Astrid Aune

Word Discovery from Unsegmented Speech

Master's thesis in MTELSYS
Supervisor: Giampiero Salvi
June 2020

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

NTNU
Norwegian University of Science and Technology

Astrid Aune

Word Discovery from Unsegmented Speech

Master’s thesis in MTELSYS
Supervisor: Giampiero Salvi
June 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Electronic Systems

NTNU
Norwegian University of
Science and Technology

Abstract

The goal of the thesis is to discover words in unsegmented speech in an unsupervised way. We experimented with two methods of latent *Factor Analysis* (FA); the *Non-Negative Matrix Factorization* (NNMF) and the *Beta Process Factor Analysis* (BPFA). By looking at the transitions between the subword units (letters or phones) and finding recurring patterns, these techniques are able to discover and estimate the words present in the utterances. The main difference between the two algorithms is that NNMF needs prior knowledge of the vocabulary size, whereas BPFA is able to infer this knowledge as well as the estimations of the words from the data set. We tested the methods using four different types of data representation, based on transitions between subword units with different complexities. The results show us that both NNMF and BPFA perform well, as long as the vocabulary size is small enough. For larger vocabularies, the most complex representation performs better than the simpler ones. However, for small vocabularies, using the 1st-order subword unit transitions is often a sufficient data representation.

Sammendrag

Hensikten til denne oppgaven er å finne ord i sammenhengende tale ved hjelp av ikke-veiledet maskinlæring. Det ble testet med to metoder av latent *faktoranalyse*; *Non-Negative Matrix Factorization* (NNMF) og *Beta Process Factor Analysis* (BPFA). Ved å se på overganger mellom basisenhetene (bokstaver eller foner) og oppdage gjentagende mønstre, klarer disse to metodene å finne estimater av ordene som befinner seg i talesekvensene. Den største forskjellen mellom de to algoritmene er at NNMF trenger å vite antall ord på forhånd, mens BPFA klarer å estimere dette tallet i tillegg til estimatene av ordene i datasettet. Metodene ble testet med fire ulike metoder å representere talesekvensene på basert på overganger mellom basisenhetene av ulik kompleksitet. Resultatene viser oss at både NNMF og BPFA presterer bra så lenge størrelsen på vokabularet er liten nok. For de større vokabularene presterer den mest komplekse datarepresentasjonen bedre enn de enklere representasjonene. Men for mindre vokabularer er det ofte tilstrekkelig med den enkleste datarepresentasjonen som kun ser på 1.ordens overganger.

Contents

List of Figures

List of Tables

Acronyms

ASR *Automatic Speech Recognition*. 10, 28

BeP *Bernoulli Process*. 15, 16

BNP *Bayesian Non-Parametric*. 11, 12, 14, 39

BP *Beta Process*. 12, 14–16

BPFA *Beta Process Factor Analysis*. v, viii, 11, 12, 14, 16, 18–20, 22, 24, 25, 37–40

CRP *Chinese Restaurant Process*. viii, 12, 13

DPGMM *Dirichlet Process Gaussian Mixture Model*. 10

DTW *Dynamic Time Warping*. 18

E2E *End-to-End*. 10

FA *Factor Analysis*. v, viii, 13, 14, 22, 39, 40

IBP *Indian Buffet Process*. viii, 12–14, 16

IPA *International Phonetic Alphabet*. 29, 44

KLD *Kullback-Leibler Divergence*. 14, 18, 23

LCM *Latent Class Model*. 12

LFM *Latent Feature Model*. 12, 13

LSA *Latent Semantic Analysis*. 10

MSE *Mean Square Error*. 14

NNMF *Non-Negative Matrix Factorization*. v, viii, 10–14, 18, 20, 22–24, 30, 31, 34, 35, 37–40, 45–52

PCA *Principal Component Analysis*. 10

SVD *Singular Value Decomposition*. 12, 16, 26

1 Introduction

Today, most *Automatic Speech Recognition* (ASR) systems make use of prior knowledge of audiology, phonology and linguistics. These are variables that change depending on language, accent, and speaker, which makes these resources expensive. To replace these traditional ASR systems, *End-to-End* (E2E) systems have been developed. E2E ASR is a single integrated approach with a much simpler training pipeline which reduces training and decoding time. However, they require enormous amounts of data annotated in order to perform as well as traditional ASRs [1]. It is thus desirable to have an ASR system which is able to recognize speech without this prior knowledge. Toddlers are able to automatically learn the acoustic, lexical and grammatical patterns of a language without any prior knowledge. Should it not be possible for machines to do the same?

Unsupervised speech recognition have been a hot topic for years and still is. For example, the *Zero Resource Speech Challenge* focuses on speech recognition without any prior linguistic expertise (e.g. transcriptions). This is a popular challenge where many participants submits their models. So far three of these challenges have been held, one in 2015 [2], one in 2017 [3] and one in 2019 [4].

There are many different aspects in modelling language acquisition ranging from *phonetic* to *lexical*, *grammatical* and *semantic*. In [5, 6] the authors focus on subword modelling from untranscribed speech. Subword modelling means constructing a representation for the speech sounds (e.g. phones or phonemes) [3]. The approach chosen both in [5] and [6] was build around the *Dirichlet Process Gaussian Mixture Model* (DPGMM) which was used to cluster speech feature vectors into a dynamically sized set of classes. Another aspect of language is the semantics, i.e. the meaning of the words and the relation between them. In [7, 8] robots were used, and the aim was for them to not only recognize what is said, but also understand it. More specifically, the model creates links between the speech utterances and the involved objects and actions. In this report, however, the focus will be on the lexical aspect of language modelling. That is, the focus is on the discovering of lexical items from transcribed speech disregarding any grammar and semantics.

For word discovery in speech, a proposed method is the *Non-Negative Matrix Factorization* (NNMF) [9, 10]. The NNMF is able to capture structures and other information hidden in the data. More specifically in this case, NNMF can be used to discover words in unsegmented speech in an unsupervised way by finding recurring patterns in the speech. In contrast to for example *Latent Semantic Analysis* (LSA) and *Principal Component Analysis* (PCA), the results of the NNMF can be given a probabilistic interpretation. This approach is also less complex than other alternatives, which is favorable. An extension to the NNMF has also been proposed [11]. *Beta Process Factor Analysis* (BPFA) is a *Bayesian Non-Parametric* (BNP) extension to factor analysis. The advantage of BPFA over NNMF is its ability to estimate the number of words in the data set while estimating the representations of each word, where NNMF needs prior knowledge of the number of words. Both of these are general methods used in various other applications than word discovery, for instance NNMF has been widely used in image processing [12–15].

In this report, we will study the properties of both NNMF and BPFA, and their abilities of word discovery in unsegmented speech. That is, we have a data set of utterances with a representation in terms of subword units. These utterances consists of one or several words, which we want to recover from the subword units. For example, from the utterance “onetwothree”, we want to recover the words “one”, “two” and “three”. The chosen methods will be tested on a data set consisting of 11 unique digits, taken from the TI-Digits database [16], in addition to an artificially constructed data set with varying size of the vocabulary. Multiple test cases will be executed to test different variables, inclusive of choice of utterance representation.

2 Theory

In order to understand the NNMF- and BPFA-methods, we will first look at the concepts of *Latent Class Model* and *Latent Feature Model*. Then NNMF will be reviewed, before we will look at the *Beta Process* and BPFA. At last, we will look at *Singular Value Decomposition*, which is a powerful concept to be used as a initialization of the factor analysis.

2.1 Latent Class and Latent Feature Models

In *Latent Class Model* (LCM) each data observation is assumed to belong to a class. Given N observations and K classes, we can represent the model as a binary matrix $Z \in \mathbb{R}^{N \times K}$, where $Z[n,k] = 1$ if an observation v_n belongs to class c_k. If the number of classes, K, is known on beforehand, we have a finite model. However, in some applications K is not known. For these cases we can implement *Bayesian Non-Parametric* (BNP) latent class models. This approach assumes that there is a infinite number of classes, i.e. $K \to \infty$, while defining the prior over these classes $P(\boldsymbol{c})$ to favor only a small group of classes [17].

This prior is called *Chinese Restaurant Process* (CRP). Imagine a restaurant with an infinite number of tables and a sequence of customers coming in and sitting down at one of the tables. The first customer comes in and sits at the first table. Then the second customer comes in and sits at the first table with probability $\frac{1}{1+\alpha}$ and the second table with probability $\frac{\alpha}{1+\alpha}$, where α is positive real. The n'th customer sits at the occupied tables with probability proportional to the number of customers already sitting at the table, and the next unoccupied table with probability proportional to α [17]. With this prior, the first tables have a higher prior probability and are favored in the random partitioning.

On the other hand, *Latent Feature Model* (LFM) model each observations as a composition of different features. With this model, the elements of the binary matrix Z is equal to 1, if the observation v_n possesses feature c_k. The major difference between LCM and LFM, is that in the class models each observation is assigned only one component, while in feature models each observation is assigned multiple components.

With LFM too, the number of features, K, is not always known and the BNP approach assumes an infinite number of features. LFM also have a prior which favors a small group of the infinitely many features, similarly to CRP. This prior, however, is called *Indian Buffet Process* (IBP). The buffet has an infinite number of dishes (features), and a sequence of costumers enters the buffet and samples the dishes. The first costumer samples the first

Poisson(α) dishes. The n'th costumer samples the previously sampled dishes with probability m_k/n, where m_k is the number of previous costumers sampling dish k. The costumer then samples Poisson(α/n) new dishes [17]. Draws from both CRP and IBP are illustrated in Figure 2.1.

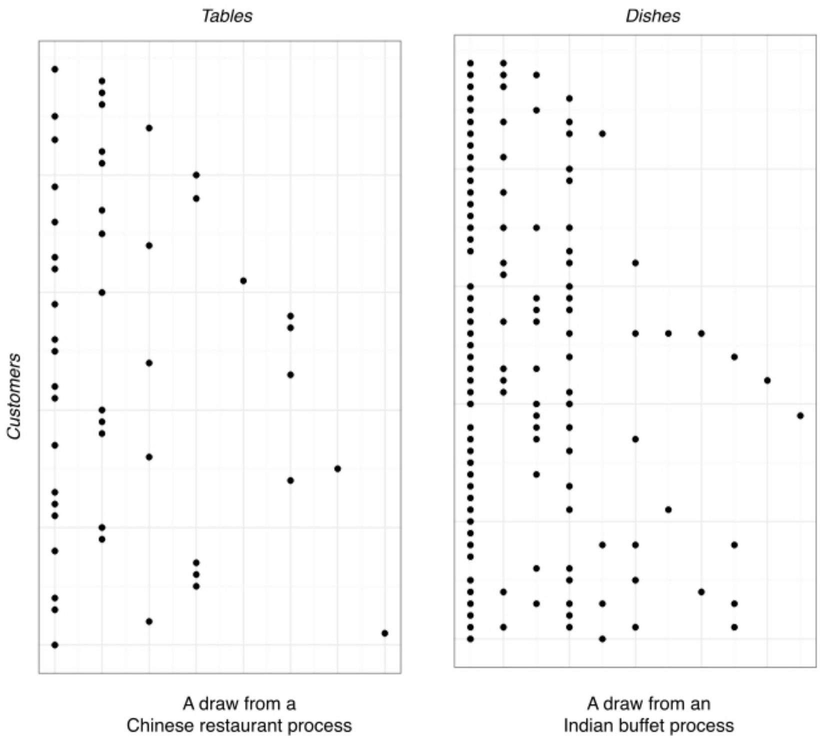


Figure 2.1: Draws from *Chinese Restaurant Process* (CRP) and from *Indian Buffet Process* (IBP) [17].

A popular method of LFM is classical *Factor Analysis* (FA), where the number of components, K, is known. Assume we have N observations, $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N$, all of them of dimension M, $\boldsymbol{v}_n = [v_{1n}, \ldots, v_{MN}]$. We have then the matrix $V = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N]$ which can be assumed to be generated from a noisy weighted combination of latent features, such that

$$\boldsymbol{v}_n = W \cdot \boldsymbol{h}_n + \boldsymbol{\varepsilon}_n, \tag{2.1}$$

where $W \in \mathbb{R}^{M \times K}$ is a feature loading matrix which weights how much feature k influences observation dimension m. $\boldsymbol{h}_n$ is a K-dimensional vector expressing the activity of each feature in the observation, and $\boldsymbol{\varepsilon}_n$ is the noise [17]. This is illustrated as a matrix factorization for all N observations in Figure 2.2. An algorithm in this field is is the *Non-Negative Matrix Factorization* (NNMF) which will be discussed further in Section 2.2.

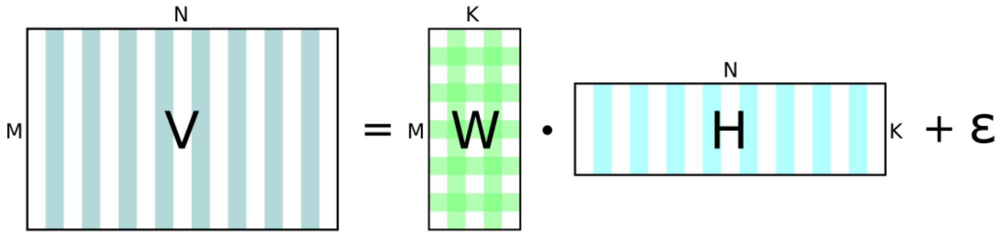


Figure 2.2: Illustration of *Factor Analysis* (FA)

The FA can be extended to a BNP version by defining the feature loading matrix as $W \circ Z$, such that

$$\boldsymbol{v}_n = (W \circ Z) \cdot \boldsymbol{h}_n + \boldsymbol{\varepsilon}_n, \tag{2.2}$$

where Z is a binary *mask* matrix of the same size as W and $\circ$ denotes element-wise multiplication. Z can then be initialized as IBP [17]. The algorithm *Beta Process Factor Analysis* (BPFA) is a realization of this and is based on the IBP and the *Beta Process* (BP). BPFA will also be discussed further later in Section 2.4.

2.2 Non-Negative Matrix Factorization

Assume you have a non-negative data matrix $V \in \mathbb{R}^{M \times N}$, where each element is $v_{mn} \geq 0 \ \forall \ m \in [1, M], n \in [1, N]$. The objective of the *Non-Negative Matrix Factorization* (NNMF) is to decompose V into two non-negative matrices $W \in \mathbb{R}^{M \times K}$ and $H \in \mathbb{R}^{K \times N}$ [18], such that

$$V \approx W \cdot H. \tag{2.3}$$

W and H are found by optimizing a cost function, under the constraint that all elements of W and H are non-negative. There are several possibilities for the cost function, for example the *Kullback-Leibler Divergence* (KLD), *Frobenius norm*, *Itakura-Saito divergence*, or the *Mean Square Error* (MSE) [18].

The *Kullback-Leibler Divergence* (KLD) criterion has been proven to be a good choice for NNMF [9], and is defined as a measure of the difference between two probability distributions [19]. Given two discrete probability distributions P and Q defined on the same probability space $\mathcal{X}$, the KLD is defined to be

$$D_{KL}(Q||P) = \sum_{x \in \mathcal{X}} P(x) \log \frac{P(x)}{Q(x)}. \tag{2.4}$$

In the case where we do not have two probability distributions but two matrices X and Y, the divergence can be expressed as

$$D_{KL}(X||Y) = \sum_{i,j} \left(x_{ij} \log \frac{x_{ij}}{y_{ij}} - x_{ij} + y_{ij} \right). \tag{2.5}$$

When $\sum_{i,j} x_{ij} = \sum_{i,j} y_{ij} = 1$, the matrices X and Y can be regarded as normalized probability distributions and Eq.(2.5) is equivalent to Eq.(2.4) [20].

