A bag-of-features framework for incremental learning of speech invariants in unsegmented audio streams

Olivier Mangin
École polytechnique / INRIA, France

Pierre-Yves Oudeyer
INRIA, France

David Filliat
ENSTA ParisTech, UEI, France

Abstract

We introduce a computational framework that allows a machine to bootstrap flexible autonomous learning of speech recognition skills. Technically, this framework shall enable a robot to incrementally learn to recognize speech invariants from unsegmented audio streams and with no prior knowledge of phonetics. To achieve this, we import the bag-of-words/bag-of-features approach from recent research in computer vision, and adapt it to incremental developmental speech processing. We evaluate an implementation of this framework on a complex speech database.

1. Introduction

1.1 Constraints, difficulties and learning scenarios

The goal of this article is to introduce a computational framework that allows a machine to bootstrap flexible autonomous learning of speech recognition skills. Technically, this framework shall enable a robot to incrementally learn speech invariants from unsegmented audio streams and with no prior knowledge of phonetics (i.e. without re-using for example a probabilistic phone recognizer). This requires to bring together various conventional speech recognition techniques with the constraints of developmental learning.

This goal implies several important challenges and issues. First of all, in order to achieve **developmental and cognitive plausibility**, as for example discussed by Brent, (Brent, 1999), our system must be incremental and self-organised. Furthermore it must start with little and generic knowledge on the environment it has to explore, i.e. in our case no knowledge of phonetics, phonotactics or lexicons. Another important issue is the design of interaction channels between the system and its environment, that is to say the kind of **(un-)supervision** it is (un-)exposed to and the **multimodality** of the input channels, as for example coupling speech learning with visual information. Finally the integration of motor channels in the learning process connects learning and action. This introduces issues ranging from goal oriented analysis of the audio stream, to studies of more intrinsic structures of this stream.

Choosing a standpoint between these aspects defines a learning scenario. A broad spectrum of such experimental setups have been developed in previous work and shows how they influence the choice of methods and algorithms. Organizing those scenarios through the degree of supervision they include gives a good way to identify categories of machine learning methods they integrate. First, systems with no interaction with outside word, thus unsupervised, extract intrinsic structure of the speech flow. Such scenarios lead to a kind of "autistic" systems: as they do not share any convention with the outside world, or other sensorimotor channels, no real communication is possible (Park and Glass, 2008). Anyway, the mechanism used in those scenarios are of real interest, for example to be used for bootstrapping (Iwahashi, 2003, Brandl et al., 2008), or as intermediate modules of larger architectures. The second category of scenarios is the one for which supervision takes the form of a reward, thus leading to reinforcement learning frameworks. It is for example the case when Gorin et al. (Gorin et al., 1994) train their system to a specific task: the *How may I help you ?* problem. Finally, the most common scenarios take the form of a completely supervised problem, with for example labeled data. Those labels may be semantic tags (Gorin et al., 1999, ten Bosch et al., 2008) or language tags for language recognition (Ma and Li, 2005). Self-labeled data are also possible, for example, when dealing with multiple sensory-channels.

1.2 Existing architecture for speech learning

We have just seen that the methods and algorithms to be used are quite influenced by these learning sce-

narios. Those methods must be integrated in some learning architecture, designed to fit the requirement of a specific scenario. Those are quite different but we may extract some key principles of their design, some of which are essential in the framework we present in this article.

Sub-lexical and lexical classification These are the abilities to classify sounds into groups that share similarities, such as phonemes, and to build a representation of words on top of those groups of basic sounds, which includes the compliance to phonotactic rules. To achieve lexical and sub-lexical classification, two heuristics have to be used: topological information, representing the proximity between sounds, and statistical information used to assimilate different sounds, that have the same function in the language. It has indeed been shown (Kuhl, 2004) that, as young infants are initially able to distinguish different sounds, this ability disappears for sounds that are different but functionally equivalent in their mother language. This phenomenon is necessary to achieve robustness against the great variability of pronunciation of a given phoneme.

