Structural Representation of Speech for Phonetic Classification

Alexander Gutkin and Simon King
Centre for Speech Technology Research, University of Edinburgh
2 Buccleuch Place, Edinburgh, EH8 9LW, United Kingdom
Alexander.Gutkin@ed.ac.uk

Abstract

This paper explores the issues involved in using symbolic metric algorithms for automatic speech recognition (ASR), via a structural representation of speech. This representation is based on a set of phonological distinctive features which is a linguistically well-motivated alternative to the "beads-on-a-string" view of speech that is standard in current ASR systems. We report the promising results of phoneme classification experiments conducted on a standard continuous speech task.

1. Introduction

Current automatic speech recognition (ASR) systems are usually based on Hidden Markov models (HMMs) of phones; speech is modelled as a linear sequence of these phones, like "beads on a string" [11]. Phones have no explicit internal structure in these systems beyond the topology of the HMMs used to model them (usually 3 emitting states in a simple left-to-right arrangement). The accuracy of such systems appears to have reached a plateau, motivating many researchers to look for alternative approaches. In this paper, an attempt to devise a classification framework based on a structural representation of speech is presented. The method we propose is motivated by the fact that a symbolic space is well-suited for capturing and exploiting structural properties of speech which HMM systems fail to capitalise on. Since speech waveforms are not symbolic, we must make a transformation into a symbolic representation. At a very low level, frame-based vector quantisation will do this, but we reject this approach since the symbol set is chosen purely on acoustic grounds. Other techniques, such as generalised feature extraction based on structure detectors [10], have emerged to extract symbolic information, providing it as an input to syntactic pattern recognition frameworks, which operate on time series data. Such approaches might potentially offer a natural way of extracting symbolic features and further research is needed to establish whether they are suitable for ASR tasks. We have chosen to make the transition from vector-space to symbolic representation at a linguistically well-motivated level: phonological features. Phonological features are a representation of speech which has several attractive properties, described in section 2. Furthermore, it has been shown [7] that recurrent neural networks can be successfully used to perform accurate phonological feature detection from speech signals.

Two questions now arise: what set of phonological features should be used? And how should we treat the real-valued outputs of the feature-detecting neural networks? These are addressed in section 2.1.

This paper is organised as follows. In section 2 we introduce the structural representation based on phonological feature structure along with a brief description of the related linguistic formalism and describe the way it is extracted from speech and some of its properties, section 3 presents results of the experiments. We summarise the paper in section 4.

2. Phonological feature structure

The lowest (i.e. closest to the speech waveform) level of our framework which is symbolic consists of a set *phonological distinctive features*. Phonological distinctive features are seen in most varieties of phonology as the fundamental units out of which phonemes are constructed. The principle of distinctive features was first proposed by Jakobson, Fant and Halle [4] and re-worked by Chomsky and Halle (cited in [7]) who showed that what were otherwise complex phonological processes (such as assimilation – for example, the [n] sound in "in" becomes [m] when followed by [b], e.g. "in-between") could be written concisely in terms of distinctive features (the process which transforms [n] into [m] is simply the spreading of one feature – place of articulation – from the [b] backwards into the [n]). Many systems of features can be used to represent speech; we use one of the most popular: multivalued features.

A. Gutkin and S. King: Structural Representation of Speech for Phonetic Classification. In: Proc. 17th International Conference on Pattern Recognition (ICPR), Cambridge, UK (August 2004), vol. 3, pp. 438–441

2.1. Multivalued features

We use five multi-valued features: front-back, place of articulation, manner of articulation, roundness and voicing (this particular choice is motivated in [7]). Each of these features takes one of several possible values – for example, manner of articulation is one of: approximant, fricative, nasal, stop, vowel, silence. The neural networks that recover these features from speech use a 1-of-N_j encoding on their output units, hence there are N_j real-valued outputs (ranging $0 \to 1$) for each feature (for manner of articulation $N_j = 6$). The total number of such values produced by the neural network for each frame is $N = \sum_{j=1}^{5} N_j = 25$. Since the training data is fully labelled and segmented, it is possible to label each frame in the data with the corresponding phonological features. Network training is achieved by specifying canonical targets (0 or 1) for each labelled frame, but at runtime the output activation values take continuous values between 0 and 1, and the features change value asynchronously. We map these continuous activation values into the symbols using simple quantisation over N separate finite alphabets of equal size (quantisation level) for each of the N values separately.