2.3 Beta Process

The *Beta Process* (BP) was first introduced in [21], and is a "distribution on distributions" for random measures with weights between 0-1 [17]. Assume we have a measurable space Ω, and $\mathcal{B}$ its σ-algebra. Let G_0 be a continuous probability measure of on $(\Omega, \mathcal{B})$ and α a positive scalar. For all disjoint, infinitesimal partitions of Ω, $\{B_1, \dots, B_K\}$, the *Beta Process* $G \sim BP(\alpha G_0)$ is defined as;

$$G(B_k) \sim \text{Beta}(\alpha G_0, \alpha(1 - G_0(B_k))), \tag{2.6}$$

where $\text{Beta}(\cdot)$ denotes the Beta distribution, and with $K \to \infty$ and $H(B_k) \to 0$ for $k = 1, \dots, K$ [22]. This *Beta Process* can also be written as

$$G(\omega) = \sum_{k=1}^{\infty} \pi_k \delta_{\omega_k}(\omega) \tag{2.7}$$

with $G(\omega_k) = \pi_k$. π is then used as a parameter of the *Bernoulli Process* (BeP). Let the vector $\boldsymbol{z}_i$ be infinite and binary with the k'th value, z_{ik}, generated by

$$z_{ik} \sim \text{Bernoulli}(\pi_k) \tag{2.8}$$

A new measure $X_i(\omega) = \sum_k z_{ik} \delta_{\omega_k}(\omega)$ is then drawn from the *Bernoulli Process* $X_i \sim BeP(G)$ [22].

In a *Beta Process*, K is assumed to infinitely large, i.e. $K \to \infty$. However, a sufficiently large number for K can be used as a finite approximation [22]. This finite approximation can be written as;

$$G(\omega) = \sum_{k=1}^{K} \pi_k \delta_{\omega_k}(\omega) \tag{2.9}$$
$$\pi_k \sim \text{Beta}(a/K, b(K-1)/K)$$
$$\omega_k \overset{\text{iid}}{\sim} G_0$$

where we have introduced two new scalar parameters, a and b, to the *Beta Process*. The vectors $\boldsymbol{z}_i$ are then drawn from a finite *Bernoulli Process* parameterized by G [22].

2.4 Beta Process Factor Analysis

An extension to the factor analysis is the *Beta Process Factor Analysis* (BPFA), and is based on IBP and Beta priors. The goal of BPFA is to optimize the matrices $H \in \mathbb{R}^{M \times K}$, and $Z \in \mathbb{R}^{K \times N}$ such that they best describe the input matrix $V \in \mathbb{R}^{M \times N}$ in the form:

$$V = HZ + \varepsilon, \tag{2.10}$$

where Z is a binary matrix. The matrices H and Z are modeled as N draws from the *Bernoulli Process*, parameterized by the *Beta Process* G [22]. That is;

$$\begin{aligned} z_{ik} &\sim \text{Bernoulli}(\pi_k) \qquad (2.11) \\ \pi_k &\sim \text{Beta}(a/K, b(K-1)/K) \\ h_k &\overset{\text{iid}}{\sim} G_0 \end{aligned}$$

for observation $i = 1, \ldots, N$ and latent feature $k = 1, \ldots, K$. In BPFA, the probability measure G_0 are often defined to be multi-variate normal distribution [22].

In order to construct a more accurate model for the factor analysis, we also include a weight matrix W of the same size as Z [22], such that

$$V = H(Z \circ W) + \varepsilon. \tag{2.12}$$

2.5 Singular Value Decomposition

Singular Value Decomposition (SVD) is a powerful concept of linear algebra and is often the choice of method for solving most linear least-squares problems [23]. Given an arbitrary matrix $A \in \mathbb{R}^{M \times N}$, it can be decomposed into three factors such that

$$A = USV^T \tag{2.13}$$

where $U \in \mathbb{R}^{M \times M}$ and has columns corresponding to the eigenvectors of AA^T. Similary, the columns of $V \in \mathbb{R}^{N \times N}$ corresponds to the eigenvectors of A^TA. $S \in \mathbb{R}^{M \times N}$ is a diagonal matrix of the form;

$$S = \begin{bmatrix} \sigma_1 & & & & \\ & \ddots & & \text{\huge 0} & \\ & & \sigma_r & & \\ & & & 0 & \\ & \text{\huge 0} & & & \ddots & \\ & & & & & 0 \end{bmatrix} \tag{2.14}$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$ and $r = \text{rank}(A)$. $\sigma_1, \ldots, \sigma_r$ are the square roots of the eigenvalues of $A^T A$ and are called the *singular values* of A [23].

3 Related Work

Non-Negative Matrix Factorization (NNMF) was first introduced in [24] by Paatero and Tapper in 1994, but it was not before 1999, when Lee and Seung published their article [25], that NNMF became a popular and widely used method. NNMF has been used in many applications, especially within image processing [12–15], where the input matrix V is interpreted as the image. The NNMF is a good choice in for example medical imaging, as NNMF is designed to capture alternative structures inherent in the data and possibly to provide more biological insight which is hidden for us in the original image [15]. Lately, NNMF has also been used in speech recognition in order to discover words from continuous speech [9, 10]. Other methods for unsupervised word discovery have also been proposed, for example a *multigram model* [26] and a variation of *Dynamic Time Warping* (DTW) [27]. These use clustering instead of a mathematical factorization like NNMF.

In [9], the authors aimed to build a system that retrieves the phone patterns within the speech input without prior knowledge of a pre-defined set of patterns linked to a fixed and pre-defined set of concepts. As the core of their system they used the NNMF method. First, a dense phone network is constructed from each utterance, given the set of subword units (phones). The NNMF-algorithm with the *Kullback-Leibler Divergence* criterion as the objective function was then applied to find the recurring patterns in the data. For the test, they used speech utterances taken from the TI-Digits database [16] which contains only 11 unique words. For this small vocabulary, they were able to obtain basis vectors that corresponded to each of the words. Pronunciation variants of each word was also automatically discovered. These were promising results, and they concluded that this opens up perspectives to deviate from the conventional beads-on-a-string approach to model speech.

In [22] a non-parametric extension to the factor analysis is proposed, i.e. *Beta Process Factor Analysis* (BPFA) which is based on Beta priors. The authors tested this method in several different applications, but not in speech recognition. This method was also tested [28], where they used it in gene-expression analysis. To use a non-negative version of BPFA to discover words in unsegmented speech was proposed in [11]. The authors argued that this method is more effective than NNMF because it can estimate the number of words in the data set at the same time as it estimates the representation of each word. They concluded that the BPFA manages to find all the words a small vocabulary case. In fact, comparing their results for BPFA with their results using NNMF shows that the BPFA is able to not only find all the correct words but also the correct number of words in the data set. When the vocabulary increases and includes words of the same origin, e.g. “grasp”, “grasping” and “grasped”, the

BPFA merges these words into the same representation. Still, BPFA has proven itself to be a good solution for unsupervised word discovery.

4 Method

Firstly, we will in this section describe the different approaches for the data representation and how the utterances in the data set are converted into the non-negative matrix V. Then comes the implementations of *Non-Negative Matrix Factorization* (NNMF) and *Beta Process Factor Analysis* (BPFA)[1]. The method for the initialization is the same for both of the factorization algorithms and will also be described, before the metrics used to evaluate the results are discussed.

4.1 Data Representation

We have a set of utterances with a corresponding set of transcriptions. Each utterance varies in length whith some consisting of only one word and other multiple words, and some words are also longer than others. However, these words are assumed to be unknown, the only thing we do know is the subword units the utterances are represented by in their transcriptions, either from text (letters) or speech (phones). Based on these subword units, we want to recover the words. Assuming the number of subword units is U, the utterances are each represented by a sequence of such symbols.

As mentioned, these sequences of subword units are of varying length, but when using factor analysis we want a fixed-length representation of each sequence. A good option is to represent the sequences as transitions from one subword unit to another, as was done in [9, 11]. The information in each utterance is summarized into the columns of $V \in \mathbb{R}^{M \times N}$, where M is the number of possible subword unit transitions and N is the number of utterances we have available in the database. To construct the matrix V, a counter $c(m, n)$ which counts the number of times the subword unit transition m occurs in the utterance n is used.

The different types of data representations will depend on how we define the transitions $m = [1, \dots, M]$. These four different representations are defined as explained in the following subsections.

[1]The implemented code for NNMF and BPFA can be found at `https://github.com/astriaun/word_discovery.git`

4.1.1 1st-order transition

The simplest form of subword unit transition is the transition from one unit to the next, i.e. 1st-order transition. $m = [a, b]$ then denotes the transition *"$a \rightarrow b$"*. As an example, the word "one" can be represented as two letter transitions, *"$o \rightarrow n$"* and *"$n \rightarrow e$"*, and similarly by the phones. Given the number of subword units, U, the number of possible transitions with this representation is $M = U^2$.

4.1.2 2nd-order transition

Another approach is to look at the 2nd-order transition. That is, the transition from one unit to the second next unit, disregarding the unit in between. In this case, $m = [a, b]$ denotes the transition *"$a \rightarrow$? $\rightarrow b$"*. Using the same example word "one", this can be represented as one 2nd-order letter transition *"$o \rightarrow$? $\rightarrow e$"*. This representation, however, can cause problems for words shorter than three units as they do not have any 2nd-order transitions. Given U, the number of possible transitions is still $M = U^2$.

4.1.3 1st-order & 2nd-order transition

A richer representation is using both 1st- and 2nd-order subword unit transitions. For example, "one" can be represented as the transitions *"$o \rightarrow n$"*, *"$n \rightarrow e$"* and *"$o \rightarrow$? $\rightarrow e$"*. The number of possible transitions with this representation is $M = 2U^2$, as we use both 1st- and 2nd-order.

4.1.4 Three unit transition

As the most complex representation in this report, we can use the transitions between three units. The three unit representation can also be seen as a 3-gram (trigram), and this representation is similar to the n-grams often used in grammar models, where we use counts instead of probabilities. Then $m = [a, b, c]$ denotes the transition *"$a \rightarrow b \rightarrow c$"*. With the same example, "one" can be represented as the transition *"$o \rightarrow n \rightarrow e$"*. Similarly with the the representation using 2nd-order transitions, this one too can cause problems for words shorter than three units. Here, the number of possible transitions is $M = U^3$.

4.1.5 Construction of transition count matrix

Given the different choices of data representation, we can construct the transition count matrix V. Figure 4.1 illustrates a simplified example of how the matrix V is constructed using 1st-order transitions for utterances constructed by an alphabet of 3 subword units. Note that the transition count matrix in Figure 4.1 is of the form $\mathbb{R}^{N \times M}$, where ths number of utterances is $N = 3$ and the number of possible transitions is $M = 3^2 = 9$.

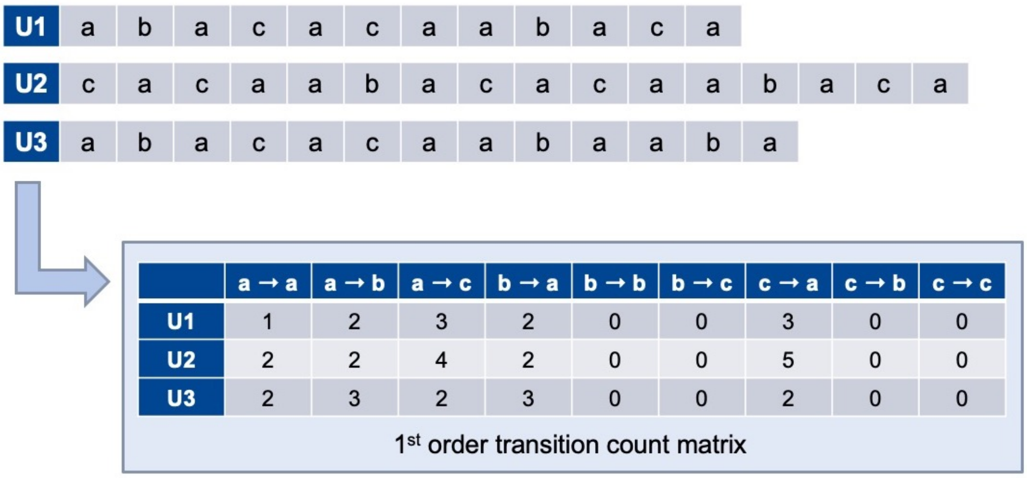


Figure 4.1: Example of the construction of 1st-order transition count matrix V from three example utterances represented by the subword units a, b and c.

The transition count matrix is constructed by counting the occurrences of the possible transitions. It is from this matrix we want to recover the words hidden in the utterances. That is, we want to go from this representation of the utterances in V to a similar representation of the words, i.e. in the matrix W in Figure 2.2. This can be done by *Factor Analysis* (FA) discussed in Section 2.1, or more specific by the algorithms NNMF and BPFA, discussed in Section 2.2 and 2.4, respectively.

4.2 Non-Negative Matrix Factorization

We have the subword unit transition count matrix $V \in \mathbb{R}^{M \times N}$ and we want to find the two matrices $W \in \mathbb{R}^{M \times K}$ and $H \in \mathbb{R}^{K \times N}$ such that

$$V \approx W \cdot H \tag{4.1}$$

M and N are the number of possible transitions and the number of samples in the data set, respectively. K is the number of expected features (i.e. words) in the data set. The algorithm for the NNMF is illustrated in Algorithm 1, where the functions `update_W`$(\cdot)$ and `update_H`$(\cdot)$ calculates the updated matrices using the update equations defined later in Eq.(4.3) and (4.4).

Algorithm 1: NNMF training

Data: matrix $V \in \mathbb{R}^{M\times N}$, integer $K < \min(M, N)$

Result: matrix $W \in \mathbb{R}^{M\times K}$, matrix $H \in \mathbb{R}^{K\times N}$

initialize $W^{(0)}$, $H^{(0)}$ non-negative;

for t = *1, 2, 3, ...* **do**

  $W^{(t)}$ = update_W(V, $W^{(t-1)}$, $H^{(t-1)}$);

  $H^{(t)}$ = update_H(V, $W^{(t)}$, $H^{(t-1)}$);

end

The update equations can either be additive or multiplicative. An additive update equation of the matrix A, is defined as $A^{(t)} = A^{(t-1)} + \Delta A$, whereas a multiplicative update equation is defined as $A^{(t)} = A^{(t-1)} \cdot \Delta A$. Multiplicative update equations are a good compromise between speed and ease of implementation [20], and will be used in this project. As these update equations do not change their sign, the initialization of the matrices are critical in order to satisfy the non-negative constraints. This is discussed further in Section 4.4.

As W and H have a probabilistic interpretation, the *Kullback-Leibler Divergence* (KLD) is an appropriate choice as the cost function [9]. The KLD function in this specific case is given as:

$$D_{KL}(V||WH) = \sum_{n,m} \left([V]_{nm} \log \frac{[V]_{nm}}{[WH]_{nm}} - [V]_{nm} + [WH]_{nm} \right) \tag{4.2}$$

where $[A]_{nm}$ is the element of matrix A in the n'th row and the m'th column. It can be shown that this objective function converges to a local optimum with the update equations for W and H given in Eq.(4.3) and (4.4) [20].

$$[W]_{mk} \leftarrow [W]_{mk} \frac{\sum_{i=1}^{N} [H]_{ki} [V]_{mi} / [WH]_{mi}}{\sum_{j=1}^{N} [H]_{kj}} \tag{4.3}$$

$$[H]_{kn} \leftarrow [H]_{kn} \frac{\sum_{i=1}^{M} [W]_{ik} [V]_{in} / [WH]_{in}}{\sum_{j=1}^{M} [W]_{jk}} \tag{4.4}$$

4.3 Beta Process Factor Analysis

We have the non-negative subword unit transition count matrix $V \in \mathbb{R}^{N \times M}$ which we want to be factorized such that

$$V = H \cdot (Z \circ W) + \varepsilon, \tag{4.5}$$

where W, $Z \in \mathbb{R}^{K \times M}$ and $H \in \mathbb{R}^{N \times K}$. Note that the subword unit transition count matrix, V, here is the transpose of the matrix used for NNMF, and the numbers M, N and K have the same definitions as for NNMF in Section 4.2.

