Unsupervised acquisition of acoustic models for speech-to-text alignment

Master-Thesis von Benjamin Milde
10. April 2014

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Unsupervised acquisition of acoustic models for speech-to-text alignment

vorgelegte Master-Thesis von Benjamin Milde

Supervisor: Prof. Dr. Chris Biemann
Coordinator: Dr. Dirk Schnelle-Walka

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den April 10, 2014

_____

(Benjamin Milde)

Zusammenfassung

Computersprachtechnologie ist heutzutage weitestgehend durch überwachtes Lernen geprägt. Die beiden prominentesten Computersprachtechnologien, Automated Speech Recognition (ASR) und Text-to-Speech (TTS) Systeme, benötigen erheblichen manuellen Aufwand, um Trainingsdaten zu erstellen und Phonemwörterbücher zu pflegen. Die Anpassung bestehender Systeme auf neue Sprachen und die Verbesserung der Daten, um die Erkennungsgenauigkeit zu erhöhen, ist ein zeitaufwendiger und fehleranfälliger Prozess. So entstanden in den letzten Jahren Ideen, um auch unüberwachtes Lernen in der Sprachtechnologie zu nutzen.

In dieser Arbeit stellen wir eine Methode vor, um mit Hilfe des unüberwachten Lernens phonemähnliche Einheiten allein aus Sprachdaten zu lernen, die unabhängig der verwendeten Sprache funktioniert und nicht auf vorherige Kenntnis setzt. Unser Algorithmus arbeitet in zwei Schritten: Das Sprachsignal wird zunächst an markanten Stellen segmentiert, die so entstandenen Sprachsegmente werden dann durch eine akustische Ähnlichkeitsfunktion in n Einheiten geclustert. Die so gewonnen Einheiten werden zunächst manuell mit einem deutschen Nachrichtenkorpus ausgewertet und dann automatisch mit bereits bekannten Transkripten aliniert. Für unsere automatische Alinierung verwenden wir ein Erwartungsmaximierungs-Verfahren (EM), das wiederholte Anwendungen von Dynamic Time Warping (DTW) beinhaltet. In dem Verfahren wird auch eine Korrelation zwischen Einheiten und Buchstaben aus den Transkripten der Sprache automatisch festgestellt, die manuelle Beobachtungen an unserem deutschen Korpus bestätigt. Wir haben unsere Methode anhand von Wortsegmentierungen eines deutschen, englischen, polnischen und ungarischen Referenzkorpus verglichen. Die Qualität der Alignierung hängt einerseits von der Menge der verfügbaren Sprachdaten ab, anderseits an der Anzahl der Einheiten die verwendet werden. Auch die verwendete Sprache hat Einfluss auf die Ergebnisse, mit besseren Ergebnissen in Deutsch und Ungarisch verglichen mit Englisch und Polnisch. Wir haben auch versucht, unsere Einheiten anstatt linguistischer Phoneme in einem ASR-System zu verwenden. Erste Ergebnisse in dieser Richtung geben Anlass zur weiteren Forschung.

Abstract

Current speech technology is largely characterized as being a supervised learning task. The two most prominent computer speech technologies, automated speech recognition (ASR) and text to speech (TTS), rely on considerable manual effort to generate and maintain training data, phoneme sets and phoneme dictionaries. Furthermore, adapting existing systems to new languages and maintain the data to increase accuracy is a time consuming and error prone process. Thus, ideas to also leverage unsupervised learning in speech technology are emerging in recent years.

In this thesis, we present a way to induce an unsupervised phoneme-like unit representation, which works independent of language and does not rely on prior knowledge. Our algorithm works in two steps: we first determine distinctive segmentation boundaries in speech and propose an acoustic similarity function, which we use to cluster all segments into a set of k units. These units are first manually evaluated with a German news corpus and then automatically aligned to known transcriptions of the utterances in which they occur. For our automatic alignment we use an expectation-maximization (EM) process that involves repeated applications of dynamic time warping (DTW). In the process, a letter to unit correlation is established, which also confirms manually made observations on our German corpus. We also evaluated our alignment process on German, English, Polish and Hungarian segmented reference corpora and compared them to the word segmentation produced by our unsupervised method. The quality of the alignment depends on the amount of available speech data, how many units are used and the target language, with better alignments in German and Hungarian than in English and Polish. We also tried to use our unit representation in lieu of linguistic phonemes in an actual ASR system. First results in this direction are encouraging further research.

Contents

1 Introduction **6**

2 Overview of core speech processing technologies **8**
2.1 ASR systems . . . 8
2.2 TTS systems . . . 9
2.3 Performance ceiling in ASR technology? . . . 9

3 Related Work **11**

4 Speech corpora used in this work **13**
4.1 Timit corpus . . . 13
4.2 DW-news corpus . . . 13
4.3 Simple4all corpora . . . 13

5 Unsupervised speech units **14**
5.1 Blind segmentation . . . 14
5.1.1 Results from related work . . . 14
5.2 Blind segmentation of speech . . . 15
5.2.1 Classical MFCC feature representation . . . 16
5.2.2 Feature representation signal processing steps . . . 16
5.2.3 RPCA feature representation . . . 16
5.2.4 Delta-features . . . 17
5.2.5 Sliding window jumpfunction . . . 17
5.2.6 Clustering units . . . 17
5.2.6.1 Euclidean acoustic similarity function . . . 18
5.2.7 Implementation details . . . 18
5.2.8 Evaluation . . . 19
5.2.9 Unit n-grams . . . 21

6 Unsupervised speech to text alignment **23**
6.1 Character-to-unit correlation . . . 23
6.2 EM algorithm . . . 23
6.3 Dynamic time warping . . . 23
6.4 Iterative method to align and correlate units with DTW-paths . . . 24
6.5 Results . . . 26
6.5.1 Convergence issues . . . 27
6.6 Evaluation . . . 30
6.6.1 Window Diff . . . 30
6.6.2 Evaluation of word alignments . . . 31
6.6.3 Time measurements and scaling . . . 34
6.6.4 ASR integration . . . 36

7 Conclusion **37**

8 Acknowledgement **39**

Bibliography **40**

9 Appendix **43**

1 Introduction

With speech being so ubiquitous and natural to us humans, it does not come as a surprise that researchers were interested in building speech processing systems very early in the history of computing. First electronic machines appeared as early as in the 1930ies for a simple vocalization task and 1950ies for a simple spoken number recognizer [18].

Ever since, humans dreamed of speech technology that is able to understand and transcribe fluently spoken speech as good as themselves and of computer systems that are able to produce natural and realistic sounding language, undistinguishable from human speech. This holy grail of speech technology would be a breakthrough and would change the way we interact with computers. With the proliferation of smart phones and commercial systems like Siri[1], this dream seems to come closer to be fulfilled and yet, a sense of disappointment comes easily as we recognize first-hand that this technology is still in its infancy. After all, it might still take a while until speech technology is on par with human speech capabilities.

Automated speech recognition (ASR) and text to speech (TTS) are the two prominent speech technologies that emerged over the decades. Statistical methods dominate in ASR, notably the hidden Markov model (HMM) framework that was introduced in the 1980's [17,23]. Modern ASR systems are still build using the same mathematical framework at its core.

As of today, ASR and TTS both work on subunits of words called phonemes, the linguistic building blocks of spoken languages. In the current approach, a lexicon is required to map words to a series of phonemes. The set of phonemes a speech system operates on, is usually static and lists of valid phonemes are compiled by linguistic experts for a target language. Both ASR and TTS have in common that usually a large corpus of aligned speech and text is required to build and train them. This means that current ASR and TTS is a supervised learning task and considerable expert knowledge is needed to train such a system. Another problem is that accurate transcribed training material is expensive to generate: In the U.S.A., the resulting cost is $90–$150 per hour of speech [42].

This is contrasted by human speech recognition (HSR): We grow up and learn a language by processing large amounts of unannotated speech and we still manage to learn word inventories and boundaries and can use these abilities to acquire new words [12]. Why shouldn't we try to build systems that imitate these learning abilities?

This would equate to a paradigm change, turning speech processing into an unsupervised learning task, where less expert knowledge is needed. This could prove to be beneficial for ASR and TTS in several ways:

We might need to look for radically different ways of doing speech processing to progress further in the long run. Unsupervised speech processing is promising, because unannotated speech data is much easier to acquire than transcribed speech data. This also means that bigger corpora could be build and used than currently possible, simply because more data can be acquired with less cost. Research in this area could also potentially yield to ASR and TTS systems that outperform current technology significantly one day, even if they perform worse at first.

Promising could also be the field of multi-lingual speech processing: English is arguably the most researched language in speech processing and the same techniques that were developed for English have been applied and adapted to other languages with varying success. Still, there are many languages that lack these technologies, or only have very primitive ASR and TTS systems. The cost and expert knowledge needed to produce speech technology with a supervised system for a new language can be a

[1] http://www.apple.com/de/ios/siri/

barrier of entry, particularly for languages of poorer countries. That barrier could be lowered with less needed expert knowledge in the target language, if unsupervised learning can replace that knowledge.

The fundamental question is, how much can be learned from the available data alone, without expert knowledge of the target language? Do we really need linguistic phonemes and pronunciation lexica, or could that be learned purely from data?

In this thesis, we focus on fundamental issues of unsupervised speech processing: A way to mine an unsupervised phoneme-like unit representation is presented and we evaluate its usefulness on a speech-to-text alignment task. We segment the speech in an unsupervised fashion and use acoustic similarity to cluster those segments. Using parallel corpora in several European languages, namely German, English, Polish and Hungarian we create the alignment by inducing a simple letter to sound model using this unit representation with an Expectation-Maximization (EM) process and DTW alignments. Unsupervised alignment of speech and text can possibly help to build speech corpora in an automated fashion and could allow to build pronunciation dictionaries automatically. One hope is also that such unsupervised knowledge could replace traditional linguistic phonemes and static phoneme dictionaries. In the long run, both TTS and ASR systems could benefit from such additional knowledge with better performance and it makes it possible to adapt existing systems faster to new languages.

The remainder of the thesis is organized as follows: Chapter gives a short overview over how typical ASR and TTS systems work, followed by a discussion if current ASR technology has reached a performance ceiling. Chapter 3 discusses related work in the field of unsupervised speech processing, while Chapter 4 discusses all speech corpora used in this thesis. We show a method to segment and cluster speech units in an unsupervised fashion in Chapter 5. Our unit representation is then used in Chapter 6 to induce a text-to-speech alignment. In that chapter, we also evaluated how useful our unit representation is in an actual ASR system.

2 Overview of core speech processing technologies

Automated speech recognition (ASR) is a major speech processing technology with the objective to transcribe spoken words into text. Text-to-speech (TTS) has the opposite goal, it is the process of synthesizing speech from written text. A brief overview over how both technologies work is given in this chapter.

2.1 ASR systems

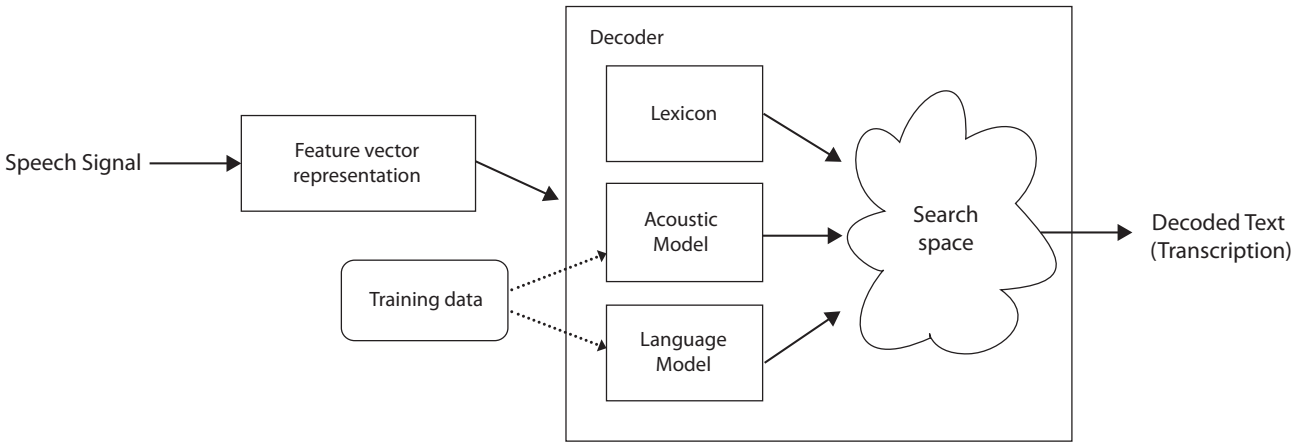

Figure 2.1: Overview of typical components in an ASR system. Figure adapted from the 2012 lecture on ASR of the University of Edinburgh [32].

