



# Automatically Learning the Units of Speech by Non-negative Matrix Factorisation

Veronique Stouten, Kris Demuynck, Hugo Van hamme

Katholieke Universiteit Leuven – Dept. ESAT  
Kasteelpark Arenberg 10, B-3001 Leuven, Belgium

{veronique.stouten, kris.demuynck, hugo.vanhamme}@esat.kuleuven.be

## Abstract

We present an unsupervised technique to discover the (word-sized) speech units in which a corpus of utterances can be decomposed. First, a fixed-length high-dimensional vector representation of the utterances is obtained. Then, the resulting matrix is decomposed in terms of additive units by applying the non-negative matrix factorisation algorithm. On a small vocabulary task, the obtained basis vectors each represent one of the uttered words. We also investigate the amount of speech data that is needed to obtain a correct set of basis vectors. By decreasing the number of occurrences of the words in the corpus, an indication of the learning rate of the system is obtained.

**Index Terms:** matrix factorisation, word segmentation, phone lattices, language acquisition.

## 1. Introduction

This work is motivated by the observation that infants appear to be remarkably good at acquiring new words from a spoken language. An ability that is intrinsically required for this learning process is that the continuous stream of speech can be segmented into words. Evidence exists that this segmentation problem might be solved by keeping track of the relative consistency in the sound sequences. Over a speech corpus, those sequences with relatively high conditional probabilities are likely to be inside words, while the sequences with low conditional probabilities are likely to be the accidental juxtapositions of sounds at word boundaries [1, 2]. Human learners are able to segment elementary units out of large and apparently unsegmented streams, using the statistical structure embedded within this stream.

On the other hand, current automatic speech recognition (ASR) systems incorporate expert knowledge that arises from audiology and phonology. It is decided beforehand what the words are, and how the words are composed of sub-word units. This knowledge is a necessary prerequisite for training statistical models of the primitive elements and of the patterns to be recognised. In such a hierarchical way of representing speech (the ‘beads-on-a-string’ approach [3]), sub-phones are combined to phones, which are linked together to words, and finally to sentences. Nevertheless, children acquiring language do not know words and speech sounds until after the acquisition process is completed. This has motivated researchers to develop a technique in which the subword units are automatically selected and the lexicon is learnt from data [4, 5]. In these methods, the design of a unit inventory consists of an acoustic segmentation followed by a clustering step.

Also our aim is to let a computer system automatically detect and learn the units that make up speech, starting from a database of speech utterances. These units manifest themselves

as recurrent patterns in the data. The core of our system consists of an algorithm that has shown great potential in discovering a compact set of building blocks into which the input can be decomposed, namely non-negative matrix factorisation (NMF) [6, 7]. By the use of non-negativity constraints, NMF allows only additive (not subtractive) combinations, and thereby it is distinguished from other matrix factorisation techniques such as principal component analysis (PCA).

We have shown [8] that NMF is able to discover the structure that is latent in the speech data. To construct the input matrix of this algorithm, a fixed-length high-dimensional vector representation must be obtained from the utterances, such that the model underlying the latent variable method applies. In this step, phonetic knowledge sources are used to compute phone lattice transition probabilities. Experiments on the TI-Digits database revealed that the basis vectors represent each of the eleven words that are uttered. This illustrates that the lexical information can be extracted automatically.

In this paper, we investigate the amount of data that is needed to obtain a correct set of basis vectors. Instead of using the whole training data set, we select a representative subset as input to our algorithm. This decreases the number of occurrences of the words, and will give us an indication of the learning rate of the system in general. Also, it will reveal which digits are more difficult to acquire.

In the next section, we briefly review the basics of the NMF technique. In section 3, we explain how the NMF input matrix is constructed. The resulting basis vectors are linked to words (or parts of words) in section 4. In section 5, we describe the experiments that are conducted to investigate the required amount of speech data. Finally, conclusions can be found in section 6.