Sub-lexical classification may be achieved by Recurrent Neural Networks, as in (Roy and Pentland, 2002), or Hidden Markov Models as in (Ma and Li, 2005); but in those examples, this classifier is trained offline, before the experiment, in a static manner, as a *universal sound recognizer* which does not fit our developmental requirements. Aimetti (Aimetti, 2009) build a system that learns a lexical representation, computed using similarity measures on segments of speech; one example of those segments is used to represent each keyword. Park and Glass used a graph clustering method to group similar sound segments as sub-lexical entities (Park and Glass, 2008), thus autonomously learning such a representation. The learning of this classification is often treated as a clustering problem.

Segmentation Whereas utterance segmentation is quite easy through silence recognition, word segmentation is indeed a difficult task (even for standard written text when spaces are removed). Many speech recognition systems are based on the ability to find a word segmentation, whereas others recognise utterance without performing such a segmentation. For example *segmental dynamic time warping* methods uses dynamic programming to find similar sound segments between speech examples. Then they define sub-lexical units as those segments (Aimetti, 2009, Park and Glass, 2008, Gajjar et al., 2008). On the other hand non-negative matrix factorization methods, such as in (ten Bosch et al., 2008, ten Bosch et al., 2009) also show great results in speech recognition. This method builds an internal word representation from whole utterances, without being designed on the ability to segment words. It is anyway, afterward, able to find such a segmentation, as a consequence of the recognition process.

Semantic structure Matching a lexical representation of spoken utterances to a more structural representation, which may include grammar extraction, or syntactic analysis, requires specific integration of such structure in the recognition system. Methods designed toward this goal often use a predefined structure, to which the utterances are mapped. For example in Iwahashi's' experiment (Iwahashi, 2003), this semantic consists of (object, action, position) associations and is analysed using a graph structure adapted to this grammar. Without predifined implementation, the system must find an origin for these semantic representations.Other input channels such as the vision channel or motor channels, in the case of action oriented goals, are generally part of this process. It may then be difficult to design a system that autonomously builds its own grammar representation without dealing with the corresponding structures in other sensorimotor channels. For example Gorin et al. (Farrell et al., 1993, Gorin et al., 1994) use multilayer neural networks to map the recognition of some words to an action. In this experiment the semantic of the environment was based on action choices, that is to say on a kind of motor channel.

Memory-like architectures To build larger and more realistic systems, that are capable of long-term learning, it often becomes necessary to work with a model of memory. Actually, having an incremental system often brings growth in data size, leading to memory usage and computation. The ACORNS[1] (Aimetti, 2009, ten Bosch et al., 2008, ten Bosch et al., 2009) project have explored some of these issues by introducing memory levels. The principle is to separate data storage in different levels, where depth in memory is correlated to an increase in organization of the data. This may, to a certain extent, be seen as a compression problem, related to highly organised data in the long term memory, but also introduce an attention mechanism, associated with a short term memory.

2. Applying the bag-of-words method to spoken language invariants discovery and recognition

The main contribution of this article is to adapt the bag-of-words method to a developmental approach of the learning and bootstrapping of speech recognition

[1] **A**cquisition of **Co**mmunication and **R**ecogn**i**tion **S**kills, http://www.acorns-project.org

skills. In this approach, the bag-of-words method will be used to bootstrap and maintain incrementally new, potentially multi-scale and multi-type, acoustic representations of speech invariants from unsegmented speech streams and with no prior phonetic knowledge. A lower-level will build those representations in an unsupervised manner, while a higher level of the architecture will consist in re-using those low-level representations to learn to predict a general semantic tag associated to whole utterances. We did not introduce memory handling, but we believe it is a matter of adapting the algorithms from each part of the framework, without changing its global design.