Figure 2.1 shows the typical components of a traditional ASR system. The signal is first transformed into a feature representation of acoustic observations, e.g. the most popular choice is Mel-Frequency Cepstral Coefficients (MFCC) [6]. Given a sequence of acoustic observations, the role of the decoder is to find the most probable word combination in the search space of all possible word combinations. An intermediate unit representation is used to describe the pronunciation of words, current ASR system typically use linguistically motivated phonemes.

The decoder uses several sub-components: trained acoustic models provide evidence to the likelihood of a unit sequence given the acoustic observations. A lexicon maps phonemes, the intermediate unit representation, to words of a target language. The language model incorporates a prior knowledge about the target language, i.e. it provides evidence for likely and unlikely sequences of words in the target language, independent from acoustic observations. Popular methods to train acoustic models are Gaussian Mixture Models (GMM) and Hidden Markov Models (HMM) [29], while e.g. N-gram language models are used to model the target language. Both acoustic models and language models are trained from transcribed corpora, where parallel speech and text is available. On the other hand, the lexicon component that maps words to phonemes is usually static and is often created and maintained by hand.

The search for the most likely word sequence given the acoustic observations is guided by all of these components. A form of Viterbi decoding [28] is usually used to select the most probable word sequences out of all possible word sequences, which is the output text transcription of the ASR system.

2.2 TTS systems

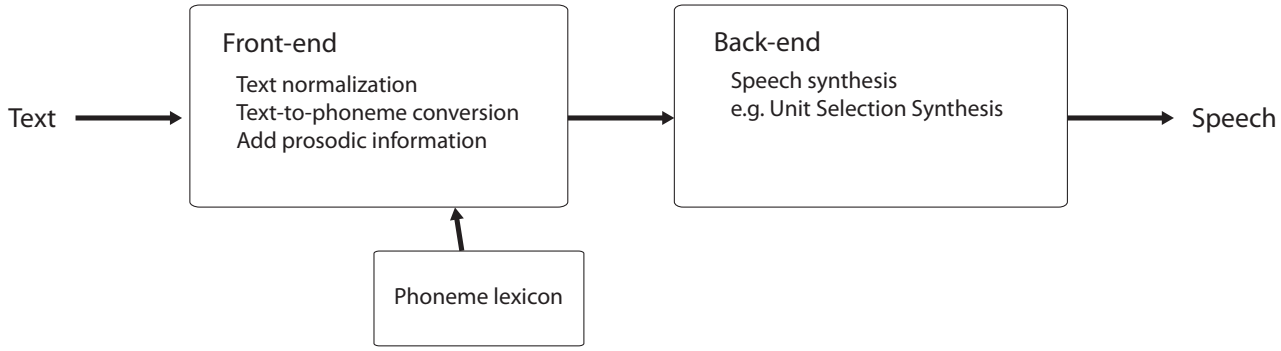

Figure 2.2: A typical TTS system is composed of a front-end and a back-end. The front-end converts the data into an immediate phoneme representation with prosodic information, while the back-end synthesis speech.

A typical TTS system is usually divided into front-end and back-end [36]. The front-end translates raw text into phonetic transcriptions and prosody information. In a first step, the text is normalized, e.g. numbers, abbreviations are converted to words. The normalized text is then converted into an immediate form consisting of phonetic transcriptions and prosodic information. The former is generated in the text-to-phoneme conversation process, which usually uses a phoneme dictionary and incorporates also models to predict the pronunciation of unknown words and proper names. The phonetic transcriptions are augmented with prosodic information, which can mark e.g. prosodic units, like phrases, accentuated word, pitch modulation and many other additional speech features besides pronunciation.

The back-end turns the immediate representation of phonemes and prosodic information into synthesized speech. A popular method in current TTS systems is Unit Selection Synthesis [15]. A large database of recorded sound examples of phones, diphones, syllables, and sometimes even larger units is needed. Speech is then constructed, given phonetic transcription and prosodic information, by concatenating the best matching units from the database. Minimal digital signal processing (DSP) is applied to concatenate the units, but the goal is to reduce artificial DSP as much as possible so that speech sounds as natural as possible.

Since segmenting speech into appropriate units takes considerable manual effort, forced alignment is often used to create an alignment automatically [24], from which unit segments can be extracted. A trained ASR model in the target language is needed for force alignment and is modified to use a known transcription of speech utterances instead of searching for most likely transcription. The accuracy of segmentation boundaries is very important, otherwise incorrect speech fragments are used in the synthesis of speech. Thus, automatic alignments are often manually corrected in the generation of a TTS corpus for Unit Selection Synthesis [10]. Having a larger database is advantageous, since more units from which a TTS can chose from, means also that finding better matching units is more likely when speech is synthesized. This directly improves the quality of synthesized speech because less DSP is needed.

2.3 Performance ceiling in ASR technology?

While the performance of TTS systems is more difficult to measure systematically due to the subjective nature of what constitutes to natural speech, word-error-rate (WER) has been established in ASR to

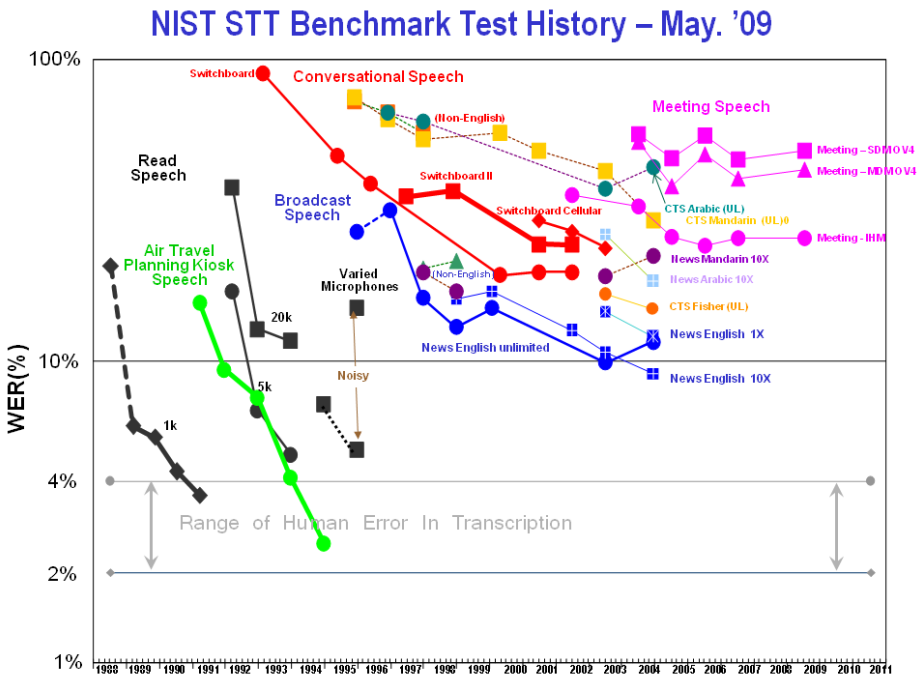

Figure 2.3: Improvements of word-error-rates on various NIST ASR-tasks over the years. Taken from the nist.gov website on ASR history.

measure performance objectively. The national institute of standards and technology (NIST) has been tracking the performance of mainly English ASR technology for the past decades in regular time intervals. Figure 2.3 shows the improvements of word-error-rates on various NIST ASR-tasks over the years. Advancements were quickly made on more isolated and simpler tasks, where the vocabulary is limited and speech quality is good (air-travel kiosk, read speech and broadband speech), with 10% WER being an acceptable goal for a task. A word error rate of 2-4% is considered usual for human transcribers.

In recent years, the attention of NIST has shifted to ASR tasks with an open vocabulary and potentially multiple speakers. These increasingly more challenging tasks, like conversational and meeting speech, show according to NIST diminishing returns of improvements. It might be, that with the current technology and learning paradigm a performance ceiling has been reached. This underlines the necessity to investigate alternate form of doing ASR. Exploring unsupervised learning in the context of TTS and ASR gives rise to the promise to overcome this performance ceiling one day. In the next chapter, we give an overview over the current progress in unsupervised speech processing.

3 Related Work

In the previous years, there is a growing interest in unsupervised speech computing among the speech community. The problem of acoustic speech pattern discovery has received most interest. Park et al. developed a method called segmented dynamic time warping (S-DTW) to find acoustic keywords in speech [26, 44]. S-DTW extends the idea of dynamic time warping (DTW), so that the strict requirement of DTW to match sequences of full length (see also Section 6.3) is lifted and the best matching sub-sequence to a sequence can be found. This makes it possible to use S-DTW for acoustic information retrieval, i.e. acoustic search of a query word, topic analysis, or to discover motifs in speech. This is very different from traditional speech processing, since no transcriptions are needed. The employed algorithms work entirely on the speech signal itself and work on any language. S-DTW has been continuously refined, to also allow discovery of larger patterns [16]. Muscariello et al. used this for word discovery of unannotated speech [25], with the aim of producing audio summaries by keyword extraction. The method produced good preliminary results but needs further adaptation to handle multiple speakers. Jansen et al. showed that term discovery can be used to model phoneme-like speech units across speakers, but notes that the corpus size is limited given the inherent $\mathcal{O}(n^2)$ nature of the term discovery algorithm with S-DTW.

Research has been carried out in identifying linguistic phoneme boundaries in an unsupervised fashion, also called blind segmentation of speech. Aversano et al. [3] used spectral instability detection to achieve this text-independent segmentation of speech. Their algorithm uses spectral analysis of the FFT-signal directly, instead of relying on more common speech feature representations like MFCC. Estevan et al. used maximum margin clustering within a window function over the signal [9]. Both methods segment speech locally with limited context. Also common to these approaches are similar performance scores, despite drastically different methods [30].

Wang et al. [40] uses unsupervised acoustic unit mining to mine phoneme-like units automatically. They improved segment clustering with Gaussian posteriorgram representations and evaluated their work on several languages with various cluster methods.

Recently, there is also growing interest in learning unit representations and pronunciation lexica automatically from data, targeted for use in ASR systems. Glass et al. [12] gives an overview over progress and future in this domain. The ultimate goal is to incorporate more unsupervised knowledge into ASR systems to improve performance significantly one day and to reduce human intervention in training models and creating appropriate corpora. He also introduced different data scenarios, where different degrees of potential unsupervised learning scenarios are explored. In the data-based approach, speech units and words are discovered from parallel text and speech. A more extreme learning approach is the decipher-based approach, where separate and non-parallel text and speech corpora are available and the goal is to decipher the relationship between acoustic words and text.

Lee et al. [21] developed a method to jointly learn phonetic units and word pronunciations with hierarchical Bayesian models and used it to train speech recognizers. The method had been tested on an English corpus. While this method does not outperform traditional ASR system, it can outperform grapheme based recognizers (in English), paving the way for future improvements.

The task of aligning speech and text automatically helps to build larger speech corpora. Traditionally, this can be done with forced alignment, when a trained HMM acoustic model is available for the target language [24]. When the goal is to align speech and text at the phoneme level, for example to train TTS systems, then manual segmentations are usually of better quality then automatic ones [5]. They are, however, very impractical for larger corpora due to their high cost of manual annotation at the phoneme level.

Related is the alignment of 'found speech' with imperfect transcriptions. This is readily available for many languages in the form of audio books, news transcriptions and so on; and thus particularly useful. Hazen [14] developed a supervised technique to align long audio segments in this context. A particular challenge is that 'found speech' can often contain small errors in the corresponding transcriptions. After obtaining anchor points with traditional force alignment, Hazen used finite-state transducers (FST) networks that account for insertions, deletions, and substitutions of words to pseudo-align the transcriptions, which greatly improves the alignment in the presence of errors.

