SEARCH IN SPEECH FOR PUBLIC SECURITY AND DEFENSE

Jan Černocký, Igor Szöke, Michal Fapšo, Martin Karafiát, Lukáš Burget, Jiří Kopecký, František Grézl, Petr Schwarz, Ondřej Glembek, Ilya Oparin, Pavel Smrž and Pavel Matějka

Speech@FIT group, Department of Computer Graphics and Multimedia
Faculty of Information Technology, Brno University of Technology (FIT BUT)
Božetěchova 2, 61266 Brno, Czech Republic
cernocky@fit.vutbr.cz

ABSTRACT

The paper defines "search in speech" techniques for security and defense and concentrates on spoken term detection (STD). It presents NIST STD 2006 evaluations and discusses the functionality and results of Brno University of Technology (BUT) STD system.

1. INTRODUCTION

Speech is the most important modality in human to human communication. Even in face-to-face communication, it might contain more than 80% of the information and in case of a telephone conversation, this proportion goes up to 100%. Speech is omni-present in many electronic media (land-line and mobile networks, IP telephony, dedicated channels, ...).

Although most of the voice traffic in these media is legitimate, the operators can not avoid their use by persons and groups willing to commit crimes, terrorism and other malicious activities. Processing of voice calls consumes important share of time, budget and efforts of intelligence (both civilian and defense) services, police forces and private investigators. In their work, the problem is usually not to **obtain** the speech, as there are many ways to tap the wire, radio or optical cable communications. The problem is to **efficiently process** thousands of hours of voice communications running in parallel with the aim to deliver results in reasonable time so that they are still useful for an investigation or preventive action.

Typically, speech communications are processed by human experts, but this processing capacity is always limited due to:

- lack of qualified personnel,
- lack of budget,
- insufficient knowledge of foreign languages
- insufficient security clearances

or combination of the above. The problem is especially serious for languages spoken in potentially dangerous regions, for which the lack of native speakers authorized to access sensitive data can impair efficient action against criminal groups.

Automatic speech search techniques can help the security and defense specialists to find the requested information – the "needle in a haystack" – in reasonable time and without extensive human labor. However, they should not be considered almighty – they *are not able* to replace qualified personnel that will always have to do the final analysis and decisions. They *are able* to automate some processing steps and "limit the search space" for humans.

Let us first define the categories of speech search:

- The task of **Large Vocabulary Continuous speech recognition (LVCSR)** is to determine "what was said" or to provide textual transcription of speech. The best performing LVCSR systems are based on complex acoustic models trained on thousands hours of transcribed speech and on language models trained on gigabytes of text. The biggest challenge in LVCSR is processing spontaneous data for which developers lack speech and language resources.
- In some situations, LVCSR is not the best suited to find the information in speech — as it is constrained by recognition vocabulary, it is hard to find rare information such as new proper names (companies, politicians, geographical). In this case, it is necessary to use **phonetic search** that operates not on the output of word recognizer, but rather on oriented graphs containing smaller units – phonemes.
- As important as the contents of speech is the information "who said it" – this is addressed by **speaker recognition**. We speak about **speaker identification** if the task is to choose one out of a set of N speakers, or about **speaker verification**, where it s necessary to confirm the claimed identity of a speaker.
- Last but not least, **language identification** is needed in speech search to be used as search criterion itself or for routing speech segments to appropriate language-dependent recognizer.

The research of our group – Speech@FIT – is closely related to public security and defense through our cooperation with Czech Ministry of Defense. Starting from 2007, it is also part of a a 1.3 M$/year research project "Security-oriented research in information technology" at Faculty of Information Technology of BUT sponsored by Czech Ministry of Education.

Speech@FIT is involved in many aspects of speech search and has recorded success in NIST speaker [2] and language recognition [1] evaluations. However, this paper is devoted to the first two mentioned topics — LVCSR and phonetic search, jointly called **spoken term detection**, mainly in light of first edition of NIST Spoken term detection (STD) evaluations [3].

The paper is organized as follows: section 2 defines the techniques of spoken term detection. Section 3 deals with indexing and search. Section 4 describes the data from 2006 NIST STD evaluations, provides details of BUT system and its results. Section 5 concludes the paper.