2.1 Background and principle

Bag-of-words methods originate in text classification applications (Joachims, 1997) and have been used with great success in image categorization applications after the seminal work of Sivic and Zisserman, 2003 (Sivic and Zisserman, 2003). The general idea is to represent the text or the image as an unordered collection of local elements chosen in a dictionary (the words in a text and local visual features in an image), thus ignoring the global structure. Using this representation, a classification algorithm can then be used to predict the associated category. In computer vision applications, this representation is very compact thanks to the quantization of local feature representation in the dictionary, while preserving the stable local information and ignoring more unstable global geometry. In most applications, the dictionary is static and requires an initial training phase, but in previous work, we have developed an incremental approach that meet the requirements of developmental systems (Filliat, 2008). We will therefore transpose this method to the speech recognition problem. Yet, for the sake of clarity, we will use the terminology "bag-of-features" instead of "bag-of-words", since the "words" in the bag-of-words approach are not at all equivalent to "linguistic words" in the speech stream and which constitute important speech invariants to be discovered and learnt in our framework.

2.2 Presentation of the framework

Our framework is composed of three distinct layers that we describe below.

- **Continuous Acoustic Feature Vectors (CAF) extraction:** this layer transforms the input audio signal into a set of vectors, each associated with some position information. The goal of this process is to transform the signal into a set of local descriptors, which are more adapted to similarity comparison. Actually the next step requires to have a kind of distance on these vectors, in order to be able to access to a notion of acoustic similarity. This first layer typically uses static sound processing methods (e.g. MFCC or RASTA-PLP, see Section 3.2).

- **Unsupervised clustering:** the role of this layer is to transform each CAF vector from the set obtained above, into a *discretized acoustic feature* (DAF), that is to say a single number. This transformation is accomplished through a clustering process. More precisely this clustering must build incrementally a representation of this DAFs, using the similarity measure inherent to the CAF space. This representation has to both allow retrieval of the DAF corresponding to a given CAF vector and the learning of new DAFs when a CAF vector does not match any known feature.

- **Higher level semantic treatment:** the two previous layers may be seen as a pre-processing, which goal is to transform the input audio signal into a bag of discretized acoustic features, more precisely we get a set of couples, each composed of a DAF and its position. This semantic layer introduces a new representation of the audio signal that allows to efficiently set up higher level statistical treatment, such as keyword recognition or more complex analysis.

Mathematically, this process may be described as following: given an input audio sequence $a \in \mathcal{A}$, a continuous feature vector space $\mathcal{F}$, a set of localization data, such as time position in the utterance, $\mathcal{P}$, a discrete acoustic feature dictionary $\mathcal{D}$:

- extract CAFs: $a \in \mathcal{A} \longrightarrow (v_i, p_i) \in (\mathcal{F} \times \mathcal{P})^\star$
- find corresponding DAFs: $(v_i, p_i) \longrightarrow (f_i, p_i) \in (\mathcal{D} \times \mathcal{P})^\star$

Where, i is a free variable, and for any set $\mathcal{E}$, we call $\mathcal{E}^\star = \bigcup_{k \in \mathbb{N}} \mathcal{E}^k$ the set of finite sequences over $\mathcal{E}$.

In the case of tag inference, the statistical process is then, given a set $\mathcal{T}$ of tags, a mapping: $(\mathcal{D} \times \mathcal{P})^\star \to \mathcal{T}$.

2.3 Modularity and cognitive plausibility

Bag-of-sounds approach has already been used by Ma and Li (Ma and Li, 2005), but with a sub-lexical model built offline and from labelled examples. The novelty of our work is to present a framework based on the ability to learn autonomously a new representation of sound, which enables a completely generic statistical treatment. Indeed the modularity of the framework is present at each of the previous layers.

The first level, whose role is to extract CAFs from input audio sound, may implement a large variety of signal processing treatment on this input stream. It may for example implement windowed spectral or

cepstral analysis of the sound, but may also be composed of more elaborated pieces of information such as pitch or stress patterns. The position labels p_i associated with those vectors may also be of different natures, either just an index in a sequence or a more precise position, and may be completed by other information, such as the width of the input signal relevant to this feature vector. Furthermore, CAFs of different natures may be simultaneously computed, they are then clustered by *distinct* dictionaries, but may be grouped at the end in the *same* bag of DAFs representation. This is a completely transparent way to mix pre-processings of completely different natures.