The MAUS-Framework [20] is a ongoing research effort to use forced alignment for constructing automatically aligned corpora. A web version, called WebMAUS [20] is available to the public and produces TextGrid word and phoneme alignments. TextGrid is a popular annotation format for speech, which can be edited with the open-source Praat-Software [1].

Stan et al. [37] developed a lightly supervised grapheme based method for the alignment of unsynchronised and imperfect transcripts, considering that no initial acoustic models and no phoneme lexicon is available. The TUNDRA corpus is a European speech corpus of 14 languages and is a result of using this method on free audio books. It contains approx. 100 hours of total speech, aligned at the sentence level. There is also growing interest to build complete TTS systems with unsupervised or semi-supervised techniques. Watts et al. [41] build automatic TTS systems for these 14 languages with minimal expert knowledge, relying on letters as acoustic units instead of phonemes.

[1] http://www.fon.hum.uva.nl/praat/

4 Speech corpora used in this work

Several speech corpora have been used in this thesis to evaluate the proposed algorithms. All corpora are divided into multiple files, which are either sentence transcriptions and their corresponding recorded speech, or are divided by pauses in the audio signal and the corresponding transcriptions. Table 4.1 gives an overview over all speech corpora.

Corpus	Language	Hours	Alignment	Avg. words per file
Timit	English	1h	phonetic alignment by linguistic experts	8.68
DW-News	German	3.5h	automatic, WebMAUS	6.84
Simple4All	English	2.6h	automatic, semi-supervised	19.2
	Polish	2.3h	automatic, semi-supervised	12.82
	Hungarian	5h	automatic, semi-supervised	14.39

Table 4.1: Speech corpora used in this thesis.

4.1 Timit corpus

The TIMIT corpus is designed for acoustic-phonetic studies in English [11]. While this corpus is over 20 years old, it is an invaluable resource for several reasons. It contains reliable hand annotated transcriptions on the phoneme level for over 6000 sentences, which is a tedious annotation task. Almost all work on phoneme recognition and boundary prediction for English in the past 20 years has been evaluated using this corpus, which makes results comparable. A smaller part of this corpus is also publicly available and used in this thesis: 160 sentences, consisting of 6185 hand-made phoneme segmentations.

4.2 DW-news corpus

We have harvested 36 hours of speech from dw.de, which has German spoken daily news with its corresponding transcriptions. Typically, a daily news item is 5 minutes long. A total of three and a half hours of speech belonging to one particular speaker have been cleaned from radio jingles by hand and were then aligned with WebMAUS [20]. WebMAUS divided the corpus into files belonging to speech segments separated by audible pauses in the speech signal. It produced Textgrid files (which can be viewed with the popular speech analysing software Praat [1]). Subjectively, the automatic segmentation on word level is of good quality.

4.3 Simple4all corpora

The simple4all corpora consists of automatically aligned audio books to their imperfect transcripts. They have been aligned by using the semi-supervised approach described in [37]. All data can be freely downloaded, which consists of sentences and the corresponding speech. Each corpus is a single audio book read by a single speaker and all utterances are recorded under the same circumstances.

[1] http://www.fon.hum.uva.nl/praat/

5 Unsupervised speech units

The goal of finding unsupervised speech units is to find units that are representative of the basic building blocks of a language. This is traditionally done by segmenting the speech into pieces, which are then clustered [4]. A simply method would be to find appropriate and distinctive boundaries locally in the speech signal, which are then boundaries between units. We can then use an acoustic distance measure to compare and cluster these unlabelled units.

Recorded speech is a continuous signal in the time domain, which makes finding distinctive boundaries difficult. A common approach in speech technology is to transform the signal into the frequency domain first, where changes in spectral energies can be identified.

It can be observed that maximum spectral transition positions correlate with phone boundaries in speech signals and that mid-signal phonemes are often stationary [8, 9]. A simple algorithm to identify these maximum spectral transition positions is shown in Section 5.2. An acoustic similarity function is build in Section 5.2.6, that can be applied to any speech signal.

5.1 Blind segmentation

There have been a few attempts to find the boundaries between phonemes in speech signals, without using any additional information than the speech signal itself. This is generally called a blind segmentation of speech. Most previous work in this area has focused on comparing an automatic segmentation to a human made one (indicating phoneme boundaries). A high correlation to the existing phoneme inventory is good, however not the ultimately goal for a data-driven inventory. In fact, an unsupervised method could potentially find a better representation than traditional phonemes, thus comparing the exact boundaries of both methods gives only a vague performance indicator.

5.1.1 Results from related work

Algorithm	**HR (%)**	**OS (%)**
Räsänen et al. (2009) [30]	71.9	-6.90
Almpanidis and Kotropoulos (2008)	80.72	11.31
Aversano et al. (2001)	73.58	0.00
Esposito and Aversano (2005)	79.30	9.00
Estevan et al. (2007)	76.00	0.00
This thesis	74.86	3.87

Table 5.1: Results on blind segmentation from different authors. Taken from [30].

Räsänen et al. did a comparative study on such local blind segmentation algorithms, which compares them in terms of over segmentation rate OS (%) and hit rate HR (%), see Table 5.1. 'Hit rate' is the percentage of correctly identified segmentation boundaries, which lie within 20 ms of a reference segmentation. 'Over segmentation' refers to whether the algorithm produces more segmentation boundaries compared to a reference segmentation, or less, i.e. over-segments or under-segments the speech signal.

The authors employ radically different strategies to identify maximum spectral transition positions and yet they have achieved similar results compared to the phoneme segmentation of the TIMIT corpus

at 70-80% hit rate with 0% over-segmentation, i.e. no over or under-segmentation. Thus, Räsänen et al. [30] argues that this indicates that there might be no significantly better method using a pure bottom-up blind segmentation technique and that 70-80% is the performance ceiling for it.

With this in mind, any approach to blind segmentation that lies within the 70-80% hit rate range is viable approach. Based on the comparative study, we decided to use and reimplement the approach proposed by Estevan et al. However it soon turned out that maximum margin clustering [43] (MMC) is difficult to implement or come by and does place a burden on runtime. In the proposed method, it is needed for every window computation. Zhang et al. [43] notes that even for small data sets (100 data points, with 64 features each), the original MMC formulation takes considerate time for computation in the range of minutes. We tried to use the faster SVR based method proposed by Zhang et al., but we could not reproduce the same results as with MMC-clustering. However, we found out that we could drop the MMC requirement altogether with just a slight loss in performance compared to traditional phonemes, using a slightly different method to blind segmentation. Since our main goal is not necessarily to segment speech into linguistic phonemes, but to induce unsupervised units, a slight performance drop in respect to linguistic phonemes is not of great concern for us.

In the next sections, a slightly simpler method is proposed for blind segmentation of speech. It is inspired by the window approach used by Estevan et al., but does not need maximum margin clustering. Thus, an advantage of our method is that it is very fast and can potentially be used on 1000's of hours of speech, see also Section 6.6.3.

5.2 Blind segmentation of speech

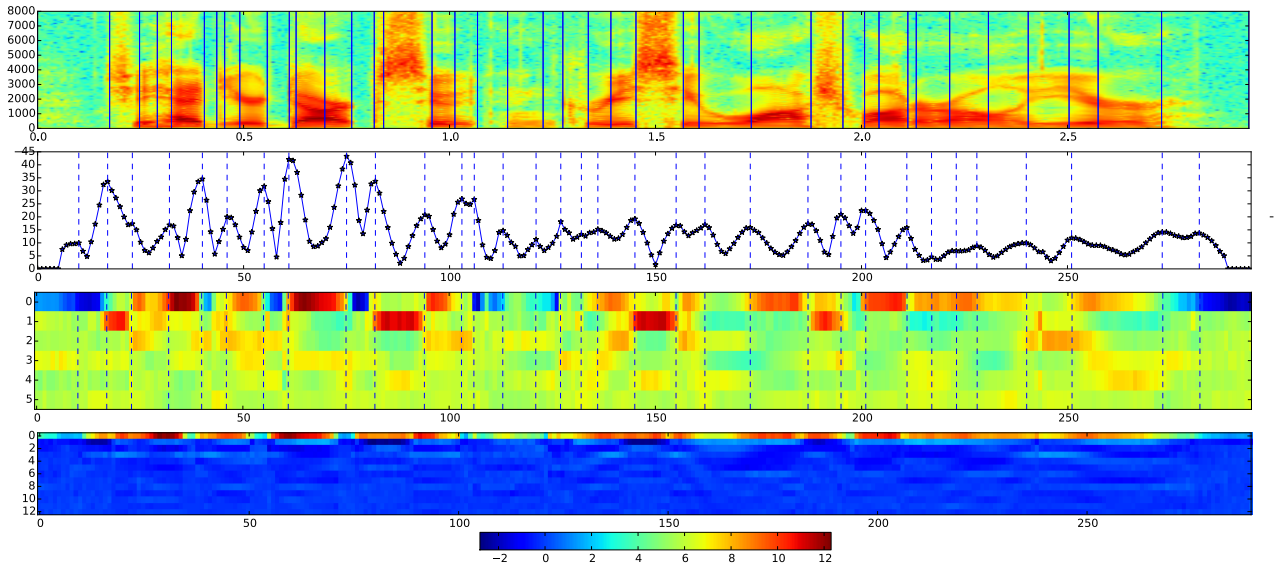

Figure 5.1: Blind segmentation of speech. From top to bottom: audio source as logarithmic spectrogram with manually marked phoneme boundaries from the TIMIT corpus, jump function with highlighted maxima, blind boundaries in RPCA-6 feature space, MFCC features of the signal.

Figure 5.1 shows segmentation results on the phrase "she had your dark suit in greasy wash water all year" from the Timit corpus. The source audio signal is displayed as spectrogram with manually marked phoneme boundaries from human annotators in the second row. The spectrogram shows the intensity of frequencies in audio signal. In the third row of the diagram is a function whose maxima are used to predict boundaries, see Section 5.2.5. RPCA-6 feature space and MFCC feature, shown in the last two rows are explained in Section 5.2.3 and Section 5.2.1.

5.2.1 Classical MFCC feature representation

Mel Frequency Cepstral Coefficients (MFCC) [29] are the most widely used feature representation in speech processing. They have been modelled after how humans perceive speech, with the goal to adequately capture phonetically important characteristics of speech.

5.2.2 Feature representation signal processing steps

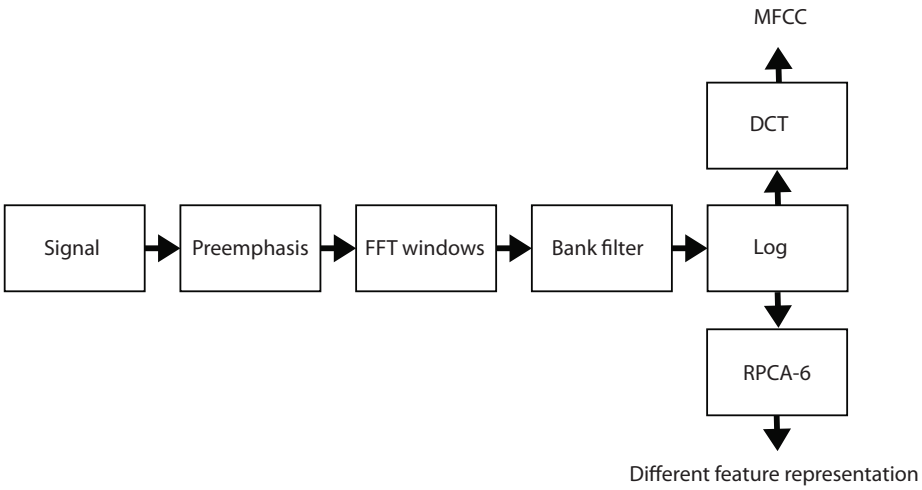

Figure 5.2: Signal pipeline to generate MFCC features and experimental (R)PCA features as an alternative.