In order to approximate this matrix factorization, the variables we want to infer are the following; [28]

$$\begin{aligned}
\boldsymbol{v}_m &\sim \left|\mathcal{N}(H(\boldsymbol{z}_m \circ \boldsymbol{w}_m), \operatorname{diag}(\psi_1, \ldots, \psi_N)^{-1})\right| \\
z_{km} &\sim \text{Bernoulli}(\pi_k) \\
\pi_k &\sim \text{Beta}(\alpha/K, \beta(K-1)/K) \\
h_{nk} &\sim \left|\mathcal{N}(0, \gamma_{nk}^{-1})\right| \\
\boldsymbol{w}_m &\sim |\mathcal{N}(\mathbf{0}, I_K)| \\
\psi_n &\sim \text{Gamma}(e, 1/f) \\
\gamma_{nk} &\sim \text{Gamma}(c, 1/d)
\end{aligned}$$

Note that in order to keep the matrices non-negative, we use *folded* Normal distribution by simply taking the absolute value [11]. The parameters α, β, c, d, e and f have been set to $\alpha = \beta = c = 1$ and $d = e = f = 10^{-6}$. These have not been tuned. As with the NNMF-algorithm, initializing W and H non-negative is crucial for the BPFA-algorithm. The initialization for these two matrices used is the same as for the NNMF, and is discussed in Section 4.4. Algorithm 2 illustrates the training process for the BPFA, and the update equations are defined in Eg.(4.6)-(4.18).

To infer these variables, *Gibbs sampling* has been used as in [28], but where we use folded Normal distribution in order to keep the matrices strictly non-negative as was done in [11]. The update equations are as follows;

Update of $\boldsymbol{Z}$:

The elements of the binary matrix Z are sampled from the Bernoulli distribution, $z_{km} \sim \text{Bernoulli}(p)$, where p is the probability of $z_{km} = 1$. This probability is proportional to

$$P(z_{km} = 1 \mid -) \propto \ln(\pi_k) - \frac{1}{2}\left(\boldsymbol{h}_k^T \operatorname{diag}(\boldsymbol{\psi})\boldsymbol{h}_k w_{km} - 2\boldsymbol{h}_k^T \operatorname{diag}(\boldsymbol{\psi})\boldsymbol{\varepsilon}_m^{-k} w_{km}\right) = p_1, \tag{4.6}$$

Algorithm 2: BPFA training

Data: matrix $V \in \mathbb{R}^{N \times M}$

Result: matrix $H \in \mathbb{R}^{N \times K}$, matrix $W \in \mathbb{R}^{K \times M}$, matrix $Z \in \mathbb{R}^{K \times M}$

initialize $H^{(0)}$, $W^{(0)}$ non-negative;

initialize $Z^{(0)} = 1$;

initialize $\pi^{(0)} = 10^{-6}$;

initialize $\psi^{(0)}$, $\gamma^{(0)} = 1$;

for t = *1, 2, 3, ...* **do**

$Z^{(t)}$ = update_Z(V, $H^{(t-1)}$, $W^{(t-1)}$, $Z^{(t-1)}$, $\pi^{(t-1)}$, $\psi^{(t-1)}$);

$\pi^{(t)}$ = update_π($Z^{(t)}$);

$W^{(t)}$ = update_W(V, $H^{(t-1)}$, $W^{(t-1)}$, $Z^{(t)}$, $\psi^{(t-1)}$);

$H^{(t)}$ = update_H(V, $H^{(t-1)}$, $W^{(t)}$, $Z^{(t)}$, $\psi^{(t-1)}$, $\gamma^{(t-1)}$);

$\gamma^{(t)}$ = update_γ($H^{(t)}$);

$\psi^{(t)}$ = update_ψ(V, $H^{(t)}$, $W^{(t)}$, $Z^{(t)}$);

end

where ε^{-k} is the residual error disregarding the k'th feature. The probability of $z_{km} = 0$ is proportional to

$$P(z_{km} = 0 \mid -) \propto \ln(1 - \pi_k) = p_0. \tag{4.7}$$

We thus have

$$p = \frac{1}{1 + p_0/p_1} = \frac{1}{1 + e^{p_0 - p_1}}. \tag{4.8}$$

Update of $\boldsymbol{\pi}$:

π_k is sampled from $\pi_k \sim \text{Beta}(\hat{\alpha}, \hat{\beta})$, where

$$\hat{\alpha} = \sum_{m=1}^{M} z_{km} + \frac{\alpha}{K} \tag{4.9}$$

$$\hat{\beta} = M - \sum_{m=1}^{M} z_{km} + \frac{\beta(K-1)}{K} \tag{4.10}$$

Update of $\boldsymbol{W}$:

Each column of W is sampled from $\boldsymbol{w}_m \sim |\mathcal{N}(\boldsymbol{\mu}_m, \Sigma_m)|$.

$$\Sigma_m = \left[\widetilde{H}_m^T \text{diag}(\boldsymbol{\psi}) \widetilde{H}_m + I_K\right]^{-1}, \tag{4.11}$$

$$\boldsymbol{\mu}_m = \Sigma_m \widetilde{H}_m^T \text{diag}(\boldsymbol{\psi}) \boldsymbol{v}_m, \tag{4.12}$$

where $\widetilde{H}_m = H \circ \widetilde{Z}_m$ where $\widetilde{Z}_m = [\boldsymbol{z}_m, \boldsymbol{z}_m, \ldots, \boldsymbol{z}_m]^T$ with the K-dimensional vector $\boldsymbol{z}_m$ repeated N times.

Update of $\boldsymbol{H}$:
Each element of H is sampled from $h_{nk} \sim |\mathcal{N}(\mu_{nk}, \Sigma_{nk})|$.

$$\Sigma_{nk} = \left[\gamma_{nk} + \psi_n \sum_{m=1}^{M} z_{km} w_{km}^2\right]^{-1}, \tag{4.13}$$

$$\mu_{nk} = \Sigma_{nk} \psi_n \sum_{m=1}^{M} z_{km} w_{km} v_{nm} \tag{4.14}$$

Update of $\boldsymbol{\psi}$:
The noise $\boldsymbol{\psi}$ is sampled from $\psi_n \sim \text{Gamma}(\hat{e}, 1/\hat{f})$, where

$$\hat{e} = e + \frac{M}{2} \tag{4.15}$$

$$\hat{f} = f + \frac{1}{2} \sum_{m=1}^{M} |\varepsilon_{nm}|^2 \tag{4.16}$$

Update of $\boldsymbol{\gamma}$:
The covariance for H is sampled from $\gamma_{nk} \sim \text{Gamma}(\hat{c}, 1/\hat{d})$

$$\hat{c} = c + \frac{1}{2} \tag{4.17}$$

$$\hat{d} = d + \frac{1}{2} h_{nk}^2 \tag{4.18}$$

4.4 Initialization

A common method to initialize the matrices W and H in matrix factorization is the *Singular Value Decomposition* (SVD). However, SVD can result with negative elements in the matrices. Aiming to initialize to strictly non-negative matrices, a non-negative variant of SVD proposed in [29] is used.

Algorithm 3 illustrates the implementation of this method. Here `svds`$(\cdot)$ calculates the K largest singular values and vectors of a sparse matrix and `norm`$(\cdot)$ calculates the *Euclidean norm* (also called the *2-norm*). The functions `pos`$(\cdot)$ and `neg`$(\cdot)$ extract the positive and negative sections of their arguments, respectively. That is:

$$\begin{aligned} \text{pos}(A) &= (A \geq 0) \circ A \\ \text{neg}(A) &= (A < 0) \circ A \end{aligned} \tag{4.19}$$

Algorithm 3: NNDSVD initialization of non-negative matrix

Data: matrix $V \in \mathbb{R}^{M \times N}$, integer $K < \min(M, N)$

Result: matrix $W \in \mathbb{R}^{M \times K}$, matrix $H \in \mathbb{R}^{K \times N}$

```
U, S, V = svds(X, K);
for j = 0 : K-1 do
    x = U[:, j]; y = V[j, :];
    xp = pos(x); xn = neg(x);
    yp = pos(y); yn = neg(y);
    xpnrm = norm(xp); ypnrm = norm(yp);
    xnnrm = norm(xn); ynnrm = norm(yn);
    mp = xpnrm · ypnrm; mn = xnnrm · ynnrm;
    if mp > mn then
        u = xp/xpnrm; v = yp/ypnrm;
        sigma = mp;
    else
        u = xn/xnnrm; v = yn/ynnrm;
        sigma = mn;
    end
    W[:, j] = sqrt(S[j] · sigma) · u;
    H[j, :] = sqrt(S[j] · sigma) · v;
end
```

where $\circ$ denotes element-wise multiplication.

4.5 Evaluation

Different metrics are used to evaluate the performance of our system. These will be discussed in this subsection.

4.5.1 Euclidean distance

The *Euclidean distance* is a common evaluation metric. In this case, it is suitable to calculate the Euclidean distance between the estimated basis vectors in the feature load matrix W and the true basis vectors obtained from sequences with isolated words. If we let $\boldsymbol{w} = [w_1, w_2, \ldots, w_M]$ be the true basis vector and $\hat{\boldsymbol{w}} = [\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_M]$ be the estimate, then the

Euclidean distance is defined as;

$$d(\boldsymbol{w}, \hat{\boldsymbol{w}}) = \sqrt{(w_1 - \hat{w}_1)^2 + (w_2 - \hat{w}_2)^2 + \cdots + (w_M - \hat{w}_M)^2} \tag{4.20}$$

We want the value of $d(\boldsymbol{w}, \hat{\boldsymbol{w}})$ to be as close to 0 as possible, where $\hat{\boldsymbol{w}} \approx \boldsymbol{w}$.

4.5.2 Word Accuracy

A common metric to use when reporting the performance of an ASR system is *Word Accuracy*, W_{acc}, defined in Eq.(4.21).

$$W_{\text{acc}} = \frac{C - I}{N} \tag{4.21}$$

where C is the number of correctly detected words, I the number of falsely detected words, and N is the total number of words. We want this metric to be as close to 1 as possible where $C = N$ and $I = 0$.

An activation matrix is constructed which presents the true words being present in each of the sequences. A word is assumed present for all corresponding elements of the estimated matrix H which is greater or equal to 1, i.e. $[H]_{ij} \geq 1$. H is then compared to the activation matrix, and W_{acc} is calculated from this comparison.

5 Experiments

5.1 TIDIGITS

In this section, we will go through the test cases and results where the database TI-Digits[16] was used. A more detailed description of the database used and the test cases is given in the Section 5.1.1 and 5.1.2, respectively. At last the results are presented and discussed in Section 5.1.3.

5.1.1 Data

The data used for these experiments is taken from the TI-Digits database [16], which contains recordings of spoken digits. The same database was also used in [9] and [11]. The TI-Digits database consists of in total 326 unique speakers from the U.S, where 111 of them are classified as **Man**, 114 as **Woman**, 50 as **Boy** and 51 as **Girl**. The U.S. was divided into 21 dialectical regions in order to obtain a dialect balanced database.

The utterances consist of 11 digits and are listed in Table 5.1 with corresponding orthographic and phonetic transcription. The phonetic transcriptions presented in this table are just a chosen “canonical” pronunciation for each word and the speakers’ pronunciation may deviate from these transcriptions. Also note that we use the ARPABET[2] phonetic symbols to represent pronunciations. A mapping between these symbols to the *International Phonetic Alphabet* (IPA) can be found in Appendix A.

Table 5.1: Orthographic and phonetic transcription of the digits

Z	zero	z iy r ow	6	six	s ih k s
1	one	w ah n	7	seven	s eh v ah n
2	two	t uw	8	eight	ey t
3	three	th r iy	9	nine	n ay n
4	four	f ao r	O	oh	ow
5	five	f ay v			

The speakers provided each with 77 sequences of these digits. These 77 sequences include;

22 isolated digits

11 two-digit sequences

[2]https://en.wikipedia.org/wiki/ARPABET

11 three-digit sequences
11 four-digit sequences
11 five-digit sequences
11 seven-digit sequences

Hence, the database has a total of $77 \cdot 326 = 25102$ utterances. However, for the factor analysis only the test set have been used. The data used in the test cases therefore consists of $K = 11$ unique words and $N = 8700$ sequences. The train set of this database was used to train the phonetic recognizer used in some of the test cases.

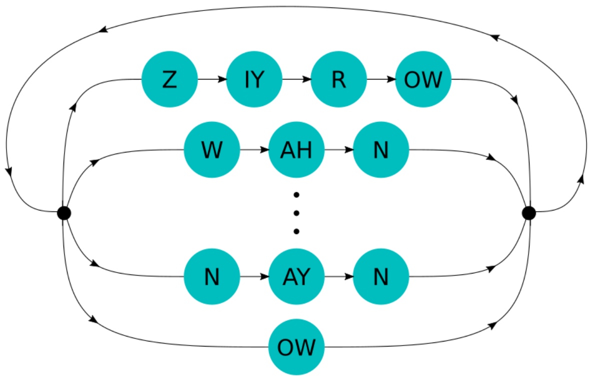
5.1.2 Test Cases

There are four main test cases that will be executed using the NNMF-method. One where we train on orthographic transcriptions of the utterances, and three where we train on phonetic transcriptions. The difference between the last three test cases is the approach taken in order to obtain the phonetic transcriptions from the speech recordings. These four transcriptions are then represented by all of the subword unit transition count matrices described in Section 4.1. That is, each transcription produces four matrices which are factorized with the NNMF-algorithm.

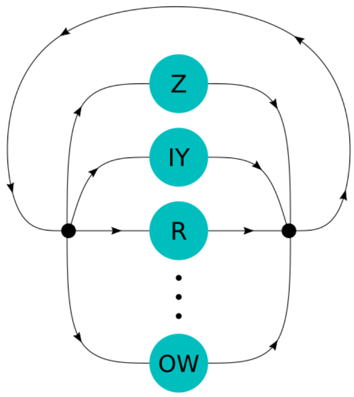
Firstly, an orthographic transcription of the utterances will be constructed. This transcription is obtained by simply writing out the sequences in the database and disregarding the recordings of the spoken digits. The sequences are converted into unsegmented text by removing all spaces between the words. For example, a typical sequence like "5O217" becomes "fiveohtwooneseven". These sequences will then be represented in a letter transition count matrix constructed as explained in Section 4.1.

The first phonetic transcription is also obtained disregarding the recordings and is simply obtained from the orthographic transcription by means of the pronunciation dictionary in Table 5.1. The example sequence from earlier, "5O217", will then be converted into "f ay v ow t uw w ah n s eh v ah n". This test case will illustrate a perfect phonetic transcription, and will be used to compare to the erroneous transcription gained from a phonetic recognizer.

The last two test cases use the phonetic transcription obtained from the speech and a phonetic recognizer from the Kaldi recipe [30]. The phones the phonetic recognizer look for are exclusively the ones listed in Table 5.1. This recognizer will be run twice. The reason for this is the original phonetic recognizer in the Kaldi recipe works like illustrated in Figure



(a) Original Kaldi[30] phonetic recognizer.



(b) Modified phonetic recognizer.

Figure 5.1: Phonetic recognizer where (a) recognizes sequences of phones corresponding to the words present in the database, and (b) recognizes each phone independently of each other.

5.1a, where the system recognizes sequences of phones corresponding to the words in the data set. An alteration was made to get the recognizer work as illustrated in Figure 5.1b, where each phone is recognized independently of each other. This is a more realistic structure for our application as we assume that the words are not known, and will perhaps result in a more noisy phonetic transcription. The original recognizer is added to the experiments as a reference.

5.1.3 Results and discussion

The results of the four test cases are illustrated in Figure 5.2-5.5. They show the plotting of the metrics discussed in Section 4.5, the Euclidean distance and the word accuracy, of each of the words in the data set for each of the representations discussed in Section 4.1. The average values for distance and word accuracy are illustrated by a horizontal line. In Appendix B, you can find truncated versions of the estimated basis vectors found by the NNMF sorted by the weights given to the subword unit transitions.

For the first two test cases, the transcriptions are simply written out, without any use of a speech recognizer, and are therefore noise free. Figure 5.2 shows the results using the orthographic transcription of the digit-sequences and Figure 5.3 the results using phonetic transcription.

