GTTS Systems for the Query-by-Example Spoken Term Detection Task of the Albayzin 2012 Search on Speech Evaluation*

Amparo Varona, Mikel Penagarikano,
Luis Javier Rodríguez-Fuentes, Germán Bordel, and Mireia Diez

GTTS (http://gtts.ehu.es), Department of Electricity and Electronics, ZTF/FCT
University of the Basque Country UPV/EHU
Barrio Sarriena, 48940 Leioa, Spain
amparo.varona@ehu.es

Abstract. This paper briefly describes the systems presented by the Working Group on Software Technologies (GTTS)[1] of the University of the Basque Country (UPV/EHU) to the Query-by-Example (QbE) Spoken Term Detection task of the Albayzin 2012 Search on Speech Evaluation. GTTS systems apply the state-of-the-art Brno University of Technology phone decoders for Czech, Hungarian and Russian in two different ways: System A looks for approximate matchings of the best decoding of a spoken query in the phone lattice of the target audio document; System B represents both the query and the audio document in terms of frame-level phone log-likelihoods and scans the audio document for possible matchings, by minimizing the distance between feature vectors and applying heuristic strategies to prune the search space and to validate the hypothesized segments.

Index Terms: Spoken Term Detection, Phone Lattice, String Matching, Phone Log-Likelihoods, Cosine Distance, Heuristic Matching.

1 Introduction

In the Query-by-Example (QbE) Spoken Term Detection task of the Albayzin 2012 Search on Speech Evaluation, the input to the system is an acoustic example per query and hence a prior knowledge of the correct word/phone transcription corresponding to each query is not available [1]. The locations and durations of all the occurrences of spoken queries in the audio documents must be obtained. The task is defined in the same terms as MediaEval 2012 Search on Speech [2].

* This work has been supported by the University of the Basque Country, under grant GIU10/18 and project US11/06, by the Government of the Basque Country, under program SAIOTEK (project S-PE11UN065), and the Spanish MICINN, under *Plan Nacional de I+D+i* (project TIN2009-07446, partially financed by FEDER funds).

[1] http://gtts.ehu.es

2 System A: Exact Matching on Phone Lattices

2.1 System description

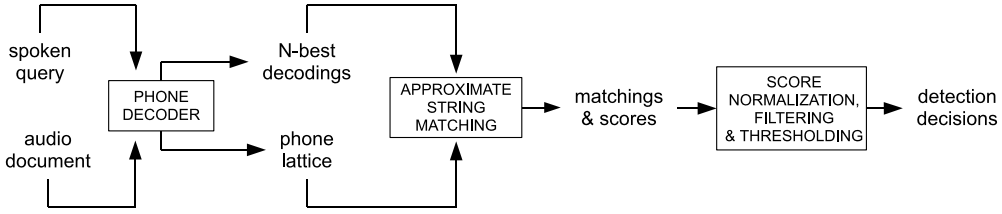

Fig. 1. Processing steps of the GTTS System A for the QbE Spoken Term Detection task of the Albayzin 2012 Search on Speech Evaluation.

Computing phone lattices. As a first step, the open-software Brno University of Technology (BUT) phone decoders for Czech, Hungarian and Russian [3] are applied to decode both the spoken queries and the audio documents. BUT decoders have been trained on 8kHz (SpeechDat) databases, so both the spoken queries and the audio documents are downsampled from 16 kHz to 8kHz.

BUT decoders feature 45, 61 and 52 phonetic units for Czech, Hungarian and Russian, respectively. For each unit, a three-state model is used, so three state posterior probabilities per frame and unit are computed. Since exact (or almost exact) matchings are required to detect queries, the number of phonetic units may be too high for this application. Note that the same sound may be decoded in different ways, in terms of similar (but different) units. To compensate for this effect, the set of units is reduced by defining groups of similar (i.e. highly confusable) units, according to their characterization in the International Phonetic Alphabet (IPA). Besides, the three non-phonetic units used by BUT decoders are fused into a single non-phonetic unit model. Eventually, we use 25 units for Czech, 23 for Hungarian and 21 for Russian.