Figure 5.2 shows the typical processing steps to transform a sampled audio signal into a sequence of feature vectors. In a first step, called preemphasis, higher frequency energy in the signal is emphasized. This is done so that the characteristics of lower frequency energy is not overrepresented in the speech signal. Then, a window is shifted over the samples of the audio signal, which is multiplied with a window function (e.g. a hanning function), to minimize signal discontinuities in the beginning and end of the windows. In this work we use 25 ms large windows which are shifted by 5 ms. Fourier transformations are then applied successively to the windows, which convert the samples of the window from the time domain into the frequency domain and yield the spectrum of the signal. Studies show that humans perceive audio not linearly and the mel-scale [38] has been invented to model perceived pitch better than at a linear scale. In a next step triangular bank filters are used, those bandwidths occupies constant intervals on the mel-scale, to imitate how humans would perceive sound. The power outputs of the bank filter produce the mel-spectrum, which can also be expressed logarithmically. When MFCC features are computed, a discrete cosine transformation (DCT) is performed as a last step to decorrelate the dimensions.

5.2.3 RPCA feature representation

Instead of using DCT to decorrelate the feature space, PCA can also be used. It projects the output of the bank filters into a smaller feature space with linearly uncorrelated dimensions, which showed to be beneficial for our segmentation algorithm. A disadvantage is that the feature representation must be learned from data and that it is unique to the corpora it is trained on. However, after training, new signals can also be transformed. We use a randomized version of PCA, called RPCA [33], which has a much better asymptotic runtime of $\mathcal{O}(n)$ instead of PCA's $\mathcal{O}(n^3)$.

5.2.4 Delta-features

The rate of change in a spectral feature space gives additional cues to identify sub-word units (like phonemes). For speech recognition, the feature vector is usually augmented by delta features. These estimate first and higher-order derivatives of each feature dimension by using discrete derivatives. Usually, the first two derivatives of each feature dimension are used, these additional features are called delta and double-delta features.

5.2.5 Sliding window jumpfunction

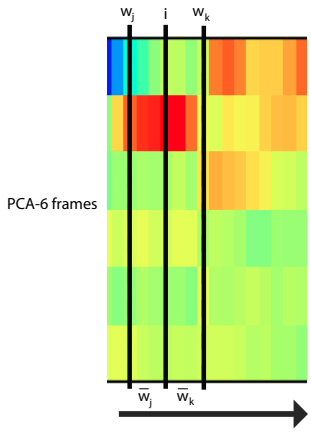

Figure 5.3: Jumpfunction: difference of averages of $\bar{w}_j$ and $\bar{w}_k$ in i.

We use a *jump function* to detect changes in the feature space over time, to find candidates of unit boundaries. We move a window $w = (w_j, w_k)$ successively over all frames of a feature vector representation and calculate the average window halves $\bar{w}_j$ and $\bar{w}_k$. Figure 5.3 illustrates this. The difference between $\bar{w}_j$ and $\bar{w}_k$ can be interpreted as a jump function $f(i)$ that measures the averaged change in position i. Maxima of $f(i)$ can be interpreted as points of highest change in the feature space between w_j and w_k, which we use as potential unit boundaries. A larger window size will lead to smoothing of the jump function, resulting in less maxima and less potential boundaries. Likewise, a smaller window size produces more potential boundaries, since smaller fluctuations in the feature space are picked up as maxima of the jump function.

5.2.6 Clustering units

We can interpret signal fragments between boundaries as units and these units can be clustered to obtain a labelled unit inventory. In order to cluster in a meaningful way, an acoustic distance function needs to be defined. One problem is that the units will have different lengths. Thus, dynamic time warping (DTW) over the feature vectors of units would be a natural choice to compare the units. DTW is a distance-like measure commonly used to compare series of data of different length. It can also be used to find a non-linear warping path, which best matches both data series. DTW can be used to compare speech, where different speeds of pronunciation need to be taken into account.

However, simpler and faster cluster algorithms, like K-means, require an euclidean distance function that compares vectors of fixed length. It is not possible to use K-means with DTW directly since there

(currently) is no approach to meaningfully average signals of varying lengths, so that K-means still convergences without an euclidean distance measure [31]. Other cluster algorithms can work with different distance functions, but would require a full cost matrix between all units to be calculated, which will quickly become impractical for larger data sets, due to the quadratic runtime and memory requirement.

5.2.6.1 Euclidean acoustic similarity function

There are several lower bounds for DTW in regard to one dimensional sequences. E.g., the lower bounding function `LB_Kim` introduced by Kim et al. [19] works by extracting a four-tuple feature vector from each sequence. The features are the first and last elements of the sequence, maximum and minimum values. These feature vectors can than be compared with an euclidean distance measure. In a similar fashion, we extend the idea to approximate DTW on multi-dimensional sequences, by constructing fixed-dimensional feature vectors.

Let n be the length of one vector from the sequence. We use the feature representation shown in Table 5.2. This fixed-dimensional feature representation can be used with an euclidean distance function and thus it can be used to cluster the units with fast clustering algorithms like Kmeans.

Feature	**Dimension**
average vector (across sequence)	1...n
averaged delta vector	n+1 ... 2n
averaged double delta vector	2n+1 ... 3n
first vector in sequence	3n+1 ... 4n
last vector in sequence	4n+1 ... 5n
vector where first dimension is max	5n+1 ... 6n
vector where first dimension is min	6n+1 ... 7n
length of sequence	7n+1

Table 5.2: Feature vector to compare similarity of acoustic units.

5.2.7 Implementation details

We implemented our proposed blind segmentation and unit clustering in Python 2.7, with a Numpy/Scipy stack[1] and Sci-Sklearn[2] as add-on. We used the K-means++ implementation of Sci-Sklearn and also its implementation of RPCA, which are used for the unit generation. Sci-Sklearn's implementation of K-means++ does several runs on the data to avoid local minima. It returns the best run, as determined by 'inertia', the main criterion that K-means minimizes. These runs are independent from each other and run in parallel if multiple cores are available. K-means++ improves K-means by selecting initial cluster centers in a more advantageous way, which usually results in a significant improvement in cluster quality and runtime. [2]

Furthermore, we use the Python MFCC implementation of Sphinxtrain[3] to generate MFCC feature vectors. Sphinxtrain is part of CMU Sphinx, a well tested open source ASR framework. Its Python implementation is used for research purposes and the MFCC features that are generated with it are compatible to the ones that the more popular Java/C++ versions use.

[1] `http://www.numpy.org/`
[2] `http://scikit-learn.org/stable/`
[3] `http://sourceforge.net/projects/cmusphinx/files/sphinxtrain/`

5.2.8 Evaluation

We can evaluate the segmentation results by comparing it to a phoneme segmentation. The TIMIT corpus is a good choice for this task, since it is annotated and segmented on the phoneme level. But, as discussed earlier, this gives only a vague performance indicator and only shows how similar the segmentation is to a phonetic one. The segmentation and labelling used in Chapter 6 to align speech to text and the evaluation in Section 6.6 gives better performance estimates, because no comparisons to linguistic phonemes are made.

Feature	OS %	Hit rate
MFCC	3.93	0.7246
RPCA-6	3.87	0.7487

Table 5.3: Segmentation results on the TIMIT corpus (subset) with a window of 20 speech frames.

Table 5.3 shows segmentation results on the TIMIT corpus. These numbers are based on a publicly available subset of the TIMIT corpus, consisting of only 160 sentences. However, these 160 sentences have 6185 phoneme segmentations, which should still give a good estimate. The rate of over segmentation (OS), in comparison to the TIMIT corpus, can be adjusted by varying the window size parameter in the generation of the jump function. In Figure 5.4 hit rate (HR) and OS is compared with window size, where a smaller window size results in a higher HR and higher OS.

That a higher OS means also a higher HR, is due to the effect that the probability of finding the right segmentation boundaries is higher when more potential segmentation boundaries are proposed. See Figure 5.5 which shows this relationship. A negative OS rate is also possible, which means that the algorithm under-segments the data in comparison to the gold segmentation from the TIMIT-corpus. At a window size of 20, OS is near zero, which means that a similar number of segmentations is proposed by our Algorithm, compared to the number of gold segmentations in the TIMIT corpus.

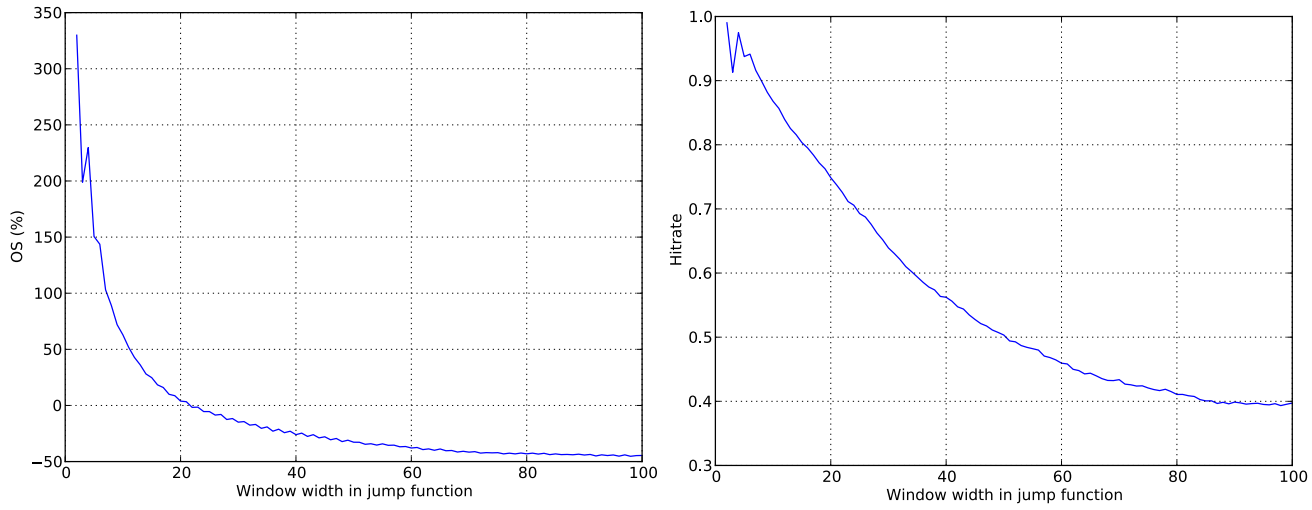

Figure 5.4: The effect of varying window width in generation of the jump function on over segmentation (OS) and hit rate (HR), as compared to phoneme boundaries in the TIMIT corpus.

The unit clusters can also be made audible, to give a subjective impression of them and to inspect them by hand. We created audio files for each unit that contain each occurrence found in a corpus, followed by a short silence signal. The occurrences are sorted by their distance to the cluster center, so that

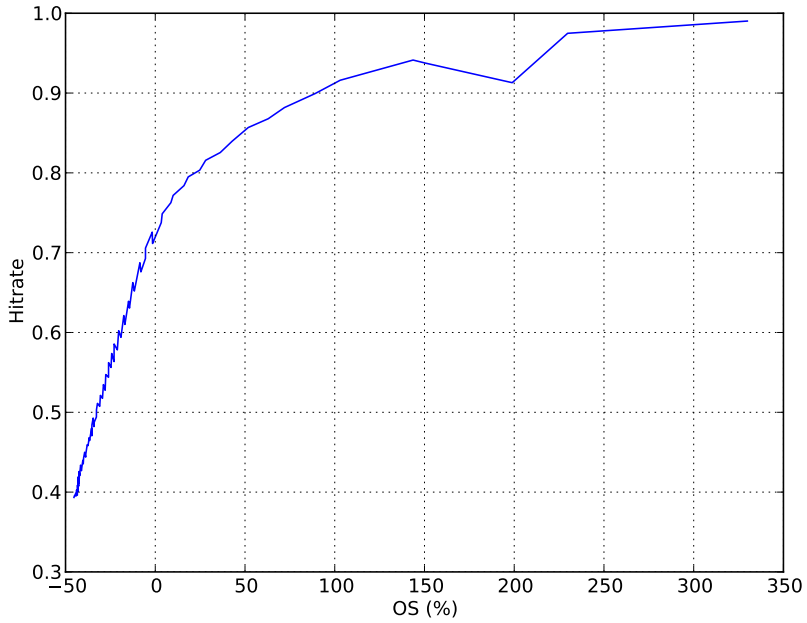

Figure 5.5: The effect of different window widths in generation of the jump function on over segmentation (OS), as compared to phoneme boundaries in the TIMIT corpus. With a window width of 20, a similar number of boundaries can be achieved.

the most characteristic sound examples of a unit are played first. To reduce clipping sounds by abrupt changes between silence and active signal, we introduced a short fade in and fade out of the active signal (3 msec). Some software players, like VLC, allow to lower the playback speed without changing the frequency of the signal. Thus, we used VLC to listen to the audio at reduced speed, which helped especially to better understand very short sound fragments. In the next section, we use this method of manual introspection to analyse common n-grams of the signal.