2.2. Structural representation

The speech has now been transformed into a sequence of vectors of symbols; this can be seen as a sequence of symbolic matrices, each identifying a phone in terms of its distinctive phonological features. A phone realisation (token) p of class (type) P, $p \in P$, is thus represented as

$$\begin{array}{cccc} f_1^{t_p} & f_1^{t_p+1} & \dots & f_1^{t_p+k_p-1} \\ f_2^{t_p} & f_2^{t_p+1} & \dots & f_2^{t_p+k_p-1} \\ \dots & \dots & \dots & \dots \\ f_N^{t_p} & f_N^{t_p+1} & \dots & f_N^{t_p+k_p-1} \end{array} \quad \rightarrow_t$$

where t_p is the start time, k_p is the duration of p in frames and N is the fixed number of distinctive phonological feature-values which henceforth will be referred to as *streams*. Each of the five features has multiple possible values and hence multiple corresponding streams.

Our phone classification system is template based: it consists of a set of templates learnt from the training data, one or more per phone class to be recognised. The templates may be actual tokens from the training data, or may be constructed; in either case, each template will be represented by the structure described above.

This representation has a number of attractive features. It accounts for duration and contextual effects. Since the durations of tokens vary, even within a class, templates of various durations can be used for a given class. Aspects of co-articulation (such as assimilation, described above) can be accounted for, since the features are represented explicitly and independently. They can change value anywhere within a given template. Finally, this representation is amenable to human examination since its components have explicit linguistic interpretations. One of the shortcomings of this representation is, that at present, we have no way of modelling the feature spreading from one template on to the next one since each template only has the knowledge of its own speech frames.

Using every token in the training data as a template is computationally prohibitive for our data set, so it is necessary to perform clustering to produce a reduced number of templates to represent each class.

2.3. Similarity measures

Once the structural representation is obtained by means of quantisation of neural network outputs, the next step is to define a distance between pairs of templates, or between a template and a token to be classified. An assumption which we make in this paper is that the streams are entirely independent of one another and all have equal importance. For a single token, each stream is a *string* of symbols from our alphabet.

We proceed by defining a metric space corresponding to our structural representation:

Definition 2.1. A *phonological metric space* is a pair $(\mathbb{P}, \mathfrak{d})$ where $\mathbb{P}$ is a set of all possible templates having N streams and $\mathfrak{d} : \mathbb{P} \times \mathbb{P} \to \mathbb{R}^+$ is a mapping of the Cartesian product $\mathbb{P} \times \mathbb{P}$ into the set of non-negative real numbers $\mathbb{R}^+$, such that $\mathfrak{d} = \sum_{i=1}^{N} d_i$, where d_i can be any chosen string similarity measure, satisfying the metric axioms.

Note that the resulting properties of the metric space are essentially dictated by the per-stream distance functions d_i. An additional assumption we make is that the same type of distance function is used for all the streams, i.e. $\forall i \in [1, N] : d_i = d$. Thus, given any two templates, p and q, the distance between them is defined by $\mathfrak{d}(p, q) = \sum_{i=1}^{N} d(s_i^p, s_i^q)$, where s_i^p and s_i^q are the two strings representing stream i of p and q.

The most popular distance function defined on strings is the *Levenshtein Distance*, which introduces a set of edit operations and defines the distance between two strings of lengths m and n as the minimum number of edit operations needed to transform one string into the other. The standard algorithm [3] uses dynamic programming to find this efficiently in polynomial time $O(m \cdot n)$. A more general approach is to assign weights to each of the operations, in which case the optimal sequence is the one that obtains the lowest sum of the weights of editing operations used. In our system, we set all the weights of all insertion/deletion operations to be equal; weight of the substitution is taken to be the sum of insertion and deletion weights. All the weights are normalised by the cardinality of the alphabet.

An alternative distance we used is the *Normalised Edit Distance* [9], which is defined as $\min_{\mathcal{T}} \frac{W(\mathcal{T})}{L(\mathcal{T})}$ where $W(\mathcal{T})$ is the sum of the editing weights in an editing trace $\mathcal{T}$ and $L(\mathcal{T})$ is the number of editing operations described by $\mathcal{T}$. It was shown to outperform its un-normalised counterpart on a hand-written digit [9] and chromosome [8] recognition tasks. This algorithm has $O(m \cdot n^2)$ time complexity.