The second layer may implement any clustering algorithm, since we have a relevant metric on CAFs produced by the previous one. It indeed corresponds to the extraction of acoustic building blocks.

The third layer is completely general as our goal is to provide an intermediate representation for the audio input. We may for example plug onto this representation any classification algorithm, or structured output algorithm.

What we show in the following is that the particular representation we have built retains enough information to enable powerful statistical treatment, and simplifies enough the signal representation to allow such treatment to be efficient. Furthermore, studies on the ability of children to distinguish between sounds seem to indicate that such a representation, even if it is a lot simplified, is a reasonable one for speech recognition (Kuhl, 2004).

The following experiment implements this framework in a very simplified manner: position information is dropped, statistical analysis is reduced to a quite rudimentary scoring method, far from state-of-the-art statistical machine learning approaches. Yet, results will show that this representation is sufficient to predict semantic tags with great accuracy in a large complex database, even with this implementation, thus showing the robustness of the general approach.

3. Specific implementation

3.1 Our experimental scenario

As explained above, in this paper, we adopt a framework where the goal is to allow a robot to progressively learn to predict semantic tag(s) associated to a given speech utterance. For example the robot is incrementally provided with examples of associations between speech utterances and semantic tags, and should accordingly incrementally update its internal representations in order to predict better these semantic tags in new utterances. Semantic tags are technically encoded as keywords referring either to general topic(s) of the utterance, sometimes corresponding to the presence of a particular word in the utterance or to the speaker style or language.

The framework presented in the previous section 2.2 is illustrated by the following steps, in the case of this particular application:

- extract CAFs from the input channel. The CAFs used in our application are described in the next section.
- match these vectors to the dictionary(-ies) and drop the position information, thus creating a bag of DAFs representation of the input signal, and update the dictionary(-ies) if necessary. This particular process is described in section 3.3.
- infer the semantic tag associated with the utterance through a scoring method.

We now present an implementation of this framework and associated experiments. It should be considered as a specific implementation of the general framework that we presented.

3.2 Continuous feature vectors extraction

In our experiment we use Mel-Frequency Cepstral Coefficients (MFCC) and Relative Spectral Transform - Perceptual Linear Prediction (RASTA-PLP) features over a short time window, from Ellis (Ellis, 2005) implementation. The former feature vectors, which are actually time sequences of successive feature vectors, are compared with respect to a Dynamic Time Warping (DTW) distance (Sakoe and Chiba, 1978). This distance takes into account possible insertions and deletions in the feature sequence. It is adapted for sound comparison but does not correspond to an inner product in CAF space, since it is not an euclidean distance, which leads to some new issues.

One other interest of using a DTW distance is to be able to compare sound feature vectors of different length or of varying rhythm. However, in our experiments we used fixed length feature vectors (but the rhythm varies): for each sound utterance we first compute the MFCC sequence corresponding to this audio stream. After extracting this MFCC sequence, we cut it into fixed length features, using a 80 or $150ms$ sliding window. The sliding length used in most of our experiments is one third of the length of the window. However, it is also completely possible to mix several lengths in the same vocabulary or to extract features of random lengths. This may result in a more multiscale-like approach. Those lengths are here around the scale of a phoneme length and give a good trade-off between sufficiently long sequence of MFCC vectors and the DTW quadratic complexity. Furthermore it is relevant to limit this length to get really local descriptors, which we tried

to implement, even if it is not a requirement of the framework.

3.3 Incremental unsupervised clustering

The dictionary must group similar CAF vectors according to the DTW distance into discretized acoustic features. This requires two processes : the dictionary construction and the retrieval of the DAF matching a specific CAF. In our developmental approach we need an incremental dictionary construction able to learn new DAFs, that is to say new vector clusters. We also face a computation time issue for the matching process, thus requiring a dictionary data structure that enables both : efficient algorithms for matching, and the possibility to perform incremental clustering.

