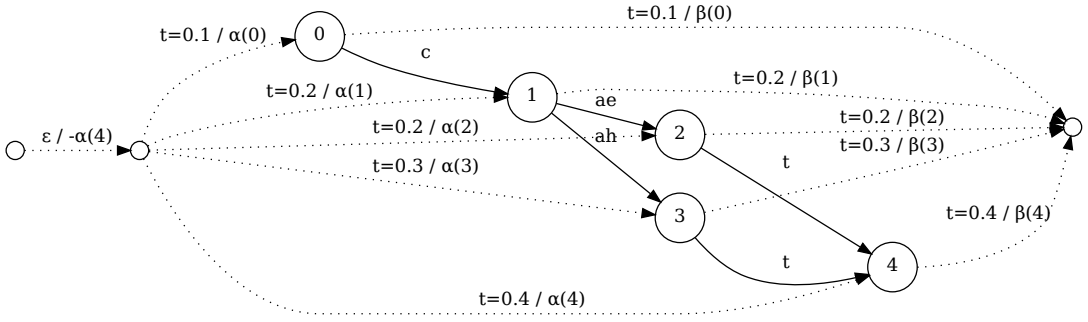
$$(S \circ T)(r, t) = \sum_{s \in \Gamma^*} S(r, s)T(s, t).$$

For example, if S represents $P(s_k|s_j)$ and T $P(s_j|s_i)$ in (7.3), it is clear that $S \circ T$ represents $P(s_k|s_i)$.

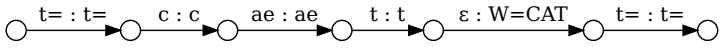
Query-by-example systems based on WFSTs usually convert phone lattices into transducers. The search is then performed by a composition of query transducer with utterance transducer. The resulting transducer contains all occurrences of the query in the utterance with corresponding probabilities. This process will be described in more detail in Section 7.2.3.

7.2 Using Lattices (WFSTlat)

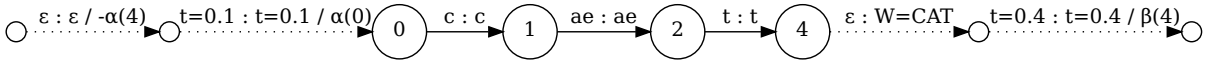
Using only a one-best string output of a phone recognizer for a query-by-example system would lead to a low number of detections and consequently to a low performance when data are not



(a) An example of a WFST E.



(b) An example of a term "CAT" encoded into Q transducer.



(c) Result of a composition $R = E \circ Q$.

Figure 7.1: Examples of R, E and Q transducers generated from lattices.

perfectly clean (noise, mismatched training/evaluation data, etc.). It can be overcome by using phone lattices which preserve also hypotheses with lower probabilities not included in the one-best string output.

Our phone recognizer generated tri-state phone posteriorgrams, which we converted to lattices by the HVite decoder[3] [Young et al., 2006]. A certain amount of information from the posteriorgrams is lost during such conversion caused by pruning in the decoder. Because of the loss, query-by-example systems based on phone posteriorgrams should perform better than the ones based on pruned lattices. However, the pruning factor and thus the amount of lost information can be tuned arbitrarily to keep only necessary information in lattices (see Section 7.5.1).

Both query and evaluation lattices are converted to a WFST representation. By a composition of query and evaluation transducers we obtain a transducer R which represents all detections of the query in the evaluation data, [Allauzen et al., 2007]:

$$R = E \circ Q, \tag{7.4}$$

where E is a transducer for evaluation data, and Q represents the query. We present some examples of transducers in Figures 7.1a, 7.1b, and 7.1c. Let us now describe the process of converting lattices to E and Q transducers in more detail.

7.2.1 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

[3]Included in the Hidden Markov Model Toolkit (HTK), `http://htk.eng.cam.ac.uk`

Algorithm 7.1 A detailed algorithm for creating the transducer E from a traditional lattice.

1. Each link $L_{ij} : N_i \to N_j$ 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: $L_{in} : N_{first-1} \to N_i$ with $weight(L_{in}) = \alpha(N_i)$ (forward likelihood to node i) and $L_{out} : N_i \to N_{last+1}$ (N_{last} is the lattice's final node) with $weight(L_{out}) = \beta(N_i)$ (backward likelihood to node i). The labels of both links carry the time of the node i (e.g., "t=14.23").

4. A new node $N_{first-2}$ and a new link $L_{norm} : N_{first-2} \to N_{first-1}$ 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.

each possible path through an evaluation WFST E starts and ends 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 7.1a. 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. See Algorithm 7.1 for details of creating the transducer E.

Figure 7.1a 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:

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

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

$$p(path) = \frac{\alpha(start(L_1))l(L_1)l(L_2)...l(L_M)\beta(end(L_M))}{\alpha(N_{last})} \tag{7.5}$$

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

Algorithm 7.2 A detailed algorithm for creating the transducer Q from a traditional lattice.

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.

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].

7.2.2 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 (query-by-text WFST STD), we can simply create a chain of phones and add the time-consuming labels and keyword label as shown in Figure 7.1b. 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 Algorithm 7.2.

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.

7.2.3 Composing WFSTs and extracting detections

From the composition in (7.4), we obtain a transducer R, in which all paths represent detections of Q in E. The result R of composing E in Figure 7.1a with Q in Figure 7.1b is shown in Figure 7.1c. All paths in R 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 of R must then be traversed and overlapping detections of the same term must be merged.

The resulting transducer R is much more complex for real data than the one shown in Figure 7.1c. In the case when the query transducer Q is created from a lattice cut, the R transducer usually contains lots of parallel detections. A more realistic example is shown in Figure 7.2.

To avoid recursive traversal of all paths in the R transducer, we used dynamic programming. Nodes of the transducer are topologically sorted, so for each node, we just forward detection data to all node's successors, and then for each node, we have data about all paths leading to that node. Detection data contain start time of the detection, individual phone labels and collected likelihood. For nodes tagged as a detection end, time label is read from the following link together with the likelihood till the end of utterance. Then, a new detection instance is created from the data propagated into the detection end node, and this detection is added to detections pool. In the pool, overlapping detections are merged together by taking the best path's weight and timing. We also tried summing up weights of the overlapping detections (*logadd* in the *log* domain), but the results were worse and we have yet to further investigate the reason of this deterioration.

7.3 Using Confusion Networks (WFSTcn)

Phone lattices described in the previous section can be further converted to confusion networks (also called sausage lattices because of their sausage-like structure). They have a special property that each path from the start node to the end node goes through all other nodes, while all paths from the original lattice are preserved and some new paths are added. Confusion networks are more efficient than traditional lattices, in terms of size and structure, without compromising recognition accuracy. Since they force the competing phones to be in the same group, they enforce the alignment of the phones that occur at the same approximate time interval in the lattice. Posterior probability of each phone in the group is the sum of probabilities of all paths traversing that phone at around that approximate time frame. Posterior probabilities of all phones in the group sum to one. [Mangu et al., 2000; Hakkani-Tür et al., 2006].

Similarly to lattices, also confusion networks have to be converted to WFST representation. The conversion process is however much simpler, mainly because confusion networks have simpler structure and also links already carry posterior probabilities (not likelihoods).

Again, by a composition of evaluation E and query Q transducers we obtain a transducer R:

$$R = Q \circ E \tag{7.6}$$

Note that the order of Q and E is swapped in the composition when compared to (7.4) due to our particular implementation.

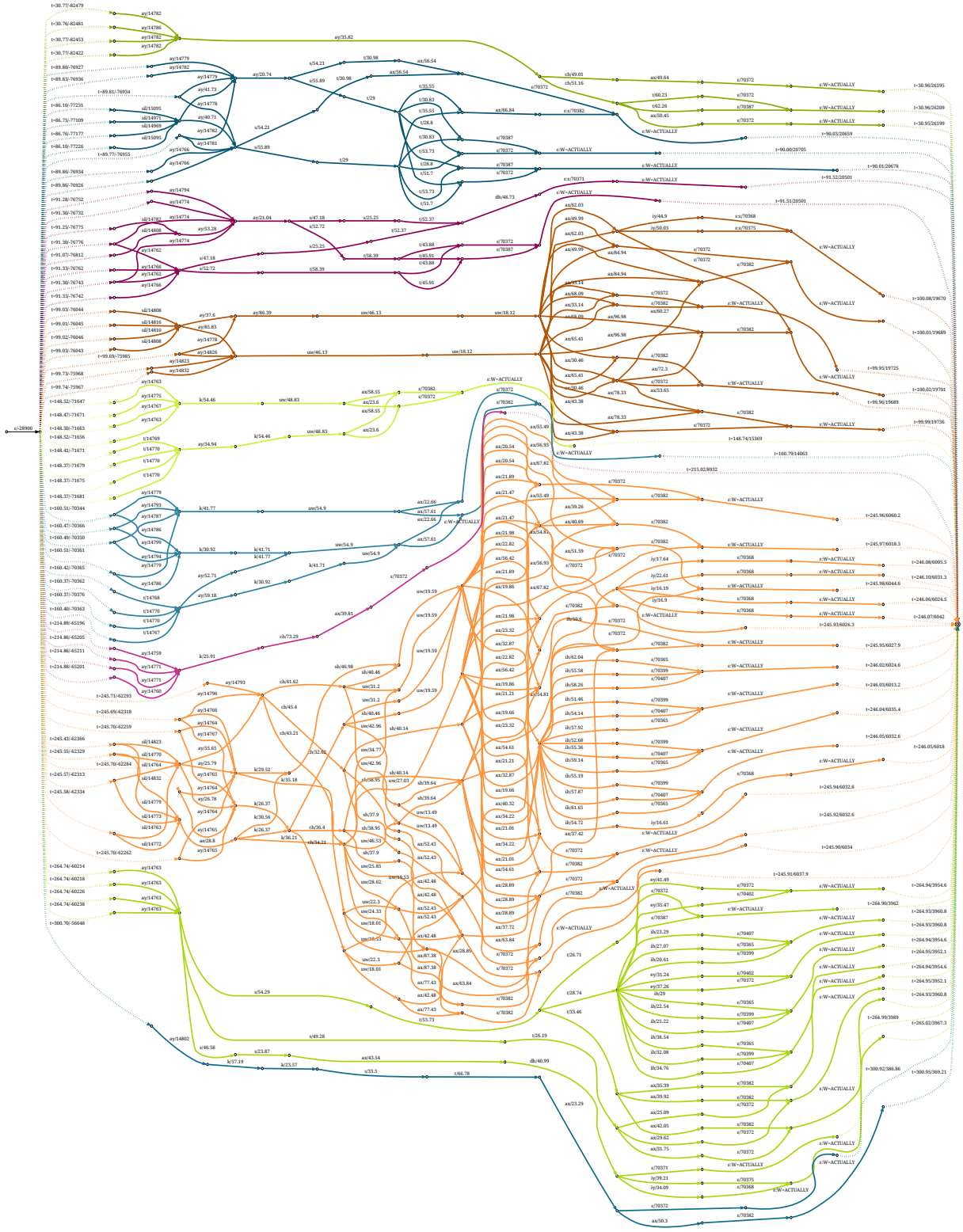


Figure 7.2: Example of the result transducer R for an example of query "ACTUALLY", visualizing complexity of the transducer. Paths overlapping in time share the same color. Start time and end time links are dotted, appearing on the left and right side of the graph. Links right before the end time links represent the query label. All other solid links represent phones.

7.3.1 Converting Evaluation Confusion Networks to WFSTs

Each link is copied from a confusion network to the transducer E, having phone in the input label and start time in the output label. The only exception are ε transitions[4] which have ε in both input and output labels in the WFST. Weight of the link remains its posterior probability from the confusion network.

7.3.2 Preparing Query WFSTs

Query transducers are slightly different from evaluation transducers. The following parts are concatenated in a sequence to form a query transducer:

1. Self-loop with ρ or σ symbols on both input and output label (ρ and σ are equivalent in this case).
2. Helper link denoting the beginning of the query (`TERM_START` in our system).
3. Confusion network converted to transducer similarly as for evaluation confusion networks, but without time labels. In this case, both output and input labels represent the link's phone.
4. Helper link denoting the end of the query (`TERM_END` in our system).
5. Same self-loop as at the beginning.

Beginning and end self-loops consume the part of an evaluation transducer which does not match the query and have a similar role as filler models in the acoustic keyword spotter (models A and C in the Figure 4.1). Helper links denoting start and end of the query are useful for extracting detections after the composition operation (7.6), specifically for assigning start and end time for each detection.