2.4. Template selection

For each class (39 in our task), we create a number of templates from the set of tokens in the training data of that class, using clustering. The "mean" of each cluster is used as a template – this requires a definition of the "mean" of a set of tokens.

The most obvious choice is *set median*, a generalisation of a traditional set median string. Given a set P of templates, the set median $\mathfrak{p}_s$ is the member of this set that satisfies

$$\mathfrak{p}_s = \operatorname{argmin}_{p \in P} \sum_{q \in P} \mathfrak{d}(p, q),$$

where $\mathfrak{d}$ is the distance from definition 2.1. The time complexity of this algorithm is $O(N \cdot |P|^2 \cdot L^2)$ where $L = \max_{p \in P} |p|$ is the maximum length (duration) found among all the templates in a set, $|P|$ is the number of a templates in a set and N is a fixed number of streams.

An alternative to the set median is the *(generalised) median template* which is based on the notion of *median string* [1] which is a string minimising the sum of distances to each string of the set, but that does not necessarily itself belong to the set. In general, given a set of strings S over a finite alphabet Σ, the median string of a set S is defined as

$$s_m = \operatorname{argmin}_{x \in \Sigma} \sum_{y \in S} d(x, y).$$

The search for the generalised median string is NP-hard. However, efficient techniques for computing an approximation exist and we have adopted a greedy algorithm described in [1].

Recall that we treat the streams independently. Hence, given a set of templates P, we can find N string medians, one for each of the N sets of streams comprising the set P. We then define a *median template* for a set P by constructing a template which consists of the discovered N string medians.

2.5. Clustering strategies

Perhaps the most widely used clustering algorithm is k-means. In this symbolic setting we use a variant called k-medians in which, instead of using the familiar vector-space cluster means, the cluster centroids are either set medians or generalised medians from the previous section.

Level	Training Set	Test Set
3	107284	42061
10	124962	46633
15	125151	46704

Table 1. Num. tokens vs. quantisation level.

The authors of [5] compared four different initialisation techniques for the k-medians algorithm for strings and favoured the symbolic version of an efficient initialisation technique called *Maxmin* which iteratively selects one centroid at a time. At each iteration, the member of the set furthest from any current centroid is added to the set of centroids. Once the centroids are chosen, each token is assigned to a cluster with the nearest centroid.

An alternative initialisation technique we have tried uses the *duration* of the training tokens. Given the size M of the training data set, the tokens are first sorted by duration and the data is then divided into k sets each containing M/k training tokens with the centroids of these k sets chosen as initial centroids.

3. Experiments

Our experiments used the TIMIT database [2]. This is a corpus of high-quality recordings of read continuous speech from North American speakers. The entire corpus is reliably transcribed at the word and surface phonetic levels. For details of the feature-detecting neural networks, please refer to [13].

The standard training/test data partition is kept, with only the `sx` and `si` sentences being used, resulting in 3696 training utterances from 462 different speakers, out of which 100 sentences were held out for cross-validation training of neural networks. The entire test set of 1344 utterances from 168 speakers was used for the classification experiment. None of the test speakers are in the training set, and hence all the experiments are open and speaker independent. There are 39 phone classes.

We quantised the neural network output activations using several different quantisation levels, each inducing a new alphabet. For each quantisation level, the redundant tokens were removed from the resulting symbolic training and test sets. Table 1 shows the number of tokens in the training and test sets obtained for quantisation levels 3, 10 and 15. It can be seen from table 1, that increase in quantisation level leads to increase in number of unique tokens in both training and test sets.

Each training class P was divided into k clusters by k-medians clustering using two different distance metrics (weighted Levenshtein edit distance `Ld` or normalised edit distance `Nd`), two template selection strategies (set median

$\lvert k\rvert$	5	10	15	30	50	100
Sm/Ld/Dc	54.7	58.4	58.8	59.0	59.6	**60.3**
Sm/Nd/Dc	47.1	54.1	55.4	56.5	56.9	**58.2**
Sm/Ld/Mc	49.9	50.6	50.0	50.6	50.7	**54.1**
Gm/Ld/Mc	45.7	49.7	49.1	49.7	**49.8**	49.5

Table 2. Classification accuracy (%) for TIMIT.

Sm and generalised median Gm) and two different initialisations (*Maxmin* Mc and duration-based Dc).