Our approach is very similar to the one we used for image processing in (Filliat, 2008). The idea is to represent DAFs by clusters of CAF vectors. Those clusters are organised in a hierarchical way. More precisely, the DAFs are hyperspheres in the continuous feature space, and their centers are organised in a tree structure inspired by the one of Nister and Stewenius (Nister and Stewenius, 2006) where leaves and nodes represent hierarchical clusters. The tree structure is organised as follows:

- each leaf or cluster C is represented by its centroid: a vector v_C,
- each cluster is associated to a hypersphere of radius r_{max} around its centroid. A CAF vector v is therefore part of a cluster C if and only if $d(v, v_C) \leq r_{max}$
- each node of the tree has a limited number of children N_{max} and has an associated centroid n_C which is the mean of its children CAF vectors.

A CAF vector is matched to a cluster by recursively following the child of the node which centroid is the nearest from the searched vector. The dictionary is built by adding these vectors to the tree: we find the nearest cluster; if the vector matches the radius condition regarding to this cluster, it is added inside this one; if not, a new cluster is created initially containing only this vector. This cluster is added as a leaf in the tree, at the same level and with the same father as the previously found nearest cluster. Then we check if the number of children is below N_{max}; if not, the node is split in k nodes, by a k-means process on the centroids of the leaves. The leaves are then distributed to those child nodes. An example of this mechanism, also described by the following pseudo-code, is shown in figure 1.

Algorithm Adding a vector to the cluster tree node

ADD_VECTOR_TO_NODE(*current_node*, *vector*, k, r_{max}, N_{max})

- *current_node* is the node where the vector is to be added,
- *vector* is the vector to add,
- k is the k-means parameter,
- r_{max} is the threshold distance that is used to decide if two vectors are considered identical,
- N_{max} is the maximum number of vectors that a leaf may contain.

```
. if current_node is a leaf
.    let v be the nearest vector in current_node
.    distance (vector, v) ≥ r_max
.    add vector to current_node
.    let n be the number of vectors in current_node
.    if n ≥ N_max
.       new_leaves ← k_means(k, current_node)
.       let new be a new internal node with the
.            elements of new_leaves as children
.       replace current_node by new
.    else
.       add vector to current_node
. else
.    let child be the nearest child from vector
.       in current_node
.    add_vector_to_node(child, vector, k, r_max, N_max)
```

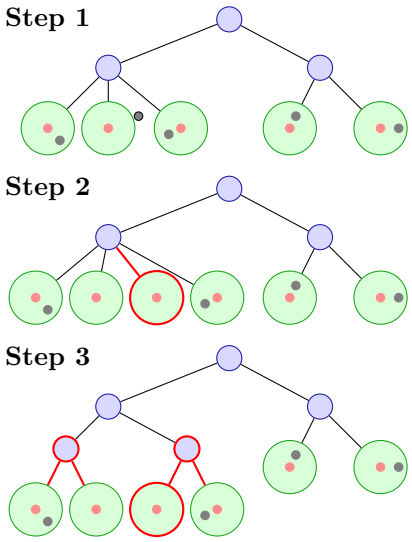

Figure 1: Insertion of a new vector in the hierarchical structure. The nearest leaf is found, but the vector is too far from the center (first step) so a new leaf is created (second step). The new leaf father has now too many children ($N_{max} = 3$) so the node is split in two parts(third step).($k = 2$)

This structure leads to approximate nearest neighbour search, and thus the processes of learning a CAF or retrieving the corresponding DAF are ap-

proximate. Since CAF vectors are themselves noisy, this approximation is naturally handled by the statistical treatment in layer 3. In order to reduce the impact of orientation errors while exploring the tree, which may result in an important final error, for example, if it occurs near the root of the tree, we added the following improvement to the search algorithm.

The idea is to launch more than one search for each request and then select the best results. This is close to *branch-and-bound* techniques and may be implemented in many ways. We tried two implementations of this method. In the first one, for each node reached during the search process, the search is launched again on its b best children, instead of just the best child. By best children we mean the b sons with the lowest distance between their centroid and the requested vector. b is called the backtracking parameter. This method leads to a complexity of $\mathcal{O}\left(n^{\log_k(b)}k\right)$, where n is the number of nodes, k the k-means parameter used to create the tree and b the backtracking parameter. This may be long compared to the $\mathcal{O}\left(k\log(n)\right)$ original complexity.