## 2. The NMF algorithm

In this section, we briefly review the theory of non-negative matrix factorisation. Let us consider an  $(n \times t)$  input matrix  $\mathbf{V}$  that contains only elements that are positive or zero. The NMF algorithm looks for an approximate factorisation of  $\mathbf{V}$  into an  $(n \times r)$  matrix  $\mathbf{W}$  and an  $(r \times t)$  matrix  $\mathbf{H}$ :

$$\mathbf{V} \approx \mathbf{W}\mathbf{H} \tag{1}$$

by optimising an objective function under the constraint that all matrix elements should be non-negative. The exact form of the cost function is not very crucial to the success of the algorithm. In this paper we use the divergence criterion [7]:

$$D(\mathbf{V}||\mathbf{W}\mathbf{H}) = \sum_{i,j} \left( [\mathbf{V}]_{ij} \log \frac{[\mathbf{V}]_{ij}}{[\mathbf{W}\mathbf{H}]_{ij}} - [\mathbf{V}]_{ij} + [\mathbf{W}\mathbf{H}]_{ij} \right) \tag{2}$$

The notation  $[\mathbf{A}]_{ij}$  denotes the element in the  $i$ th row and  $j$ th column of matrix  $\mathbf{A}$ . Eq. (1) indicates that each data vector  $[\mathbf{V}]_{:j}$  is written as a linear combination of the columns of  $\mathbf{W}$  weighted by the coefficients  $[\mathbf{H}]_{:j}$ . Since relatively few basis vectors  $[\mathbf{W}]_{:k}$  are used ( $r \ll t$ ), a high quality approximation can only be achieved if the basis vectors discover structure that is latent in the data. Hence,  $\mathbf{W}$  contains a set of recurring patterns, while the columns of  $\mathbf{H}$  indicate when these are active.

It can be shown [7] that the algorithm converges to a local optimum of this objective function by iterating the following update rules for  $\mathbf{W}$  and  $\mathbf{H}$ :

$$[\mathbf{H}]_{k\ell} \leftarrow [\mathbf{H}]_{k\ell} \frac{\sum_i [\mathbf{W}]_{ik} [\mathbf{V}]_{i\ell} / [\mathbf{W}\mathbf{H}]_{i\ell}}{\sum_j [\mathbf{W}]_{jk}} \quad (3)$$

$$[\mathbf{W}]_{ik} \leftarrow [\mathbf{W}]_{ik} \frac{\sum_\ell [\mathbf{H}]_{k\ell} [\mathbf{V}]_{i\ell} / [\mathbf{W}\mathbf{H}]_{i\ell}}{\sum_m [\mathbf{H}]_{km}} \quad (4)$$

Because these update rules are multiplicative and do not change the sign of  $\mathbf{W}$  or  $\mathbf{H}$ , it is sufficient to initialise the matrix elements of  $\mathbf{W}$  and  $\mathbf{H}$  to strictly positive values in order to satisfy the non-negativity constraints. Usually, sparse representations are produced such that the input can be interpreted in terms of a few components. Another interesting property is that the algorithm is able to handle feature representations of large dimensionality. Note that NMF with divergence criterion is closely related to the well-known Expectation-Maximisation (EM) algorithm with multiplicative update rules.

### 3. Constructing the input matrix

In this section, we explain how the input matrix  $\mathbf{V}$  is constructed. For our purposes, the columns of  $\mathbf{V}$  contain the fixed-length high-dimensional vector representation of the utterances. Hence the value of  $t$  equals the number of speech sentences. This vector representation is obtained from the corresponding phone networks that result from an acoustic-phonetic decoding.

Given the acoustic features  $Z$  of the incoming signal, the network of most probable phone strings  $F$  is determined using the first layer of the FLVoR architecture [9]. The employed knowledge sources are an acoustic model  $p(Z|F)$  and a phone transition model  $p(F)$ . In this directed acyclic graph, the arcs correspond to phones and the nodes impose time and context-dependency constraints. The acoustic score that is associated with each arc is transformed to a posterior probability. The ratio of the acoustic model likelihood and the language model likelihood is adjusted as to optimise the mutual information between the arc probabilities and the true phone identity.