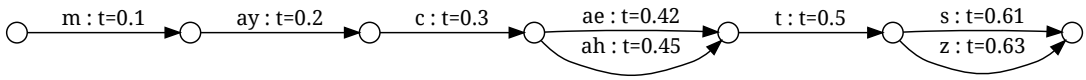
Since there is no query label in the transducer, we can search only for a single query in a single utterance. Query transducers generated from lattices (Section 7.2.2) include also query labels and thus a set of queries can be merged into a single query transducer which is then searched as a whole in a single utterance. However, according to our experiments, composition of these merged query transducers requires significantly more memory than in the case of a single query.

7.3.3 Composing WFSTs and extracting detections

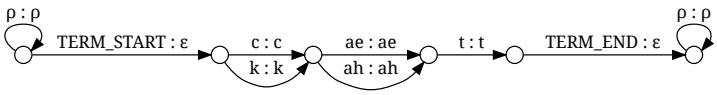
The process of composing WFSTs and extracting detections is similar to WFSTs generated from lattices in the WFSTlat system (Section 7.2). From the composition in Eq. (7.6), we obtain a transducer R, in which all paths represent detections of Q in E. However the paths are a bit different in this case. Time information of all phones in the evaluation WFST is preserved in the output symbols, so we do not only know the start and end time of a detection, but also of all phones the detection contains. So, every path in R consists of:

1. An optional sequence of the filler-model's output containing only ϵ links or ρ or σ links.

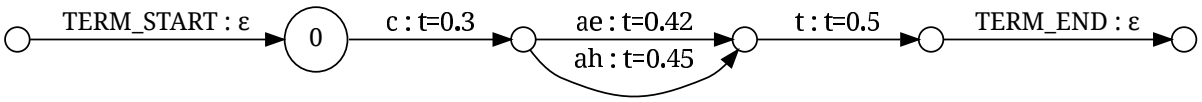
[4] ε transitions are denoted by `*DELETE*` label in confusion networks generated by SRILM's lattice-tool. OpenFST uses `<eps>` for that purpose.



(a) An example of a WFST E.



(b) An example of a term "CAT" encoded into Q transducer.



(c) Result of a composition $R = Q \circ E$.

Figure 7.3: Examples of R, E and Q transducers generated from confusion networks.

2. A link denoting the detection's start (with label `TERM_START` in our system).
3. Links with phones matching both query and evaluation transducers. Each of these phone links has phone symbol in the input label and its start time in the output label. When there are more phones in a sausage (aligned group) matching in both query and evaluation transducers, the sausage structure is preserved also in R. This is helpful for optimization during traversing the R transducer and extracting detections.
4. A link denoting the detection's end (with label `TERM_END` in our system).
5. Again an optional sequence of the filler-model's output.

Traversing, extracting and merging of detections is similar to the case of transducers generated from lattices (Section 7.2.3), with few differences. In the WFSTlat system, time links injected into lattices in the process of creating evaluation transducers, allow the composition algorithm to jump directly to the matching part of evaluation transducer, and thus the R transducer contains only parts representing the query. On the other hand, in the WFSTcn system, the filler ρ or σ-loops at the beginning and end of the query transducer, together with ϵ links added into overlapping groups in confusion networks, make the R transducer significantly larger. Another difference is that for overlapping paths in the WFSTlat system, only the best path is taken into account, while in the WFSTcn system, weights of links in the same sausage (aligned group) are summed up.

The result of composing Q in Figure 7.3b with E in Figure 7.3a is shown in Figure 7.3c.

7.4 Combining Query Examples

In QbE systems and their applications, a query may be represented by more than one example. There are several techniques for dealing with it in WFST-based QbE systems:

Merge posteriorgrams of all examples of a query into one posteriorgram as was described in Section 6.1. The rest of the processing (lattice generation, transducer composition and

detections extraction) remains the same as for a single example per query. We used this approach for the WFSTcn system.

Merge transducers of all examples of a query into one transducer. All examples share the same start node, but if we want to know which detection came from which example, we need to add example's ID to `TERM_START` or `TERM_END` labels. This technique can be also used when the search needs to be performed for a set of queries. In that case, in addition to example ID, query ID needs to be also included in each query label. This would lead to a huge query transducer consisting of many parallel paths. It seems like an elegant solution, but during our experiments we found that a composition with such huge query transducers requires too much memory and is infeasible for our setups.

Merge detections of all examples of a query into one detection when detections overlap. This method can also be denoted as a post-search merging. All query examples are treated as separate single-example queries during all phases of the query-by-example system, till all detections are extracted. Each evaluation utterance is composed with each example of each query separately. Each R transducer is then converted to a list of non-overlapping detections containing the term name, timing, and posterior probability. To merge overlapping detections of all examples of each query, we tested two approaches: we either summed up weights of the overlapping detections or we took only the best weight. In both cases, timing of the best detection was taken. We used this approach for WFSTdict and WFSTlat and experimented with it also for the WFSTcn system.

7.5 Initial Results

In this section, we present initial results of our WFST-based query-by-example STD system which worked with lattices, as was described in Section 7.2. These results were published in our paper [Tejedor et al., 2012].

We present two different WFST-based approaches: WFST based on pronunciation dictionaries (WFSTdict) related to query-by-text STD, serving as our WFST baseline, and WFST based on examples from lattices related to QbE STD (WFSTlat). These two systems differ only in input queries, which are either taken from the pronunciation dictionary as phone strings in case of WFSTdict, or they are cut from speech data in case of WFSTlat.

7.5.1 Lattice Pruning

During the decoding process involved in generating lattices from posteriors, certain amount of information is lost. This amount can be tuned to fit a particular application. Too aggressive pruning produces very narrow lattices with low number of alternative hypotheses. On the other hand, too vague pruning produces very wide lattices which consume a lot of disk space and also a lot of processing power during all operations including search in WFST-based QbE systems.

For our experiments, we generated very wide lattices by the HVite decoder and then we used STK toolkit[5] to prune them to various densities. The pruning factor in STK toolkit depends on both the feature extractor and the target language, so we measured an average number of links per second in speech segments and tuned the pruning factor to produce the desired amount of

[5] `http://speech.fit.vutbr.cz/software/hmm-toolkit-stk`

links per second in lattices. Pruning factors were tuned separately for all combinations of four feature extractors, four target languages and five desired counts of links per second (70, 230, 380, 540 and 700). Performance of these selected lattice densities is evaluated in the following section.

7.5.2 Dictionary Pronunciations

The acoustic keyword spotting STD system described in Chapter 4, works with full posterior features and thus it can serve as a reliable baseline for the DTW and GMM/HMM-based query-by-example STD systems. However, the WFST-based system works with lattices which contain only a subset of the information present in full posterior features. To cancel the effect of the information loss introduced by converting posterior features to phone lattices, we created another baseline which is more comparable to our WFST query-by-example experiments, the query-by-text WFST STD system. Similarly to the AKWS system, it also derives queries from a pronunciation dictionary, leading to a simple phone string representing each query.

Because the system searches in phone lattices, their width is one of important factors affecting the system's performance and can be controlled by posterior pruning (also described in Section 7.6.1). Results for various pruning factors are summarized in Figure 7.4. It shows that, as expected, the performance increases with denser lattices since more information is kept. The system performance saturates with 700 links per second of speech for all the languages. Therefore, we chose the 700 links per second as the configuration for the rest of WFST experiments. This configuration allows us to be sure that enough information is kept in lattices, which is especially needed for language-independent experiments, where the 3-state phone posteriors become blurred. However, as we will see later in this work, ideal lattice densities might be different for language-independent setups.

Results of the two baseline systems, AKWS and WFSTdict (with 700 links per second lattices), are summarized in Figure 7.5 for an easy direct comparison across languages. Similarly to what we have seen in previous chapters, Hungarian presents the best overall performance, and Levantine again shows the worst performance, due to the nondiacritized approach, where we model graphemes instead of phones. We can also see in the figure, that the WFSTdict system performs almost as well as the AKWS system for all languages except English, where the performance dropped significantly. The oracleFOM and unnpFOM for English WFSTdict system are very close to its npFOM which means that the system is very well calibrated across different utterances and almost all hits have higher scores than false alarms. However, generating more detections could possibly improve the performance. This could be achieved with lattices even wider than 700 links per second.

For a deeper insight into the density of lattices, we present the number of unique parallel links per frame in Figure 7.6. It shows that even such a high number of parallel links per second represents only around 30 parallel links per frame, while only around 4 of them have unique labels (the rest of the links differs only in timings), which can be seen approximately also in the posteriorgram example in Figure 7.12d. By inspecting the figure, we can see that for all languages, the total number of parallel links per frame rises linearly with the number of links per second. The only exception are English lattices with 700 links per second, where the numbers of total and unique parallel links per frame dropped. This could also be the reason for the lower performance of WFSTdict on English when compared to the AKWS system.

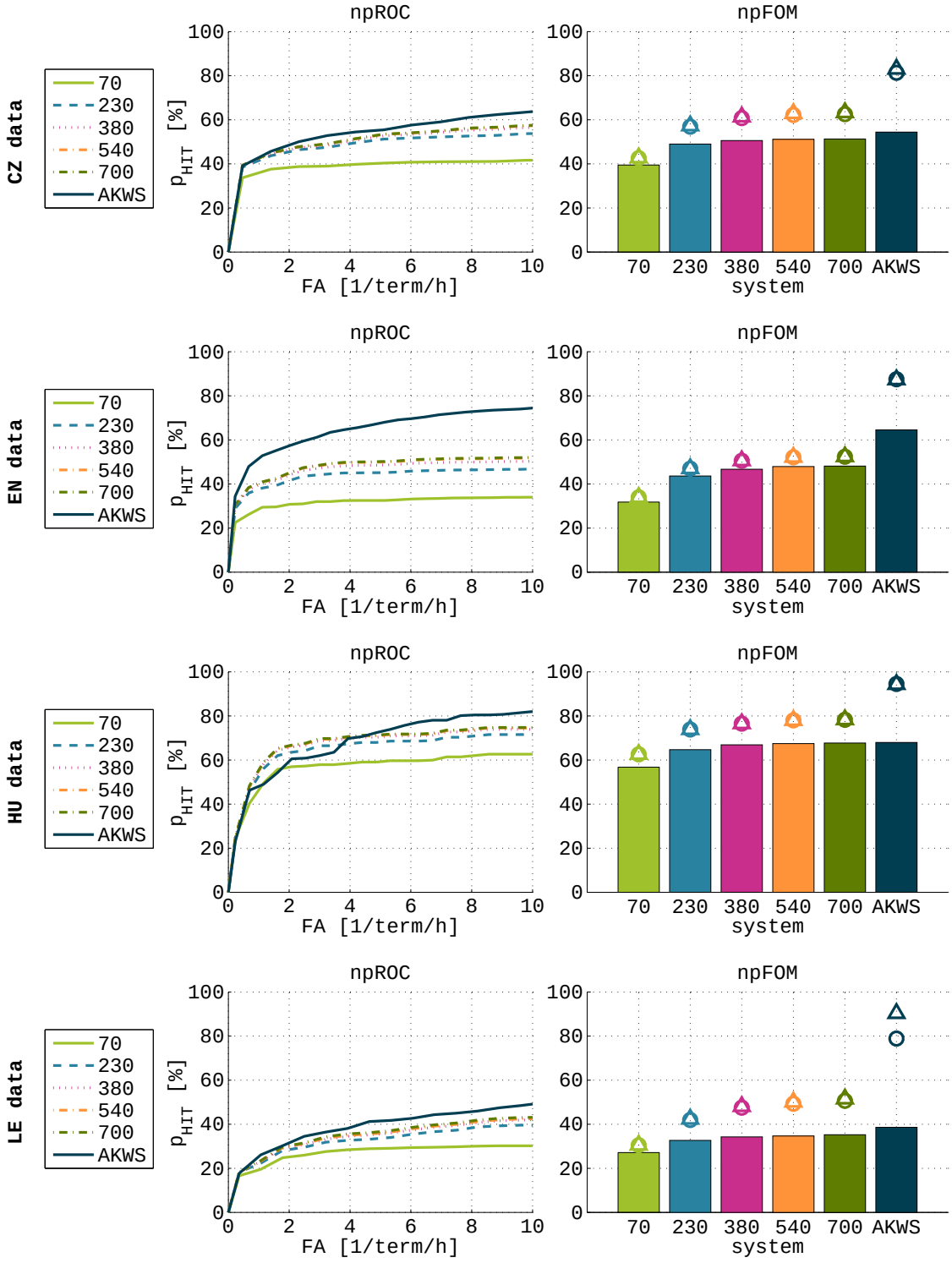


Figure 7.4: Results of the query-by-text WFST QbE STD systems for Czech, English, Hungarian and Levantine datasets. The systems search for *dictionary pronunciations* in lattices. System labels "70", ..."700" represent width of lattices, in the average number of links per second in lattices. The AKWS system is shown as the baseline. For each dataset, only results for the system corresponding to the evaluation data language, are shown.