5.2.9 Unit n-grams

By clustering all previously identified segments of a corpus, a unit representation of the corpus can also be derived. Each segment gets the cluster number it belongs to assigned as label. Then, we can look through all unit labels and identify common recurrences of unit sequences in the form of unit n-grams. Since our copy of the TIMIT-corpus contained only a 160 sentences, we used the German DW-corpus to analyse common German unit n-grams. We used $k = 30$ to cluster and derive a unit representation and looked for all unit n-grams where $n \geq 2$ and $n \leq 20$. We created audio files for the 100 most common n-grams for each n, sorted by the number of occurrences in the corpus. Each audio file contains each occurrence of the n-gram followed by a short silence signal.

Unit representation	**Occurences**	**Words/sounds**
15 15 15 6	74	an, am, ein
3 15 15 15	73	ba, wa, la
6 15 15 15	71	na, ma
12 5 2 12	69	erkl(ä), gek(l), eltw(ei), ...
6 8 1 1	63	long n sound, n(e)
18 12 5 23	61	setz, sitz, seit, selbs
2 29 15 15	56	fa, ta, kla
12 18 12 5	56	präsid, gese(tz)

Table 5.4: 8 most common 4-grams units from the German DW-corpus which are not repetitions of a single unit. Transcriptions are made by listening to the 4-grams.

The most common 4-grams of the German DW-corpus are shown in Table 5.4, along with possible text transcriptions that could by identified manually by listening to all 4-gram occurrences. An observation is that a certain degree of variance occurs and that distinct sounds/transcriptions can get mapped to the same cluster. However, this mapping between text transcriptions and unit appears to have a stable correlation for most units. For example, the 4-grams "15 15 15 6" and "6 15 15 15" show that 15 correlates highly to 'a' and 6 to 'n' and 'm'. Another observation is that especially vowels like 'a', which usually have longer sounds in the speech signal than consonants, get split into successive repeating unit labels of the same unit. Some 4-grams, like "12 8 12 5" show some variance where a dominant representation ("präsid") is observed along side a less occurring but different sounding occurrence ("gese(tz)").

Table 5.5 shows the most common 8-grams of the German DW-corpus. Unsurprisingly, 8-grams units occur a lot less often than 4-grams in the corpus. On one hand, transcription variance among all occurrences of a particular 8-gram is lower, with many 8-grams pointing to distinct words. E.g. The two 8-grams "12 23 1 15 15 15 15 15" and "23 1 15 15 15 15 15 15" point to 7 occurrences of the word USA. On the other hand, the word USA appears 44 times in the German DW-corpus. Likewise 'Hisbollah' appears 5 times in the corpus with 3 occurrences sharing the same 8-gram and 'verfassungs' 10 times, with 3 occurrences having the same 8-gram. This suggests that there is some variance in the exact unit representation of a particular word. Noteworthy is also the unit 21, which is highly correlated to various breathing and smacking noises.

Table 5.6 shows the most common 10-grams from the German DW-corpus. Naturally, some of 8-grams reappear as part of a 10-gram (verfassungs → sverfassungs, zwanzig men → zwanzig mensch, spräsiden → spräsident, ...), since any 10-gram unit also contains 8-grams units. Both Table 5.5 and Table 5.6 do not show all 8-grams and 10-grams with two or three occurrences of which there are of course many more. There seems to be a strong correlation between the unit 23 and 's' and 'z'. Again, unit 21 models noise.

Unit representation	Occurences	Words/sounds
12 23 1 15 15 15 15 15	4	USA
2 29 2 29 23 3 14 23	3	verfassungs
23 3 14 23 16 25 14 12	3	zwanzig men
23 24 2 12 18 12 5 12	3	spräsiden
23 1 15 15 15 15 15 15	3	SA
21 29 15 15 15 15 15 15	3	ba, ta, pa
21 21 21 21 21 21 1 1	3	<breathing noises>
1 21 21 21 21 21 21 21	3	<smacking noises>
18 12 5 23 24 2 29 15	3	sechstau, selbstr, setz hab
16 25 24 29 15 15 15 29	3	die Staa
16 23 24 3 3 15 15 15	3	Hisbollah
16 23 23 12 8 2 12 18	3	isterpräsi

Table 5.5: 12 most common 8-grams units from the German DW-corpus which are not repetitions of a single unit. Transcriptions are made by listening to the 8-grams.

Unit representation	Occurences	Words/sounds
21 21 21 21 21 21 21 21 21 1	4	<breathing and smacking noises>
29 14 14 25 16 18 12 10 16 11	2	antisemit
26 22 10 12 12 23 24 2 12 18	2	US-Präsid
25 29 2 24 16 25 14 20 19 14	2	schaftlichen Dien(s)
24 2 12 18 12 5 12 14 14 8	2	Präsident
24 12 23 24 16 25 16 25 12 8	2	Das griechische
23 3 14 23 16 25 14 12 20 25	2	zwanzig mensch
23 2 29 2 29 23 3 14 23 24	2	sverfassungsge
23 24 2 12 18 12 5 12 14 14	2	spräsident
23 24 16 25 16 25 12 8 29 29	2	s griechische Pa(r)
23 23 29 5 25 24 29 8 25 2	2	stag statt, zeit statt
22 10 12 12 23 24 2 12 18 12	2	US-Präsid

Table 5.6: 12 most common 10-grams units from the German DW-corpus which are not repetitions of a single unit. Transcriptions are made by listening to the 10-grams.

In Chapter 6, these units are automatically correlated to letters from speech transcriptions. The observations made here through manual introspection are also supported by this data, see Section 6.5.

6 Unsupervised speech to text alignment

Usually a phoneme lexicon, phoneme inventory and a trained speech recognizer is needed to do phoneme-level automatic alignment of speech and corresponding transcripts (see also Chapter 2.1). Normally, a speech recognizer searches for the most likely phrase matching to a sound recording. With forced alignment, a speech recognizers model can be used to align speech with an already known speech transcription. This is very helpful for generating aligned corpora automatically. The MAUS-system uses this approach successfully for a number of languages.

However, a trained acoustic model might not be available for every language. In the following sections we evaluate an unsupervised method which does not need a pre-trained speech model in the target language. The method can be used to establish a correlation between single letters and our unsupervised speech units and to align the speech units to text transcripts.

6.1 Character-to-unit correlation

In many languages, there is a correlation of its alphabet's characters to perceived speech sounds of the language. This hidden structure can be exploited to align speech and its transcriptions in an unsupervised fashion. The correlation is stronger in some languages than in others, and thus the approach shown in the next sections has different outcomes for different languages. Of course, there are also languages which do not have a written form which correlates to its pronunciation, like Mandarin or Japanese, for which the approach would need to be adapted in order to work. In the following sections we present a procedure to correlate single transcription letters to speech sounds by using our unsupervised unit inventory. While many languages have more complex pronunciation rules that also depend on multiple letters, a correlation between single letters and sounds can still be established. We combined two well known algorithms, the EM-algorithm and dynamic time warping (DTW) to estimate this correlation. A short introduction is given to both algorithms in the two following sections.

6.2 EM algorithm

The EM-algorithm [7] is an iterative method to the find the maximum-likelihood parameters of a (statistical) model. It is often used when missing or hidden data is involved which makes it impossible to calculate the maximum-likelihood equations directly. Often two interlocked equations can describe the problem, with two sets of unknown parameters. Each set can be estimated when the parameters of the other set are known. The EM-algorithm is based on the idea, that there is an iterative way to solve this problem: In an initialization step, arbitrary values can be chosen for the first set of unknowns. Then the second set of unknowns can be estimated from the first set. A better estimate than the initially chosen arbitrary values for the first set can be calculated from second set. This process is repeated until both sets of unknowns converge. In Section 6.4, such an iterative method is used to find the hidden structure between unsupervised speech units and corresponding text transcripts.

6.3 Dynamic time warping

Dynamic time warping (DTW) is commonly used to align and compare sequences of different length. Its originates from the problem of comparing short speech utterances, which can vary in length and

pronunciation speed [39]. DTW uses dynamic programming to solve this problem. Its objective is to compare two sequences $X := (x_1, x_2, ..., x_n)$ and $Y = (y_1, y_2, ..., y_m)$. The sequences can have discrete values or they can also be sequences of vectors. A local cost measure $c(x, y)$ needs to be defined, that calculates the cost of matching two entities from the sequences. Computing DTW requires $\mathcal{O}(n^2)$ steps. The algorithm returns a warping path, which indicates how the elements of X are mapped to the elements of Y and the cost of matching the sequence X to Y.

6.4 Iterative method to align and correlate units with DTW-paths

Data: Sequences of unsupervised units s_i, Corresponding text transcriptions t_i
Result: Costs[units][characters], alignment paths p_i
Init:
costs[][] ← normalize(ones(no. units, no. characters))
repeat
  E-step:
  forall the *Segments* s_i *and transcriptions* t_i **do**
    p_i = alignpath from DTW(s_i,t_i,$c(x, y)$ = costs[x][y])
  end
  M-step:
  costs ← zeros(no. units, no. characters)
  forall the *Alignpaths* p_i **do**
    costs[unit(p_i)][char(p_i)] += 1
  end
  costs ← normalize(1-(cost/len(s)))
until *convergence of costs*

Algorithm 1: Inducing an alignment of sequences of speech units to corresponding text transcriptions.

If a correlation cost matrix between a single letter of a language and its unsupervised speech units were available, we could find an alignment path directly by applying DTW once. A problem is that we do not know the correlation costs directly. But we could calculate correlation costs out of alignment paths from DTW, if they are available: We use the paths between speech units and transcription letters and count how often a particular unit gets assigned to a particular transcription letter in an alignment path. From these counts, a correlation cost matrix can trivially by computed. However, we also do not have any alignment paths without applying DTW first.

This is a problem of interlocked equations, neither of two needed sets is directly available, but once one of the two is available, the other can be estimated. Thus, it is a good use case for an expectation-maximization (EM) type algorithm. In Algorithm 1 we use this iterative method to align and correlate unsupervised units with single transcription letters. Specifically, we define a local cost function $c(x, y) = costs[x][y]$ that uses a cost matrix $costs$ to store matching costs between all possible letters and units. Alternating steps of DTW-alignment of all segments and alignment cost estimation from aligned paths are used, until the cost matrix used to calculate DTW-paths converges. The initialization-step is a DTW-alignment were we assume that all units have the same matching cost to all letters, so that $c(x, y)$ is constant for every x and y. While being a terrible assumption for virtually every language, it is sufficient as an initialization point that can be iteratively refined until a cost matrix is found that better reflects the hidden structure between speech units and transcription letters.

As with many iterative methods, if the algorithm converges, it indicates that a (local) maximum has been found. It is also possible that the algorithm never converges and that no result can be obtained.

Our iterative algorithm is stopped if the cost matrix convergences or a certain number of iterations is reached.

If convergence is reached, a final alignment with DTW can be used to assign unit positions to letters of the utterance's transcription. Also, the final cost matrix reflects the correlation between letters and unsupervised speech units in the target language. Furthermore, we can also derive the position of words in the unit representation, by using the DTW path segments that get assigned to words from the text transcriptions. The endpoints of a path segment that contains a word points to distinct units in the unit representation. By using positional information of the units we can also establish temporal start and end positions of words in the audio source and segment the original audio source into words. We use a word segmentation obtained in this way in Section 6.6 to evaluate the quality of alignment in several languages. In the next section we show results of our iterative process and how our EM-process converges.