The information that is contained in each phone network is summarised into a  $n$ -dimensional vector. The first  $n_m$  elements of the vector contain the accumulated posterior probability of the individual phones  $\phi$ ,

$$p(\phi) = \sum_{\{\alpha: h(\alpha)=\phi\}} p(\alpha) \quad (5)$$

The remaining  $n_b$  elements contain the probability of every two consecutive phones  $\phi$  and  $\psi$ , accumulated over the network:

$$p(\phi, \psi) = \sum_{\{\alpha: h(\alpha)=\phi\}} \sum_{\{\beta: h(\beta)=\psi\}} p(\alpha) p(\beta) \Delta_{\alpha\beta} \quad (6)$$

in which  $h(\alpha)$  and  $h(\beta)$  return the phone identity, and  $p(\alpha)$  and  $p(\beta)$  the posterior probability of the arcs  $\alpha$  and  $\beta$ , respectively. If the start node of  $\beta$  is equal to the end node of  $\alpha$ , then  $\Delta_{\alpha\beta}$

is the inverse of the probability of the common node, else it is zero. This node probability is given by the sum of the posterior probabilities of the incoming (or outgoing) arcs of the corresponding node. The value of  $n_m$  equals the number of phone identities, while the value of  $n_b$  equals the square of  $n_m$ .

These  $n$ -dimensional vectors form the columns of the matrix  $\mathbf{V}$ . Apart from using the information about the  $n_m$  monophones and the  $n_b$  biphones in the phone network, also the distant biphones could be incorporated in these vectors. Since the elements of  $\mathbf{V}$  represent accumulated probabilities or co-occurrence counts, they will never become negative, as is required for applying the NMF algorithm.

### 4. Interpretation of the basis vectors

After convergence of the NMF algorithm, the matrix  $\mathbf{W}$  contains a set of basis vectors that can be interpreted in terms of recurrent speech units. For a small vocabulary task, these units correspond to the words in the lexicon when the value of  $r$  is chosen properly. To determine the value of  $r$ , the NMF objective function can be plotted versus the number of basis vectors  $[\mathbf{W}]_{:k}$  that are extracted. It was observed [8] that the slope of the curve changes at  $r$  equal to the number of words that are uttered in the database.

The assignment of labels to basis vectors is done automatically. Let us consider the set of labels that represents all possibly discovered speech units. In our case, this set contains the lexicon of the Resource Management (RM) database, augmented with all corresponding sub-sequences of phones. For instance, the word ‘any’ gives rise to the following labels: {‘SIL EH N IY SIL’, ‘SIL EH N IY’, ‘SIL EH N’, ‘SIL EH’, ‘EH N IY SIL’, ‘EH N IY’, ‘EH N’, ‘N IY SIL’, ‘N IY’, ‘IY SIL’}. Since RM contains not only the 11 digits but also more than 900 other words, the quality of the basis vector must be high in order to assign a correct label.

For each label  $b$  the  $(n \times 1)$  vector  $\mathbf{c}_{:b}$  is constructed that contains the number of occurrences of every monophone and every biphone in this label. Both  $\mathbf{c}_{:b}$  and  $[\mathbf{W}]_{:k}$  are given unit L1-norm. Then, basis vector  $[\mathbf{W}]_{:k}$  is assigned to the label  $b$  for which the Kullback-Leibler divergence between the vectors  $\mathbf{c}_{:b}$  and  $[\mathbf{W}]_{:k}$ :

$$D(\mathbf{c}_{:b} || [\mathbf{W}]_{:k}) = \sum_{\{i: \mathbf{c}_{ib} \neq 0\}} \left( \mathbf{c}_{ib} \log \left( \frac{\mathbf{c}_{ib}}{[\mathbf{W}]_{ik} + \epsilon} \right) \right) \quad (7)$$

is minimal. Here,  $\epsilon (= 10^{-50})$  is a small number.

An interesting property is that the basis vectors  $[\mathbf{W}]_{:k}$  automatically group different pronunciation variants of a word into one vector. For instance, the phone transitions of ‘W AH N’, ‘W AA N’ and ‘W AO N’ are all present in the basis vector of the digit ‘one’. However, because the first variant is observed more frequently, its probability dominates.