Let us consider one of the BUT decoders, featuring M phone units, each of them typically represented by means of a left-right model of S states. The posterior probability of each state s ($1 \leq s \leq S$) of each phone model i ($1 \leq i \leq M$) at each frame t, $p_{i,s}(t)$, is directly provided by the phone decoder. When considering a reduced set of units, each unit j clusters a number of similar units, and its posterior probability at each state s and each frame t can be computed by adding the posterior probabilities of all of them:

$$p_{j,s}(t) = \sum_{\forall i \in S_j} p_{i,s}(t) \tag{1}$$

with $1 \leq j \leq R$, R being the number of clusters in the reduced set and S_j the subset of phone units in cluster j. Finally, posterior probabilities are used to produce phone lattices —which encode multiple hypotheses with acoustic likelihoods—, by means of the HTK tool *HVite* [4].

Searching phone lattices. For each spoken query, the phone decoding with the highest likelihood is extracted from the phone lattice by means of the *lattice-tool* of *SRILM* [5]. Only those strings containing more than two phones are considered. Then, the *Lattice2Multigram* (L2M) tool by Dong Wang [6][7][8][2] is applied. L2M takes two inputs: a list of phone strings (the queries) and a list of phone lattices (the documents), and outputs detections in MLF format [4]. The behaviour of L2M is controlled by several parameters, which have been tuned on the development dataset. In particular, *LogLikeliBaumWelch* has been chosen as lattice score computation method and matchings are located under the *ExactMatch* setup (i.e. no edition operations are allowed in matchings).

Handling the scores. Three filters are sequentially applied to MLF detection files:

- *mlf2mlf*: for each detected segment i, a new normalized score is computed in the following way:

$$new\_score_i = \log \frac{e^{\frac{score_i}{length_i}}}{\sum_{\forall j \neq i} e^{\frac{score_j}{length_j}}} \quad (2)$$

  where $length_x$ stands for the length (in frames) of a detected segment x, and the summation in the denominator extends over all the detections $j \neq i$ in the audio document.
  Given a set of spoken queries and a set of audio documents, three MLF files are produced, based on the Czech, Hungarian and Russian BUT decoders, respectively. Detection files can be either mixed and processed jointly, or processed independently. In any case, for each audio document, overlapping detections are processed such that only the most likely detection is kept, the remaining ones being discarded.
- *mlf2std*: detection information is converted to the final STD format.
- *std2std*: for each query, only the K most likely detections in all the audio documents are retained, scores are z-normalized and a threshold is applied.

Preliminary Experiments. Table 1 summarizes the results obtained in preliminary experiments on the development set, using different configurations (in all cases, MLF files are mixed and jointly processed, and $K = 50$). The *Actual Term Weighted Value* (ATWV) is used as primary evaluation measure [9], but false alarm and miss error probabilities are shown too. Since best performance was attained when mixing the Hungarian and Russian decoders (HU + RU), this was the setup chosen for the *primary system*. For the contrastive system, we chose the mix of the three decoders (CZ + HU + RU). In both cases, the final threshold was set to 0.20.

[2] http://homepages.inf.ed.ac.uk/v1dwang2/public/tools/index.html

Table 1. Performance of the GTTS System A in preliminary experiments on the development set under different setups.

BUT decoders	Max ATWV	Threshold	P(FA)	P(Miss)
CZ	0.002	0.168	0.00046	0.970
HU	0.004	0.236	0.00056	0.974
RU	0.006	0.222	0.00116	0.947
CZ + HU	0.010	0.125	0.00075	0.956
CZ + RU	0.006	0.256	0.00137	0.935
HU + RU	0.013	0.222	0.00134	0.933
CZ + HU + RU	0.009	0.247	0.00145	0.928

2.2 Training and development data

BUT decoders: training data.

- Czech Decoder (CZ) - 8 kHz, trained on the Czech SpeechDat(E) database, containing 12 hours of speech from 1052 (526 male, 526 female) Czech speakers, recorded over the Czech fixed telephone network.
- Hungarian Decoder (HU) - 8 kHz, trained on the Hungarian SpeechDat(E) database, containing 10 hours of speech from 1000 (511 male, 489 female) Hungarian speakers, recorded over the Hungarian fixed telephone network.
- Russian Decoder (RU) - 8 kHz, trained on the Russian SpeechDat(E) database, containing 18 hours of speech from 2500 (1242 male, 1258 female) Russian speakers, recorded over the Russian fixed telephone network.

Development data. The GTTS System A was developed based exclusively on the materials provided by organizers for this evaluation, consisting of a set of talks extracted from the Spanish MAVIR workshops: 60 spoken queries and 7 audio documents amounting to about 5 hours of speech [1].