6.5 Results

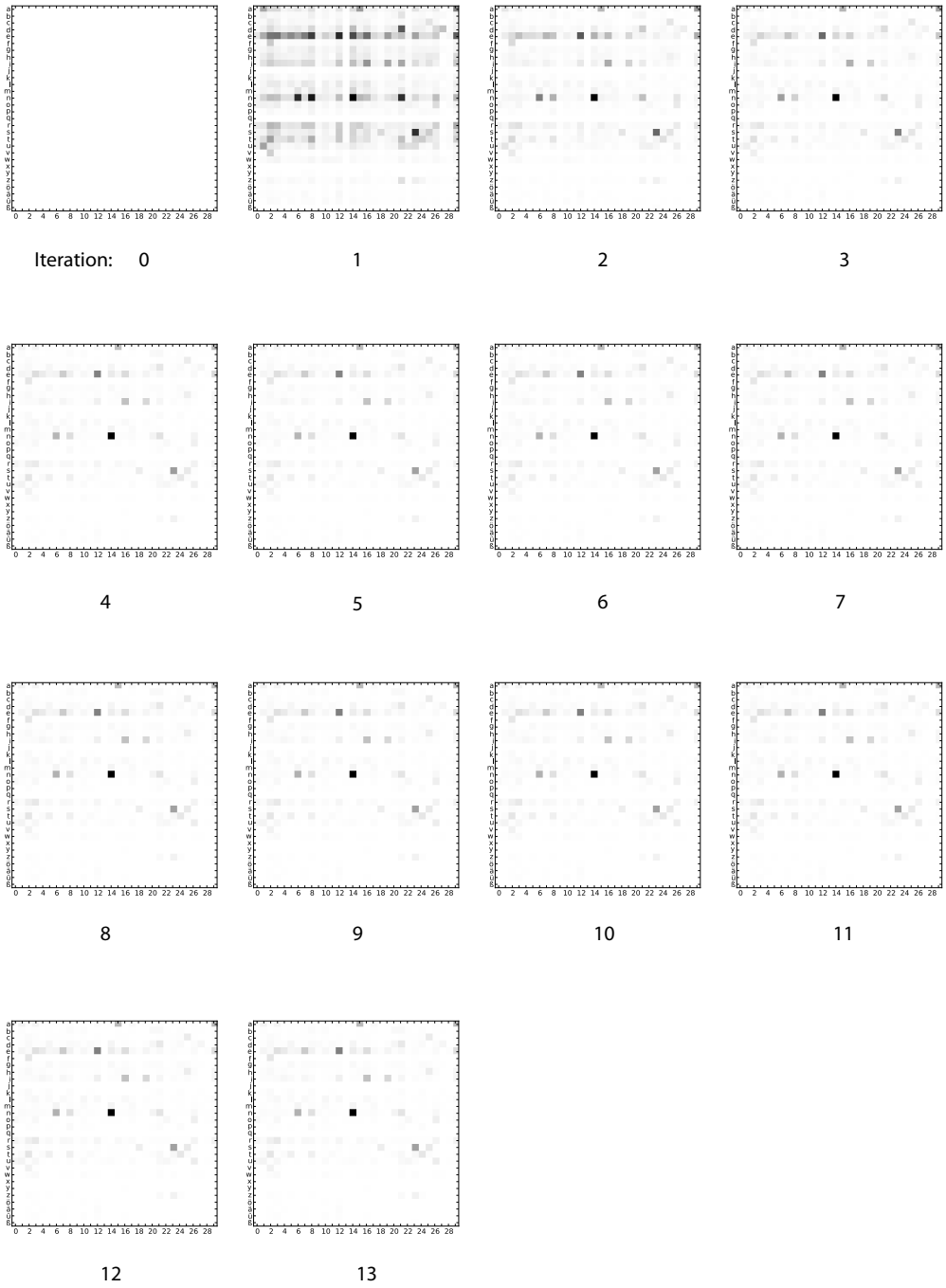

Figure 6.1: Changes in the cost matrix after each iteration of the EM-alignment on the German DW-corpus. The algorithm starts with a cost matrix that assumes the same correlation for all units and letters and converges after 13 iterations.

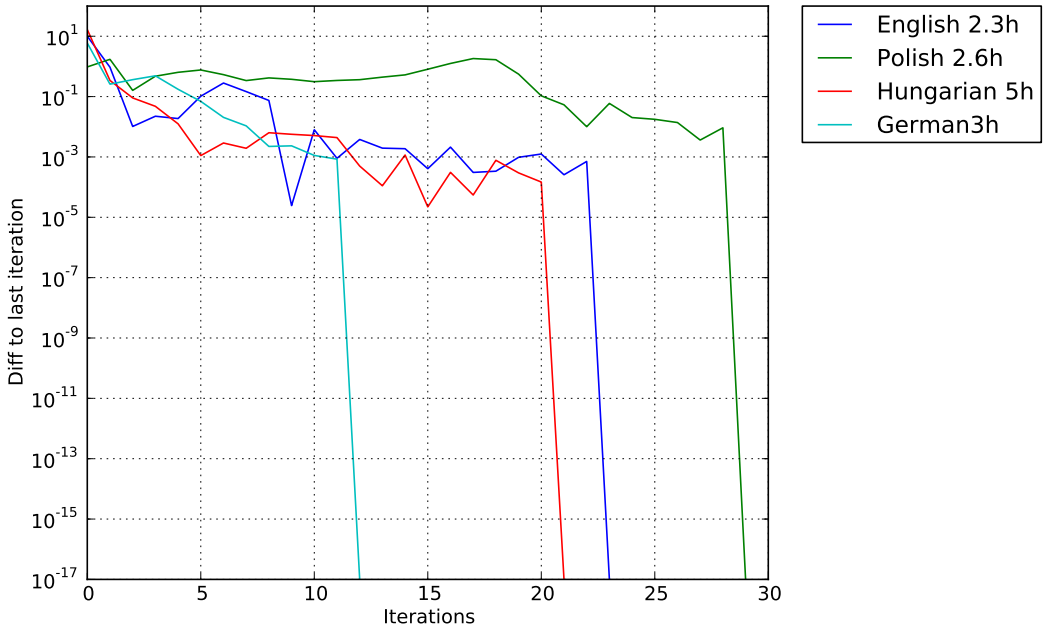

Figure 6.2: Total difference of all entries in the cost matrix after each iteration for the different corpora. For all corpora, it took at most 30 iterations to reach convergence.

Figure 6.1 shows the progression of the cost matrix from EM-alignment on the German DW-corpus in different iterations. The cost matrix shown here correlates single German letters to speech units. For the purpose of better visibility the cost colors are inverted, i.e. dark spots indicate low matching cost and thus high correlation and lighter or white spots indicate high matching cost and low correlation. In the first iteration, all letters and units are assigned to the same correlation costs. The cost matrices are subsequently recalculated in each iteration and convergence usually occurs quickly. In each iteration, all speech utterances of a corpus are aligned at once, so that as many DTW alignments are calculated as there are utterances. Since one iteration then means the realignment of many DTW-paths at once, it might explain why convergence of cost matrix is reached after just a few iterations, see Figure 6.2.

Figure 6.3 shows the correlation cost matrices for all corpora used in the work, namely for the languages German, English, Polish and Hungarian. For the German corpus, some correlations between letters and units have already been identified by listening to and analysing common n-grams in Section 5.2.9. The automatic correlation supports these findings: E.g. `15` correlates highly with the letter `'a'`, `'n'` and `'m'` correlate highly with 6. Some short comings can also be seen in all four final cost matrices and not all correlations are meaningful or clearly assign some sound to a specific letter. Also, very common letters like 'e' correlate with many units. Some of these correlations are probably due to misalignments or segmentation inaccuracies. Also, in many languages a simple letter to sound does not model pronunciation rules very well.

6.5.1 Convergence issues

Especially the corpora with a higher number of average words per file, i.e. longer sequences that need to be aligned, showed convergence problems occasionally. Figure 6.4 demonstrates such a case with the Polish corpus. Here the iterative algorithm got stuck in a local maxima that trivially assigns all correlations only to the most common unit ('14') and most common letter ('a') of the corpus. This

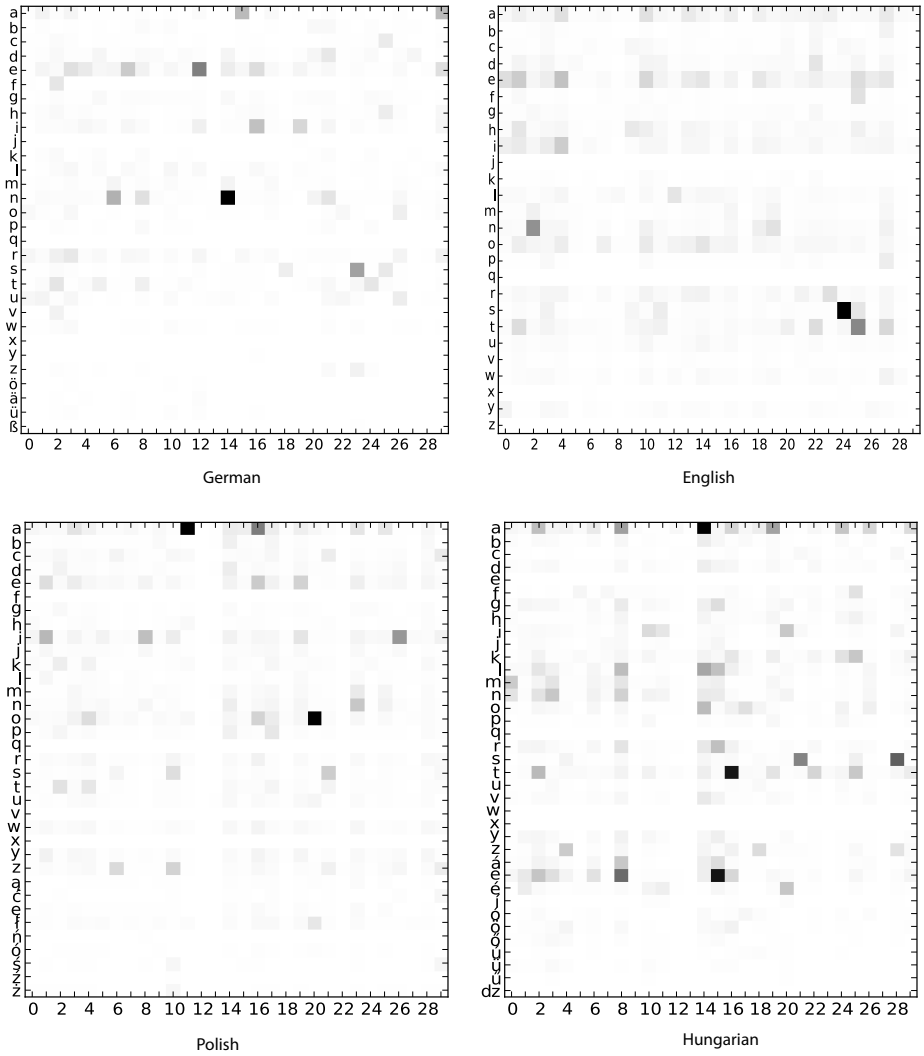

Figure 6.3: Final letter/unsupervised unit correlation matrix for the German DW-corpus and for the simple4all corpora in English, Polish and Hungarian.

did never happen with the German DW-corpus, which has a much lower number of average words per sequence that has to be aligned, compared to the Polish simple4all corpus (see Chapter 4).

We solved this problem by giving shorter sequences more weight in the recalculation of the cost matrix. Specifically, we used the length (number of characters) l_i of transcription t_i to calculate weights w_i for all sequences. We used $w_i = k/l_i$, where k is some constant that can be used to control the impact of shorter sequences.

We modified the M-step in Algorithm 1 so that instead of the constant one, w_i is added to the cost matrix. Algorithm 2 shows the modified version. In the normalization step, the cost is divided by $\sum_{i=0}^{len(s)} w_i$ instead of $len(s)$. By experimentation we found $k = 200$ to be a good value for our corpora, which avoided the problem shown in Figure 6.4.