The second method uses the same idea, but instead of deciding locally which node deserves to be explored, it runs full searches, at the end of which it launches again a search from some node on the tree, where a good candidate path may have been missed. More precisely, during the search, each time a child node is chosen for the proximity of its centroid to the requested vector, its siblings are memorized with some value representing how far they were from the chosen child. When a candidate leaf is finally found, the system is able to reconsider the choices it has made during the search and explore the node which had the best value.

By repeating this process b times, and finally choosing the best candidate nearest neighbor from those found, we are able to minimize the impact of the approximate nature of our structure. The actual complexity of this method is roughly $\mathcal{O}\left(bk\log(n)\right)$.

The second method gave a better trade-off between the number of explored nodes, which corresponds to computation complexity, and the quality of the retrieved approximate nearest neighbor.

3.4 Semantic tag inference

While previous steps were able to build an internal representation for the system, based on topological information, this process had no relation to the final goal of classification. Actually, all the semantics related to the classification task is created in a last step. We used a vote implementation to score DAFs and examples regarding semantic tags.

The idea of the voting scheme is to associate a weight w_i to each DAF i. Let f_i^t be the frequency of DAF i regarding tag t, $f_i^t = \dfrac{n_{i,t}}{n_t}$ where $n_{i,t}$ is the number of co-appearances of DAF i and tag t and n_t the number of appearances of t.

For a query utterance q, where acoustic DAF i appears q_i times, i votes as $V_i = q_i \cdot f_i^t \cdot w_i$, where w_i are weights.

A common way of setting weights w_i is to use a Time Frequency - Inverse Document Frequency (TF-IDF) approach by setting $w_i = log\left(\frac{N_{tags}}{N_{tags}^{(i)}}\right)$, where N_{tags} is the total number of tags and $N_{tags}^{(i)}$ the one of tags whose examples contain DAF i at least once.

From this basis, conditions may be added such as setting all node weights to zero except from leaves, which rely entirely on the a priori chosen size of clusters, that is to say the r_{max} parameter, in our case. One may also choose to allow only nodes near the leaves to have a nonzero weight or to rely entirely on TF-IDF weights. This kind of modifications may bring more scalability and robustness to the system. It also defines which clusters are DAFs: either only leaves or all nodes, and thus the use or not of hierarchical and multi-scale DAFs.

In order to be able to compute this score we store the number of appearances of each DAF in an utterance associated to a particular semantic tag: this corresponds to previously introduced $n_{i,t}$.

The following process is used: while training, for a given utterance with tag t, transformed in a bag of DAFs, for each DAF i, $n_{i,t}$ is increased by one.

During a testing phase, we extract the bag of DAFs corresponding to the utterance. Then, for each tag we compute its score on the utterance, by summing the votes of each DAF. Votes are computed as explained previously, using only the count of co-occurrences, by simple operations over the $(n_{i,t})_{i,t}$ matrix.

4. Results, analysis and further directions

4.1 Databases and protocols

We restricted our work on labeled classification problems, that is to say, sets of utterances associated with a semantic label. Those labels may be words contained in the utterance as well as more general themas, levels of speech, or speakers. The system is trained with such a learning database and then evaluated on its label prediction performance.

During our experiments we worked with two databases. The first one was a home made database in which utterances were single words. This database, which contains twenty three examples of ten different words, was used to evaluate the performances of the nearest neighbor retrieval with word-long features. The second one is a database provided by the ACORNS project, composed of 1000

utterances containing 13 keywords, each spoken by 4 speakers in English adult directed speech; which makes a total of 4000 utterances. An example of sentences used in the database is *Angus is lazy today.* where the semantic tag/keyword is *Angus*.