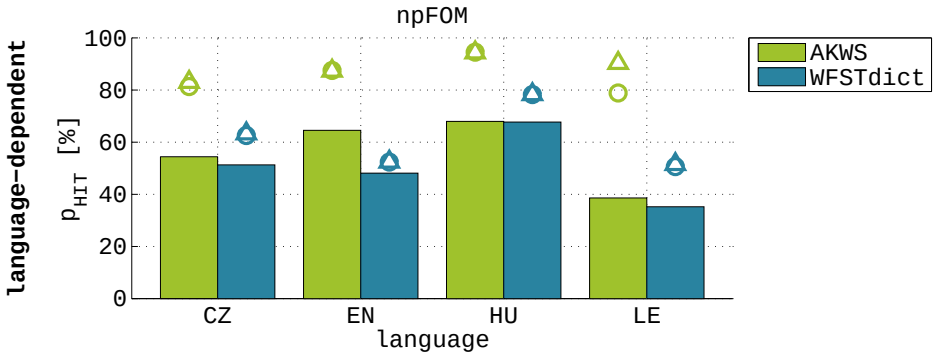


Figure 7.5: Comparison of results of the AKWS and WFSTdict baseline STD systems searching for *dictionary pronunciations* in Czech, English, Hungarian and Levantine datasets. The WFSTdict systems search in lattices with 700 links per second.

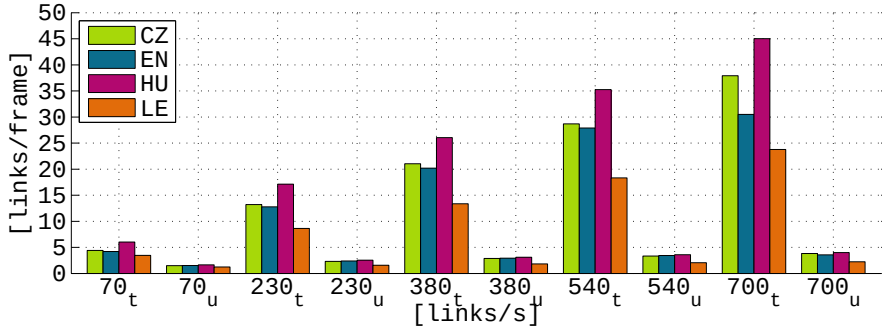


Figure 7.6: Numbers of links (hypotheses) per frame of speech kept in lattices with different densities. Subscript "t" stands for total number of parallel links per frame, subscript "u" for number of parallel links with unique labels per frame. Many parallel links differ only in time alignment, which 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.

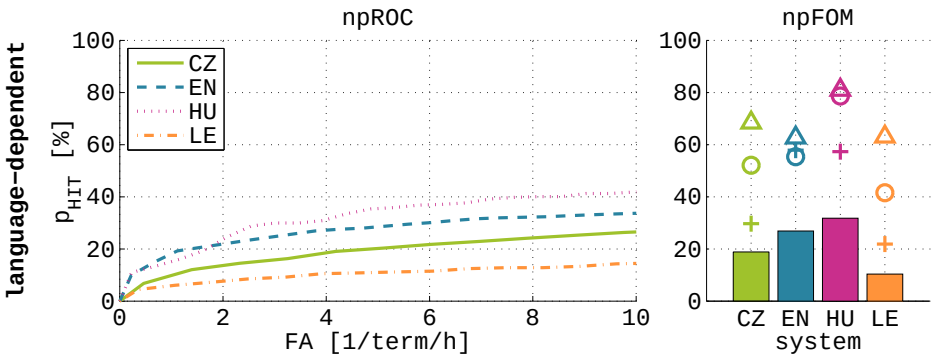


Figure 7.7: Results of WFSTlat systems with single example per query in language-dependent setups.

7.5.3 Examples from Lattices

Our basic WFST query-by-example STD system is very similar to the query-by-text WFSTdict STD system. The only difference is that queries are now represented by lattices cut from utterance lattices generated from audio data, instead of simple phone strings looked up in a pronunciation dictionary.

Results for the language-dependent setup with a single example for each query, are shown in Figure 7.7. It can be seen that example-based queries perform worse than queries extracted from dictionary pronunciations (see Figure 6.5 for comparison). This is due to the inherent advantage of the pronunciation dictionary, which was used for training the feature extractor itself. Also, the amount of variation across examples of a single query is non-negligible and examples taken from lattices are noisy, while the dictionary contains only "clean" data.

When we combine five examples of each query using the post-search "merge detections" strategy, the results improve significantly. However, even for five examples per query, the performance is still lower than that of the WFSTdict baseline. Comparison of the WFSTdict baseline and WFSTlat systems for a single example and five examples per query is shown in Figure 7.8. The English system with five examples per query exhibits the closest performance to the query-by-text WFST baseline. 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.

By comparing npFOM values for combined five examples per query and maximum npFOM values among those five examples ("+" markers in Figure 7.8), we can see that the method for combining examples is suboptimal for all languages except Czech, where five combined examples perform very close to a single best example. In the comparison figure, we can also see that for English and the single example case, when we take only the best example of the five for each query ("+" marker), npFOM result is even higher than for the WFSTdict system, which is a very promising observation. This might be caused by some queries pronounced a bit differently than in the pronunciation dictionary, and it shows that QbE systems could be complementary to query-by-text STD systems in some cases.

All language-dependent and language-independent results for searching five examples per query with WFSTlat QbE STD systems are shown in Figure 7.9. We can see a considerable degradation of performance for language-independent setups when compared to language-dependent

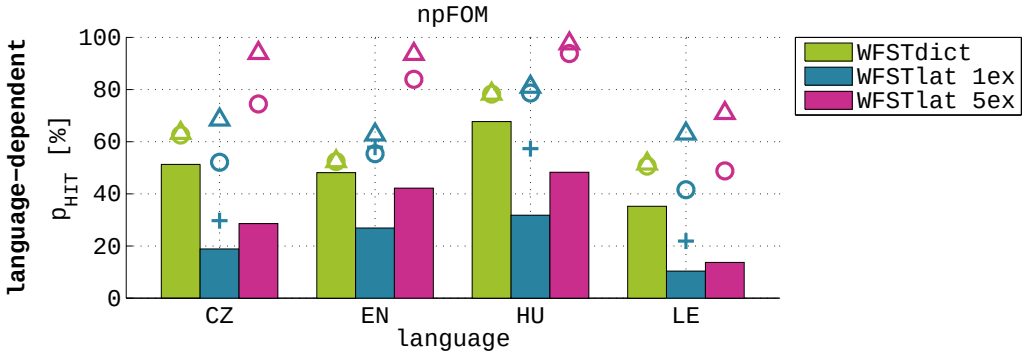


Figure 7.8: Comparison of results of query-by-text WFSTdict systems and WFSTlat QbE STD systems with a single example and five examples per query for the language-dependent setups. The systems search in lattices with 700 links per second. For WFSTlat systems with a single example per query, the "+" marker shows npFOM of the best example among those five.

setups, especially for Czech and English data. However, for the clean speech in Hungarian data, the degradation is less significant. For Levantine data, all systems achieve quite poor performance due to the nondiacritized approach described earlier, which results in more errors, even when the feature extractor matches the target language.

By inspecting unnpFOM values (circle markers in the figure), we can see that for Hungarian data, all systems perform well when only a single evaluation utterance is considered. According to this metric, Czech system works well across all evaluation languages. This is probably caused by mixed audio conditions of training data for the Czech feature extractor which make it more robust.

7.6 Analysis

Comparison of the WFSTlat and DTW-based QbE systems for language-dependent setups is shown in Figure 7.10. We can see that DTW systems significantly outperform WFSTlat systems for both a single example and five examples per query. The performance gap for the single example case is the main issue of the WFSTlat system. The case of five combined examples is influenced also by different combination strategies used for the two QbE systems, since the DTW system combines posteriorgrams of five examples to create a single posteriorgram representing the query, and WFSTlat system searches for each of the five examples and combines the detections afterwards.

In this section, we analyze possible causes of the performance gap between the DTW and WFSTlat systems.

7.6.1 From Posteriorgrams to Lattices and Back

During the conversion from phone-state posteriorgrams to lattices, certain amount of information is lost. In Section 7.5.2 we already compared the performance of the AKWS and WFSTdict systems, where the former one works with full posteriorgrams and the second one with phone lattices, but queries for these systems were simple phone strings taken from a pronunciation

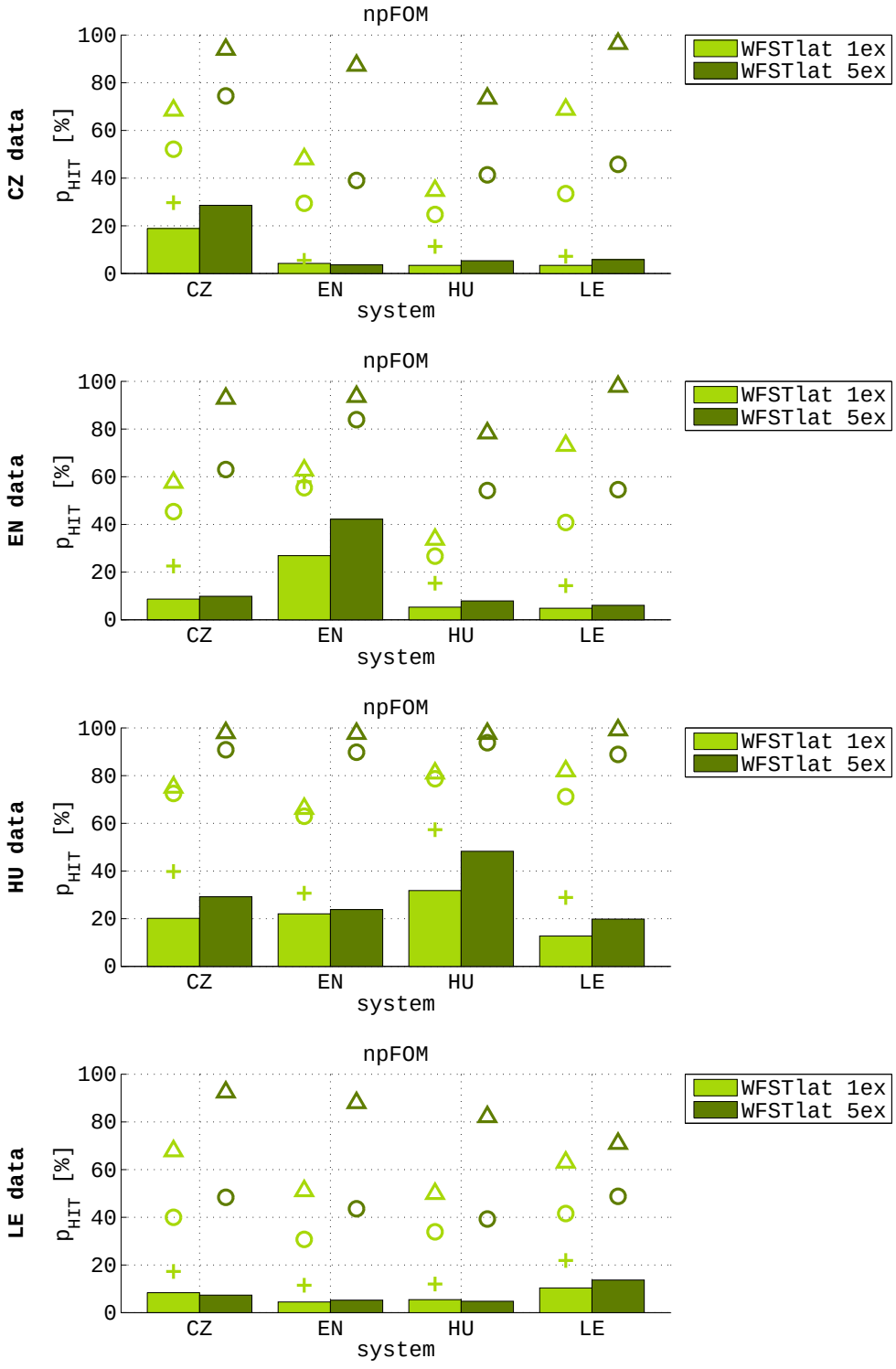


Figure 7.9: Comparison of results of the lattice-based WFST QbE STD systems for a single and five examples per query in lattices with 700 links per second.