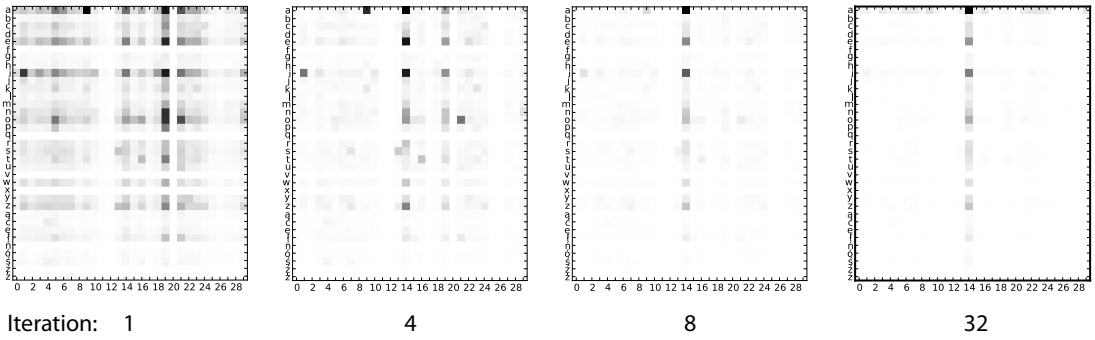

Figure 6.4: Alignment problems with a trivial solution of assigns all correlations only to the most common unit ('14') and most common letter ('a') of the corpus.

Data: Sequences of unsupervised units s_i, Corresponding text transcriptions t_i, Lengths of corresponding text transcriptions l_i
Result: Costs[units][characters], alignment paths p_i
Init:
costs[][] ← normalize(ones(no. units, no. characters))
repeat
  E-step:
  forall the *Segments* s_i *and transcriptions* t_i **do**
    p_i = alignpath from DTW(s_i,t_i,$c(x,y)$ = costs[x][y])
  end
  M-step:
  costs ← zeros(no. units, no. characters)
  forall the *Alignpaths* p_i **do**
    w_i ← k / l_i
    costs[unit(p_i)][char(p_i)] += w_i
  end
  costs ← normalize(1-(cost/ $\sum_{i=0}^{len(s)} w_i$))
until *convergence of costs*

Algorithm 2: Inducing an alignment of sequences of speech units to corresponding text transcriptions, improved with sequence cost reweighing.

6.6 Evaluation

In order to objectively evaluate how good our word alignment is, we need a reference segmentation and a metric to compare segmentations. A human made reference segmentation would be an optimal solution, but only a small portion of the TIMIT was available for this thesis. Large corpora with good hand-made reference segmentations are hard to obtain.

Thus, we use automatic word alignments produced by WebMAUS as reference segmentation, which come quite close to the quality of a human made segmentation. Compared to a phoneme label agreement between three trained phoneticians MAUS achieves a normalized performance of 96.99% [34]. Out of all correctly identified phonemes, the segmentation accuracy of the boundaries follows a Gaussian distribution [35], see Figure 6.5. Usually, the phoneme boundary lies within ±30 ms compared to the reference segmentation of a phonetician. In our German corpus, words have an average duration of 500ms, thus we deemed the automatic segmentation of MAUS to be good enough to be used as reference word segmentation.

By comparing the gold word segmentation with our unsupervised word segmentation, we can measure the effects of parameters in the generation of unsupervised speech units and we see how they will affect the alignment. Another important question is, if alignment quality is influenced by the amount available data. Optimally, using a larger corpus should result in an improvement of alignment quality. Using word alignments makes it possible to accurately measure how useful the unsupervised speech units are for an alignment task, independent of how they compare to actual linguistic phonemes (which was evaluated in Section 5.2.8). A good metric to compare word segmentations is described in the next section.

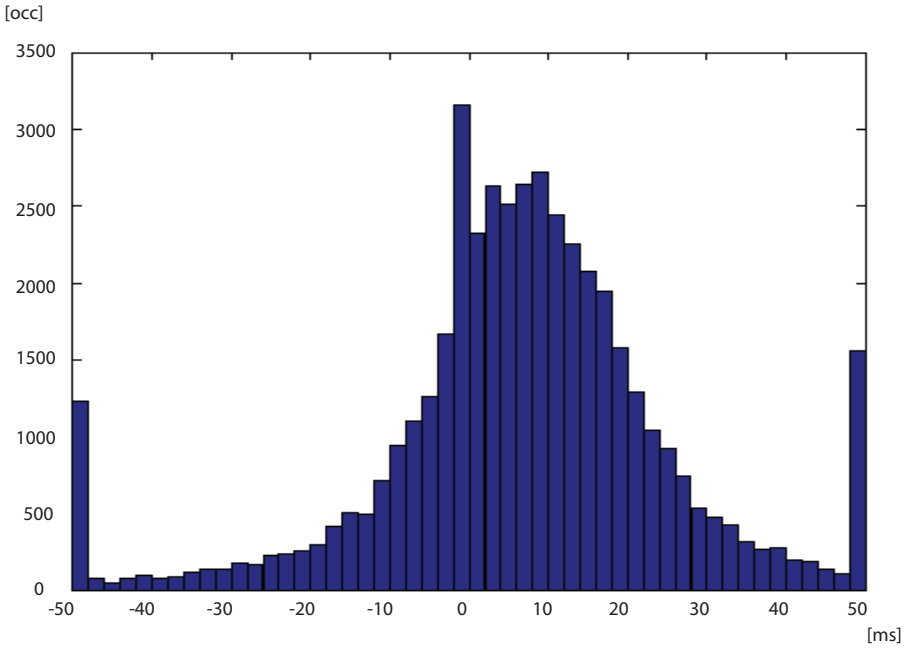

Figure 6.5: Phoneme-boundary accuracy of the MAUS system, compared to a hand-made German reference segmentation. Taken from [35].

6.6.1 Window Diff

The Window Diff (WD) metric, proposed by Pevzner and Hearst [27], aims to be soft measure to compare two segmentations. It uses a sliding window approach, there the window size is typically chosen as half

the average segment size. It is then shifted over both reference and hypothetical segmentations and a counter is increased if the number of segmentations differs inside the window. This way, reference and hypothetical segmentation boundaries that are not identical, but would fit inside the sliding window, are penalized according to how far they are apart, instead of being counted as a missed boundary. Thus, an absolutely identical segmentation would yield the best possible WD score of 0. This score gets bigger, as reference and hypothetical segmentation are more dissimilar.

More formally, for a reference segmentation and hypothetical one, WindowDiff is defined as:

$$WindowDiff(ref, hyp) = 1\frac{1}{N-k}\sum_{i=1}^{N-k}(|b(ref_i, ref_{i+k}) - b(hyp_i, hyp_{i+k})| > 0)$$

where k is the length of the window and N the number of all window positions.

6.6.2 Evaluation of word alignments

We have evaluated the relationship of different parameters on WD scores comparing our word segmentation to the one from WebMAUS. After alignment, we calculated WD scores separately for each utterance and averaged the results. A central question is if using more speech data and transcriptions improves alignment scores. Since we are using a K-means algorithm to cluster data, another question is the effect of choosing different cluster sizes (i.e. choosing different k). Lastly, another parameter, the window size chosen in the generation of the jump function, affects the number of produced boundaries. This should also have an effect on word segmentation, since it has an effect on unit size and unit accuracy. E.g. too large units might have a negative impact on word segmentation accuracy, as larger units could span across words. We also looked at the distribution of WD scores for all utterances in a corpus, to see how useful it is to compare segmentation outcomes by the mean of all separate WD scores.

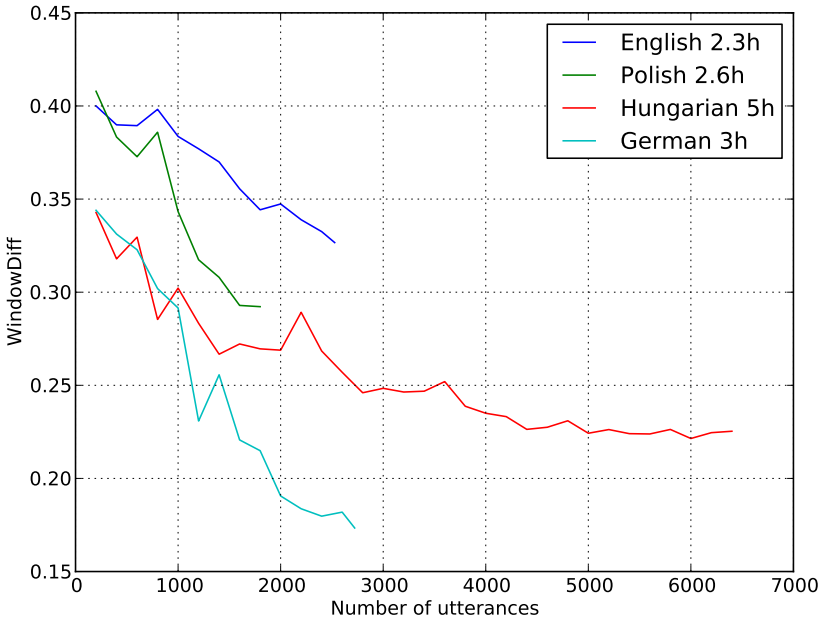

Figure 6.6: WindowDiff (WD) scores for different amounts of available utterances used. Smaller WD indicates better agreement.

Figure 6.6 shows how using different amounts of data (as measured in the amount of utterances used) to generate the unsupervised alignment affects WD scores. With all corpora, a positive effect on WD scores can be observed. Most likely, an additional benefit could be observed for the shorter English, German and Polish corpora if more data were available, while the benefit of adding more data seems to slow down after 5000 utterances for the Hungarian corpus. It can be observed that with the same amount of utterances, the German and Hungarian corpora tend to work better than the English and Polish ones.

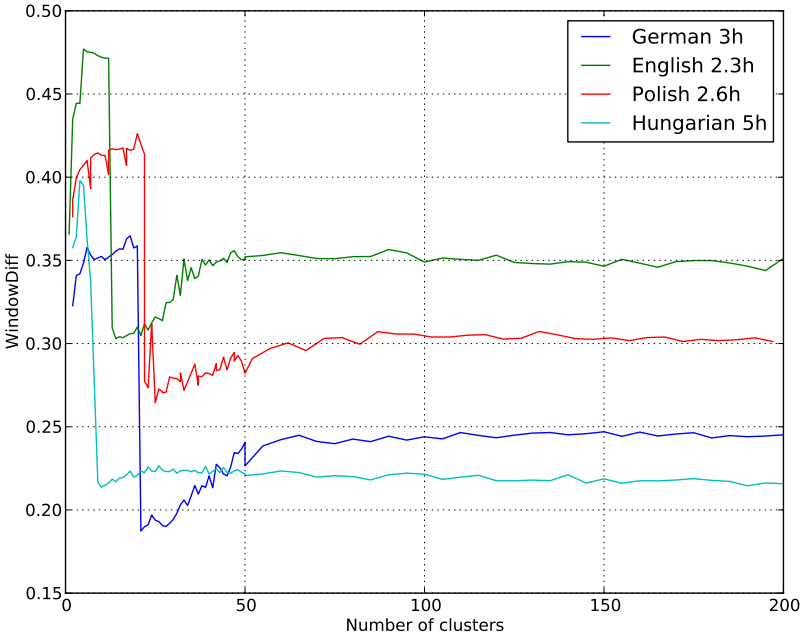

Figure 6.7: WindowDiff (WD) scores for our corpora and different values of k, as chosen in the generation of unit labels. Smaller WD indicates better agreement. For all languages, very small values for k (<10-20) did not lead to a convergence of the EM algorithm.

Figure 6.7 shows the relationship of number of clusters chosen in the unit generation phase and achieved WD score of our unsupervised method in the word alignment task compared to MAUS. For German and Polish, the best WD is obtained for 20-35 clusters and for Hungarian and English 15-25 clusters are better. A similar pattern can be observed for nearly all languages: reducing the number of clusters even further prevents convergence of the EM algorithm and yields worse WD scores. Also, using more clusters than 25-35 results in slightly worse WD scores, with no change after more than 70 clusters.

Figure 6.8 shows the effect of different values of window width, as chosen in the generation of segmentation boundaries. Smaller values for the window width lead to more boundaries and thus smaller units, while larger values for the window width will produce less segmentation points and thus larger units. A window size of 20 produces boundaries at a similar rate as linguistic phoneme boundaries, see also Section 5.2.8. For all languages, very small values of the window width do not lead to convergence. This creates a larger spike to higher WD values, similarly to the spikes in Figure 6.7.