### 5. Experiments

In this section, we illustrate the performance of the system on a small vocabulary, speaker independent database. The speech data are taken from the TI-Digits database [10] which contains recordings of male and female US-American adults. These data are downsampled to 16 kHz. The training set consists of 6159 connected digit sequences of length 1 through 7 ( $\approx 212$  min. of speech data in total).

First, a random selection of  $U$  utterances out of the 6159 training utterances is generated. This step is repeated 50 times

Table 1: Minimum, maximum and average number of digit occurrences over 50 trials in which 100 (resp. 1600) utterances are selected from the TI-Digits training set.

digit	100 utterances			1600 utterances		
	number of occurrences			number of occurrences		
	min	max	avg	min	max	avg
one	27	57	38.92	564	669	617.88
two	28	49	38.58	558	672	614.84
three	26	55	39.92	568	654	607.54
four	27	52	38.30	578	658	614.54
five	27	56	37.90	562	645	599.80
six	24	61	39.18	572	661	619.40
seven	25	54	37.48	560	669	610.74
eight	27	53	38.60	556	664	610.66
nine	25	47	36.92	553	662	599.86
oh	22	52	38.20	539	671	613.74
zero	20	50	37.10	565	662	613.72

such that the differences in length of the utterances and the differences in the number of occurrences of each of the digits can be averaged out.

Some statistics of these 50 trials are summarised in table 1 for  $U = 100$  and  $U = 1600$ . For each digit, we calculated the number of occurrences per trial. The corresponding minimum, maximum and average number of occurrences over the 50 trials are given. As can be seen, none of the digits is significantly disadvantaged w.r.t. the others. This random selection is done for several values of  $U$  in the range [100, 3500].

### 5.1. Processing the input

For each of these subsets of speech data, an input matrix  $\mathbf{V}$  is constructed with  $U$  columns. Because of the small vocabulary of the task, the phone networks are generated without too much prior information. A set of 43 different phone identities is used, including the phone ‘SIL’ (= silence). The acoustic-phonetic decoding makes use of a state-of-the-art acoustic model  $p(Z|F)$  that consists of a HMM with cross-word context-dependent tied states (GMMs). This model is trained on the Wall Street Journal (WSJ0 plus WSJ1) database such that it is general enough. For the phone transition model  $p(F)$ , a unigram is estimated on the same database.  $\mathbf{W}$  and  $\mathbf{H}$  are initialised with random positive values. Then, a set of basis vectors is extracted by the NMF algorithm with divergence criterion. The update rules (eq. (3) and (4)) are applied for 1500 iterations and the resulting matrix  $\mathbf{W}$  is evaluated as explained in section 4.

Sometimes, not all digits are found because the NMF algorithm gets stuck in a local minimum (e.g. the phone transitions of two digits are present in one basis vector, while the remaining basis vector is used to model transitions between words). It was observed that the algorithm can not always recover from this situation even when the number of iterations is further increased. A way to circumvent this situation, is to start from multiple initialisations of the matrices  $\mathbf{W}$  and  $\mathbf{H}$ . After convergence, only the decomposition with the lowest value of the objective function (divergence) is retained. In our case, seven different initialisations of the algorithm are considered.

### 5.2. Evaluating the results

For each value of  $U$ , the number of discovered digits is averaged over the 50 trials. The corresponding average number of

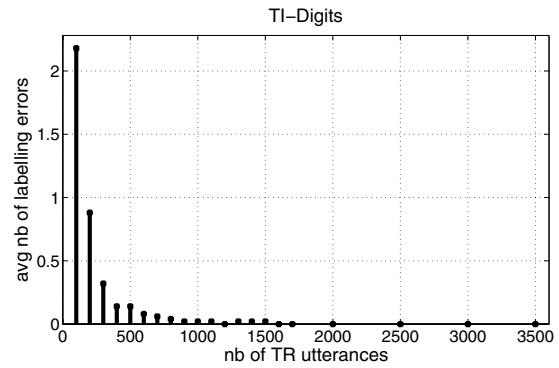


Figure 1: Average number of labelling errors as a function of the number of training utterances that are selected from the TI-Digits training set. Averaging is done over 50 trials.

labelling errors is plotted in figure 1. A label that differs only by a circular shift from a digit embedded in silence, is considered correct because our representation is invariant to these shifts. For instance, the label ‘SIL S EH V AH N SIL’ and the label ‘S EH V AH N SIL S’ are equivalent in their phone transitions, and hence both are assumed correct. Currently, the labelling algorithm does not try to detect which phone has to come first (resp. last) in the sequence, although this could be determined based on the phone transitions to (resp. from) every phone.