In the case of orthographic transcription, all of the representations perform well, as can be

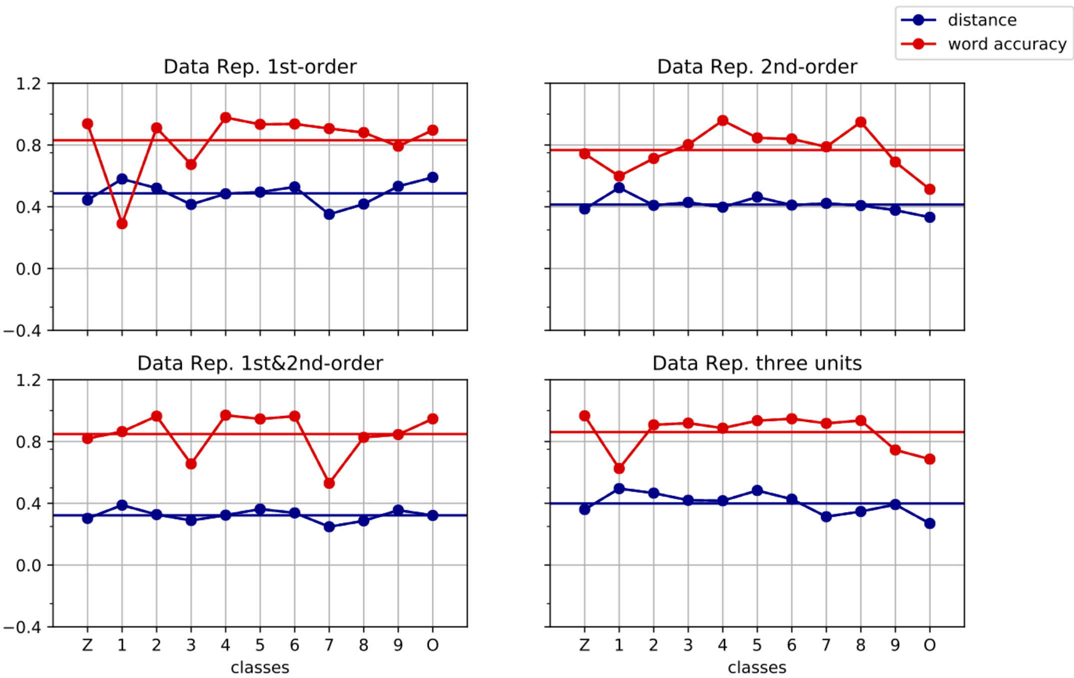


Figure 5.2: Results with orthographic transcription.

seen in both Figure 5.2 and in Table B.1-B.4 in Appendix B.1. In all of these tables, the letter transitions which are associated with the current word are given the largest weights. From Figure 5.2 we can see that the average word accuracy and distance for the 2nd-order representation scores worse than for the other representations, whose average values for word accuracy are similarly high. However, the word accuracy for 1 with the 1st-order representation is lower than the rest, this is because "one" is too similar to "nine" causing 1s to be falsely detected when there is a 9 present. As we can see in Table B.1, the estimated basis vector for 9 does not have a large weight on the transition "n $\rightarrow$ e", but the basis vector for 1 does, and this is what is causing the confusion. On average, considering both word accuracy and Euclidean distance, the three units representation might be the best choice for this particular data. But it is also the most complex representation and more time-consuming than the other three.

From the results in Figure 5.3, we can see that all the representations have problems with O with a low word accuracy and high Euclidean distance. This can also be seen from Table B.5-B.8 in Appendix B.2, where estimated basis vectors of O have not given the largest weights to the subword unit transitions associated with O. The poor performance for O is, however, to be expected, as it is a short word with represented by only one phone, see Table 5.1. In this case, the best choice of representation seems to be either 1st-order or 1st&2nd-order

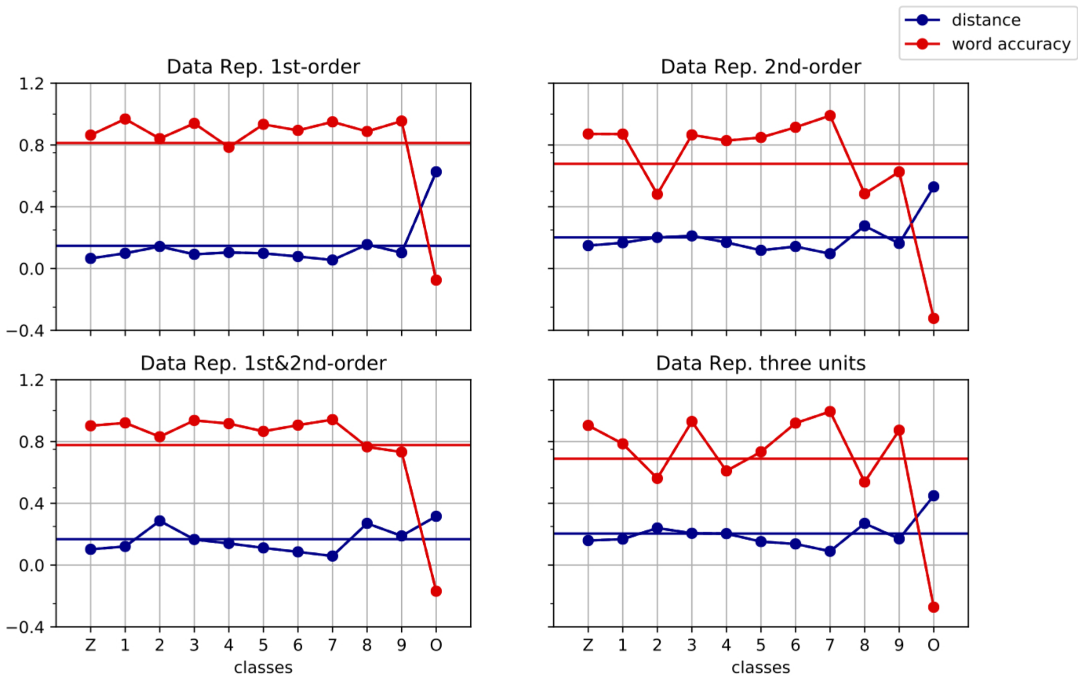


Figure 5.3: Results with perfect phonetic transcription.

representation, considering the average values of word accuracy and distance. For the other two representations, the results show a higher variability for the words in respect of the word accuracy.

The results from the experiments using the speech recordings are illustrated in Figure 5.4 and 5.5. These transcriptions have some noise as there might be errors from the phonetic recognizer. First Figure 5.4 shows the results from using the original Kaldi phonetic recognizer, illustrated in Figure 5.1a, and then Figure 5.5 shows the results from using a modified Kaldi phonetic recognizer, illustrated in Figure 5.1b. Here, the true basis vectors used to calculate the Euclidean distance is defined to be the average of the transcriptions of the utterances with a single isolated digit.

With a noisy transcription obtained from the original phonetic recognizer, the 2nd-order representation is not able to recover 2 and O from the utterances, which is illustrated in Figure 5.4 by a dotted line and absence of bullet points, in addition to the absence of the estimated basis vectors for 2 and O in Table B.10. The word accuracy for 8 is also lower than for the other words. Considering the average word accuracy and distance, the 1st-order and 1st&2nd-order representation perform well. The metrics are also similar for all the words, though they both perform a bit worse for O. Also the estimated basis vectors for 1st-order

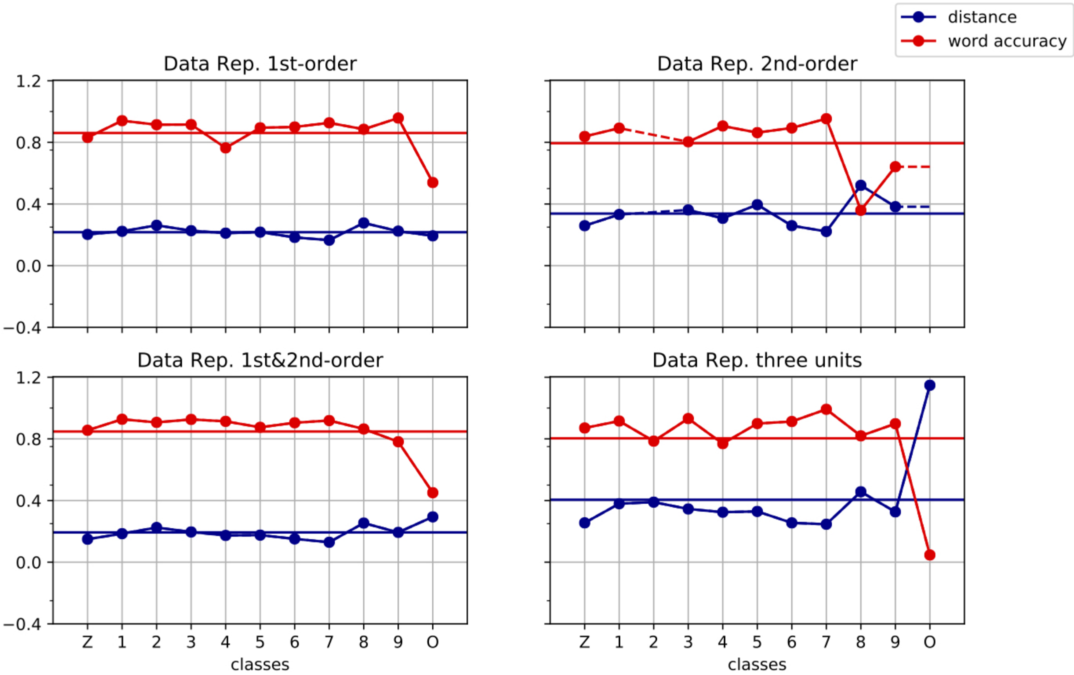


Figure 5.4: Results with phonetic transcription obtained from original Kaldi phonetic recognizer.

and 1st&2nd-order in Table B.9 and B.11, respectively, are good, as the transitions given the largest weights corresponds to the current words. Similarly, the three units representation obtain good results for all the words with the exception of O. Because the representations 1st-order and 1st&2nd-order score better for the word O than the three units representation, they are to be preferred in this test case as well.

The results in Figure 5.5 are similar to the results in Figure 5.4, with the 1st-order and 1st&2nd-order representations having similar values for all words and the 2nd-order and three units representation having problems with O. However for this transcription, the 2nd-order representation is able to recover all of the words. Considering the average word accuracy and Euclidean distance, the 1st-order and 1st&2nd-order representations perform better than the other two. The 1st-order representation also has higher word accuracy than 1st&2nd-order on average.

Overall, the NNMF-algorithm was able to find the words for all the test cases, from both text and speech. Even with noisy data, NNMF is a good method to discover words in unsegmented speech in an unsupervised way. However, in the NNMF the number of words to discover was set to 11 manually, which also helped in getting acceptable results. Also,

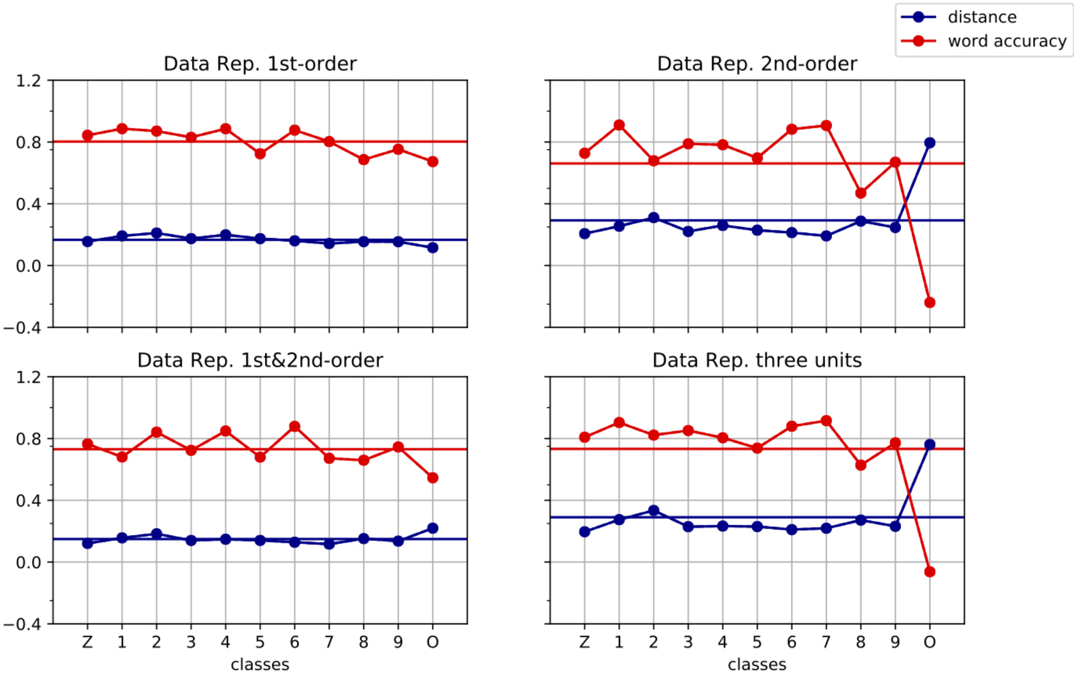


Figure 5.5: Results with phonetic transcription obtained from modified Kaldi phonetic recognizer.

judging from the results above, the 1st-order and 1st&2nd-order are to be preferred between the data representations. They often perform the best on average and does not have as much problems recovering the shorter words as the other two representations. As the vocabulary is so small, with only 11 unique words, using the simplest representation which uses only 1st-order transition have proven itself efficient enough and can be preferred over the more complex representation. In Section 5.2 we will test the NNMF further with different vocabularies, in order to determine how well this method works for larger vocabularies.

5.2 Larger Vocabularies

The results reported in the previous section were based on the TI-Digits database that has a small vocabulary. In order to test for word discovery for larger vocabularies, a different data set with increasing vocabularies has been artificially constructed which is described further in Section 5.2.1. The test cases and the results for these data sets are then presented in Section 5.2.2 and 5.2.3.

5.2.1 Data

In order to create data sets with increasing vocabulary size, vocabulary sets have been sampled from a large set of words. First 10 words were sampled from this large set to construct a vocabulary of 10 words. Then 10 new words were sampled, which with the previous words assemble a vocabulary of 20 words. This is repeated until we have vocabularies with 10, 20, 30, 40 and 50 words. The vocabulary sets are presented in Table 5.2.

Table 5.2: Vocabularies

10 words
hotline, conducting, watch, voter, judged, spurred, pressed, rap, conversations, lawyer
20 words
billion, examine, minimize, mark, franks, impeachment, artillery, winds, innovation, advanced
30 words
reviewing, prisons, awareness, resonate, chose, montclair, methods, inspector, photographs, teresa
40 words
ears, development, hearing, legislators, referendum, sink, format, suspicious, transformed, palestinians
50 words
metaphor, hand, feeling, adopted, urban, ahmad, label, finishing, gobbell, qualified

The sequences of words are then artificially constructed for each of the vocabulary sets by sampling the words in current the vocabulary. The number of words in each utterance is also randomly sampled to be between 1 and 7. In addition to these sampled sequences, sequences consisting of only one word are also added to the data set for each of the words in the vocabulary. With this approach, a database of totally 10 000 sequences was constructed for each of the vocabulary sets. An example of a sequence containing five words is “rapwatchconversationsconductingjudged”.

5.2.2 Test Cases

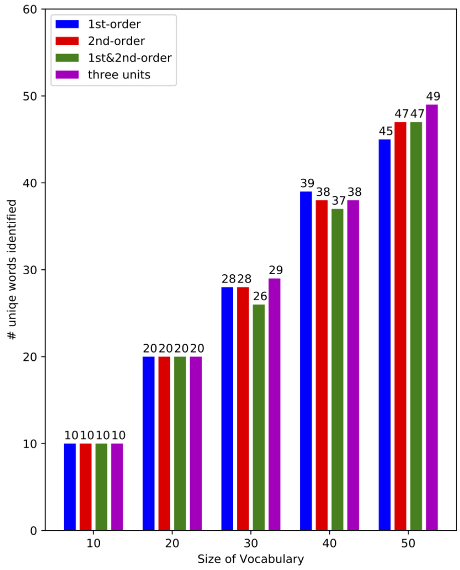
The NNMF-method is tested on the orthographic transcription of the sequences in the constructed database. As with the TI-Digits database in Section 5.1, the letter transition count matrix will be constructed with the four different approaches discussed in Section 4.1. This will be done for each of the vocabulary sets in order to test how many words NNMF can find as the vocabulary size increases. It is important to note that the amount of data, i.e. the number of sequences, are the same for all vocabularies, such that the smallest vocabulary have more data per word than the largest vocabulary. So to be specific, what we are testing is how well NNMF works for larger vocabularies, but for the same amount of data.