2. SPOKEN TERM DETECTION TECHNIQUES

Unlike search in text, where the indexing and search is the only "science", spoken term detection is a more complex process that needs to address the following points:

- conversion of speech to discrete symbols that can be indexed and searched – LVCSR and phoneme recognizers are used. Using phoneme recognizer allows to deal with out-of-vocabulary words (OOVs) that can not be handled by LVCSR.
- accounting for inherent errors of LVCSR and phoneme recognizer – this is usually solved by storing and searching in word or phoneme lattices (Fig. 1) instead of 1-best output.
- determining the confidence of a query – in this work done by evaluating the likelihood ratio between the path with searched keyword(s) and the optimal path in the lattice.
- processing multi-word queries, both quoted (exact sequences of words) and unquoted.
- providing an efficient and fast mechanism to obtain the search results in reasonable time even for huge amounts of data.

2.1. LVCSR-based search

LVCSR lattices (upper panel in Fig. 1) contain nodes carrying word labels and arcs, determining the timing and acoustic (L_a^{lvcsr}) and language model (L_l^{lvcsr}) likelihoods generated by an LVCSR decoder. Usually, each speech record is first broken into segments (by speaker turn or voice activity detector) and each segment is represented by one lattice. The confidence of a keyword KW is given by

$$C^{lvcsr}(KW) = \frac{L_\alpha^{lvcsr}(KW) L^{lvcsr}(KW) L_\beta^{lvcsr}(KW)}{L_{best}^{lvcsr}}, \tag{1}$$

where the $L^{lvcsr}(KW) = L_a^{lvcsr}(KW) L_l^{lvcsr}(KW)$.

The forward likelihood $L_\alpha^{lvcsr}(KW)$ is the likelihood of the best path through lattice from the beginning of lattice to the keyword and the backward likelihood $L_\beta^{lvcsr}(KW)$ is the likelihood of the best path from the keyword to the end of lattice. For node N, these two likelihoods are computed by the standard Viterbi formulae:

$$L_\alpha^{lvcsr}(N) = L_a^{lvcsr}(N) L_l^{lvcsr}(N) \max_{N_P} L_\alpha^{lvcsr}(N_P) \tag{2}$$

$$L_\beta^{lvcsr}(N) = L_a^{lvcsr}(N) L_l^{lvcsr}(N) \max_{N_F} L_\beta^{lvcsr}(N_F) \tag{3}$$

where N_F is a set of nodes directly following node N (nodes N and N_F are connected by an arc) and N_P is a set of nodes directly preceding node N. The algorithm is initialized by setting $L_\alpha^{lvcsr}(first) = 1$ and $L_\beta^{lvcsr}(last) = 1$. The last likelihood we need in Eq. 1: $L_{best}^{lvcsr} = L_\alpha^{lvcsr} = L_\beta^{lvcsr}$ is the likelihood of the most probable path through the lattice.

2.2. Phonetic search

The main problem of LVCSR is the dependence on recognition vocabulary. The phonetic approach overcomes this problem by conversion of query to a string of phonemes and searching this string in a phoneme lattice (lower panel in Fig. 1). The lattice has similar structure as word lattice (section 2.1), but phonemes P populate nodes instead of words.

The confidence of keyword KW consisting of string of phonemes $P_b \dots P_e$ is defined similarly as in Eq. 1 by:

$$C^{phn}(KW) = \frac{L_\alpha^{phn}(P_b) L_\beta^{phn}(P_e) \prod_{P \in P_b \dots P_e} L_a(P)}{L_{best}^{phn}}, \tag{4}$$

where $L_\alpha^{phn}(P_b)$ is the forward Viterbi likelihood from the beginning of lattice to phoneme P_b, the product is the likelihood of the keyword, and $L_\beta^{phn}(P_e)$ is the likelihood from the last phoneme till the end of the lattice. L_{best} is the likelihood of the optimal path.