In the experiments we split the database into a training set and an independent test set to evaluate the system. In order to characterize the efficiency of the learning process as its improvement through training, that is to say the convergence speed of the algorithm, we regularly test the process during the training and visualize its performance at each step.

4.2 Global results

In order to demonstrate the cognitive efficiency of our system we set up the following experiment: for each speaker we randomly split the database in two sets: a train set consisting of 900 examples and a separate test set of 100 examples. The system is trained incrementally with each utterance of the training set; after each 100 train examples, the system is tested on the whole test set. This protocol, which allows us to monitor its progress, is represented in figure 2.

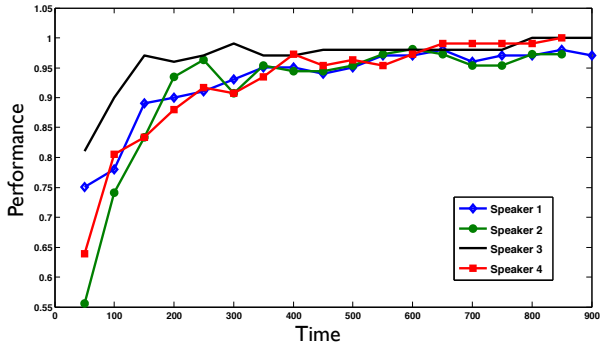

Figure 2: Accuracy against number of training examples for incremental learning and testing, 1000 examples: 900 for training and 100 for testing, one 100 examples test cycle each 50 train examples. (80ms MFCC features)

The same experiment can be made with the 4000 examples coming from all four speakers, to demonstrate that the method is, in some way, robust to multi-speakers learning. In this experiment, the training sessions are 200 examples long and after each training session the process is tested with a constant set of 400 examples: 100 from each speaker. The training set is a succession of 900 examples from each speaker, presented by order of speakers. Such results are presented in figure 3.

These experiments show the good accuracy of our system on the keyword recognition problem. We may compare these results with those from Bosch et al. (ten Bosch et al., 2008) within the ACORNS project, whose database we used. Actually our results are quite similar to the ones they obtained us-

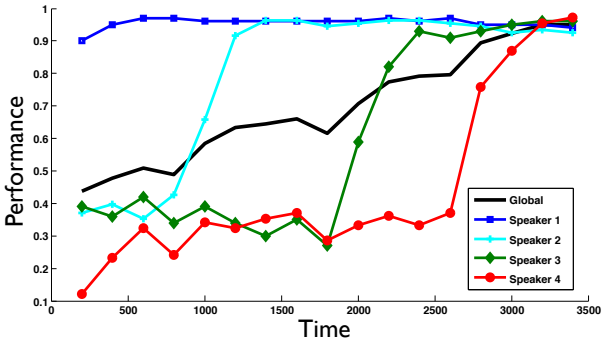

Figure 3: Success rate against number of training examples for incremental learning and testing, 4000 examples: 3800 for training and 200 for testing, one 200 examples test cycle each 100 train examples. (80ms MFCC features) One curve represents global accuracy; the other ones are for each speaker, accuracy reached on the test examples from this speaker.

ing non-negative matrix factorization, which method is also not centered on segmentation and proved to reach maximal performances among a variety of various competing technical approaches.

Those results demonstrate, first of all, the ability of our system to build an internal representation of speech units, in an unsupervised manner (information about keywords is not used in the building of the dictionary), and then to use this internal representation to achieve a keyword recognition task, performed by a kind of semantic engine, which in our experiments is the score system.

4.3 Limitations and further directions

We have presented in this article a framework for autonomous discovery of speech invariants that are useable for speech recognition. This framework essentially builds a new representation of the input signal, as a bag of discretized acoustic features. However despite the word *bag* inherited from text processing, it is completely possible to keep position information on the feature. In this article we have ignored this information in order to demonstrate the efficiency of the extracted local features but it would be of interest to take into account sequential information in future work. For example, hidden Markov models may be built on top of the introduced discretized acoustic features, or sequence analysis methods.