The second test case is to the BPFA-method on these vocabularies with different data representations. Unlike the first test case, we will not pre-define K, the number of words extracted from the sequences, but will let the BPFA infer this number as well as the basis vectors. The goal of this test case is the same as the first, we want to test out how well BPFA works as the vocabulary increases, but the same amount of data.

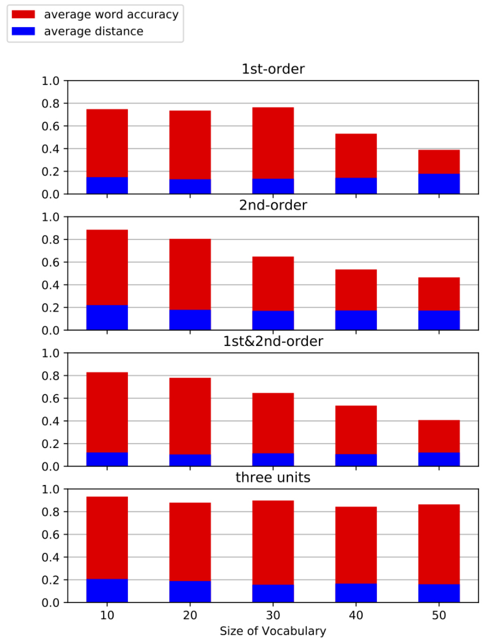
5.2.3 Results and discussion

The results from the NNMF-method are summarized in Figure 5.6, where K is set to the number of words in the vocabulary. Figure 5.6a illustrates the number of unique words present in the vocabulary the basis vectors corresponds to, and Figure 5.6b shows the average Euclidean distance and word accuracy for each vocabulary and data representation. As we can see in Figure 5.6a, all of the words are found for the two smallest vocabularies, i.e. 10 and 20 words, and the metrics for these are also good on average. For the larger vocabularies, it does not find all the words. Still, only a few words are missing, and these are often words that have similar attributes. For example, “conducting” and “conversations” both contain a “con”, and “conversations” and “innovation” both contain a “tion”. For the vocabulary with 30 words, the metrics perform still good. However, the metrics for 40 and 50 words, especially the word accuracy, are not as good as for the other vocabularies.

In Figure 5.7, the results from the BPFA-method are summarized, where again Figure 5.7a illustrates the number of unique words present in the vocabulary the basis vectors corresponds to, and Figure 5.7b shows the average Euclidean distance and word accuracy for each vocabulary and data representation. From Figure 5.7 we can see that BPFA can be used to infer the number of words, at least for smaller vocabularies or with complex data representations. As with NNMF, BPFA is able to find the words for the two smallest vocabularies,



(a) Number of unique words found.

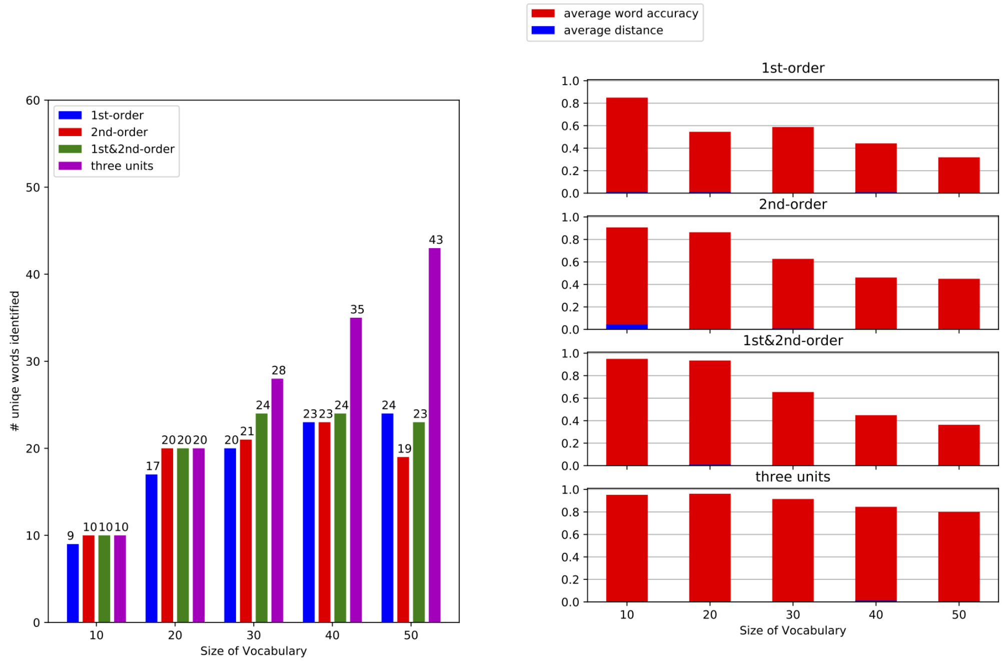


(b) Average Euclidean distance and word accuracy.

Figure 5.6: Results for each of the vocabularies and data representations using the NNMF.

with good evaluation metrics for all the representations. For the vocabulary with 30 words, the performances of the two more complex representations, 1st&2nd-order and three unit, are acceptable. The three unit representation also does well for the two largest vocabularies, in terms of both number of unique words found and the evaluation metrics.

From the results from both NNMF and BPFA, we may conclude that both methods work for vocabularies up to 30 words at most. With vocabularies of larger size, only the most complex data representation gives acceptable results. Even though the three unit representation gives good results for all the vocabularies, this is not a preferred representation to use, as it is so complex with a large number of possible transitions. The algorithm is much more time consuming for this data representation than for the other three representations. To train word discovery on a larger vocabulary than 30, a larger number of utterances are required. Maybe then the algorithms will give better results for the less complex data representations too?



(a) Number of unique words found.

(b) Average Euclidean distance and word accuracy.

Figure 5.7: Results for each of the vocabularies and data representations using the BPFA.

6 Conclusion

The goal of this project was to recover the words present in the data set from a set of utterances with corresponding transcriptions in subword units. There are several approaches in order to do this, but in this thesis we chose to focus on *Factor Analysis* (FA). We investigated how to perform this task both from text (letters) and speech (phones). In order to represent the transcriptions (i.e. sequences of subword units) in fixed-length, non-negative vectors, we used transition counts from one subword unit to another. These vectors would form the non-negative matrix V which would be factorized.

For the *Factor Analysis*, we implemented two different factorization algorithms, namely *Non-Negative Matrix Factorization* (NNMF) and *Beta Process Factor Analysis* (BPFA). The NNMF factorized the non-negative matrix V into two non-negative matrices W and H. By this factorization, it was able to recover hidden structures (i.e. words) in the data set of utterances. A drawback with NNMF is that we need to pre-define the number of words to extract manually. That is, we need to know the number of words in the vocabulary on beforehand. The BPFA is a *Bayesian Non-Parametric* (BNP) version of NNMF, and the advantage of this algorithm over NNMF is its ability to also infer the number of words as well as the representations of the words.

In the experiments with the TI-Digits database [16], the NNMF was able to recover all of the words in most cases. When we tested with increasing vocabularies, NNMF was still able to find most of the words, though it did struggle more as the vocabulary increased. These results were though acquired when the number of words to discover was set manually. For the smaller vocabularies, the BPFA was able to infer the number of words and the words themselves. But as the vocabulary increased, it was not able find all the words anymore.

Four different types of data representations were also tested with different degrees of complexity. The most complex representation usually performed better than the simpler ones, especially when the vocabulary was large and the other representations performed poorly. However, this representation struggled with the shorter words, for instance with O (oh, /ow/). It is also more complex and time-consuming compared to the other three. The 2nd-order representation often performed worse than the others, and the results using 1st-order and 1st&2nd-order representation were often similar. Between those two, using only 1st-order transitions is the simplest representation and therefore is preferred when it gives similar results anyway.

From the experiments done in this thesis, we can conclude that NNMF and BPFA can be used in word discovery for small vocabularies. The results for larger vocabularies are, however, harder to conclude from. Further work should therefore include more testing of NNMF and BPFA with larger vocabularies, perhaps even larger than what was tested here. Also, several of the words in the larger vocabularies used in the tests were relatively long, which might make it easier for the *Factor Analysis* to recover the words. It would therefore be interesting to run the tests with another data set consisting of shorter words but of the same vocabulary size. The larger vocabularies should also be tested by using speech too and not just text as was done in this thesis, as using speech cause more erroneous transcriptions which may affect the *Factor Analysis* negatively.

References

[1] G. Kurata, K. Audhkhasi, and B. Kingsbury. (Oct. 18, 2019). Ibm research advances in end-to-end speech recognition at interspeech 2019, [Online]. Available: `https://www.ibm.com/blogs/research/2019/10/end-to-end-speech-recognition/` (visited on 12/15/2019).

[2] M. Versteegh, R. Thiollière, T. Schatz, X.-N. Cao Kam, X. Anguera, A. Jansen, and E. Dupoux, "The zero resource speech challenge 2015", Sep. 2015.

[3] E. Dunbar, X.-N. Cao Kam, J. Benjumea, J. Karadayi, M. Bernard, L. Besacier, X. Anguera, and E. Dupoux, "The zero resource speech challenge 2017", Dec. 2017.

[4] E. Dunbar, R. Algayres, J. Karadayi, M. Bernard, J. Benjumea, X.-N. Cao, L. Miskic, C. Dugrain, L. Ondel, A. Black, L. Besacier, S. Sakti, and E. Dupoux, "The zero resource speech challenge 2019: Tts without t", Apr. 2019.

[5] M. Heck, S. Sakti, and S. Nakamura, "Feature optimized dpgmm clustering for unsupervised subword modeling: A contribution to zerospeech 2017", in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, Dec. 2017, pp. 740–746.

[6] H. Chen, C. Leung, L. Xie, B. Ma, and H. Li, "Multilingual bottle-neck feature learning from untranscribed speech", in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, 2017, pp. 727–733.

[7] M. Spranger, "Incremental grounded language learning in robot-robot interactions — examples from spatial language", in *2015 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, 2015, pp. 196–201.

[8] G. Salvi, L. Montesano, A. Bernardino, and J. Santos-Victor, "Language bootstrapping: Learning word meanings from perception–action association", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 3, pp. 660–671, 2012.

[9] V. Stouten, K. Demuynck, and H. Van hamme, "Discovering phone patterns in spoken utterances by non-negative matrix factorization", *Signal Processing Letters, IEEE*, vol. 15, pp. 131–134, Feb. 2008.

[10] J. Driesen, L. Bosch, and H. Van hamme, "Adaptive non-negative matrix factorization in a computational model of language acquisition", Jan. 2009, pp. 1731–1734.

[11] N Vanhainen and G. Salvi, "Word discovery with beta process factor analysis", *13th Annual Conference of the International Speech Communication Association 2012, INTERSPEECH 2012*, vol. 1, Sep. 2012.

[12] J.-H. Ahn, S.-K. Kim, J.-H. Oh, and S. Choi, “Multiple nonnegative-matrix factorization of dynamic pet images”, Jan. 2004.

[13] P. Sajda, S. Du, and L. Parra, “Recovery of constituent spectra using non-negative matrix factorization”, *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 5207, pp. 321–331, Jan. 2003.

[14] Hualiang Li, T. Adali, Wei Wang, and D. Emge, “Non-negative matrix factorization with orthogonality constraints for chemical agent detection in raman spectra”, in *2005 IEEE Workshop on Machine Learning for Signal Processing*, Sep. 2005, pp. 253–258.

[15] A. Cichocki, R. Zdunek, and S.-i. Amari, “New algorithms for non-negative matrix factorization in applications to blind source separation”, vol. 5, Jun. 2006, pp. V –V.

[16] R. Leonard, “A database for speaker-independent digit recognition”, *Proc. IEEE ICASSP*, vol. 9, pp. 328–331, Mar. 1984.

[17] S. Gershman and D. Blei, “A tutorial on bayesian nonparametric models”, *Journal of Mathematical Psychology*, vol. 56, Jun. 2011.

[18] N. Gillis, “The why and how of nonnegative matrix factorization”, *Regularization, Optimization, Kernels, and Support Vector Machines*, vol. 12, Jan. 2014.

[19] S. Kullback and R. A. Leibler, “On information and sufficency”, *The Annals of Mathematical Statistics*, vol. 22, pp. 79–86, Mar. 1951.

[20] D. Lee and H. Seung, “Algorithms for non-negative matrix factorization”, *Adv. Neural Inform. Process. Syst.*, vol. 13, Feb. 2001.

[21] N. Hjort, “Nonparametric bayes estimators based on beta processes in models for life history data”, *The Annals of Statistics*, vol. 18, Sep. 1990.

[22] J. Paisley and L. Carin, “Nonparametric factor analysis with beta process priors”, *Proc. Annual Int. Conf. on Machine Learning*, pp. 777–784, Jun. 2009.

[23] Y.-B. Jia, *Singular value decomposition*, Lecture Notes, Iowa State University, Sep. 2019.

[24] P. Paatero and U. Tapper, “Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values”, *Environmetrics*, vol. 5, pp. 111–126, May 1994.

[25] D. Lee and H. Seung, “Learning the parts of objects by non-negative matrix factorization”, *Nature*, vol. 401, pp. 788–91, Nov. 1999.

[26] S. Deligne and F. Bimbot, “Inference of variable-length linguistic and acoustic units by multigrams”, *Speech Communication*, vol. 23, pp. 223–241, Nov. 1997.

[27] A. Park and J. R. Glass, “Towards unsupervised pattern discovery in speech”, in *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005.*, Nov. 2005, pp. 53–58.

[28] B. Chen, M. Chen, J. Paisley, A. Zaas, C. Woods, G. Ginsburg, A. Hero, J. Lucas, D. Dunson, and L. Carin, “Bayesian inference of the number of factors in gene-expression analysis: Application to human virus challenge studies”, *BMC Bioinformatics*, vol. 11, pp. 1–16, Nov. 2010.

[29] C. Boutsidis and E. Gallopoulos, “Svd based initialization: A head start for nonnegative matrix factorization.”, *Pattern Recognition*, vol. 41, pp. 1350–1362, Apr. 2008.

[30] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, “The kaldi speech recognition toolkit”, in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, IEEE Catalog No.: CFP11SRW-USB, Hilton Waikoloa Village, Big Island, Hawaii, US: IEEE Signal Processing Society, Dec. 2011.