3. INDEXING AND SEARCH

3.1. LVCSR lattice indexing

The **indexing** of LVCSR lattices is inspired by [4]. It begins with the creation of lexicon which provides a transformation from word to a unique number (ID) and vice versa. Then, a forward index is created storing each hypothesis (the word,

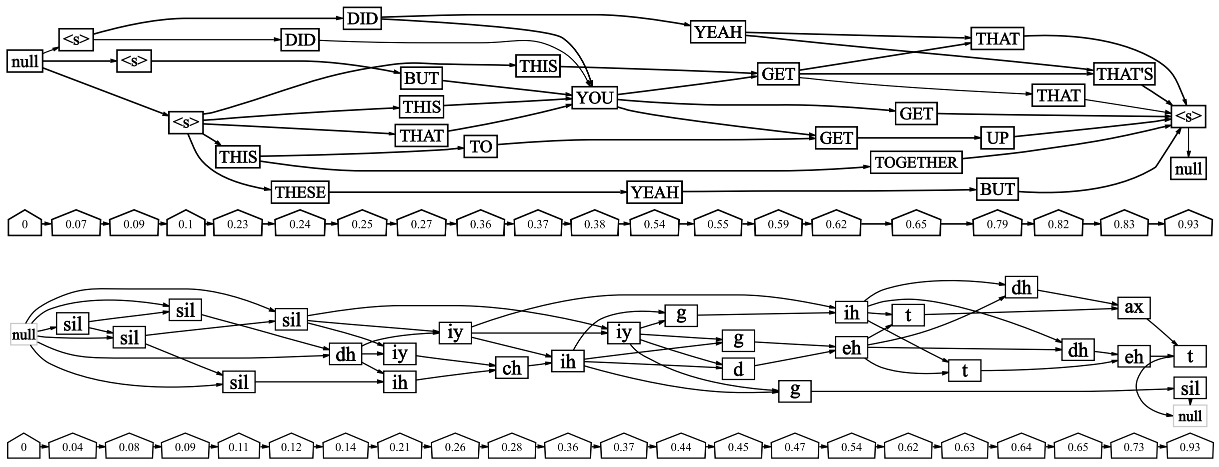

Fig. 1. Example of a word and phoneme lattices

its confidence, time and nodeID in the lattice file) in a hit list. From this index, an inverted index is created (like in text search) which has the same structure as the forward index, but is sorted by words and by confidence of hypotheses. Each speech record is represented by many lattices. The inverted index tells us, in which lattice and at which time the keyword appears.

In the **search** phase, the inverted index is used to find occurrences of words from query. An important feature of our system is the generation of the most probable **context** of the found keyword – a piece of the Viterbi path from the found keyword forward and backward. For all matching occurrences, the searcher therefore loads into the memory a small part of lattice within which the found word occurs. Then, the searcher traverses this part of lattice in forward and backward directions selecting only the best hypotheses; in this way it generates the most probable string which traverses the found word.

3.2. Multi-word queries

A usable system for STD should support queries of type `word1 word2 word3` and `"word1 word2 word3"` with the former one representing finding words in random order with optional spaces in between (in opposite to text-search where we work within a document, we specify a time-context) and the later one representing the exact match. Provided the query Q is found in the lattice, we again need to evaluate its confidence $C^{lvcsr}(Q)$. Similarly to Eq. 1, this is done by evaluating the likelihood of the path with all the words w_i belonging to the query and dividing it by the likelihood of the optimal path:

$$C^{lvcsr}(Q) = \frac{L^{lvcsr}_{rest} \prod_i L^{lvcsr}(w_i)}{L^{lvcsr}_{best}}, \tag{5}$$

where L^{lvcsr}_{rest} is the likelihood of the "Viterbi glue": optimal path from the beginning of the lattice to $w_{earliest}$, connections between words w_i (for unquoted query) and optimal path from w_{latest} to the end of the lattice. In other words, L^{lvcsr}_{rest} represents everything except the searched words. We should note, that each time we deviate the Viterbi path from the best one, we loose some likelihood, so that $C^{lvcsr}(Q)$ is upper-bounded by $\min_i C^{lvcsr}(w_i)$ — actually the confidence of the worst word in the query.

The same index as for single-word queries (keywords) is used. Processing of a query involves the following steps:

1. Based on frequencies of words, the least frequent one from the query, w_{lf}, is taken as first and all its occurrences are retrieved.

2. The search proceeds with other words and verifies if they are within the specified time interval from w_{lf} (for non-quoted queries) or joint to w_{lf} (for quoted ones). The internal memory representation resembles again a lattice. In such way, a candidate list is created.

3. The list is pre-sorted by the upper-bound of query confidence, as described above. The list is then limited to the pre-determined number of candidates (usually 10).

4. For these candidates, the evaluation of correct confidence is done according to Eq. 5. While looking for the "Viterbi glue", the Viterbi algorithm is extended before and after the part of lattice containing Q in order to obtain the left and right contexts.