3 System B (late submission): Heuristic Matching

3.1 System Description

The contrastive (late) GTTS system uses a frame-level sequence of phone log-likelihoods to represent both the query and the audio document. The BUT decoders for Czech, Hungarian and Russian are applied to downsampled (8 kHz) signals to get frame-level phone log-likelihoods, at a rate of 100 frames per second. Phone log-likelihoods are stacked in a single feature vector, those corresponding to non-speech units being left out. Thus, feature vectors include 42 log-likelihoods from Czech, 58 from Hungarian and 49 from Russian, which amounts to 149 dimensions.

Based on the above described representation, multiple occurrences of the query inside the audio document could be found just by defining a suitable distance measure between two feature vectors and applying a Dynamic Time

Warping (DTW) approach which minimizes the accumulated distance. The time and space complexities of this DTW-based approach would be in $\Theta(m \cdot n)$, where m: length of the query and n: length of the audio document, with $m \ll n$. We tried DTW using different distances (Euclidean, Mahalanobis, cosine), with and without z-normalization, but got unsatisfactory results on a set of toy examples. Then we tried a slightly different approach, based on a frame-by-frame greedy search for matchings (that is, taking locally optimal decisions) and on some heuristic pruning and validation criteria. Though space and time complexities were still in $\Theta(m \cdot n)$, this *heuristic matching* approach yielded much better results than DTW on the toy examples used for development.

Feature normalization. Let us consider a spoken query Q, represented by the sequence of D-dimensional feature vectors $Q = \{q_1, q_2, \ldots, q_m\}$, and the audio document $A = \{a_1, a_2, \ldots, a_n\}$ on which the search is performed. First, we *z-normalize* the feature vectors of both Q and A according to the means $\mu = [\mu_1, \mu_2, \ldots, \mu_D]^t$ and standard deviations $\sigma = [\sigma_1, \sigma_2, \ldots, \sigma_D]^t$ estimated on the audio document A. Given a feature vector v, the normalized vector $\hat{v}$ is given by:

$$\hat{v} = \left[\frac{v_1 - \mu_1}{\sigma_1}, \frac{v_2 - \mu_2}{\sigma_2}, \ldots, \frac{v_D - \mu_D}{\sigma_D} \right]^t \tag{3}$$

Distance measure. We implemented three distance measures to be applied on the z-normalized features: Euclidean, Mahalanobis and cosine. Best results in preliminary experiments on a set of toy examples were obtained with the cosine distance, defined as follows:

$$d(v, w) = \frac{\sum_{i=1}^{D} \exp(v_i + w_i)}{\sqrt{\sum_{i=1}^{D} \exp(v_i + v_i)} \sqrt{\sum_{i=1}^{D} \exp(w_i + w_i)}} \tag{4}$$

Note that dot products are performed in the space of likelihoods, so the z-normalized log-likelihoods used as features are added and the result is exponentiated and accumulated.

Within-query maximum distances. In order to define query-dependent thresholds, for each query Q the maximum distance between two feature vectors being k frames away from each other, q_i and q_{i+k}, is computed and stored, for $k = 1, 2, \ldots, 10$:

$$dmax(Q, k) = \max_{i=1,\ldots,m-k} d(q_i, q_{i+k}) \tag{5}$$

Search procedure. The search procedure consists of a frame-by-frame sequence of matching attempts, so that up to n different matchings are tried in the worst case. To avoid repeated distance computations, the whole matrix of distances between all the pairs of feature vectors (q_i, a_j), q_i from the query Q

($i \in [1, m]$) and a_j from the audio document A ($j \in [1, n]$), is computed in first place, so time and space complexities are both in $\Theta(m \cdot n)$.

The length of segments matching the query is heuristically bounded by two warping factors, w_{min} and w_{max}. Segments with less than $w_{min} \cdot m$ frames are discarded. On the other hand, if a partial match starting at frame i involves more than $w_{max} \cdot m$ frames, the current search is abandoned and a new search is started at frame $i+1$ (see Figure 2). In this work, $w_{min} = 0.5$ and $w_{max} = 2.0$.