During the recognition stage, an efficient k-NN AESA search technique [6] was used throughout and simple nearest neighbour search based on the score of the top candidate (in terms of the smallest distance to the test token) in the k-best list outperformed the majority voting schemes.

Classification accuracy for the data obtained using a quantisation level of 10 is shown in table 2 for various values of k, which is the number of centroids per class. As can be seen from table 2, the schemes using weighted Levenshtein distance outperform those using normalised edit distance. The schemes using duration-based initialisation outperform those using *Maxmin*. These two findings indicate that accounting for duration is important. Also, the schemes using set median outperform the one using generalised median, suggesting that the construction of medians (as opposed to selecting an existing member of the set) is problematic.

The result of 60.3% obtained in our experiments compares favourably with the results of *basic* vector-space models reported in the literature, such as performance of context-independent three-state single-mixture Gaussian HMM models (61.7% reported in [14]).

4. Conclusions and future work

In this paper we have introduced a linguistically motivated *structural approach* to continuous speech recognition based on *symbolic representation* of distinctive phonological features. We have shown how existing algorithms over strings can be adapted for a phonological feature structure framework and have presented preliminary results for a pattern classification recognition experiment.

Whilst the accuracy of the system is currently somewhat lower than those reported for state-of-the-art vector-space approaches, like Support Vector Machines [12], we are reasonably optimistic since: the structural framework we have used is both intuitive and interpretable; the results were obtained using standard algorithms widely used in the structural pattern recognition community, especially bioinformatics; the system is currently very simple and there is considerable scope for improvement.

Future directions of research will involve improving the modelling power of the framework, particularly with respect to temporal processes within features both within and across phone boundaries (such as assimilation). We will introduce weights on the streams and develop a weight learning component.

References

[1] F. Casacuberta and M. de Antonio. A Greedy Algorithm for Computing Approximate Median Strings. In *VII Spanish Symposium on Pattern Recognition and Image Analysis*, pages 193–198, Apr. 1997.

[2] J. S. Garofolo. *Getting Started with the DARPA TIMIT CD-ROM: an Acoustic Phonetic Continuous Speech Database*. National Institute of Standards and Technology (NIST), Gaithersburgh, Maryland, 1988.

[3] D. Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, New York, 1997.

[4] R. Jakobson, G. M. Fant, and M. Halle. *Preliminaries to Speech Analysis*. MIT Press, Cambridge, MA, 1952.

[5] A. Juan and E. Vidal. Comparison of Four Initialization Techniques for the K-Medians Clustering Algorithm. In *Advances in Pattern Recognition: Joint IAPR International Workshops, SSPR 2000 and SPR 2000*, volume 1876, pages 842–852, Alicante, Aug. 2000. Springer.

[6] A. Juan and E. Vidal. On the Use of Normalized Edit Distances and an Efficient k-NN Search Technique (k-AESA) for Fast and Accurate String Classification. In *15th ICPR*, volume 2, pages 680–683, Sept. 2000.

[7] S. King and P. Taylor. Detection of Phonological Features in Continuous Speech using Neural Networks. *Computer Speech and Language*, 14(4):333–353, 2000.

[8] C. D. Martínez-Hinarejos, A. Juan, and F. Casacuberta. Median strings for k-nearest neighbour classification. *Pattern Recognition Letters*, 24:173–181, 2003.

[9] A. Marzal and E. Vidal. Computation of Normalized Edit Distance and Applications. *IEEE PAMI*, 15(9):926–932, Sept. 1993.

[10] R. T. Olszewski. *Generalized Feature Eextraction for Structural Pattern Recognition in Time-Series Data*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, Feb. 2002.

[11] M. Ostendorf. Moving beyond the ‘beads-on-a-string’ model of speech. In *Proc. IEEE ASRU Workshop*, 1999.

[12] J. Salomon, S. King, and M. Osborne. Framewise Phone Classification using Support Vector Machines. In *Proc. ICSLP*, Denver, 2002.

[13] M. Wester. Syllable classification using articulatory acoustic features. In *Proc. Eurospeech*, pages 233–236, Geneva, 2003.

[14] S. J. Young. The General Use of Tying in Phoneme-Based HMM Speech Recognisers. In *Proc. IEEE ICASSP-92*, pages 569–572, San Francisco, Mar. 1992.