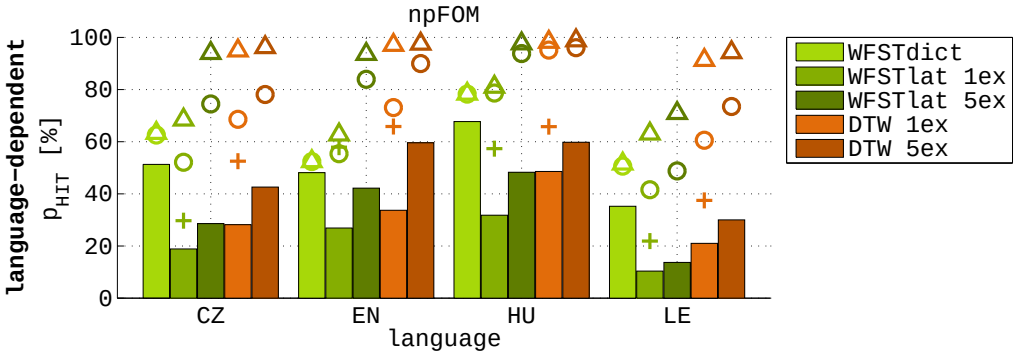


Figure 7.10: Comparison of results of query-by-text WFSTdict systems and WFSTlat QbE STD systems with a single example and five examples per query for the language-dependent setups. The systems search in lattices with 700 links per second.

dictionary. In this section, we will analyze the difference between full posteriorgrams and lattices from the perspective of a query-by-example system. We used DTW search over original posteriorgrams and over posteriorgrams generated from lattices to analyze the information loss introduced by converting posteriorgrams to lattices. This way we can isolate the cause of potential performance degradation only to the information loss introduced by lattices themselves, since the WFSTlat system is not present at all in this experiment. The original full phone-state posteriorgrams (Figure 7.12a) were considered a baseline for this experiment.

7.6.1.1 Posteriorgrams to Lattices

Phone-state posteriorgrams were converted to phone-state lattices using the HVite decoder. We experimented with two types of decoding networks. One where the order of phone states is not constrained, so that any phone state can be active in any frame. And another one where the order of phone states is constrained, so that after the first state of a phone, only the second state of the same phone can follow and switching to another phone is possible only from the last state of the phone.

7.6.1.2 Lattices to Posteriorgrams

Phone-state lattices were used for pruning the original posteriorgrams. We traversed lattices and for each link we copied the corresponding posteriors from the original posteriorgram to a new pruned posteriorgram. This way we created posteriorgrams pruned by lattices. Examples of posteriorgrams generated from an unconstrained and constrained phone-state lattices are shown in Figures 7.12b and 7.12c respectively.

Another set of posteriorgrams was created by computing each link's posterior probability in lattices and adding the probability to a new posteriorgram on the corresponding position. In this case we did not use any direct information from the original posteriorgrams, those new posteriorgrams were generated directly from lattices.

Posteriors with zero probability were raised to slightly above zero to better fit the distance metric of the DTW system.

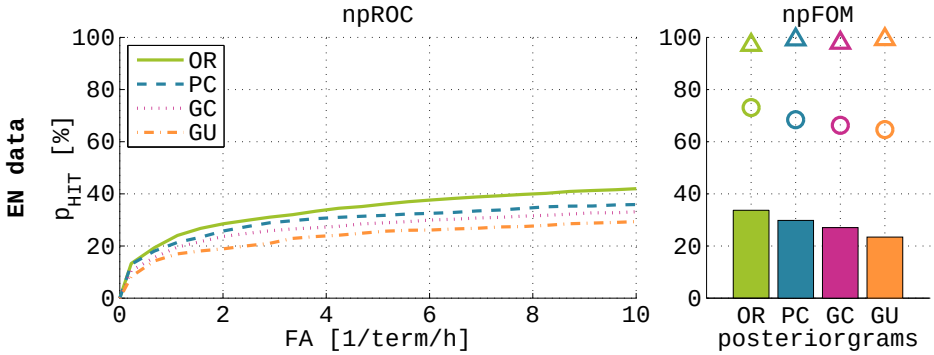


Figure 7.11: Results of DTW-based QbE system with language-dependent setup for English searching for a single example per query in four different versions of posteriorgrams: original full posterior features (OR), posteriorgrams pruned by constrained phone-state lattices (PC), posteriorgrams generated from constrained phone-state lattices (GC), posteriorgrams generated from unconstrained phone-state lattices (GU).

7.6.1.3 Results

We run this experiment only for the English language-dependent setup. Results in Figure 7.11 show that the performance drops for posteriorgrams pruned by constrained phone-state lattices, it drops further when posteriorgrams are directly generated from constrained phone-state lattices, and even further when posteriorgrams are generated from unconstrained phone-state lattices. Thus the loss of information introduced by converting posteriorgrams to lattices is not negligible and we can not expect same results for WFST QbE system as for the DTW QbE system which works with full posteriorgrams.

7.7 Improvements

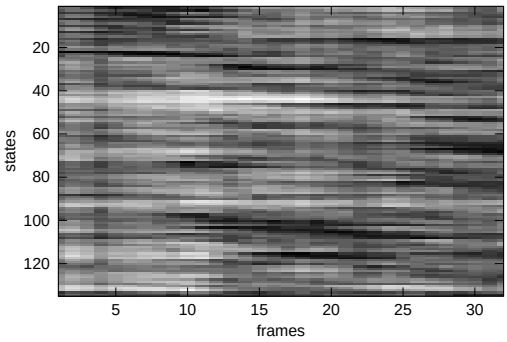
In this section, we describe several improvements over the basic WFSTlat system published in [Tejedor et al., 2012].

7.7.1 Confusion Networks

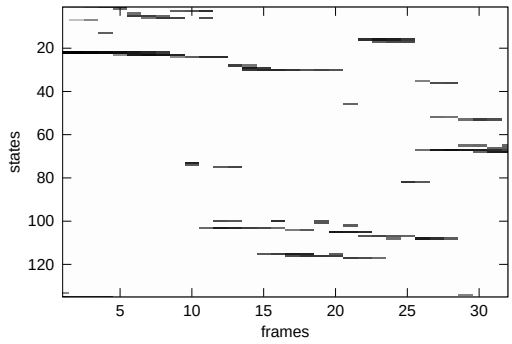
As the WFSTlat system working with lattices performed considerably worse than the DTW system working with full posteriorgrams, we decided to develop and experiment with a confusion network-based WFST system as described in Section 7.3.

7.7.1.1 Single Example per Query

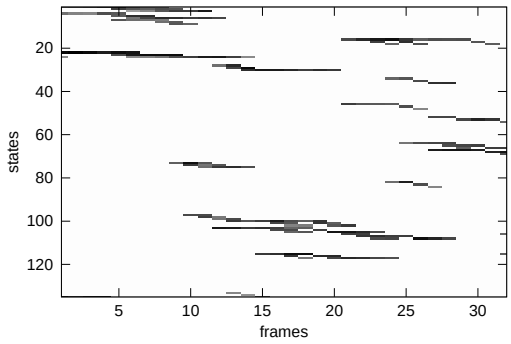
We first consider the basic language-dependent setup where we search only for a single example per query. The results for the confusion network-based WFST QbE system (WFSTcn) together with the WFSTdict, WFSTlat and DTW systems for comparison, are shown in Figure 7.13. We can see in the figure, that the WFSTcn system significantly outperformed the WFSTlat system, and is now on par with the DTW system for Czech and English, slightly better for Hungarian and slightly worse for Levantine. Also, the unnpFOM metric is now very similar for both WFSTcn and DTW systems.



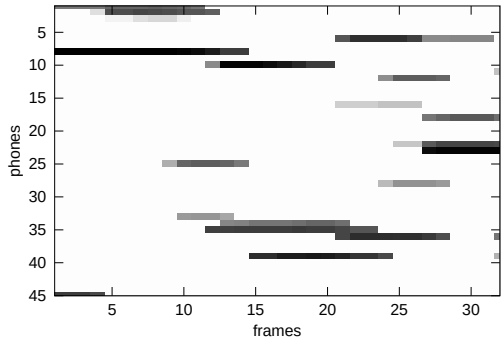
(a) Original posteriorgram.



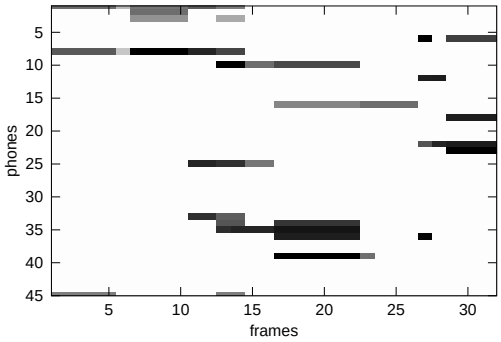
(b) Posteriorgram generated from an unconstrained state lattice.



(c) Posteriorgram generated from a constrained state lattice.



(d) Posteriorgram generated from a phone lattice.



(e) Posteriorgram generated from a phone confusion network.

Figure 7.12: Comparison of posteriorgrams of one example of the term "ACTUALLY". Pruning factor for generating lattices was chosen to emit approximately 700 links per second. These figures visualize the information loss during conversion of posteriorgrams to lattices and further to confusion networks.

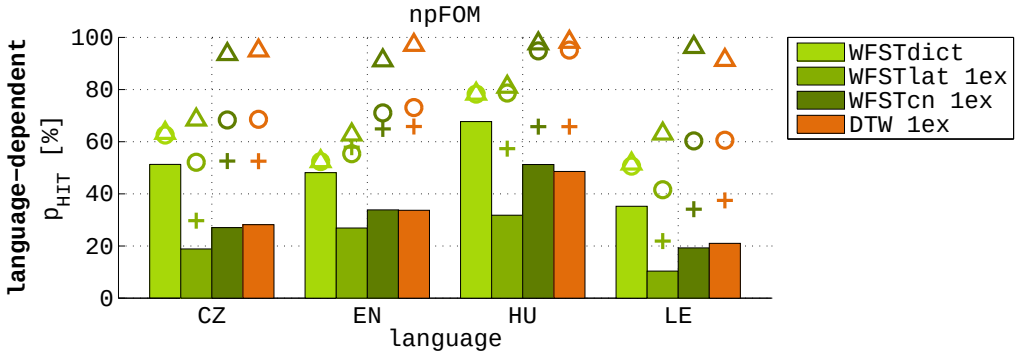


Figure 7.13: Comparison of results of WFSTlat and WFSTcn systems with language-dependent setup searching for a single example per query. WFSTdict

The reason why using confusion networks instead of lattices caused such a significant increase in performance, could be in extracting detections from the R transducer. For lattices, there are significantly more parallel links than for confusion networks, which makes the technique for combining overlapping parallel paths much more important. As shown in Figure 7.2, each detection consists of many overlapping paths. For lattices, only the best path of the overlapping paths is taken as the detection, while for confusion networks, probabilities of overlapping links in the R transducer, belonging to the same time-aligned group (sausage), are summed up, and then the best "sausage-path" is taken as the detection. Thus for confusion networks, many paths contained in the best sausage-path are considered, which seems to improve confidence scores of detections. As we wrote before, the technique used for merging overlapping detections, seems to be an important factor influencing the WFST QbE system's performance. We have yet to investigate techniques for considering all overlapping sausage-paths, which could improve the performance even further. In addition to these reasons, confusion networks were reported to outperform lattices also in query-by-text STD [Mangu et al., 2014].

Let us now analyze the results for language-independent setups, still with only a single example per query. The comparison of WFSTlat, WFSTcn and DTW systems is shown in Figure 7.14, where we can see that for language-independent setups, the WFSTcn system still performs considerably worse than the DTW system. However, we should also note that for the best example case ("+" marker in the figure), for Czech data with Hungarian and Levantine feature extractors, the WFSTcn system performs even better than the DTW system, although the unnpFOM values are still better for the DTW system. Investigating the unnpFOM metric across the whole figure, we can see that the difference between WFSTcn and DTW systems is relatively low, which means that the WFSTcn system is worse calibrated across different evaluation utterances than the DTW system.