A Mapping of International Phonetic Alphabet

Table A.1: Mapping between ARPABET symbols used in the TI-Digits experiments and standard phones from *International Phonetic Alphabet* (IPA)

vowels		**consonants**	
TI-Digits symbol	IPA symbol	TI-Digits symbol	IPA symbol
ah	ʌ	f	f
ao	ɔ	k	k
ay	aɪ	n	n
eh	ɛ	r	ɹ
ey	eɪ	s	s
iy	i	t	t
ih	ɪ	th	θ
ow	oʊ	v	v
uw	u	w	w
		z	z

B TIDIGITS Results - Basis Vectors

B.1 Orthographic transcription

Table B.1: Basis vectors of the orthographic transcription taken from NNMF, where 1st-order data representation is used.

zero		one		two		three		four		five	
r o	**0.2557**	**n e**	**0.4800**	**t w**	**0.3518**	**e e**	**0.2304**	**f o**	**0.2675**	**f i**	**0.2751**
z e	**0.2380**	**o n**	**0.2651**	**w o**	**0.3518**	**r e**	**0.1961**	**o u**	**0.2675**	**i v**	**0.2751**
e r	**0.2380**	i n	0.0814	e t	0.0627	**h r**	**0.1853**	**u r**	**0.2675**	**v e**	**0.2662**
e z	0.0604	n i	0.0814	o t	0.0588	**t h**	**0.1853**	e f	0.0562	e f	0.0824
o z	0.0455	e o	0.0776	o o	0.0478	e t	0.0578	r f	0.0367	o f	0.0232
o s	0.0261	o o	0.0045	o s	0.0429	h t	0.0384	r s	0.0340	n e	0.0164
o f	0.0176	e t	0.0028	i x	0.0270	i g	0.0256	r t	0.0338	e t	0.0153
o t	0.0169	t w	0.0023	s i	0.0270	g h	0.0256	o f	0.0201	w o	0.0140
r z	0.0147	w o	0.0023	o f	0.0135	e i	0.0256	r n	0.0053	t w	0.0140
x z	0.0144	e f	0.0011	x t	0.0104	t t	0.0112	w o	0.0050	o n	0.0099

six		seven		eight		nine		oh	
s i	**0.3566**	**v e**	**0.1681**	**h t**	**0.2042**	**n i**	**0.3704**	**o h**	**0.5525**
i x	**0.3566**	**e n**	**0.1646**	**g h**	**0.1965**	**i n**	**0.3704**	e o	0.1362
e s	0.0923	**e v**	**0.1522**	**i g**	**0.1965**	e n	0.0838	h o	0.1126
x f	0.0471	**s e**	**0.1522**	**e i**	**0.1965**	o h	0.0411	h s	0.0668
x o	0.0450	e s	0.0414	n e	0.0350	h n	0.0306	h f	0.0646
x s	0.0448	n e	0.0362	o e	0.0272	r n	0.0213	h e	0.0272
x t	0.0390	e e	0.0346	t o	0.0257	x n	0.0198	h t	0.0197
x n	0.0084	r e	0.0291	t s	0.0250	n n	0.0168	r o	0.0146
x e	0.0069	h r	0.0269	t f	0.0241	t n	0.0150	t o	0.0030
x z	0.0019	t h	0.0269	o n	0.0161	t w	0.0103	n o	0.0017

Table B.2: Basis vectors of the orthographic transcription taken from NNMF, where 2nd-order data representation is used.

zero		one		two		three		four		five	
e - o	**0.2731**	**o - e**	**0.4906**	**t - o**	**0.3715**	**r - e**	**0.2476**	**f - u**	**0.2664**	**f - v**	**0.3146**
z - r	**0.2369**	e - n	0.1064	o - i	0.1051	**h - e**	**0.2312**	**o - r**	**0.2664**	**i - e**	**0.3042**
o - e	0.0814	n - o	0.0763	e - w	0.0798	**t - r**	**0.2210**	r - i	0.0701	**e - i**	**0.1037**
o - i	0.0598	n - t	0.0700	o - w	0.0475	e - o	0.0809	u - f	0.0373	v - f	0.0419
r - z	0.0327	t - o	0.0697	w - f	0.0462	e - h	0.0629	u - s	0.0359	v - o	0.0411
n - z	0.0291	o - n	0.0487	w - s	0.0457	e - t	0.0320	o - o	0.0350	v - t	0.0389
r - f	0.0284	e - w	0.0345	w - t	0.0456	z - r	0.0274	u - t	0.0329	v - s	0.0389
r - s	0.0272	w - o	0.0311	o - h	0.0383	o - h	0.0226	r - h	0.0328	e - f	0.0380
r - o	0.0181	h - n	0.0295	e - t	0.0346	n - t	0.0145	u - o	0.0323	n - f	0.0306
r - t	0.0165	o - o	0.0200	w - o	0.0277	e - z	0.0117	e - f	0.0287	o - e	0.0230

six		seven		eight		nine		oh	
s - x	**0.3734**	**e - e**	**0.3005**	**g - t**	**0.2259**	**i - e**	**0.2524**	e - h	0.1904
x - i	0.0938	**s - v**	**0.2017**	**i - h**	**0.2259**	**n - n**	**0.2369**	h - i	0.1784
e - i	0.0630	**v - n**	**0.2009**	**e - g**	**0.2259**	e - i	0.1141	h - h	0.1242
i - f	0.0501	n - i	0.0460	t - i	0.0601	e - e	0.0681	o - o	0.1135
x - h	0.0496	e - s	0.0396	h - s	0.0309	v - n	0.0376	n - o	0.0947
x - e	0.0482	e - o	0.0334	h - t	0.0284	f - v	0.0342	o - s	0.0711
i - s	0.0481	r - e	0.0270	h - f	0.0284	e - n	0.0308	o - f	0.0633
i - o	0.0477	n - h	0.0251	t - h	0.0283	t - r	0.0296	o - t	0.0542
n - s	0.0467	h - e	0.0246	n - e	0.0280	s - v	0.0247	o - n	0.0402
i - t	0.0439	n - e	0.0240	h - o	0.0264	r - e	0.0242	h - o	0.0280

Table B.3: Basis vectors of the orthographic transcription taken from NNMF, where 1st&2nd-order data representation is used.

zero		one		two		three		four		five	
r o	**0.1348**	**o - e**	**0.2233**	**w o**	**0.1781**	**e e**	**0.1221**	**o - r**	**0.1389**	**i - e**	**0.1581**
e - o	**0.1347**	**o n**	**0.2104**	**t w**	**0.1781**	**r e**	**0.1046**	**u r**	**0.1389**	**f i**	**0.1558**
z e	**0.1252**	**n e**	**0.2065**	**t - o**	**0.1773**	**h - e**	**0.1045**	**o u**	**0.1389**	**f - v**	**0.1558**
e r	**0.1252**	e o	0.0629	e - w	0.0461	**r - e**	**0.0996**	**f - u**	**0.1389**	**i v**	**0.1558**
z - r	**0.1252**	e - n	0.0531	o - i	0.0416	**t h**	**0.0988**	**f o**	**0.1389**	**v e**	**0.1526**
o - e	0.0334	n - o	0.0365	e t	0.0390	**t - r**	**0.0988**	r - i	0.0356	e - i	0.0531
e z	0.0319	e - o	0.0346	o t	0.0292	**h r**	**0.0988**	e f	0.0253	e f	0.0434
o - i	0.0278	o - n	0.0226	w - o	0.0232	e t	0.0307	r f	0.0192	v - t	0.0208
o z	0.0248	n - f	0.0212	w - s	0.0222	e - h	0.0288	u - f	0.0192	v - f	0.0199
r - z	0.0162	o o	0.0157	w - t	0.0215	h t	0.0211	r s	0.0185	v - s	0.0185

six		seven		eight		nine		oh			
s i	**0.1843**	**e - e**	**0.1257**	**e i**	**0.1041**	**n - n**	**0.1692**	**o h**	**0.2762**		
i x	**0.1843**	**e n**	**0.0949**	**g - t**	**0.1041**	**n e**	**0.1595**	h - i	0.0693		
s - x	**0.1843**	**v e**	**0.0822**	**g h**	**0.1041**	**i - e**	**0.1536**	e - h	0.0618		
e s	0.0467	**v - n**	**0.0810**	**i g**	**0.1041**	**i n**	**0.1476**	e o	0.0591		
x - i	0.0466	**s e**	**0.0764**	**i - h**	**0.1041**	**n i**	**0.1476**	h - h	0.0532		
e - i	0.0285	**s - v**	**0.0764**	**e - g**	**0.1041**	e - i	0.0556	h o	0.0499		
i - f	0.0246	**e v**	**0.0764**	**h t**	**0.1027**	o n	0.0285	o - o	0.0491		
x f	0.0246	e s	0.0206	t - i	0.0288	e - n	0.0166	o - t	0.0360		
x - h	0.0245	n - i	0.0201	o e	0.0152	n - f	0.0158	h s	0.0345		
x s	0.0236	n e	0.0199	n - e	0.0132	o - e	0.0108	o - s	0.0345		

Table B.4: Basis vectors of the orthographic transcription taken from NNMF, where three unit data representation is used.

zero		one		two		three		four		five	
e r o	**0.2403**	**o n e**	**0.4646**	**t w o**	**0.4331**	**t h r**	**0.2359**	**f o u**	**0.2829**	**f i v**	**0.3295**
z e r	**0.2403**	e o n	0.1165	e t w	0.1064	**r e e**	**0.2359**	**o u r**	**0.2829**	**i v e**	**0.3295**
e z e	0.0617	n e o	0.0723	w o f	0.0557	**h r e**	**0.2359**	e f o	0.0601	e f i	0.0790
o z e	0.0466	o o n	0.0602	o t w	0.0545	e t h	0.0639	u r f	0.0395	v e t	0.0458
r o f	0.0306	t w o	0.0493	w o t	0.0545	e e t	0.0322	u r s	0.0380	v e s	0.0416
r o z	0.0300	x o n	0.0313	w o s	0.0538	n e t	0.0289	u r t	0.0377	v e f	0.0409
n e z	0.0295	w o o	0.0271	n e t	0.0447	e e o	0.0264	u r o	0.0340	v e o	0.0396
r o s	0.0286	s i x	0.0265	w o o	0.0372	o t h	0.0264	o f o	0.0337	o f i	0.0378
r o t	0.0284	n e s	0.0233	w o n	0.0285	e e f	0.0260	r f o	0.0207	n e f	0.0283
i n e	0.0192	i x o	0.0217	o n i	0.0283	e e s	0.0214	r s e	0.0195	t f i	0.0058

six		seven		eight		nine		oh			
s i x	**0.3931**	**v e n**	**0.1799**	**i g h**	**0.1932**	**n i n**	**0.2621**	e o h	0.1611		
e s i	0.1004	**e v e**	**0.1696**	**e i g**	**0.1932**	**i n e**	**0.2621**	o h o	0.1119		
i x f	0.0522	**s e v**	**0.1696**	**g h t**	**0.1932**	e n i	0.0667	h o h	0.0765		
i x t	0.0506	e s e	0.0432	e e i	0.0488	n e f	0.0545	o h t	0.0756		
i x s	0.0505	i n e	0.0230	o e i	0.0255	f o u	0.0433	o h f	0.0691		
o s i	0.0470	n i n	0.0230	h t f	0.0254	o u r	0.0433	o h s	0.0678		
n e s	0.0427	e n t	0.0215	h t o	0.0250	n e n	0.0377	n e o	0.0611		
x f i	0.0265	e n s	0.0214	n e e	0.0249	i v e	0.0242	h t w	0.0377		
x t w	0.0257	e n f	0.0213	h t t	0.0229	f i v	0.0242	h t h	0.0346		
x s i	0.0256	e n o	0.0200	h t s	0.0194	e f o	0.0223	h f i	0.0333		

B.2 Perfect phonetic transcription

Table B.5: Basis vectors of the perfect phonetic transcription taken from NNMF, where 1st-order data representation is used.

zero		one		two		three		four		five	
r ow	**0.2575**	**w ah**	**0.3645**	**t uw**	**0.4900**	**r iy**	**0.3260**	**f ao**	**0.3090**	**ay v**	**0.3679**
iy r	**0.2473**	**ah n**	**0.3638**	n t	0.0933	**th r**	**0.3260**	**ao r**	**0.3090**	**f ay**	**0.3679**
z iy	**0.2473**	n w	0.0641	ow t	0.0604	n th	0.0607	n f	0.0562	n f	0.0546
n z	0.0395	ow w	0.0485	uw f	0.0589	iy s	0.0434	r f	0.0486	v s	0.0481
ow z	0.0322	uw w	0.0242	uw s	0.0584	ow th	0.0415	f ay	0.0281	v f	0.0381
ow s	0.0195	t w	0.0239	ey t	0.0462	iy f	0.0334	ay v	0.0281	v t	0.0259
s z	0.0158	n ow	0.0226	uw ey	0.0350	iy t	0.0228	th r	0.0262	v ow	0.0226
r z	0.0155	r w	0.0226	uw t	0.0332	iy th	0.0217	r iy	0.0262	v th	0.0201
iy z	0.0152	s w	0.0200	t t	0.0332	iy ow	0.0211	r th	0.0238	v ey	0.0181
uw z	0.0146	n f	0.0120	uw ow	0.0322	uw th	0.0209	n ay	0.0193	v w	0.0144

six		seven		eight		nine		oh			
ih k	**0.2549**	**ah n**	**0.2038**	**ey t**	**0.5184**	**n ay**	**0.3714**	n s	0.5331		
k s	**0.2549**	**s eh**	**0.1963**	n ey	0.1050	**ay n**	**0.3714**	ow f	0.3079		
s ih	**0.2549**	**eh v**	**0.1963**	t f	0.0692	n n	0.0734	ao r	0.0436		
ao r	0.0405	**v ah**	**0.1963**	ow ey	0.0691	ow n	0.0468	f ao	0.0436		
f ao	0.0405	ow s	0.0335	t s	0.0685	n ow	0.0269	ow ow	0.0334		
s f	0.0342	r ow	0.0188	t ey	0.0353	t n	0.0224	r n	0.0165		
s s	0.0284	z iy	0.0148	t ow	0.0300	uw n	0.0219	s ow	0.0063		
r s	0.0181	iy r	0.0148	r ey	0.0240	s n	0.0196	t uw	0.0051		
s ow	0.0179	r s	0.0131	ow ow	0.0225	iy n	0.0182	r t	0.0050		
s t	0.0134	n ow	0.0122	t th	0.0169	v n	0.0174	s z	0.0013		

Table B.6: Basis vectors of the perfect phonetic transcription taken from NNMF, where 2nd-order data representation is used.

zero		one		two		three		four		five	
iy - ow	**0.2536**	**w - n**	**0.4213**	n - uw	0.1106	**th - iy**	**0.3237**	**f - r**	**0.3788**	**f - v**	**0.4917**
z - r	**0.2517**	n - ah	0.0806	ah - t	0.0756	n - r	0.0552	ao - f	0.0561	ay - f	0.0653
r - z	0.0515	r - w	0.0601	t - t	0.0740	ah - th	0.0410	ao - s	0.0513	ow - ay	0.0460
n - iy	0.0414	ow - ah	0.0571	t - s	0.0681	r - th	0.0406	r - ay	0.0462	v - ay	0.0458
ow - iy	0.0333	ah - w	0.0551	uw - ay	0.0603	ow - r	0.0387	ow - ao	0.0455	r - f	0.0352
ay - z	0.0324	ay - w	0.0450	ow - uw	0.0540	r - s	0.0348	r - ao	0.0285	v - uw	0.0351
ah - z	0.0259	ah - s	0.0352	t - f	0.0457	f - r	0.0345	r - ah	0.0258	v - r	0.0318
r - s	0.0220	t - w	0.0314	ay - t	0.0428	r - f	0.0296	ao - w	0.0258	v - eh	0.0313
r - f	0.0184	uw - ah	0.0290	uw - t	0.0384	ay - th	0.0266	r - eh	0.0251	ay - s	0.0305
iy - iy	0.0181	iy - ah	0.0289	t - ey	0.0372	r - r	0.0246	uw - ao	0.0250	ay - ow	0.0275

six		seven		eight		nine		oh			
s - k	**0.2614**	**v - n**	**0.2163**	n - t	0.1769	**n - n**	**0.4076**	n - ay	0.3701		
ih - s	**0.2614**	**eh - ah**	**0.2155**	ah - ey	0.1146	ay - n	0.0584	ah - f	0.2727		
n - ih	0.0442	**s - v**	**0.2155**	ay - ey	0.0964	ow - ay	0.0483	n - ao	0.1480		
k - f	0.0347	n - eh	0.0403	ey - f	0.0957	f - v	0.0483	ah - n	0.1370		
s - ay	0.0346	ah - s	0.0382	t - ay	0.0926	th - iy	0.0449	ay - f	0.0294		
n - n	0.0342	ow - eh	0.0260	ey - s	0.0708	r - n	0.0447	r - s	0.0176		
ay - s	0.0282	iy - ow	0.0210	ow - t	0.0647	ay - f	0.0446	iy - eh	0.0060		
ow - ih	0.0278	z - r	0.0208	t - ao	0.0442	ay - ow	0.0352	ao - s	0.0048		
k - s	0.0273	ay - s	0.0205	v - t	0.0432	n - ay	0.0273	r - eh	0.0038		
k - ow	0.0186	r - s	0.0162	ey - n	0.0427	iy - ay	0.0262	r - w	0.0026		

Table B.7: Basis vectors of the perfect phonetic transcription taken from NNMF, where 1st&2nd-order data representation is used.