As can be seen in figure 1, the number of speech utterances can be decreased to  $U = 1600$  without affecting the performance of the system. When the amount of training data is smaller, some of the 50 trials contain one or more labelling errors. At  $U = 1000$ , the average number of errors is 0.02, meaning that for one of the 50 trials one of the digits has not been labelled correctly. From table 1, we can see that on average every digit is approximately 610 times uttered when  $U = 1600$ . In total, an average of 55 min. of speech data are used for each trial in this case. At  $U = 200$ , approximately 7 min. of speech are presented to our system, while on average less than one of the eleven digits is not correctly discovered (typically the ‘oh’).

Table 2 illustrates which digits are more difficult to discover. For each digit, it shows the percentage of trials in which this digit is correctly labelled in case  $U$  utterances are selected from the TI-Digits training set, with  $U$  equal to 100, 200, 300, 500, 700 or 1600. For instance, the label ‘OW’ is present in the labelling result of only 11 out of 50 trials (22%) at  $U = 100$ . Because ‘oh’ is a short word (only one phone transition) that is also a part of the word ‘zero’, it is easily omitted in the set of basis vectors. At  $U = 700$ , the digits ‘seven’ and ‘oh’ are sometimes omitted or only partly discovered. In this respect it can be noted that at  $U = 700$  the minimum number of occurrences of the digit ‘seven’ over the 50 trials is the smallest of all digits (except for the digit ‘zero’), although its average number of occurrences is comparable to the others. The relatively low score for digit ‘seven’ is caused by an incorrect label ‘SIL S EH V AH N S’. This label is assigned because the transition between the digits ‘7-7’ or ‘7-6’ is present in the corresponding basis vector with a non-negligible probability.

### 5.3. Comparison with human performance

Finally, we can compare the learning rate of our system with that of humans. In [1], the artificial language consisted of 6 trisyllabic ‘words’. Adults were exposed to an unbroken 21 min.

Table 2: Percentage of trials in which the corresponding digit is correctly labelled in case  $U$  utterances are selected from the TI-Digits training set.

digit	number of TR utterances ( $U$ )					
	100	200	300	500	700	1600
one	96	100	100	100	100	100
two	76	96	100	100	100	100
three	98	100	100	100	100	100
four	92	100	100	100	100	100
five	92	100	100	100	100	100
six	94	100	100	100	100	100
seven	80	76	86	92	96	100
eight	70	86	98	100	100	100
nine	74	94	98	100	100	100
oh	22	62	86	94	98	100
zero	88	98	100	100	100	100

corpus with no prosodic or acoustic markers of word boundaries. The transitional probabilities inside words were relatively high, while those spanning a word boundary were relatively low, as is the case for real languages. In the 2-alternative forced-choice test, subjects had to choose which 3-syllable sequence sounded more familiar (words versus non-words). They could recognise 76% correct. Although the experimental conditions differ considerably, figure 1 shows that our system has on average 0.08 labelling errors at  $U = 600$  (approx. 21 min. of speech data). This is equivalent to an error rate of 0.7%.