7.7.1.2 Combining Examples

As was written in Section 7.4, there are several possible strategies for combining multiple examples per query. In our initial experiments with the WFSTlat system, we used the post-search "merge detections" strategy, where probabilities of overlapping detections were summed up and the timing was taken from the best detection among them. For the WFSTcn system, we experimented also with taking only the single best detection of each overlapping group. Since

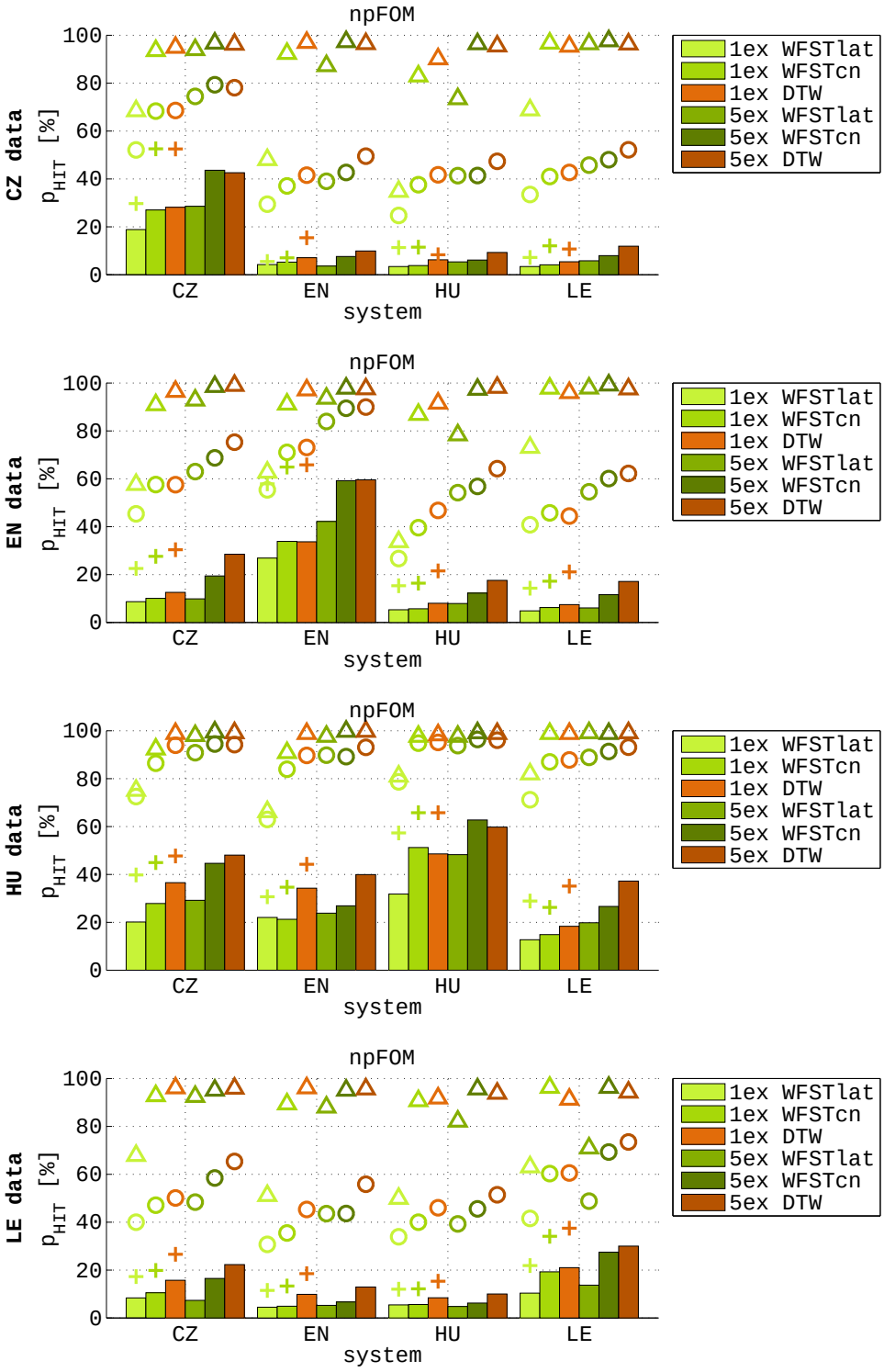


Figure 7.14: Comparison of results of WFSTlat, WFSTcn and DTW QbE STD systems searching for a single example or five examples per query in both language-dependent and language-independent setups.

the DTW system uses the "merge posteriors" strategy, we tried that one also for the WFSTcn system, where we generated query lattices from the merged posteriorgrams. Results for language-dependent setups are shown in Figure 7.15. They reveal that the posteriors merging strategy performs significantly better than the post-search merging of detections.

Both post-search merging strategies seem to perform similarly; only for Levantine, summing up probabilities of overlapping detections seems to outperform the case of taking only a single best detection of the group. It means that for Levantine, more vividly then for the other languages, the more examples agree on the same detection, the better score the detection should have, while also taking into account the actual scores of overlapping detections.

Another important observation revealed in the figure is, that the performance of WFSTlat and WFSTcn systems for the post-search merge detections strategy is similar, which does not correspond to their performance for the single example per query case (Figure 7.13). For Hungarian, the performance of WFSTlat system is even considerably better than that of the WFSTcn system. So the merging strategy seems to hurt the performance the more, the better the system performs for individual examples. However, the reason for this behavior could be of course more complex and has yet to be investigated more deeply.

Also, when we inspect the unnpFOM metric for Czech and Hungarian systems, we see that the post-search merging of detections and the merging of posteriorgrams perform similarly, which means that the main cause of their significant difference in npFOM values is caused by a better calibration of merged posteriorgrams across different utterances.

Comparison of overall results for both a single example and five examples per query cases is shown in Figure 7.14, where five examples for the WFSTlat system are combined by the post-search merging strategy and for the WFSTcn system by merging posteriors. We can see that also for language-independent setups, merging of posteriors outperforms the post-search merging of detections. Although for language-dependent setups with five examples per query, the performance of WFSTcn system is similar to that of the DTW system, for language-independent setups, WFSTcn performs worse. The possible cause of this performance deterioration seems to be again in the technique of merging overlapping detections in the R transducer. In language-independent setups, the number of overlapping detections in the R transducer is higher than in language-dependent setups. This could lead to a larger difference between considering only a single best detection out of an overlapping group of detections, and taking into account all detections in the group. As we wrote before, we tried summing up probabilities of overlapping detections, but the results were significantly worse than for taking only the best detection.

Also, we should note, that the post-search example merging technique, either taking the best path or summing up weights of overlapping paths, is basically the same merging technique as is used in the process of extracting detections from the R transducer, when they overlap in time. As we have seen in Figure 7.15, this merging technique is suboptimal, inferior to merging of posteriorgrams. When the same suboptimal technique is used in the process of extracting detections from the R transducer, we get suboptimal results as well. As we have pointed out already, there are more overlapping detections in language-independent setups, which is probably the reason for their worse performance.

7.7.2 Dealing with Silence

Silence in speech data is handled in our phone recognizers by a *sil* phone model, which is then propagated from posteriorgrams to lattices and confusion networks. It may happen that in

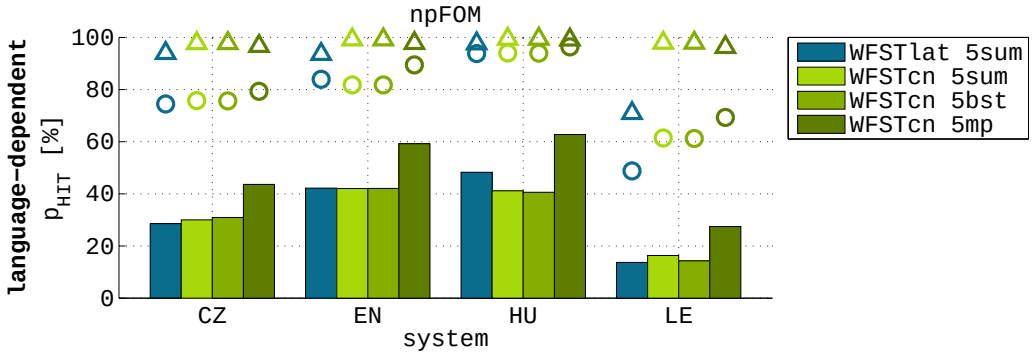


Figure 7.15: Comparison of results of various strategies for combining five examples per query in WFSTlat and WFSTcn systems and language-dependent setups. "5sum" stands for the post-search "merge detections" strategy, where scores of overlapping detections are summed up (*logadd* in our case of log scores), "5bst" selects only the best score among the overlapping detections, and "5mp" stands for the "merge posteriors" strategy where for each query, posteriorgrams of all five examples are first merged into a single posteriorgram, and then the QbE system works as in the single example per query case.

examples cut from audio data, silence appears at the beginning or end of an example. Then the silence is expected to be contained also in detections of that example in evaluation utterances. If time boundaries of our query examples were perfectly correct, there would be no silence at the beginning and end of the examples. We even know apriori that examples of our queries are all single words, so they should not contain any silence at all.

Thus we tried to set zero probability for silence in query posteriorgrams, which effectively discarded silence also from lattices and confusion networks. Results of these examples without silence (further denoted by "NOSIL") were very similar to the original examples ("ORIG"). For language-dependent setups, npFOM improved by 0.66% absolute in average, for language-independent setups by 0.2%. However, when we take best results of both NOSIL and ORIG setups for each query, npFOM improves by another 0.46% for language-dependent and by 0.37% for language-independent setups. It means that some query examples perform better with silence discarded from them, while other examples perform better with silence untouched.

Since the overall results are better for examples without silence, in all our experiments with the WFSTcn system, silence was discarded from all query examples.

Chapter 8

Overall Results and Discussion

In this section we compare and discuss results of our experiments with systems described earlier in Chapters 4, 6, 5 and 7. This chapter was previously written with Igor Szőke from the BUT Speech@FIT research group and Javier Tejedor from the HTCLab at Madrid university, and was published in [Tejedor et al., 2012], but it was completely rewritten in this thesis.

8.1 Language-Dependent

Let us first compare all systems for language-dependent setups in Figure 8.1. For the single example per query case, GMM/HMM system performs the worst, needing more than a single example of a query to reliably train its model. It is outperformed by DTW and WFSTcn systems which achieve very similar overall performance (npFOM, unnpFOM and oracleFOM) for all languages and clearly outperform the GMM/HMM in the single example per query case. Although their npFOM values are still far from the AKWS and WFSTdict baselines, when we artificially select a single best example out of the five examples for each query ("+" markers in the figure), the performance gets on par with baselines. For English, the best example case even outperforms the WFSTdict baseline which is probably caused by insufficient density of English lattices when a query is represented by only a single phone string, or it might be also caused by slightly different pronunciation in the pronunciation dictionary than in actual evaluation occurrences.

By inspecting the systems with five examples per query, we can see that the performance of all systems improved significantly, however for Hungarian, the increase was the lowest among all four languages. This is probably caused by the read prompted speech audio condition in Hungarian data, which results in high similarity of the five examples and their occurrences in evaluation data for each query.

With five examples per query, the GMM/HMM system improved significantly, especially for English and Levantine, where it reached the performance of DTW and WFSTcn systems, for Levantine it even slightly outperformed them. However, for Czech and Hungarian it again performed worse. DTW and WFSTcn systems perform consistently and similarly well for all four languages. Comparing npFOM of the AKWS baseline and the best of the three QbE systems for each language, the deterioration is around 14% in average (20% for CZ data with WFSTcn, 8% for EN data with DTW, 8% for HU data with WFSTcn and 22% for LE data with GMM/HMM).

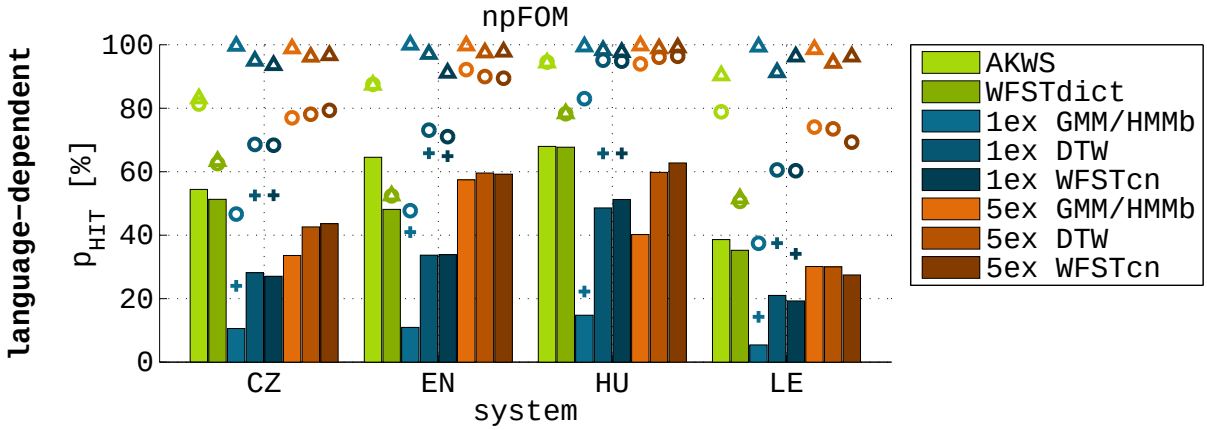


Figure 8.1: Comparison of results of QbE systems in language-dependent setups. The "b" suffix of GMM/HMM system denotes bottleneck features. All other systems use posterior features or confusion networks derived from them. AKWS and WFSTdict baseline systems are in green, GMM/HMM, DTW and WFSTcn for a single example per query are in blue and for five examples per query in brown.