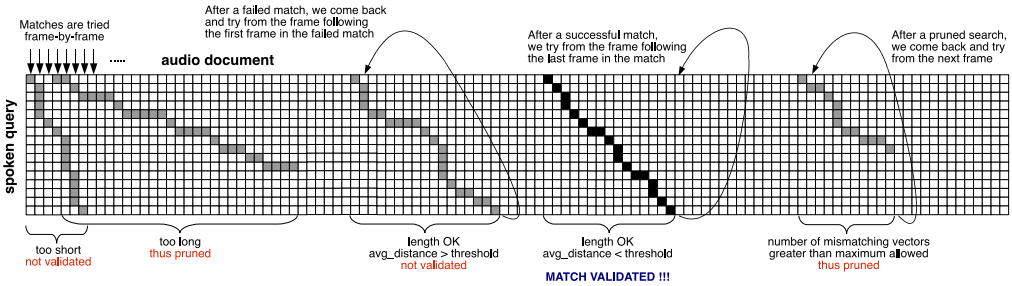

Fig. 2. Heuristic matching consists of a frame-by-frame sequence of matching attempts. Decisions are taken in a greedy fashion, minimizing the distance between the aligned vectors. Several heuristically fixed pruning and validation thresholds are applied.

A threshold θ was established so that if the distance between two feature vectors was greater than θ, we considered it a *mismatch*. We defined θ as a linear combination of within-query maximum distances, as follows:

$$\theta = \theta(Q,\alpha) = (1-\alpha) \cdot dmax(Q,1) + \alpha \cdot dmax(Q,10) \tag{6}$$

After tuning on some toy examples, best performance was found for $\alpha = 0.30$.

A maximum number of mismatches L was allowed during search, so that, given a partial match starting at frame i, if the number of mismatches exceeded L, the current search was abandoned and a new search was started at frame $i+1$ (see Figure 2). In this work, L was heuristically fixed to a fraction of the length of the query: $L = \lambda \cdot m$, with $\lambda = 0.20$.

For each segment $A[i,j]$ matching a query Q (and fulfilling the above described conditions), a threshold δ was applied, so that only if the average distance between feature vectors of Q and $A[i,j]$, $d_{avg}(Q, A[i,j])$, was lower than δ, the segment was accepted and $d_{avg}(Q, A[i,j])^{-1}$ was output as score. The threshold δ was also defined as a linear combination of within-query maximum distances, as follows:

$$\delta = \delta(Q,\beta) = (1-\beta) \cdot dmax(Q,1) + \beta \cdot dmax(Q,2) \tag{7}$$

After tuning on some toy examples, best performance was found for $\beta = 0.40$.

Preliminary Experiments. Using parameter values optimized on a set of toy examples ($w_{min} = 0.5$, $w_{max} = 2.0$, $\alpha = 0.30$, $\lambda = 0.20$ and $\beta = 0.40$), the best performance in preliminary experiments on the development set was found when the score threshold was set to 8.0.

3.2 Training and development data

As for System A, BUT decoders were applied to queries and audio documents to get frame-level phone log-likelihood feature vectors. Development data were also the same used for System A. Details have been already provided in Section 2.2.

References

1. J. Tejedor, D. T. Toledano, and J. Colas, "The Albayzin 2012 Search on Speech Evaluation," in *VIII Jornadas en Tecnologa del Habla and III Iberian SLTech Workshop, IberSPEECH*, Madrid, Spain, 2012.
2. F. Metze, E. Barnard, M. Davel, C. van Heerden, X. Anguera, G. Gravier, and N. Rajput, "The Spoken Web Search task," in *MediaEval 2012 Workshop*, Pisa, Italy, 2012.
3. P. Schwarz, "Phoneme recognition based on long temporal context," Ph.D. dissertation, FIT, BUT, Brno, Czech Republic, 2008.
4. S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Lui, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK Book (Version 3.4)*, Cambridge, 2006.
5. A. Stolcke, "SRILM - an extensible language modeling toolkit," in *ICSLP*, 2002, pp. 257–286.
6. D. Wang, S. King, and J. Frankel, "Stochastic pronunciation modelling for out-of-vocabulary spoken term detection," *IEEE Transactions on Audio, Speech, and Language Processing*, 2011.
7. D. Wang, J. Tejedor, S. King, and J. Frankel, "Term-dependent confidence normalization for out-of-vocabulary spoken term detection," *Journal of Computer Science and Technology*, 2012.
8. D. Wang, S. King, and J. Frankel, "Direct posterior confidence estimation for spoken term detection," *ACM Transactions on Information Systems*, 2012.
9. J. Fiscus, J. Ajot, J. Garofolo, and G. Doddington, "Results of the 2006 Spoken Term Detection Evaluation," in *ACM SIGIR Workshop on Searching Spontaneous Conversational Speech*, Amsterdam, 2007.