3.3. Indexing phoneme lattices

While the indexing of word lattices is straightforward, indexing phoneme lattices is more tricky: in advance, we do not

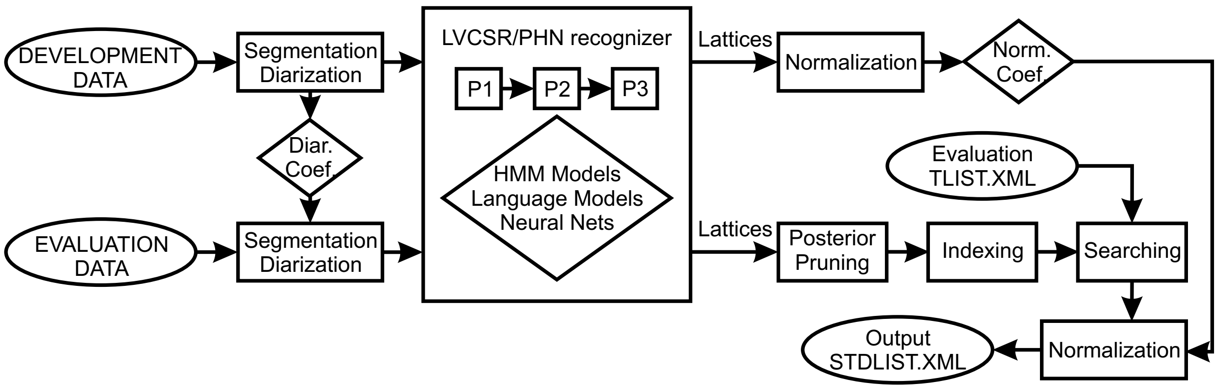

Fig. 2. BUT system for spoken term detection.

know what we will search for. Yu and Seide in [6] and Siohan and Bacchiani in [7] have chosen indexing sequences of phonemes with variable length, we have however investigated a simpler approach making use of overlapping tri-phonemes and indexing similar to multi-word queries. The use of tri-phonemes was also recommended in [5] as the best balance between number of units and number of units' occurrences in a corpus.

In the indexing phase, tri-phonemes T_i are selected in lattices. For each T_i, its confidence is evaluated by Eq. 4 as if T_i was a keyword. In case this confidence is higher than a pre-determined threshold, the tri-phoneme is inserted into the index. The search stage is similar to multi-word quoted queries:

1. The searched keyword generates a set of overlapping tri-phonemes. Based on their frequencies in the index, the least frequent one T_{lf}, is taken as first and all its occurrences are retrieved.

2. The search proceeds with other tri-phonemes and verifies that they form a chain in time (with a security margin between adjacent tri-phonemes). Similarly to multi-word queries, the internal memory representation has again the form of lattice. In such way, a candidate list is created.

3. The confidence of keyword is again upper-bounded by the confidence of the worst tri-phoneme. Based on these, the list is pre-sorted and limited to the pre-determined number of candidates (usually 10).

4. For these candidates, we go into the respective phoneme lattices and evaluate the correct confidence using Eq. 4.

We have verified, that in case no thresholds are applied in the index, we obtain exactly the same accuracy of search that in case phoneme lattices are processed directly.

4. NIST STD EVALUATIONS 2006

The first edition of Spoken term detection evaluation was organized to facilitate research and development of technology for finding short word sequences rapidly and accurately in large heterogeneous audio archives [3]. In this paper, we will deal only with US English conversational telephone speech (CTS) task, as it is the most relevant to security and defense applications.

4.1. Data and metrics

The development data (available for system tuning) consisted of 36 conversations totaling in 3 hours of speech. The evaluation data had also 36 conversations with 3 hours of speech. Development and evaluation term-lists containing 1100 of 1- to 4-word "quoted" queries were provided by NIST. The evaluation term-list contains 411 terms appearing in the CTS data, with 5856 occurrences. There were 5674 occurrences of 1-word terms, 166 occurrences of 2-word terms and only 16 occurrences of 3- and 4-word terms. Examples of terms are:

```
dr. carol lippa
bush's father george bush
thousand kurdish
senator charles
nato chief
every evening
kostunica
audio
okay
```