We can conclude that for language-dependent setups, DTW and WFSTcn QbE systems perform better than the GMM/HMM QbE system. The performance of an artificially selected single best example for DTW and WFSTcn systems shows, that the technique for merging examples is still suboptimal and can be potentially improved, which could lead to performance similar to the baseline AKWS system.

8.2 Language-Independent

Overall results for all language-dependent and language-independent setups are shown in Figure 8.2, where we also show the GMM/HMM system working with phone posteriors, for clearer comparability of the three QbE STD approaches working with exactly the same set of features – 3-state phone posteriors. In this section, we focus only on language-independent setups.

For the case of a single example per query, DTW performs the best of the three QbE systems in all language-independent setups. GMM/HMM and WFSTcn systems perform similarly in most cases, only for Hungarian data with Czech feature extractor, WFSTcn performs considerably better than the GMM/HMM system.

Let us now analyze the case of five examples per query in language-independent setups for each system separately.

The GMM/HMMb approach is the most accurate of the three QbE systems, having the best results with the Czech feature extractor. This is caused by large amount of data used to train the feature extractor and also by the mixed audio conditions of Czech data, which makes the system more robust. English feature extractor performs also very well with this system, but although its amount of training data was large, they contained only the CTS audio condition which probably made the system less robust for language-independent setups than the Czech system. We can see, that considering only the GMM/HMMb system, a language-independent feature extractor may achieve comparable results to those obtained with a language-dependent GMM/HMMb system

in case of more challenging conditions, e.g. nondiactitized data in Levantine or low amount of training data for Hungarian. Also, for the GMM/HMM system, bottleneck features were found to effectively compress the important information contained in 3-state phone posteriors. Together with their Gaussian-like distribution, it lead to better trained query models in the GMM/HMM system with five examples per query.

The DTW-based QbE STD system achieves about 50% to 75% of precision of the GMM/HMMb system in language-independent setups, except for Hungarian data, where the DTW system gets closer to the GMM/HMMb. This is caused by the small amount of training data used to train the GMM/HMMb query background model. It 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 the lowest performance of the three QbE systems for language-independent setups, where its performance is even lower than that of the DTW system, similarly to the single example per query case. This is most probably caused by a suboptimal method for merging overlapping detections.

8.3 Combining Systems

Results of a very naive combination of the three QbE systems, where for each query, we select the best result among a set of systems, is shown in Figure 8.3. It is a cheating (oracle) experiment and it shows, that especially for language-dependent setups, all three systems are complementary to some extent, so that there are some queries having best results with one particular QbE system. For language-independent setups, the perfomance gain of this simple way of combining systems is less significant and it is sufficient to combine GMM/HMM system with DTW. Combination of GMM/HMM and WFSTcn systems performs slightly worse for language-independent setups.

According to these preliminary results, it should be possible to fuse the three QbE systems to increase the performance.

8.4 Practical Considerations

In real world applications, the system accuracy may not be the only criterion. We should also take into account the speed of indexing and search and the amount of disk space needed to store the utterances.

The phase of extracting features is the same for all our QbE systems. The GMM/HMM system then has to train or adapt the background model. The WFSTcn system has to convert phone-state posteriorgrams to lattices and then further to confusion networks. However, these steps are relatively fast for all the systems. Where the real difference comes, is in the search speed. However, we have to also note here, that our implementations of the three QbE systems were not specifically optimized for speed.

As we mentioned in Section 2.4.2, there are techniques for speeding up the DTW search, but in its basic implementation, it is certainly the slowest of the three systems. GMM/HMM and WFSTcn systems are, in our implementation, approximately on par in terms of search speed. However, an efficient inverted index can be created for the WFSTcn system to significantly speedup search times [Can and Saraclar, 2011].

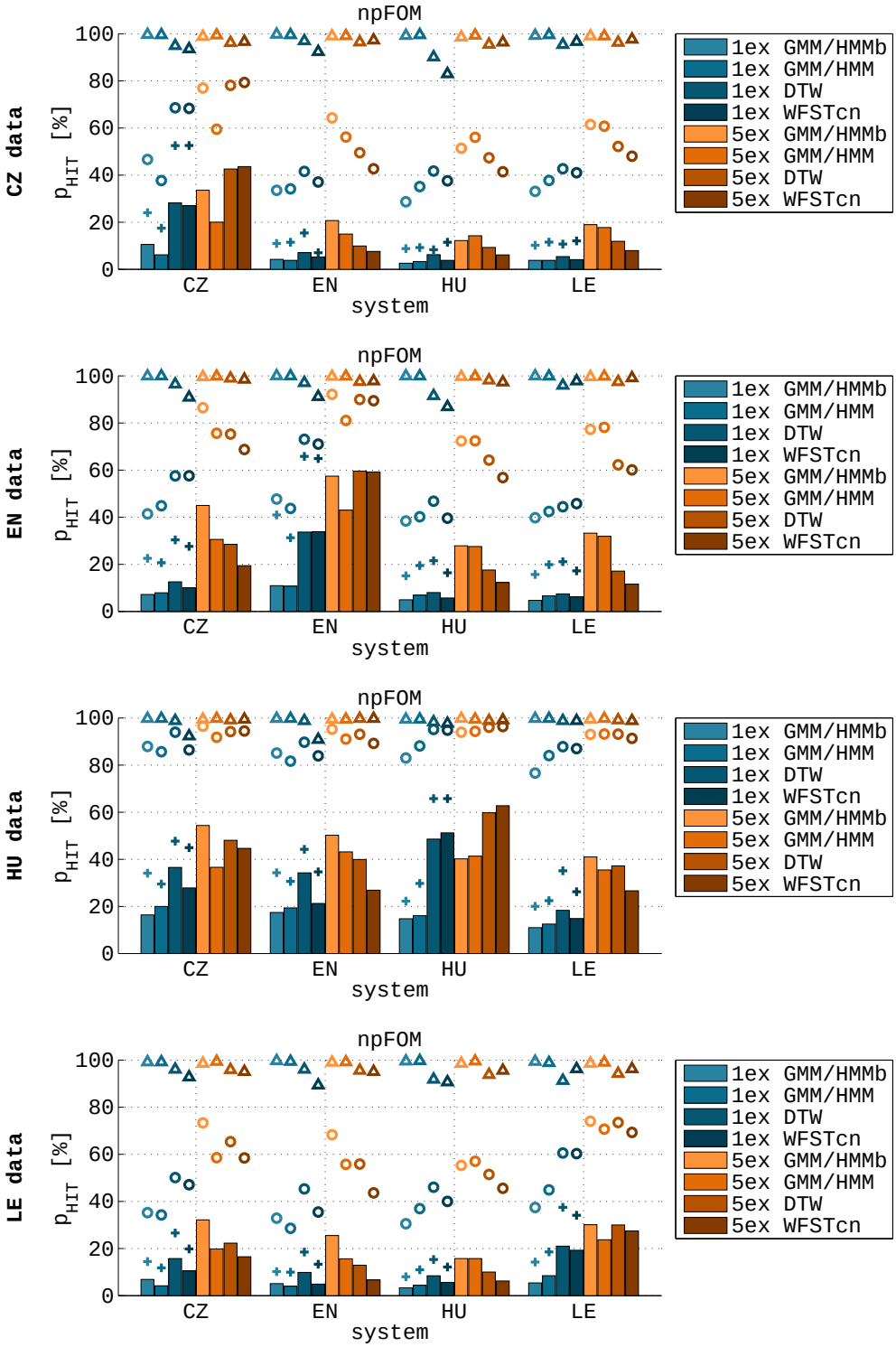


Figure 8.2: Comparison of results of the GMM/HMM, DTW and WFSTcn QbE STD systems in both language-dependent and language-independent setups. The "b" suffix of GMM/HMM system denotes bottleneck features. All other systems use posterior features or confusion networks derived from them. Single example per query cases are in blue and five examples per query in brown.

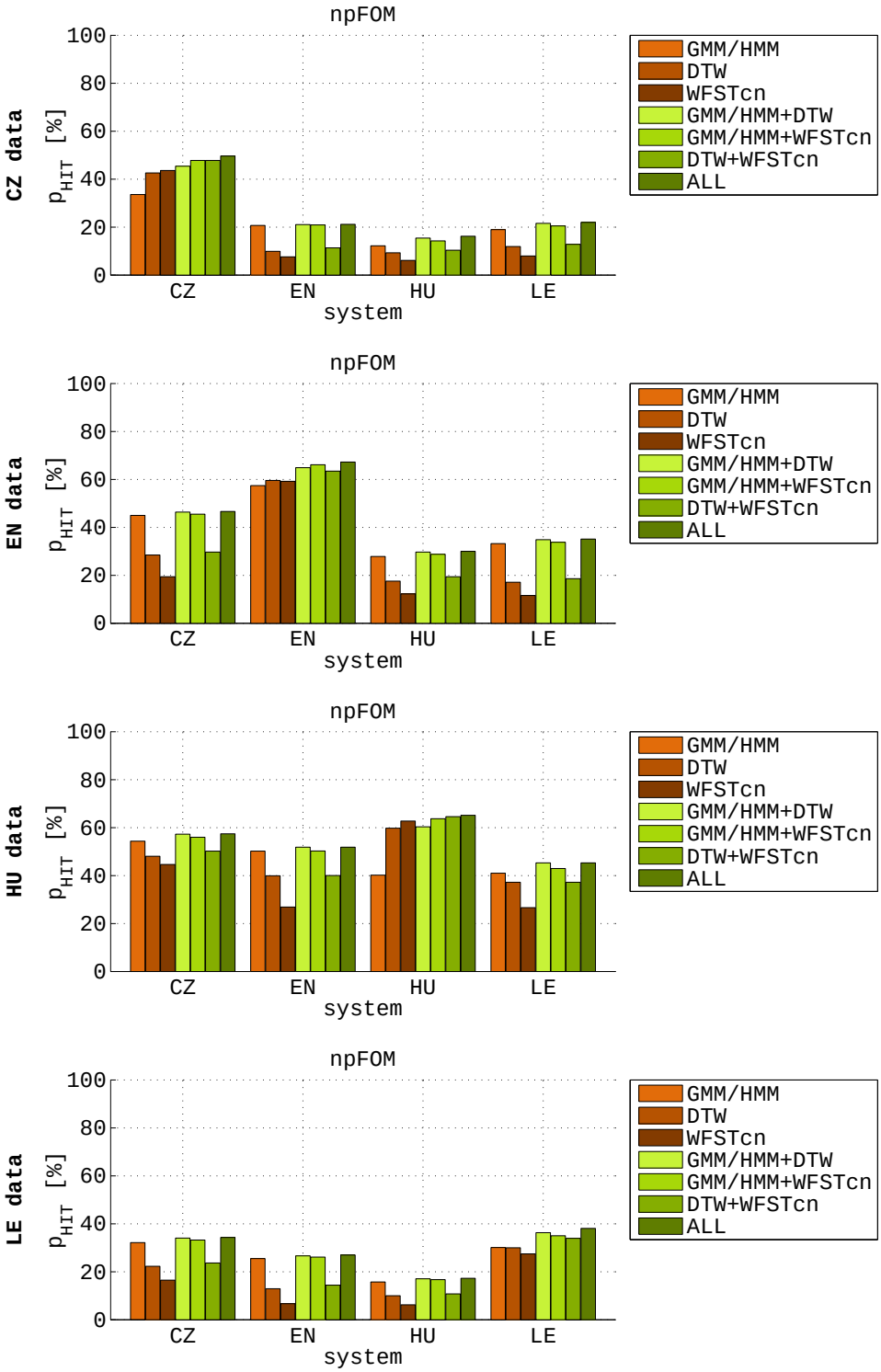


Figure 8.3: Comparison of results of the GMM/HMM, DTW and WFSTcn QbE STD systems in both language-dependent and language-independent setups. Single example per query cases are in blue and five examples per query in brown.