zero		one		two		three		four		five	
r ow	**0.1323**	**ah n**	**0.2007**	**t uw**	**0.2409**	**r iy**	**0.1652**	**f ao**	**0.1840**	**f - v**	**0.2022**
iy - ow	**0.1292**	**w - n**	**0.2001**	n - uw	0.0483	**th r**	**0.1652**	**ao r**	**0.1840**	**f ay**	**0.2022**
z iy	**0.1283**	**w ah**	**0.2001**	n t	0.0483	**th - iy**	**0.1652**	**f - r**	**0.1840**	**ay v**	**0.2022**
z - r	**0.1283**	n w	0.0378	ey t	0.0351	n th	0.0282	n - ao	0.0301	n f	0.0268
iy r	**0.1283**	n - ah	0.0378	ah - t	0.0335	n - r	0.0282	n f	0.0291	ay - f	0.0242
r - z	0.0255	ow - ah	0.0279	t - f	0.0313	iy s	0.0214	ao - f	0.0264	v s	0.0230
n z	0.0206	ow w	0.0279	uw f	0.0311	r - th	0.0208	r f	0.0264	v f	0.0216
n - iy	0.0206	r - w	0.0278	ow t	0.0304	ow th	0.0205	ao - s	0.0233	ay - s	0.0208
ow - iy	0.0166	ah - w	0.0250	ow - uw	0.0304	ow - r	0.0205	r s	0.0233	ah - f	0.0203
ow z	0.0166	ay - w	0.0168	uw - ay	0.0303	ah - th	0.0204	ow - ao	0.0222	v - ay	0.0200
six		seven		eight		nine		oh			
k s	**0.1371**	**ah n**	**0.1122**	**ey t**	**0.2672**	**n - n**	**0.1592**	n s	0.1852		
s - k	**0.1371**	**v - n**	**0.1079**	n ey	0.0560	**n ay**	**0.1571**	ah - s	0.1234		
s ih	**0.1371**	**s - v**	**0.1075**	n - t	0.0557	**ay n**	**0.1571**	n - ih	0.1026		
ih k	**0.1371**	**eh v**	**0.1075**	ah - ey	0.0380	n - ay	0.0533	n - n	0.0793		
ih - s	**0.1371**	**eh - ah**	**0.1075**	ow - t	0.0368	n n	0.0302	ay n	0.0781		
s f	0.0185	**v ah**	**0.1075**	ow ey	0.0368	ay - n	0.0237	n ay	0.0781		
k - f	0.0185	**s eh**	**0.1075**	t - ay	0.0368	n f	0.0209	n - eh	0.0690		
s - ay	0.0176	ow s	0.0183	ey - f	0.0367	ay - f	0.0202	ay - s	0.0642		
s s	0.0147	ow - eh	0.0130	t f	0.0367	f - v	0.0188	ow ow	0.0350		
k - s	0.0147	n - ay	0.0122	t s	0.0350	ay v	0.0188	ow - ay	0.0275		

Table B.8: Basis vectors of the perfect phonetic transcription taken from NNMF, where three unit data representation is used.

zero		one		two		three		four		five	
iy r ow	**0.2464**	**w ah n**	**0.4322**	n t uw	0.1314	**th r iy**	**0.3464**	**f ao r**	**0.3377**	**f ay v**	**0.4231**
z iy r	**0.2464**	n w ah	0.0903	ah n t	0.0918	n th r	0.0654	ao r f	0.0531	ay v s	0.0554
n z iy	0.0428	ah n w	0.0612	t uw f	0.0793	r iy s	0.0466	f ay v	0.0393	ow f ay	0.0509
ow z iy	0.0318	ow w ah	0.0599	t uw s	0.0758	ah n th	0.0436	ow f ao	0.0359	ay v f	0.0397
r ow z	0.0318	uw w ah	0.0312	ow t uw	0.0507	ow th r	0.0431	r f ay	0.0281	v t uw	0.0299
r ow f	0.0301	t uw w	0.0312	t uw t	0.0411	r iy t	0.0245	ao r th	0.0253	ay v t	0.0299
ah n z	0.0276	t w ah	0.0310	uw t uw	0.0411	iy t uw	0.0245	r th r	0.0253	v th r	0.0299
r ow s	0.0252	ey t w	0.0310	uw s ih	0.0400	r iy w	0.0236	v f ao	0.0250	ay v th	0.0299
v z iy	0.0171	ay n w	0.0291	uw ey t	0.0398	iy w ah	0.0236	r f ao	0.0250	v f ay	0.0279
ay v z	0.0171	ah n ow	0.0280	t uw ey	0.0398	r iy ey	0.0233	th r iy	0.0241	iy f ay	0.0278
six		seven		eight		nine		oh			
ih k s	**0.2693**	**v ah n**	**0.2263**	n ey t	0.1517	**n ay n**	**0.4163**	ah n f	0.2937		
s ih k	**0.2693**	**eh v ah**	**0.2263**	ah n ey	0.1042	n n ay	0.0805	n f ay	0.2188		
n s ih	0.0457	**s eh v**	**0.2263**	ey t s	0.0925	ah n n	0.0518	n f ao	0.2041		
f ao r	0.0395	n s eh	0.0423	ey t f	0.0872	ay n s	0.0500	ay n f	0.1304		
k s f	0.0365	ah n s	0.0399	ow ey t	0.0847	ow n ay	0.0499	w ah n	0.0832		
k s s	0.0290	ow s eh	0.0273	t s ih	0.0478	ay n ow	0.0302	v w ah	0.0141		
ow s ih	0.0286	iy r ow	0.0171	ay n ey	0.0469	ay n n	0.0279	ay v w	0.0141		
k s ow	0.0189	z iy r	0.0171	ey t ey	0.0453	uw n ay	0.0275	ay v s	0.0051		
ao r s	0.0188	r s eh	0.0154	t ey t	0.0453	t uw n	0.0275	ay v n	0.0044		
s s ih	0.0184	ah n ow	0.0150	t f ao	0.0448	ey t n	0.0272	v n ay	0.0044		

B.3 Phonetic transcription from original phonetic recognizer

Table B.9: Basis vectors of the phonetic transcription taken from NNMF, where 1st-order data representation is used.

zero		one		two		three		four		five	
r ow	**0.2274**	**w ah**	**0.2892**	**t uw**	**0.3866**	**th r**	**0.2667**	**f ao**	**0.2397**	**f ay**	**0.3027**
iy r	**0.2105**	**ah n**	**0.2830**	**sil t**	**0.1489**	**r iy**	**0.2667**	**ao r**	**0.2397**	**ay v**	**0.3027**
z iy	**0.2105**	**n sil**	**0.1050**	**uw sil**	**0.1127**	**sil th**	**0.1032**	**sil f**	**0.1017**	**v sil**	**0.1130**
sil z	**0.0771**	**sil w**	**0.1049**	n t	0.0681	**iy sil**	**0.0925**	**r sil**	**0.0856**	**sil f**	**0.1021**
ow sil	**0.0521**	ow w	0.0466	uw ow	0.0582	n th	0.0466	r f	0.0354	v s	0.0369
n z	0.0302	n w	0.0445	ow t	0.0466	ow th	0.0332	f ay	0.0327	ow f	0.0266
ow z	0.0277	n f	0.0282	uw f	0.0466	iy s	0.0323	ay v	0.0327	v f	0.0243
ow s	0.0174	t w	0.0165	uw s	0.0450	iy f	0.0229	n f	0.0290	v ow	0.0175
ow f	0.0150	uw w	0.0146	uw t	0.0220	iy th	0.0168	r iy	0.0252	v ey	0.0166
s z	0.0114	s w	0.0134	iy t	0.0169	iy ey	0.0168	th r	0.0252	v th	0.0160
six		seven		eight		nine		oh			
s ih	**0.2099**	**ah n**	**0.1753**	**ey t**	**0.3646**	**ay n**	**0.2879**	**ow sil**	**0.4556**		
k s	**0.2099**	**v ah**	**0.1622**	**t sil**	**0.1585**	**n ay**	**0.2879**	**sil ow**	**0.3300**		
ih k	**0.2099**	**s eh**	**0.1622**	**sil ey**	**0.1347**	**n sil**	**0.1042**	ow ow	0.0959		
s sil	**0.0828**	**eh v**	**0.1622**	n ey	0.0640	**sil n**	**0.1040**	n ow	0.0600		
sil s	**0.0756**	**n sil**	**0.0586**	ow ey	0.0413	n n	0.0537	ow s	0.0335		
ao r	0.0357	**sil s**	**0.0585**	t s	0.0363	ow n	0.0346	ow f	0.0169		
f ao	0.0357	n s	0.0464	t f	0.0357	n s	0.0235	v ow	0.0072		
s f	0.0272	ow s	0.0187	t uw	0.0305	n f	0.0224	t ow	0.0009		
s s	0.0221	r ow	0.0169	uw ey	0.0258	s n	0.0166	s ow	0.0000		
r s	0.0141	n f	0.0155	t ey	0.0218	t n	0.0156	uw ow	0.0000		

Table B.10: Basis vectors of the phonetic transcription taken from NNMF, where 2nd-order data representation is used.

zero		one		two		three		four		five	
iy - ow	**0.2114**	**w - n**	**0.3007**			**th - iy**	**0.2407**	**f - r**	**0.2858**	**f - v**	**0.3641**
z - r	**0.2086**	**ah - sil**	**0.1179**			**sil - r**	**0.0964**	**sil - ao**	**0.1027**	**ay - sil**	**0.1294**
r - sil	**0.0796**	**sil - ah**	**0.1053**			**r - sil**	**0.0858**	**ao - sil**	**0.0968**	ay - f	0.0416
sil - iy	**0.0764**	n - ah	0.0562			n - r	0.0357	ao - f	0.0394	ow - ay	0.0388
r - z	0.0427	ow - ah	0.0521			f - r	0.0276	r - ay	0.0366	r - f	0.0377
n - iy	0.0293	r - w	0.0411			r - th	0.0275	ao - s	0.0351	v - ay	0.0348
ow - iy	0.0286	ah - w	0.0365			ah - th	0.0265	ow - ao	0.0342	t - f	0.0282
ay - z	0.0229	ay - w	0.0334			w - n	0.0253	n - ao	0.0273	v - t	0.0274
ah - z	0.0179	ah - s	0.0270			ow - r	0.0251	ao - ow	0.0265	ey - f	0.0264
r - s	0.0174	t - w	0.0215			r - ow	0.0210	t - f	0.0199	t - ay	0.0246
six		seven		eight		nine		oh			
s - k	**0.2085**	**v - n**	**0.1655**	**ey - sil**	**0.1450**	**n - n**	**0.2596**				
ih - s	**0.2085**	**s - v**	**0.1646**	**sil - t**	**0.1277**	**ay - sil**	**0.1438**				
k - sil	**0.0826**	**eh - ah**	**0.1646**	sil - uw	0.1229	ay - n	0.0476				
sil - ih	**0.0739**	**ah - sil**	**0.0579**	t - sil	0.0897	th - iy	0.0473				
n - ih	0.0327	**sil - eh**	**0.0564**	t - ow	0.0555	n - ay	0.0442				
k - f	0.0264	n - eh	0.0298	t - t	0.0451	ow - ay	0.0359				
s - ay	0.0262	ah - s	0.0279	sil - sil	0.0439	r - n	0.0353				
k - s	0.0206	r - s	0.0215	ow - t	0.0391	f - v	0.0267				
ow - ih	0.0198	ow - eh	0.0197	n - t	0.0317	t - n	0.0240				
n - n	0.0168	iy - ow	0.0192	t - ey	0.0280	iy - ay	0.0239				

Table B.11: Basis vectors of the phonetic transcription taken from NNMF, where 1st&2nd-order data representation is used.

zero		one		two		three		four		five	
r ow	**0.1105**	**w ah**	**0.1502**	**t uw**	**0.2019**	**r iy**	**0.1334**	**f - r**	**0.1451**	**f - v**	**0.1630**
iy - ow	**0.1080**	**w - n**	**0.1502**	**sil t**	**0.0793**	**th r**	**0.1334**	**ao r**	**0.1451**	**ay v**	**0.1630**
z - r	**0.1068**	**ah n**	**0.1499**	**sil - uw**	**0.0793**	**th - iy**	**0.1334**	**f ao**	**0.1451**	**f ay**	**0.1630**
iy r	**0.1068**	**n sil**	**0.0585**	**uw sil**	**0.0594**	**sil th**	**0.0515**	**sil f**	**0.0540**	**v sil**	**0.0601**
z iy	**0.1068**	**ah - sil**	**0.0555**	**t - sil**	**0.0593**	**sil - r**	**0.0515**	**sil - ao**	**0.0520**	**ay - sil**	**0.0585**
r - sil	**0.0416**	**sil w**	**0.0546**	n - uw	0.0359	**r - sil**	**0.0465**	**r sil**	**0.0482**	**sil f**	**0.0569**
ow sil	**0.0410**	**sil - ah**	**0.0546**	n t	0.0359	**iy sil**	**0.0454**	**ao - sil**	**0.0482**	**sil - ay**	**0.0564**
sil - iy	**0.0391**	ow w	0.0254	uw ow	0.0308	n - r	0.0201	n - ao	0.0228	ay - f	0.0180
sil z	**0.0391**	ow - ah	0.0254	t - ow	0.0279	n th	0.0201	n f	0.0204	ow f	0.0170
r - z	0.0210	n - ah	0.0248	ah - t	0.0251	iy s	0.0166	ao - f	0.0198	n f	0.0169
n z	0.0154	n w	0.0248	uw f	0.0246	r - th	0.0165	r f	0.0198	v f	0.0164
n - iy	0.0154	r - w	0.0238	ow - uw	0.0239	ow - r	0.0160	ao - s	0.0173	ow - ay	0.0161
six		seven		eight		nine		oh			
s - k	**0.1064**	**ah n**	**0.0908**	**ey t**	**0.1833**	**n - n**	**0.1315**	**ow sil**	**0.2659**		
ih - s	**0.1064**	**v - n**	**0.0851**	**ey - sil**	**0.0808**	**n ay**	**0.1288**	**sil ow**	**0.1798**		
k s	**0.1064**	**v ah**	**0.0847**	**t sil**	**0.0808**	**ay n**	**0.1288**	**sil - sil**	**0.0915**		
s ih	**0.1064**	**eh v**	**0.0847**	**sil - t**	**0.0722**	**sil - ay**	**0.0536**	n ow	0.0613		
ih k	**0.1064**	**eh - ah**	**0.0847**	**sil ey**	**0.0682**	**ay - sil**	**0.0532**	ow ow	0.0602		
k - sil	**0.0418**	**s - v**	**0.0847**	n - t	0.0340	**n sil**	**0.0492**	ay - ow	0.0439		
s sil	**0.0418**	**s eh**	**0.0847**	n ey	0.0328	**sil n**	**0.0487**	ow - sil	0.0361		
sil s	**0.0380**	**n sil**	**0.0326**	ow - t	0.0246	n - ay	0.0318	ah - ow	0.0357		
sil - ih	**0.0375**	**sil s**	**0.0306**	ow ey	0.0211	n n	0.0225	n - sil	0.0322		
n - ih	0.0165	**ah - sil**	**0.0299**	ah - ey	0.0208	ay - n	0.0178	sil - ow	0.0245		
n s	0.0145	**sil - eh**	**0.0290**	t - ay	0.0200	f ay	0.0146	r - ow	0.0232		
n - n	0.0142	n s	0.0204	t f	0.0193	f - v	0.0146	v ow	0.0188		

Table B.12: Basis vectors of the phonetic transcription taken from NNMF, where three unit data representation is used.