Clearly, the setup of both experiments differs and we will not be able to make a fair comparison. As we will point out below, assessing the learning capabilities would only be possible at the systemic level because of major differences in the architecture. We identified the following differences:

- Our experiment handles voices from different speakers, which is thought to be more difficult.
- The acoustic confusability of our vocabulary is larger (10 mostly monosyllabic digits versus 6 trisyllabic words of which only one pair has 2 syllables in common).
- Our learning algorithm uses all data in batch mode, i.e. has direct access to the phone transition probabilities of the 21 min. of speech. The human brain can probably not hold the same level of detail in its episodic memory. In this sense, an incremental version of NMF would be a more accurate simulation of human acquisition of words.
- In [1], an acoustic test is designed to verify if the acquired words are correct. In our experiment, we opted for a symbolic test. The result of an acoustic word recognition test would be degraded by the inferior speech recognition performance of the HMM. We would not only be measuring the patterns discovery error rate. Word recognition error rates based on phonetic models are non-negligible relative to the 0.7% error rate observed.
- The error rates are measured differently. To simulate an experiment as in [1], a decision threshold on the divergence (eq. (7)) should be defined for every digit. Experiments show that in case of a labelling error at  $U = 600$ , the canonical transcription is always in second place. Hence, it is possible to find a word-dependent threshold (i.e. the KL-divergence of the canonical transcription plus an epsilon) that will lead to no more than 0.7% of error in an in-vocabulary test as in [1]. Here, we also need to point out that we use thousands of intruders.

Hence, the experimental evidence seems to point out that this learning algorithm achieves superhuman performance in

identifying word-sized acoustic patterns in the audio stream.

## 6. Conclusions

In this paper, we have presented an unsupervised technique to discover the (word-sized) units that are present in speech utterances. We explored the use of phone lattice transition probabilities to obtain a fixed-length high-dimensional vector representation of the speech utterances. The non-negative matrix factorisation (NMF) algorithm was applied to discover structure that is latent in the input matrix. In case of a small vocabulary task, the obtained basis vectors correspond to words.

We also investigated the amount of data that is needed to obtain a correct set of basis vectors. Experiments on the TI-Digits database revealed that the number of training utterances can be decreased to 1600 without affecting the performance of the system. In this case, every digit is uttered approximately 610 times and in total 55 min. of speech data is used. These results give us an indication of the learning rate of the system in general. The digit ‘oh’ appeared to be the most difficult to acquire when the amount of observations is small.

## 7. Acknowledgements

This research was funded by the IWT - SBO project ‘SPACE’ (project no. 040102), by ‘Research Fund (Onderzoeksfonds) K.U.Leuven’ (project no. OT/03/32/TBA), and by the European Commission under contract FP6-034362.

## 8. References

- [1] J. Saffran, E. Newport, and R. Aslin, “Word segmentation: The role of distributional cues,” *Journal of Mem. and Lang.*, vol. 35, no. 0032, pp. 606–621, 1996.
- [2] J. Saffran, R. Aslin, and E. Newport, “Statistical learning by 8-month-old infants,” *Science*, vol. 274, no. 5294, pp. 1926–1928, 1996.
- [3] M. Ostendorf, “Moving beyond the ‘beads-on-a-string’ model of speech,” in *Proc. ASRU 1999*, Dec. 1999.
- [4] A. Tsopanoglou and N. Fakotakis, “Selection of the most effective set of subword units for an HMM-based speech recognition system,” in *Proc. EUROSPEECH*, Sept. 1997, pp. 1231–1234.
- [5] M. Bacchiani and M. Ostendorf, “Joint lexicon, acoustic unit inventory and model design,” *Speech Comm.*, vol. 29, no. 2, pp. 99–114, Nov. 1999.
- [6] D. Lee and H. Seung, “Learning the parts of objects by non-negative matrix factorization,” *Nature*, vol. 401, pp. 788–791, 1999.
- [7] D. Lee and H. Seung, “Algorithms for non-negative matrix factorization,” *Advances in Neural Information Processing Systems*, vol. 13, pp. 556–562, 2001.
- [8] V. Stouten, K. Demuyne, and H. Van hamme, “Discovering phone patterns in spoken utterances by non-negative matrix factorisation,” *IEEE SP Letters*, 2007, (submitted for publication).
- [9] K. Demuyne, T. Laureys, D. Van Compernelle, and H. Van hamme, “Flavor: a flexible architecture for LVCSR,” in *Proc. EUROSPEECH*, Sept. 2003, pp. 1973–1976.
- [10] R. Leonard, “A database for speaker-independent digit recognition,” in *Proc. ICASSP*, 1984, pp. 42.11/1–4.