Considering the disk space, DTW system has highest requirements, because 3-state phone posteriorgrams are large themselves. The GMM/HMM system works with bottleneck features which require around 20% of the disk space needed for posteriorgrams. Finite state transducers for the WFSTcn system require even less: 3% of the disk space needed for posteriorgrams. Moreover, the size of transducers can be tuned for a particular application.

Chapter 9

Conclusions and Future Work

In this work, we have presented three query-by-example STD systems and two query-by-text STD baselines. All systems were described, evaluated and analyzed in language-dependent and language-independent setups, for a single example and five examples per query. In our experiments, all STD systems worked with the same set of features, while GMM/HMM worked, besides the common 3-state phone posterior features, also with bottleneck features.

We found out, that for language-dependent setups, for both a single example and five examples per query, DTW and WFSTcn QbE STD systems achieve the best performance. For five examples per query, performance of all three QbE systems increases significantly, although it is still not as good as that of the baseline AKWS system. For clean Hungarian data, WFSTcn slightly outperformed DTW, while for more challenging nondiacritized Levantine data, DTW performed slightly better. For Levantine and English data with five examples per query, also the GMM/HMM bottleneck system caught up the other two systems, but for Czech and Hungarian its performance is still considerably inferior to the other two systems.

For language-independent setups with a single example per query, the DTW system outperforms the other two systems, but with five examples per query, the GMM/HMM bottleneck system improves significantly and outperforms the others. Our experiments also showed that bottleneck features are more sensitive to amount of training data than 3-state phone posterior features.

Based on the presented results, we can conclude that query-by-example STD systems are a viable alternative to query-by-text STD systems, especially when more examples per query are available.

As we expected, language-independent setups show a significant decrease of performance, compared to language-dependent setups. However, there is an exception to this behavior, seen in Levantine data, where the Czech GMM/HMM bottleneck system outperformed even all language-dependent Levantine QbE systems. In general, we can say that the mismatch between training and target languages of a feature extractor, are the main reason for the significant decrease of performance in language-independent setups.

In this work, we have also analyzed and significantly improved our WFST system, which now performs on par with the DTW system in language-dependent setups. Moreover, the WFST system's search is considerably faster and requires only about 3% of the disk space, compared to the DTW system.

9.1 Future Work

During our work, we saw many possible directions for future research, which we were not able to investigate in more depth yet.

- The presented unnpFOM metric showed that the main problem of all our STD systems is in calibration across different utterances. We will investigate various known normalization techniques which could improve the npFOM performance towards the unnpFOM.
- Techniques for merging overlapping detections of the WFST system showed suboptimal performance. We will analyze the applied techniques more deeply to increase the performance of the WFST system in language-independent setups, to match the performance of the DTW system similarly as in language-dependent setups.
- Our WFST system is able to produce various lattice-based features for each detection. According to our preliminary experiments, training a statistical model with these features, has a potential to improve scores of detections and the whole system's performance as well.
- Comparing performance of an artificially selected single best example per query and five merged examples per query, we can see a considerable space for improvement. Thus we will also analyze and investigate methods for combining examples.
- For the GMM/HMM system, bottleneck features performed considerably better than 3-state phone posterior features. They are, however, not directly suitable for DTW and WFST systems. We will explore possibilities to use bottleneck features for all our QbE systems.
- Our preliminary experiments with combination of QbE systems showed a potential for performance improvement, but it has to be analyzed more deeply with real fusion techniques.
- All systems were evaluated with the non-pooled FOM metric. We did not yet pay attention to calibration of scores across queries, nor to providing hard decisions for detections of our QbE systems. These topics will have to be also investigated, as they are needed for many real world applications.

Bibliography

Abad, A., Astudillo, R. F., and Trancoso, I. 2012. The L2F spoken web search system for MediaEval 2012. In *MediaEval*.

Abad, A., Rodríguez-Fuentes, L. J., Penagarikano, M., Varona, A., and Bordel, G. 2013. On the calibration and fusion of heterogeneous spoken term detection systems. In *INTERSPEECH*. 20–24.

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 at HLT-NAACL 2004*. 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 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. 2013. Information retrieval-based dynamic time warping. In *INTERSPEECH*. 1–5.

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. *The Annals of Mathematical Statistics 41,* 1, 164–171.

Burget, L., Černocký, J., Fapšo, M., Karafiát, M., Matějka, P., Schwarz, P., Smrž, P., and Szöke, I. 2006. Indexing and search methods for spoken documents. In Proceedings of the Ninth International Conference on Text, Speech and Dialogue, TSD 2006. *Lecture Notes in Computer Science* 4188, 351–358.

Can, D. and Saraclar, M. 2011. Lattice indexing for spoken term detection. *Audio, Speech, and Language Processing, IEEE Transactions on 19,* 8, 2338–2347.

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

Chan, C.-A. and Lee, L.-S. 2011. Unsupervised hidden markov modeling of spoken queries for spoken term detection without speech recognition. In *INTERSPEECH*. 2141–2144.

Chen, Y.-W., Chen, K.-Y., Wang, H.-M., and Chen, B. 2013. Effective pseudo-relevance feedback for spoken document retrieval. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 8535–8539.

Chia, T. K., Sim, K. C., Li, H., and Ng, H. T. 2010. Statistical lattice-based spoken document retrieval. *ACM Transactions on Information Systems (TOIS) 28,* 1, 2.

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.

Fiscus, J. G., Ajot, J., and Doddingtion, G. 2006. The spoken term detection (STD) 2006 evaluation plan. Tech. rep., National Institute of Standards and Technology (NIST) USA.

Fiscus, J. G., Ajot, J., Garofolo, J. S., and Doddington, G. 2007. Results of the 2006 spoken term detection evaluation. In *Proceedings of Workshop on Searching Spontaneous Conversational Speech (SIGIR-SSCS'07)*.

Grézl, F. and Fousek, P. 2008. Optimizing bottle-neck features for LVCSR. In *Proceedings of ICASSP'08*. 4729–4732.

Grézl, F., Karafiát, M., and Burget, L. 2009. Investigation into bottle-neck features for meeting speech recognition. In *Proceedings of Interspeech'09*. 2947–2950.

Grézl, F., Karafiát, M., Kontár, S., and Černocký, J. 2007. Probabilistic and bottle-neck features for LVCSR of meetings. In *Proceedings of ICASSP'07*. 757–760.

Hakkani-Tür, D., Béchet, F., Riccardi, G., and Tur, G. 2006. Beyond ASR 1-best: Using word confusion networks in spoken language understanding. *Computer Speech & Language 20,* 4, 495–514.

Hazen, T. and Bazzi, I. 2001. A comparison and combination of methods for OOV word detection and word confidence scoring. In *Proceedings of 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.

Helén, M. and Virtanen, T. 2007. Query by example of audio signals using euclidean distance between gaussian mixture models. In *Proceedings of ICASSP'07*. 225–228.

Helén, M. and Virtanen, T. 2010. Audio query by example using similarity measures between probability density functions of features. *EURASIP, Journal on Audio, Speech and Music Processing 2010*, 2:1–2:12.

Iwami, K., Fujii, Y., Yamamoto, K., and Nakagawa, S. 2010. Out-of-vocabulary term detection by n-gram array with distance from continuous syllable recognition results. In *Spoken Language Technology Workshop (SLT), 2010 IEEE*. IEEE, 212–217.

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.

Knill, K. M., Gales, M. J., Ragni, A., and Rath, S. P. 2014. Language independent and unsupervised acoustic models for speech recognition and keyword spotting. *Proc Inter-Speech*.

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., Ramabhadran, B., and Siohan, O. 2007. Vocabulary independent spoken term detection. In *Proceedings of ACM-SIGIR'07*. 615–622.

Mangu, L., Brill, E., and Stolcke, A. 2000. Finding consensus in speech recognition: word error minimization and other applications of confusion networks. *Computer Speech & Language 14,* 4, 373–400.

Mangu, L., Kingsbury, B., Soltau, H., Kuo, H.-K., and Picheny, M. 2014. Efficient spoken term detection using confusion networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 7844–7848.

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. and Anguera, X. 2013. Speed improvements to information retrieval-based dynamic time warping using hierarchical k-means clustering. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 8515–8519.

Mantena, G. V. and Prahallad, K. 2013. IIIT-H SWS 2013: Gaussian posteriorgrams of bottle-neck features for query-by-example spoken term detection. In *MediaEval*.

Metze, F., Anguera, X., Barnard, E., Davel, M., and Gravier, G. 2014. Language independent search in MediaEval's spoken web search task. *Computer Speech & Language 28,* 5, 1066–1082.

Mohri, M., Pereira, F., and Riley, M. 2008. Speech recognition with weighted finite-state transducers. In *Springer Handbook of Speech Processing*. Springer, 559–584.

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"*.

Norouzian, A. and Rose, R. 2014. An approach for efficient open vocabulary spoken term detection. *Speech Communication 57*, 50–62.

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. *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*, 404–409.

Parada, C., Sethy, A., and Ramabhadran, B. 2010. Balancing false alarms and hits in spoken term detection. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE, 5286–5289.

Park, A. S. and Glass, J. R. 2008. Unsupervised pattern discovery in speech. *IEEE Transactions on Audio, Speech and Language Processing 16,* 1, 186–197.

Pereira, F., Riley, M., and Sproat, R. 1994. Weighted rational transductions and their application to human language processing. *Proceedings of the workshop on Human Language Technology - HLT '94* 2, 262.

Pinto, J., Szöke, I., Prasanna, S., and Heřmanský, H. 2008. Fast approximate spoken term detection from sequence of phonemes. In *Proceedings of the 31st Annual International ACM SIGIR Conference*. 28–33.

Salamon, J., Serra, J., and Gómez, E. 2013. Tonal representations for music retrieval: from version identification to query-by-humming. *International Journal of Multimedia Information Retrieval 2,* 1, 45–58.

Schmalenstroeer, J., Bartek, M., and Haeb-Umbach, R. 2011. Unsupervised learning of acoustic events using dynamic time warping and hierarchical k-means++ clustering. In *INTERSPEECH*. 305–308.

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. thesis, 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.

Srikanth, M. 2013. Speaker verification and keyword spotting systems for forensic applications. Ph.D. thesis, Indian Institute of Technology Madras.

Stoimenov, E. and Schultz, T. 2009. A multiplatform speech recognition decoder based on weighted finite-state transducers. *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*, 293–298.

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

Szöke, I., Burget, L., Grézl, F., Černocký, J., and Ondel, L. 2014. Calibration and fusion of query-by-example systems – BUT SWS 2013. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 7849–7853.

Szöke, I., Fapšo, M., Karafiát, M., Burget, L., Grézl, F., Schwarz, P., Glembek, O., Matějka, P., Kopecký, J., et al. 2008. Spoken term detection system based on combination of LVCSR and phonetic search. In *Machine Learning for Multimodal Interaction*. Springer, 237–247.

Szöke, I., Fapšo, M., and Veselý, K. 2012. BUT2012 approaches for spoken web search-MediaEval 2012. In *MediaEval*. Citeseer.

Szöke, I., Grézl, F., Černocký, J., and Fapšo, M. 2010. Acoustic keyword spotter - optimization from end-user perspective. In *Proceedings of the SLT'10*. 177–181.

Szöke, I., Schwarz, P., Matějka, P., Burget, L., Karafiát, M., Fapšo, M., and Černocký, J. 2005. Comparison of keyword spotting approaches for informal continuous speech. In *Proceedings of Interspeech'05*. 633–636.

Szöke, I., Tejedor, J., Fapšo, M., and Colás, J. 2011. BUT-HCTLab 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. thesis, Universidad Autónoma de Madrid, Madrid, Spain.

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, 18:1–18:34.

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

Tsai, W.-H. and Wang, H.-M. 2004. A query-by-example framework to retrieve music 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.

Velivelli, A., Zhai, C., and Huang, T. S. 2003. Audio segment retrieval using a synthesized HMM. In *Proceedings of the ACM SIGIR workshop on multimedia information retrieval, Toronto, Canada*.

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.

Wallace, R., Vogt, R., and Sridharan, S. 2007. A phonetic search approach to the 2006 NIST spoken term detection evaluation. In *Proceedings of Interspeech'07*. 2385–2388.

Wegmann, S., Faria, A., Janin, A., Riedhammer, K., and Morgan, N. 2013. The tao of ATWV: Probing the mysteries of keyword search performance. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 192–197.

Welling, L., Kanthak, S., and Ney, H. 1999. Improved methods for vocal tract normalization. In *Proceedings of ICASSP'99*. 761–764.

Xin, L. and Wang, B. 2001. Utterance verification for spontaneous mandarin speech keyword spotting. In *Proceedings of International Conference on Info-tec and Info-net*. 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.