zero		one		two		three		four		five	
iy r ow	**0.2109**	**w ah n**	**0.3601**	**sil t uw**	**0.1878**	**th r iy**	**0.2739**	**f ao r**	**0.2473**	**f ay v**	**0.2916**
z iy r	**0.2109**	**sil w ah**	**0.1301**	**t uw sil**	**0.1358**	**sil th r**	**0.1052**	**sil f ao**	**0.0928**	**ay v sil**	**0.1061**
r ow sil	**0.0825**	n w ah	0.0643	n t uw	0.0807	**r iy sil**	**0.0934**	**ao r sil**	**0.0876**	**sil f ay**	**0.1033**
sil z iy	**0.0772**	ow w ah	0.0601	t uw ow	0.0696	n th r	0.0427	n f ao	0.0436	n f ay	0.0522
n z iy	0.0317	ah n w	0.0423	ah n t	0.0545	r iy s	0.0352	ao r f	0.0358	ay v s	0.0354
ow z iy	0.0279	**ah n sil**	**0.0328**	t uw f	0.0532	ow th r	0.0342	f ay v	0.0311	ow f ay	0.0349
r ow z	0.0271	r ow w	0.0326	t uw s	0.0490	ah n th	0.0308	ao r ow	0.0282	ay v f	0.0256
r ow f	0.0258	ah n s	0.0316	ow t uw	0.0483	r iy f	0.0284	n ay n	0.0279	ah n f	0.0215
ah n z	0.0203	ay n w	0.0216	uw ow sil	0.0370	r iy t	0.0185	ow f ao	0.0248	v f ay	0.0181
r ow s	0.0197	uw w ah	0.0208	uw f ao	0.0270	iy t uw	0.0185	r f ay	0.0188	ay v ow	0.0177
six		seven		eight		nine		oh			
s ih k	**0.2139**	**eh v ah**	**0.1975**	**ey t sil**	**0.1885**	**n ay n**	**0.3030**	ah n sil	0.7220		
ih k s	**0.2139**	**s eh v**	**0.1975**	**sil ey t**	**0.1602**	**ay n sil**	**0.1131**	sil ow ow	0.0500		
k s sil	**0.0844**	**v ah n**	**0.1975**	n ey t	0.0830	**sil n ay**	**0.1097**	**sil ow sil**	**0.0337**		
sil s ih	**0.0759**	**sil s eh**	**0.0676**	ah n ey	0.0559	n n ay	0.0591	n sil ow	0.0331		
n s ih	0.0320	n s eh	0.0357	sil ow sil	0.0554	ah n n	0.0393	n sil th	0.0164		
f ao r	0.0296	ah n s	0.0325	ow ey t	0.0412	ow n ay	0.0346	sil ow s	0.0128		
k s f	0.0276	ow s eh	0.0236	ey t f	0.0307	ay n s	0.0311	n sil ey	0.0125		
k s s	0.0220	z iy r	0.0152	ey t s	0.0295	ay n f	0.0218	n sil f	0.0124		
ow s ih	0.0201	iy r ow	0.0152	t uw ey	0.0276	ay n ow	0.0198	ow sil ow	0.0118		
s s ih	0.0141	ah n f	0.0134	uw ey t	0.0276	ay n n	0.0196	n sil s	0.0107		

B.4 Phonetic transcription from modified phonetic recognizer

Table B.13: Basis vectors of the phonetic transcription taken from NNMF, where 1st-order data representation is used.

zero		one		two		three		four		five	
iy r	**0.1953**	**w ah**	**0.2770**	**t uw**	**0.3667**	**th r**	**0.2383**	**f ao**	**0.2920**	**f ay**	**0.2110**
r ow	**0.1928**	**ah n**	**0.2759**	**sil t**	**0.1794**	**r iy**	**0.1983**	**ao r**	**0.2581**	**ay v**	**0.2096**
z iy	**0.1680**	**n sil**	**0.1101**	**uw sil**	**0.1190**	**sil th**	**0.0920**	**sil f**	**0.1053**	**sil f**	**0.0883**
sil z	**0.0610**	**sil w**	**0.1055**	n t	0.0570	**iy sil**	**0.0811**	**r sil**	**0.1013**	**v sil**	**0.0852**
ow sil	**0.0575**	ow w	0.0454	uw s	0.0439	n th	0.0365	r s	0.0349	v f	0.0390
s iy	0.0305	n w	0.0343	uw n	0.0436	iy s	0.0292	n f	0.0339	th r	0.0263
n z	0.0272	uw w	0.0212	uw f	0.0422	ow th	0.0218	r f	0.0284	n f	0.0259
ow z	0.0230	n f	0.0160	ow t	0.0318	iy t	0.0213	ow f	0.0270	f ao	0.0215
ow f	0.0148	t w	0.0110	uw t	0.0297	t th	0.0197	r t	0.0187	v v	0.0192
uw z	0.0147	v w	0.0101	uw ow	0.0164	r ey	0.0192	r th	0.0130	v t	0.0182
six		seven		eight		nine		oh			
s ih	**0.1992**	**ah n**	**0.1574**	**ey t**	**0.2793**	**n ay**	**0.2699**	**sil ow**	**0.3612**		
k s	**0.1973**	**s eh**	**0.1559**	**t sil**	**0.2055**	**ay n**	**0.2138**	**ow sil**	**0.3390**		
ih k	**0.1889**	**eh v**	**0.1449**	**sil ey**	**0.1673**	**n sil**	**0.1389**	ow ow	0.0543		
s sil	**0.0907**	**v ah**	**0.1429**	**sil t**	**0.0478**	**sil n**	**0.1101**	ow f	0.0273		
sil s	**0.0765**	**sil s**	**0.0628**	t s	0.0324	ah n	0.0388	n ow	0.0270		
n s	0.0275	**n sil**	**0.0623**	n ey	0.0269	ay ah	0.0381	ao ow	0.0249		
s f	0.0265	n s	0.0359	t uw	0.0250	ow n	0.0220	ay ow	0.0200		
f ay	0.0205	ow s	0.0142	t f	0.0244	n n	0.0192	ow t	0.0144		
ay v	0.0202	r ow	0.0134	t ey	0.0207	r n	0.0173	t ow	0.0135		
s t	0.0156	iy r	0.0124	r ey	0.0198	t n	0.0112	ow ao	0.0125		

Table B.14: Basis vectors of the phonetic transcription taken from NNMF, where 2nd-order data representation is used.

zero		one		two		three		four		five	
iy - ow	**0.1826**	**w - n**	**0.2777**	**sil - uw**	**0.1684**	**th - iy**	**0.2299**	**f - r**	**0.2230**	**f - v**	**0.2248**
z - r	**0.1564**	**sil - ah**	**0.1046**	**t - sil**	**0.1510**	**sil - r**	**0.1091**	**ao - sil**	**0.0882**	**sil - ay**	**0.0920**
sil - iy	**0.0714**	**ah - sil**	**0.0926**	n - uw	0.0575	**r - sil**	**0.0922**	**sil - ao**	**0.0850**	**ay - sil**	**0.0726**
r - sil	**0.0685**	ow - ah	0.0444	uw - ay	0.0487	n - r	0.0477	ao - f	0.0326	ay - f	0.0548
r - z	0.0341	r - w	0.0307	t - t	0.0478	r - s	0.0294	n - ao	0.0296	n - ay	0.0447
s - r	0.0303	t - w	0.0286	ah - t	0.0455	r - t	0.0289	ao - s	0.0290	v - ay	0.0325
n - iy	0.0281	ah - f	0.0232	t - n	0.0441	ah - th	0.0284	ow - ao	0.0282	v - sil	0.0325
ow - iy	0.0215	uw - ah	0.0230	uw - sil	0.0414	r - th	0.0262	r - f	0.0265	ow - ay	0.0316
r - s	0.0176	n - ah	0.0227	r - t	0.0400	t - r	0.0247	th - iy	0.0259	ay - v	0.0285
t - z	0.0175	ah - w	0.0213	t - f	0.0345	ow - r	0.0229	r - ay	0.0188	n - n	0.0279
six		seven		eight		nine		oh			
ih - s	**0.1942**	**s - v**	**0.1584**	**sil - t**	**0.1858**	**n - n**	**0.3204**	ay - n	0.2596		
s - k	**0.1903**	**eh - ah**	**0.1567**	**ey - sil**	**0.1630**	**ay - sil**	**0.1189**	n - ah	0.1965		
k - sil	**0.0827**	**v - n**	**0.1506**	sil - sil	0.1572	**sil - ay**	**0.1156**	ah - sil	0.1321		
sil - ih	**0.0697**	**sil - eh**	**0.0557**	t - t	0.0350	ah - ay	0.0835	sil - ay	0.0975		
n - ih	0.0323	**ah - sil**	**0.0511**	ow - ey	0.0290	ay - t	0.0282	n - ow	0.0447		
k - f	0.0255	n - eh	0.0282	t - ey	0.0268	ow - n	0.0256	n - v	0.0237		
s - ay	0.0248	ah - s	0.0265	ey - s	0.0250	ay - ay	0.0247	w - ah	0.0208		
ay - s	0.0198	iy - ow	0.0189	n - ey	0.0232	n - sil	0.0247	ay - w	0.0203		
k - ih	0.0167	ay - s	0.0181	t - ay	0.0200	n - ow	0.0217	ow - n	0.0188		
k - eh	0.0160	ow - eh	0.0173	ey - f	0.0195	r - ay	0.0214	f - ah	0.0170		

Table B.15: Basis vectors of the phonetic transcription taken from NNMF, where 1st&2nd-order data representation is used.

zero		one		two		three		four		five	
iy r	**0.0967**	**ah n**	**0.1262**	**t uw**	**0.1879**	**th r**	**0.1221**	**f ao**	**0.1443**	**f ay**	**0.1104**
r ow	**0.0954**	**w ah**	**0.1156**	**sil t**	**0.0891**	**r iy**	**0.1052**	**ao r**	**0.1267**	**ay v**	**0.1099**
iy - ow	**0.0931**	**w - n**	**0.1119**	**sil - uw**	**0.0771**	**th - iy**	**0.1044**	**f - r**	**0.1247**	**f - v**	**0.1060**
z iy	**0.0831**	**n sil**	**0.0543**	**uw sil**	**0.0609**	**sil th**	**0.0488**	**sil f**	**0.0561**	**sil - ay**	**0.0465**
z - r	**0.0800**	**ah - sil**	**0.0504**	**t - sil**	**0.0580**	**sil - r**	**0.0488**	**sil - ao**	**0.0532**	**v sil**	**0.0442**
r - sil	**0.0368**	**sil w**	**0.0466**	n - uw	0.0321	**iy sil**	**0.0416**	**ao - sil**	**0.0513**	**sil f**	**0.0440**
sil - iy	**0.0356**	**sil - ah**	**0.0459**	n t	0.0317	**r - sil**	**0.0401**	**r sil**	**0.0498**	**ay - sil**	**0.0336**
ow sil	**0.0337**	n - ah	0.0292	uw n	0.0237	iy s	0.0173	n - ao	0.0219	**ay - f**	**0.0223**
sil z	**0.0303**	th r	0.0270	uw - ay	0.0229	iy f	0.0156	n f	0.0208	n ay	0.0201
r - z	0.0173	r iy	0.0186	uw s	0.0219	r - f	0.0154	ao - f	0.0190	n - ay	0.0199
s iy	0.0150	th - iy	0.0184	uw f	0.0219	r - th	0.0136	ao - s	0.0182	n f	0.0196
s - r	0.0147	n w	0.0177	ah - t	0.0218	n th	0.0135	r s	0.0180	v f	0.0188
six		seven		eight		nine		oh			
s ih	**0.1019**	**ah n**	**0.0915**	**ey t**	**0.1462**	**n ay**	**0.1638**	**sil ow**	**0.1701**		
k s	**0.1006**	**s eh**	**0.0790**	**t sil**	**0.1050**	**n - n**	**0.1424**	**ow sil**	**0.1387**		
ih - s	**0.0975**	**eh v**	**0.0734**	**sil ey**	**0.0887**	**ay n**	**0.1392**	**sil - sil**	**0.1050**		
ih k	**0.0970**	**s - v**	**0.0728**	**sil - t**	**0.0848**	**n sil**	**0.0682**	ow - sil	0.0323		
s - k	**0.0957**	**v ah**	**0.0724**	**ey - sil**	**0.0720**	**sil n**	**0.0674**	ow ow	0.0282		
s sil	**0.0458**	**eh - ah**	**0.0721**	**sil t**	**0.0224**	**sil - ay**	**0.0612**	sil - ow	0.0270		
k - sil	**0.0418**	**v - n**	**0.0693**	sil - sil	0.0200	**ay - sil**	**0.0504**	n - ow	0.0257		
sil s	**0.0390**	**n sil**	**0.0322**	t - t	0.0191	ay - n	0.0307	t - ow	0.0232		
sil - ih	**0.0348**	**sil s**	**0.0320**	ey - s	0.0184	ah - ay	0.0237	ow - ow	0.0209		
n - ih	0.0158	**ah - sil**	**0.0312**	t s	0.0169	r n	0.0164	ow n	0.0202		
n s	0.0136	**sil - eh**	**0.0256**	n - t	0.0149	ay ah	0.0133	ow - ay	0.0199		
s f	0.0133	n s	0.0182	n ey	0.0136	n n	0.0107	ow f	0.0168		

Table B.16: Basis vectors of the phonetic transcription taken from NNMF, where three unit data representation is used.

zero		one		two		three		four		five	
iy r ow	**0.1760**	**w ah n**	**0.2936**	**sil t uw**	**0.1572**	**th r iy**	**0.2167**	**f ao r**	**0.2559**	**f ay v**	**0.2050**
z iy r	**0.1545**	**sil w ah**	**0.1093**	**t uw sil**	**0.1181**	**sil th r**	**0.0974**	**sil f ao**	**0.1020**	**sil f ay**	**0.0784**
r ow sil	**0.0665**	**ah n sil**	**0.0613**	n t uw	0.0679	**r iy sil**	**0.0685**	**ao r sil**	**0.0957**	**ay v sil**	**0.0575**
sil z iy	**0.0569**	ow w ah	0.0476	t uw s	0.0444	n th r	0.0352	ao r s	0.0380	n f ay	0.0349
s iy r	0.0274	n w ah	0.0445	ah n t	0.0443	r iy s	0.0347	ao r f	0.0370	ay v f	0.0317
n z iy	0.0229	ah n w	0.0260	t uw f	0.0427	r iy f	0.0233	n f ao	0.0362	ay v n	0.0221
r ow z	0.0206	uw w ah	0.0240	t uw n	0.0379	ah n th	0.0218	ow f ao	0.0351	ay v v	0.0190
r ow f	0.0204	ah n s	0.0209	ow t uw	0.0345	ow th r	0.0216	ah n f	0.0196	ow f ay	0.0183
ow z iy	0.0198	t uw w	0.0172	t uw t	0.0280	r iy t	0.0216	ao r t	0.0191	v f ay	0.0182
s ih k	0.0158	ah n f	0.0166	uw t uw	0.0221	t th r	0.0186	r f ao	0.0190	ay v t	0.0167
six		seven		eight		nine		oh			
ih k s	**0.1902**	**s eh v**	**0.1706**	**sil ey t**	**0.1929**	**n ay n**	**0.2696**	ah n sil	0.3815		
s ih k	**0.1894**	**eh v ah**	**0.1661**	**ey t sil**	**0.1676**	**sil n ay**	**0.1150**	ay ah n	0.0988		
k s sil	**0.0821**	**v ah n**	**0.1584**	sil ow sil	0.0697	**ay n sil**	**0.0929**	n ay ah	0.0906		
sil s ih	**0.0687**	**sil s eh**	**0.0597**	sil t sil	0.0345	ah n ay	0.0459	n sil ow	0.0314		
n s ih	0.0327	n s eh	0.0299	ey t s	0.0288	ay n s	0.0318	sil n ay	0.0282		
k s f	0.0254	ah n s	0.0254	ow sil ey	0.0234	ay n f	0.0264	sil ow ow	0.0259		
k s ih	0.0165	ow s eh	0.0184	t sil t	0.0218	r n ay	0.0217	ow ow sil	0.0195		
k s eh	0.0162	iy r ow	0.0183	ey t f	0.0210	ay n t	0.0217	ow n ay	0.0113		
f ay v	0.0157	z iy r	0.0160	t ey t	0.0193	ay n ay	0.0199	ow ah n	0.0111		
k s t	0.0157	v s eh	0.0109	n ey t	0.0188	ay n n	0.0138	w ay ah	0.0106		

NTNU
Norwegian University of
Science and Technology