Furthermore, our framework as it is presented is a pre-processing, bringing a new sound representation which is useable with a wide variety of existing methods with a significant complexity reduction from the original input signal.

It is also important to notice that the current clustering method is not completely optimal. Actually the structure of the DTW distance is not completely exploited and experimental analysis shows

that the built clustering has difficulties to cover the whole MFCC sequence space. This could be improved by using better cluster representation, for example using adapted kernels for DTW, such as in (Shimodaira et al., 2001).

The presented framework offers the ability to separately improve each one of these components. For example, the clustering method we used can be replaced by, and thus easily compared to, non-negative factorization.

Finally, we might mention that this framework yet only targets recognition. It would thus be an important further development to integrate a generative model in order to combine perception and action.

References

Aimetti, G. (2009). Modelling Early Language Acquisition Skills : Towards a General Statistical Learning Mechanism. In *EACL (Student Research Workshop)*, pages 1–9. The Association for Computer Linguistics.

Brandl, H., Joublin, F., and Goerick, C. (2008). Towards unsupervised online word clustering. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 5073–5076, Las Vegas, NV.

Brent, M. R. (1999). Speech segmentation and word discovery: A computational perspective.

Ellis, D. P. W. (2005). PLP and RASTA (and MFCC, and inversion) in Matlab.

Farrell, K., Mammone, R. J., and Gorin, A. L. (1993). Adaptive language acquisition using incremental learning. *Acoustics, Speech, and Signal Processing, IEEE International Conference on*, 1:501–504.

Filliat, D. (2008). Interactive learning of visual topological navigation. In *Proceedings of the 2008 IEEE International Conference on Intelligent Robots and Systems (IROS 2008)*.

Gajjar, M. R., Govindarajan, R., and Sreenivas, T. V. (2008). Online Unsupervised Pattern Discovery in Speech using Parallelization. In *InterSpeech*, pages 2458–2461. ISCA.

Gorin, A. L., Levinson, S. E., and Sankar, A. (1994). An experiment in spoken language acquisition. *Speech and Audio Processing, IEEE Transactions on*, 2(1):224–240.

Gorin, A. L., Petrovska-Delacretaz, D., Wright, J., and Riccardi, G. (1999). Learning spoken language without transcription.

Iwahashi, N. (2003). Language acquisition through a human-robot interface by combining speech, visual, and behavioral information. *Inf. Sci. Inf. Comput. Sci.*, 156(1-2):109–121.

Joachims, T. (1997). Text Categorization with Support Vector Machines: Learning with Many Relevant Features.

Kuhl, P. K. (2004). Early language acquisition: cracking the speech code. *Nature Reviews. Neuroscience*, 5(11):831–43.

Ma, B. and Li, H. (2005). Spoken language identification using Bags-of-Sounds. In Dong Minghui, Li, H., and Zhang, M., (Eds.), *International Conference on Chinese Computing*, Singapore.

Nister, D. and Stewenius, H. (2006). Scalable Recognition with a Vocabulary Tree. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2161–2168, Washington, DC, USA. IEEE Computer Society.

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.

Roy, D. and Pentland, A. (2002). Learning words from sights and sounds: a computational model. *Cognitive Science*, 26(1):113–146.

Sakoe, H. and Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49.

Shimodaira, H., Noma, K.-i., Nakai, M., and Sagayama, S. (2001). Dynamic time-alignment kernel in support vector machine. In *Neural Information Processing Systems*, volume 2, pages 921–928.

Sivic, J. and Zisserman, A. (2003). Video Google: A text retrieval approach to object matching in videos. *International Conference on Computer Vision*, page 1470.

ten Bosch, L., Boves, L., Hamme, H. V., and Moore, R. K. (2009). A Computational Model of Language Acquisition: the Emergence of Words. *Fundam. Inform.*, 90(3):229–249.

ten Bosch, L. F. M., van Hamme, H., and Boves, L. W. J. (2008). Unsupervised detection of words questioning the relevance of segmentation. In *Speech Analysis and Processing for Knowledge Discovery*, ITRW ISCA. Bonn, Germany.