Zhang, Y. and Glass, J. R. 2011. A piecewise aggregate approximation lower-bound estimate for posteriorgram-based dynamic time warping. In *INTERSPEECH*. 1909–1912.

Appendices

A Implementation and Tools

All our tools and relevant scripts are freely available at `https://michalfapso.github.io`.

A.1 WFST QbE

For working with transducers, we use our own tools based on the OpenFst toolkit [Allauzen et al., 2007]. For converting lattices to confusion networks, we use lattice-tool from the SRILM toolkit[1].

A.2 Scoring and Evaluation

HTK (Hidden markov modelling toolkit[2]) with its HResults tool can be used for evaluating the npFOM metric. Besides providing the overall npFOM value and the ROC curve, HResults outputs FOM also for each query independently together with the number of hits, false alarms and reference occurrences. These query-based statistics are very useful for debugging STD systems. We only had to make a minor change to HResults to accept the total duration of test data as a command line argument, because our reference MLF files only contain occurrences of our selected queries, so HResults is not able to get the correct total duration from the reference MLF. However, for further analysis and higher flexibility, we decided to use our own scoring tool from the LSE toolkit.

LSE (Lattice search engine[3]) is a toolkit developed by the BUT Speech@FIT research group which includes a tool *kws_scorer* for scoring STD systems. Besides computing npFOM, it can output the ROC curve and supports several methods for detecting hits:

- Middle of a reference overlaps with a detection
- Middle of a detection overlaps with a reference
- Both above (at least half of the longer one of the reference and detection overlap with the shorter one)
- Detection overlaps with the reference enlarged by 0.5s to both directions

Its scoring output is similar to HResults, adding several new columns for each query:

- $npFOM_{contrib}(Q)$
- $npFOM_{oracle}$

The tool can be also run in a mode simulating an ideal calibration across evaluation utterances, for computing the $unppFOM$ metric.

B Set of queries per language

Tables B.1 to B.5 show the word transcription of each query along with its number of phones, the average time length per query (in hundredths of seconds), and the number of occurrences in the evaluation data.

[1] The SRI Language Modeling Toolkit, `http://www.speech.sri.com/projects/srilm/`
[2] HTK is available at `http://htk.eng.cam.ac.uk/download.shtml`.
[3] LSE is freely available at `https://bitbucket.org/michalfapso/lse`

Word (#ph) (dur)	#occ	Word (#ph) (dur)	#occ
akorát (6) (36.6)	18	budeme (6) (33.4)	7
budeš (5) (21.6)	13	budou (5) (32.4)	7
dneska (6) (35.2)	21	dobrý (5) (43.8)	60
dobře (5) (48.4)	51	hlavně (6) (48.4)	12
jasně (5) (71.4)	43	jasný (5) (68.0)	15
jedna (5) (29.0)	8	jedno (5) (33.2)	13
jenom (5) (29.8)	36	jestli (6) (29.6)	31
ještě (5) (24.8)	50	jinak (5) (35.4)	8
kolik (5) (38.0)	15	možná (5) (31.8)	9
musím (5) (37.2)	26	myslim (6) (30.0)	6
nějakej (7) (41.6)	15	nějaký (6) (37.8)	12
někde (5) (37.6)	19	pátek (5) (52.2)	21
potom (5) (40.0)	25	potřebuju (9) (34.8)	5
právě (6) (28.0)	24	prosim (6) (32.6)	8
prostě (6) (40.4)	21	protože (7) (39.2)	57
sobotu (6) (56.2)	6	stejně (6) (26.2)	4
štyry (5) (30.4)	13	takhle (6) (78.8)	14
takle (5) (28.8)	9	takový (6) (50.6)	7
takže (5) (23.8)	64	teďka (5) (38.6)	16
teďko (5) (27.2)	11	tejden (6) (54.0)	9
třeba (5) (23.4)	19	úplně (5) (31.0)	11
určitě (6) (45.0)	9	uvidím (6) (55.8)	13
včera (5) (42.8)	9	večer (5) (51.2)	18
vlastně (7) (30.2)	14	všechno (6) (44.6)	15
vůbec (5) (31.4)	19	vypadá (6) (56.2)	7
vždycky (7) (33.4)	4	zatim (5) (40.8)	16
zavolám (7) (46.0)	17	zejtra (6) (35.6)	39

Table B.1: Czech Queries. The column "#ph" denotes the number of phones of a query according to the pronunciation dictionary, "dur" is an average length of a query in tens of milliseconds, "#occ" is the number of query occurrences in evaluation data.

Word (#ph) (dur)	#occ	Word (#ph) (dur)	#occ	Word (#ph) (dur)	#occ
really (5) (34.6)	181	because (5) (41.6)	0	pretty (5) (21.8)	40
always (5) (35.2)	37	before (5) (39.6)	31	married (5) (47.4)	23
getting (5) (28.6)	22	trying (5) (24.6)	20	talking (5) (37.8)	17
having (5) (32.4)	16	another (5) (37.0)	14	believe (5) (36.8)	16
making (5) (27.6)	11	months (5) (30.2)	11	outside (5) (44.2)	11
brother (5) (40.6)	15	changed (5) (51.6)	10	looking (5) (34.0)	10
often (5) (35.6)	0	taking (5) (30.2)	10	women (5) (35.8)	10
living (5) (31.2)	9	without (5) (42.8)	9	dollars (5) (37.4)	8
states (5) (36.4)	9	thousand (5) (42.0)	10	travel (5) (49.6)	13
working (5) (36.2)	8	afraid (5) (42.6)	7	during (5) (25.0)	7
kinds (5) (37.6)	7	cooking (5) (38.2)	6	feeling (5) (39.6)	7
marriage (5) (42.0)	7	nothing (5) (42.8)	6	wants (5) (20.6)	6
coming (5) (25.6)	6	doctor (5) (48.8)	5	drink (5) (33.4)	9
longer (5) (43.0)	0	scary (5) (44.6)	5	telling (5) (29.6)	5
trained (5) (43.0)	5	until (5) (44.4)	5	upset (5) (53.0)	5
watching (5) (38.4)	5	woman (5) (29.6)	5	calling (5) (31.2)	4
growing (5) (41.4)	4	scared (5) (37.0)	4	sister (5) (40.4)	8
somehow (5) (57.4)	4	sounds (5) (29.6)	4	stores (5) (39.8)	5
street (5) (36.4)	5	taken (5) (39.6)	4	topic (5) (50.4)	4
totally (5) (47.4)	4	unless (5) (34.2)	4	winter (5) (36.4)	5
something (6) (38.8)	90	anything (6) (42.8)	46	family (6) (58.6)	40
friends (6) (42.2)	39	business (6) (43.2)	23	whatever (6) (38.0)	20
thinking (6) (39.8)	18	airport (6) (42.6)	20	supposed (6) (30.2)	16
together (6) (47.4)	15	almost (6) (34.0)	12	happened (6) (47.2)	12
instead (6) (39.8)	12	someone (6) (37.4)	13	market (6) (39.0)	12
certainly (6) (36.4)	9	myself (6) (54.2)	9	second (6) (37.6)	0
between (6) (52.6)	8	minutes (6) (28.8)	8	nineteen (6) (40.0)	8
texas (6) (52.0)	7	places (6) (43.0)	7	survivor (6) (69.6)	10
whenever (6) (43.2)	7	already (6) (27.0)	6	anymore (6) (47.6)	6
english (6) (41.6)	6	except (6) (26.8)	6	israel (6) (47.8)	6
nobody (6) (37.8)	6	support (6) (45.2)	8	chicago (6) (63.0)	5
against (6) (35.4)	0	anywhere (6) (41.6)	4	extra (6) (31.0)	4
happens (6) (46.6)	4	changes (6) (45.0)	4	imagine (6) (50.2)	6
partner (6) (51.0)	5	system (6) (53.4)	7	terrible (6) (48.4)	4
wanted (6) (29.4)	4	actually (7) (44.0)	50	everything (7) (61.0)	28
parents (7) (39.2)	26	different (7) (38.6)	25	children (7) (48.2)	26
divorced (7) (55.8)	18	especially (7) (45.6)	17	remember (7) (34.2)	16
sometimes (7) (54.0)	15	started (7) (46.6)	14	affected (7) (58.6)	12
basically (7) (53.0)	12	christmas (7) (60.2)	12	situation (7) (67.4)	12
somebody (7) (38.6)	10	anybody (7) (37.0)	8	countries (7) (54.6)	8
holidays (7) (54.6)	8	husband (7) (48.2)	9	italian (7) (63.4)	8
question (7) (44.8)	8	reality (7) (63.0)	8	florida (7) (50.0)	7
involved (7) (49.8)	7	usually (7) (34.6)	7	personally (7) (62.8)	6
recently (7) (44.0)	6	terrorist (7) (50.6)	10	yourself (7) (50.4)	6

Table B.2: English Queries (1/2).

Word (#ph) (dur)	#occ	Word (#ph) (dur)	#occ	Word (#ph) (dur)	#occ
depends (7) (41.6)	5	everyone (7) (48.2)	5	example (7) (57.0)	5
popular (7) (60.4)	5	united (7) (39.6)	5	control (7) (62.2)	6
difference (7) (50.4)	3	graduate (7) (45.6)	0	hundred (7) (27.8)	4
internet (7)(47.0)	4	religious (7) (56.8)	4	society (7) (49.2)	5
spending (7) (36.8)	4	subject (7) (45.2)	0	wonderful (7) (68.0)	4
probably (8) (36.8)	56	government (8) (44.6)	18	exactly (8) (69.8)	15
important (8) (45.8)	12	problems (8) (52.0)	11	american (8) (53.8)	8
september (8) (48.6)	5	successful (8) (67.6)	5	background (8) (56.2)	4
girlfriend (8) (49.0)	4	obviously (8) (61.4)	4	definitely (9) (53.2)	30
understand (9) (59.2)	14	security (9) (53.2)	12	computers (9) (64.6)	9
completely (9) (55.6)	8	expensive (9) (68.6)	6	relationship (9) (69.0)	8
interesting (10) (59.0)	15	california (10) (58.0)	9	necessarily (10) (64.6)	6

Table B.3: English Queries (2/2).

Word (#ph) (dur)	#occ
lehet (5) (38.67)	97
azonban (7) (71.4)	43
mindig (6) (34.6)	36
volna (5) (37.2)	36
férfi (5) (46.6)	37
milyen (5) (28.2)	37
nagyon (5) (35.2)	23
talán (5) (39.0)	28

Table B.4: Hungarian Queries.

Word (#ph) (dur)	#occ	Word (#ph) (dur)	#occ
AlHmd (5) (31.8)	77	mEAky (5) (43.6)	28
AlHAl (5) (36.6)	30	mbArH (5) (33.8)	23
yslmk (5) (47.0)	22	mDbwT (5) (45.0)	21
Elyky (5) (41.4)	23	Erfty (5) (38.2)	20
AlwAHd (6) (32.8)	20	mrHbA (5) (54.0)	19
ElyhA (5) (26.4)	17	mEAhA (5) (31.8)	17
sAmEk (5) (42.0)	14	Almhm (5) (36.0)	13
AxbArk (6) (53.6)	13	bAqwl (5) (33.4)	13
bEdhA (5) (31.4)	14	btErfy (6) (42.2)	14
btHky (5) (39.4)	12	AlEZym (6) (55.2)	10
bAlbyt (6) (40.4)	10	LsmEy (5) (45.0)	10
UbdAF (5) (36.2)	10	EAlbyt (6) (39.0)	9
mmtAz (5) (44.0)	9	TbEAF (5) (36.6)	8
Uhlyn (5) (44.0)	8	wbEdyn (6) (37.8)	10
AyAhA (5) (34.2)	8	bAllyl (5) (41.4)	7
bnrwH (5) (30.2)	7	mbrwk (5) (48.2)	8
qAEdyn (6) (59.0)	7	byqTE (5) (49.2)	6
Endkm (5) (44.8)	7	mEAnA (5) (34.0)	6
msAfr (5) (45.8)	5	mvlAF (5) (38.2)	6
btlAqy (6) (33.6)	6	bUqwl (5) (36.6)	4
bykwn (5) (33.2)	4	AlUmwr (6) (43.4)	3
bAErfc (6) (35.6)	6	bncwf (5) (38.4)	3
dqAyq (5) (37.0)	3	ElynA (5) (34.8)	4
Hbyby (5) (41.6)	3	mEAhn (5) (29.6)	3
TAlEyn (6) (33.2)	3	tqrybAF (7) (47.0)	3
wEcryn (6) (29.6)	3	-	-

Table B.5: Levantine Queries.