The main mean for comparison of different systems were detection error trade-off (DET) curves, displaying, for various thresholds θ, the false alarm probability $P_{FA}(\theta)$ on x-axis and miss probability $P_{MISS}(\theta)$ on the y-axis:

$$\begin{aligned} P_{MISS}(\theta) &= \underset{term}{avg}\{1 - N_{correct}(term,\theta)/N_{true}(term)\} \\ P_{FA}(\theta) &= \underset{term}{avg}\{N_{spurious}(term,\theta)/N_{NT}(term)\} \end{aligned}$$

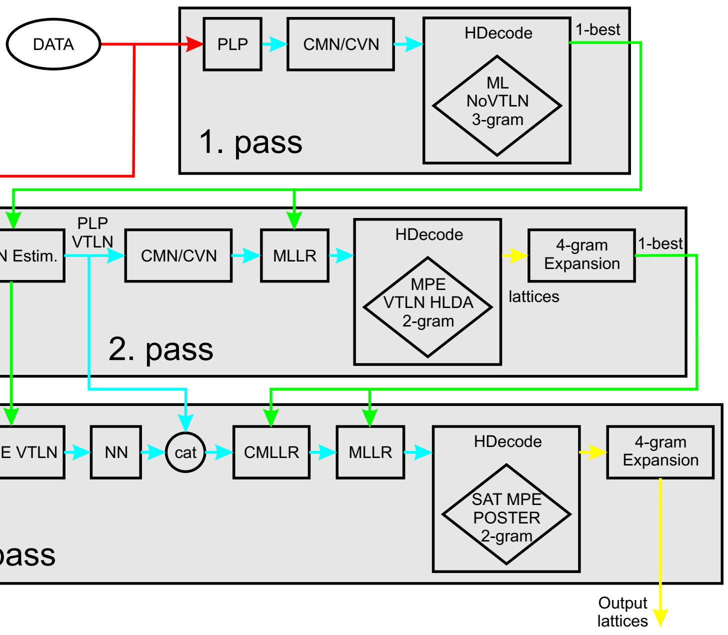

Fig. 3. LVCSR system used for spoken term detection.

where $N_{correct}(term, \theta)$ is the number of correct detections of $term$ with a score greater or equal to θ, $N_{spurious}(term, \theta)$ is the number of spurious (incorrect) detections of $term$ with a score greater or equal to θ, $N_{true}(term)$ is the number of occurrences of $term$ in corpus and $N_{NT}(term)$ is the number of opportunities for incorrect detection of $term$ which is equal to length of the corpus in seconds minus $N_{true}(term)$.

NIST defined so called $TWV(\theta)$ (Term-Weighted Value) metric to "score" a system by one number. Term weighted value is computed by first computing the miss and false alarm probabilities for each term separately, then using these and a pre-determined prior probability to compute term-specific values, and finally averaging these term-specific values over all terms to produce an overall system value:

$$TWV(\theta) = 1 - \underset{term}{avg}\{P_{MISS}(term, \theta) + 999.9\, P_{FA}(term, \theta)\}$$

Threshold θ_M is found on development data by maximization of $TWV(\theta)$. $TWV(\theta_M)$ is then computed on evaluation data with θ_M threshold and denoted as $ATWV$ (Actual Term-Weighted Value, see evaluation plan [3] for further details).

4.2. BUT system

We have submitted LVCSR, phonetic-based and merged system to STD evaluations. Full description of our system (Figure 2) is available in [8].

Recognizers The signals are first segmented into speech and non-speech regions and diarized by a Bayesian information criterion (BIC) based system [9]. Heuristics are applied not to have too long segments for the speech recognizer.

Segmented data was than processed by word (LVCSR) and phoneme (PHN) recognizer. Our LVCSR (Figure 3) is simplified version of the AMI LVCSR system [10]. It operates in three passes: in the first pass, standard MF-PLP features are derived, and the the first decoding pass yields initial transcripts. These serve for initial acoustic normalization. The second pass (P2) processes the new features and its output is used to adapt models with maximum likelihood linear regression (MLLR). Bigram lattices are produced and re-scored by trigram and fourgram language model. In the third pass (P3), posterior features [12, 11] are generated. The output from second pass is used to adapt models with Constrained MLLR (CMLLR) and MLLR. The bigram lattices with posterior features are produced and finally re-scored with trigram and fourgram language model. All systems use standard cross-word tied states hidden Markov models (HMM).

CTS acoustic models were trained on 277 hours of speech from Switchboard database, while the language model was trained on a variety of text resources and spontaneous speech transcripts [10, 8].

Phoneme recognition was based on the same features and models as LVCSR. Only the recognition network was changed to context dependent phoneme (triphone) loop with phoneme bigram language model producing context independent output (ie. the output is phonemes) .

Indexing and search followed closely the theoretical description given in section 3. In the **indexing phase**, word lattices were first converted to forward index (word unigrams).

Forward index was than sorted to inverted index. Lattice were converted to binary format. The same processing was applied to phoneme lattices.

While **searching** a query, in-vocabulary (IV) tokens are searched in inverted index to estimate their position in lattices and than they are verified in the lattice.

Out-of-vocabulary (OOV) tokens are converted to phoneme strings. Automatic grapheme-to-phoneme (G2P) tool based on rules (derived from AMI recognition vocabulary and BEEP dictionary) is used for the conversion. Then the phoneme string is split into a train of overlapped triphonemes. Then they are also searched in inverted index (phoneme) and verified in lattice (phoneme). OOVs shorter than 3 phonemes are not searched and are dropped.

If all tokens were successfully verified, the time and score is produced. The score is computed as sum of IV (LVCSR) part and OOV (PHN) part. IV scores are computed by Viterbi approximation using likelihood ratio in word lattice and then normalized. OOV scores are computed by Viterbi approximation using likelihood ratio in phoneme lattice and then normalized.

Normalization The normalization serves to make scores of different queries comparable (note that NIST scores STD systems with *one single threshold*). Our normalization is based on contributions of phonemes to normalization factors:

$$s_N(KW) = s(KW) - G - F\,l(KW) - P_1|p_1| + ... + P_K|p_K|,$$

where $s(KW)$ is raw score of the keyword, $s_N(KW)$ is the normalized score, $l(KW)$ is length of the keyword and $|p_1| \dots |p_K|$ are counts of individual phonemes in the keyword. G (a constant), F (length-dependent factor) and $P_1 \dots P_N$ (phoneme-dependent factors) need to be trained: First, for large set of keywords, we derive scores for hits and false alarms (FA) on the development set. The scores corresponding to each keyword are used to construct pairs of (HIT, FA). For each pair, an equation is generated:

$$\frac{s(HIT) + s(FA)}{2} = G + F\,l(HIT) + P_1|p_1| + ... + P_K|p_K|,$$

where the left side represents an optimal threshold for given (HIT, FA) pair. We solve the over-defined set of equations in minimum square error sense and use the resulting factors to normalize scores.

4.3. Results

The results in terms of DET curves on development data can be seen in Fig. 4. The results in terms of ATWV are are summarized in Table 1. We see minimum effect of merging phonetic search with LVCSR, this is however caused by the termlists provided – in CTS data, we have counted only 6 OOVs out of all 1100 requested terms.

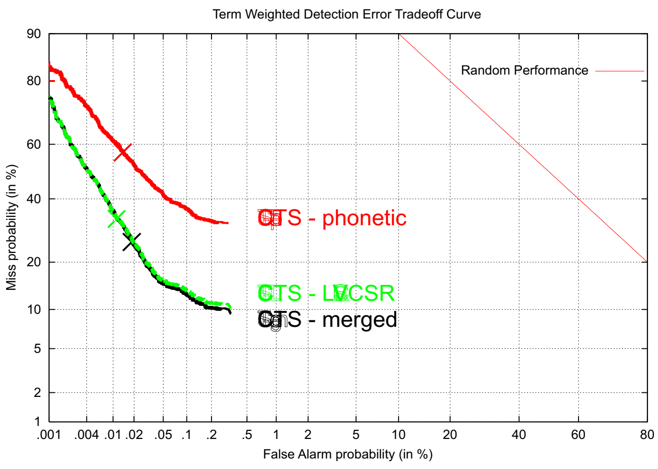

Fig. 4. DET curves for development US English CTS data: LVCSR, phoneme-based and merged systems

system	LVCSR	phonetic	merged
ATWV-Devel	0.5560	0.2910	0.5580
ATWV-Eval	0.5186	0.2977	0.5235

Table 1. ATWV values for individual systems.

To present our results in more "readable" format, think about the miss probability given some fixed proportion of false alarms (FA) per hour. For example, 1 FA/hour corresponds to 1/3600=0.028% on the x-axis of Figure 4. We can say, that at this FA rate, we are able to reach approximately 20% average miss probability, or 80% accuracy. That means, that 8 terms out of 10 will be detected.

5. CONCLUSIONS

The STD evaluation confirmed the usability of our STD system and provided us with the opportunity to compare it to other labs working in the field. The evaluation provided us also with several technical lessons, such as that using 4-gram expansion is only slightly better than 3-gram expansion, posterior pruning of LVCSR lattices shortens DET but does not decreases TWV significantly, etc.

Among the (many) issues we need to address is primarily the CPU time and memory footprint needed – despite its good accuracy, our system was far too slow compared to the others in the evaluation. When designing the system for a real security/defense oriented user, we also need to take into account other user requirements, such as signal pre-processing, entering queries and combination with other speech search modalities.

6. ACKNOWLEDGMENTS

This work was partly supported by European projects AMIDA (IST-033812) and Caretaker (FP6-027231), by Grant Agency of Czech Republic under project No. 102/05/0278 and by Czech Ministry of Education under projects Nos. MSM0021630528 and LC06008. The hardware used in this work was partially provided by CESNET under projects No. 119/2004, No. 162/2005 and No. 201/2006. Lukáš Burget was supported by Grant Agency of Czech Republic under project No. GP102/06/383.

We are grateful to David van Leeuwen (TNO, the Netherlands) for the diarization of NIST STD data, and to Vinny Wan (University of Edinburgh) for language model training. The LVCSR was developed in cooperation with AMI-project[1] partners.

7. REFERENCES

[1] P. Matějka, L. Burget, P. Schwarz, J. Černocký: Brno University of Technology System for NIST 2005 Language Recognition Evaluation, In: *Proceedings of Odyssey 2006: The Speaker and Language Recognition Workshop*, San Juan, Puerto-Rico, 2006, pp. 57-64.

[2] P. Matějka, L. Burget, P. Schwarz, O. Glembek, M. Karafiát, F. Grézl, J. Černocký, D. A. van Leeuwen, N. Brümmer and A. Strasheim: STBU System for the NIST 2006 Speaker recognition evaluation, accepted to ICASSP 2007, Hawaii, 2007.

[3] NIST Spoken Term Detection Evaluation, http://www.nist.gov/speech/tests/std/

[4] Sergey Brin, Lawrence Page: *The Anatomy of a Large-Scale Hypertextual Web Search Engine, Computer Science Department*, Stanford University, 1998.

[5] K. Ng: *Subword-Based Approaches for Spoken Document Retrieval*, PhD thesis, MIT, USA, February 2000.

[6] P. Yu and F. Seide: Fast two-stage vocabulary independent search in spontaneous speech, in *Proc. ICASSP 2005*, Philadelphia, 2005.

[7] O. Siohan and M. Bacchiani: Fast vocabulary-independent audio search using path-based graph indexing, in *Proc. Eurospeech 2005*, Lisboa, Portugal, 2005.

[8] I. Szöke, M. Fapšo, M, Karafiát, L. Burget, F. Grézl, P. Schwarz, O. Glembek, P. Matějka, S. Kontár and J. Černocký Jan: BUT System for NIST STD 2006 - English, available from http://www.fit.vutbr.cz/speech/std/2006/, file but_06_std_eval06_eng_all_spch_p-BUT-STBU-MERGED_1.txt

[9] D. A. van Leeuwen and M. Huijbregts, "The AMI Speaker Diarization System for NIST RT06s Meeting Data", in *Proc. MLMI 2006*, Lecture Notes on Computer Science 4299, pp 371–384, Washington D.C., May 2006 (Springer).

[10] T. Hain et al: The AMI Meeting Transcription System, In: *Proc. NIST Rich Transcription 2006 Spring Meeting Recognition Evaluation Worskhop*, Washington D.C., USA, 2006, p. 12.

[11] F. Grézl, M. Karafiát, S. Kontár and J. Černocký: Probabilistic and bottle-neck features for LVCSR of meetings, accepted to ICASSP 2007, Hawaii, 2007.

[12] P. Schwarz, P. Matějka, J. Černocký: Towards Lower Error Rates in Phoneme Recognition, In: *Proceedings of TSD 2004*, Brno, 2004, Springer LNCS 3206.

[1] http://www.amiproject.org