Figure 6.9 shows the spectrogram and alignment of the German phrase "Betroffen sind vor allem die Verwaltung und der Personenverkehr". The first row shows the segmentation from our unsupervised method and the second row shows the reference segmentation produced by MAUS. This phrase has a fairly typical WD score of 0.185 (average for all utterances: 0.175).

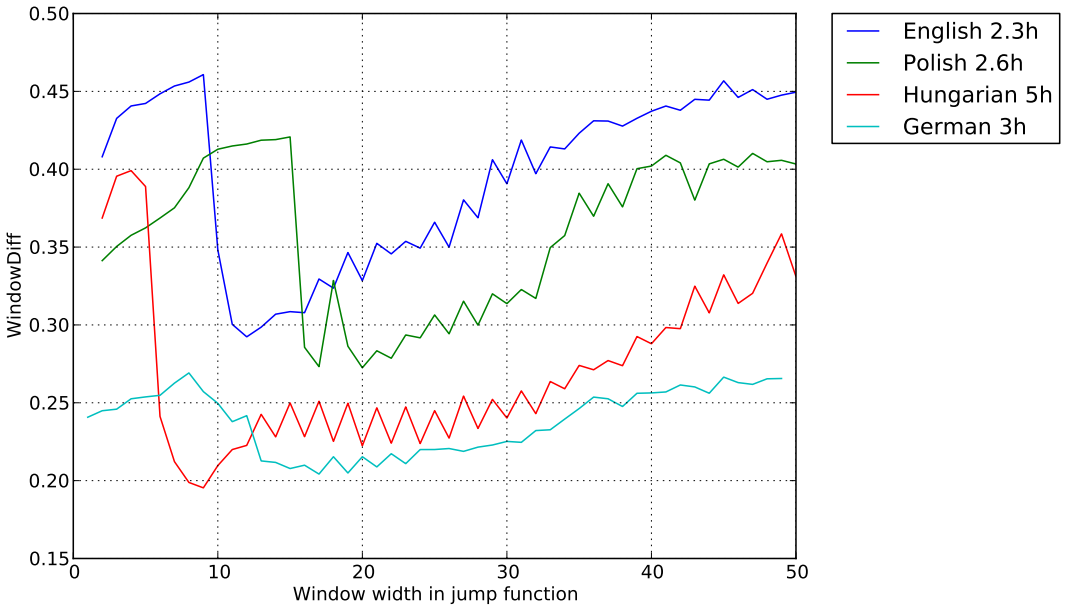

Figure 6.8: WindowDiff (WD) scores for our corpora and different values for the window width, as chosen in the generation of segmentation boundaries. Smaller WD indicates better agreement. For all languages, very small values for the window width did not lead to a convergence of the EM algorithm.

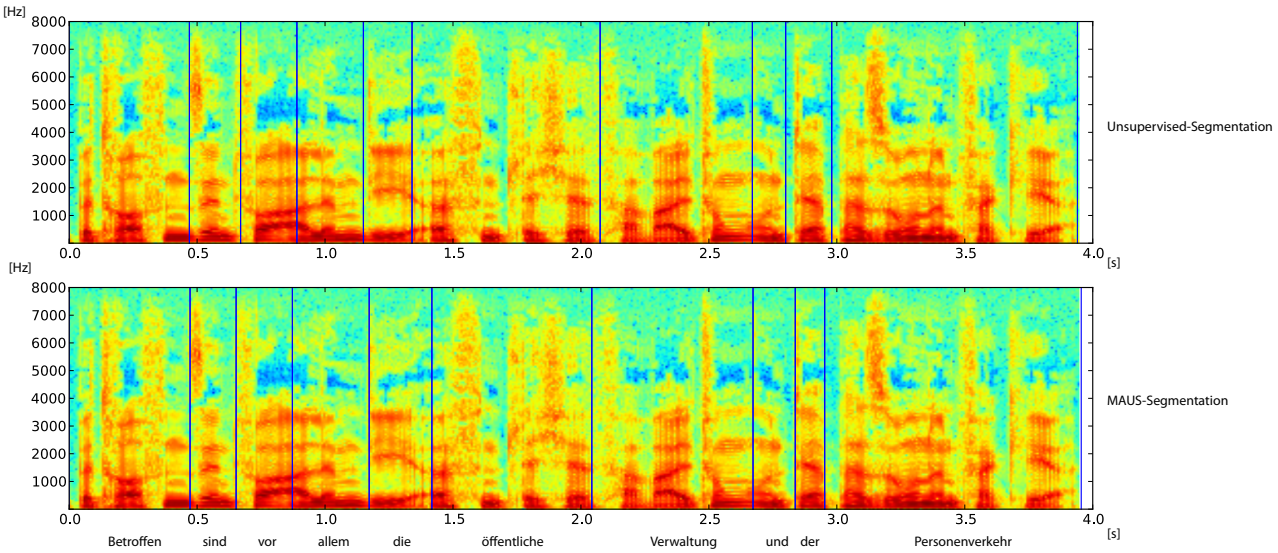

Figure 6.9: Spectrogram and word alignment for the phrase "Betroffen sind vor allem die Verwaltung und der Personenverkehr ". This alignment has a farily typical WD score of 0.185 (average for all German utterances: 0.175). The first row shows the unsupervised segmentation, the second row the reference segmentation from MAUS.

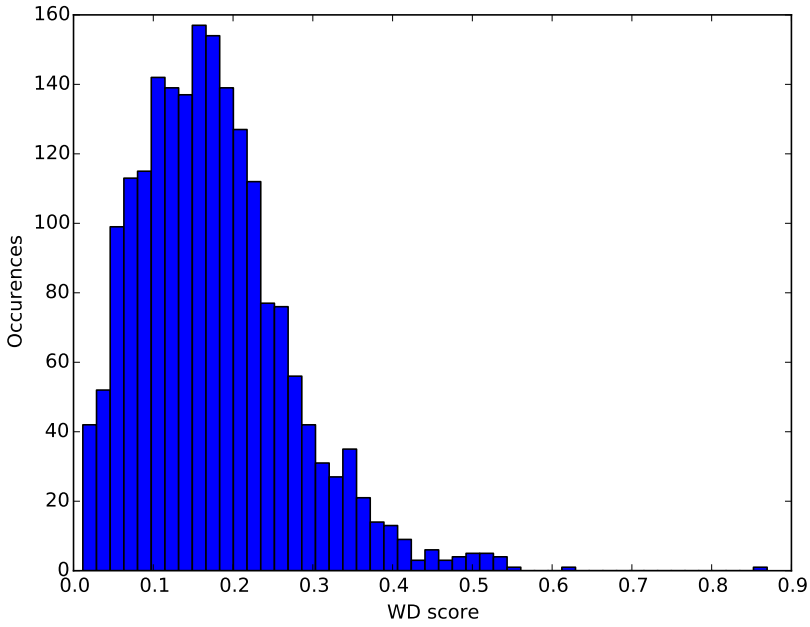

Figure 6.10: Distribution of WindowDiff (WD) scores in the alignment of the German corpus, when the alignment is computed with k=30 units. The mean WD value is 0.175.

Furthermore, Figure 6.10 shows the distribution of WindowDiff (WD) scores in the alignment of the German corpus. It roughly follows a Gaussian distribution around the mean of WD score of 0.175.

6.6.3 Time measurements and scaling

While the corpora that we used for evaluation are only in the range of a few hours, all steps to generate a unit set and to align to text have been build with scalability in mind. The first step, transforming speech into a feature vector representation performs in linear time: each window of the speech signal is processed separately. The runtime for one such window operation is constant for MFCC features, as the runtime for the FFT to generate the spectrum and its subsequent DCT to decorrelate the features is only dependent on the window size. For RPCA features, the RPCA representation must be learned from the data, which needs an additional pass over all windows. The randomized version of the PCA algorithm can be trained and used to transform data in linear time. That means that the feature vector transformation performs also in linear time for RPCA features.

Our sliding window approach to blind segmentation will also take constant time per window, as it is a local algorithm. For clustering, we have choosen K-means(++). While not many meaningful theoretical lower bound exist for K-means, it performs approximately in linear time on most datasets [1, 13]. K-means++ is even faster to some degree than K-means on most datasets [2].

The following timings have been measured on fairly typical mid-range server, with a i3-2130 CPU @ 3.40GHz, 2 physical cores (4 logical due to hyperthreading) and 8GB of RAM. Figure 6.11 shows the time needed to generate units and the time needed to align them. PCA-6 features have been already precomputed and are not included in timing results. It takes 9m26s per hour of speech to generate features, which is much more than the 75 seconds per hour of speech needed to cluster units and align them. Both segmenting and clustering units, and also the EM-alignment, show a clear linear trend.

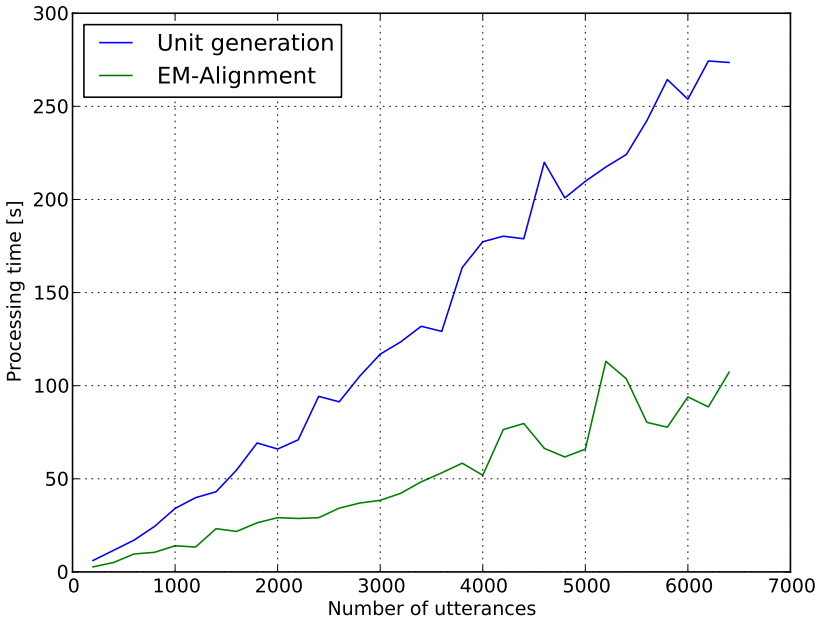

Figure 6.11: Seconds needed to segment speech and cluster units (features precomputed) and time spend in EM-alignment on the Hungarian corpus.

There is more variance with the EM-alignment, which can be explained by variance in different amounts of iterations needed for convergence.

Assuming a continuing linear trend and interpolating from these numbers, a very large corpus, e.g. 1000 hours of transcribed speech, would take 173 hours or roughly one week of processing time on the same machine. Since feature and unit generation are trivially parallelised and constitute to the majority of processing time, this could be further improved with a better machine, particularly one with more cores.

6.6.4 ASR integration

The word alignments from Section 6.6.2 can also be used to generate a unit dictionary automatically. For each word the units that have been aligned to the word have been used as dictionary entry for the word. Note that in a typical corpus words a distributed according to Zipf's Law [45], so that many words are only seen once, especially in a smaller corpus. If such a word is wrongly segmented or aligned, it makes errors in its unit transcription very likely.

Nonetheless, in a submitted, but not yet published paper with the same name as this thesis ("Unsupervised Acquisition of Acoustic Models for Speech-To-Text alignment") we integrated our unsupervised units as a drop-in replacement for standard linguistic phonemes. Dr. Dirk Schnelle-Walka did this experiment, the following results are thus not part of my thesis:

We separated one hour of German speech into 732 utterances for training, with 10% spared out for testing. The speech was segmented into a unit representation with k=30 clusters. We used CMU Sphinx [22], an open source speech recognizer, to replace linguistically motivated phonemes with our unit representation and our automatically generated word to unit dictionary.

Evaluating on our test set, we achieved a word error rate (WER) of 38.2%. By using letters as units, which results in a grapheme based recognizer, a much better recognition result can be obtained with a WER of 7.17%.

Most probably further improvements are needed in unit generation and especially dictionary generation to improve recognition results, since currently a simple grapheme based recognizer shows better performance.

7 Conclusion

In this thesis, we showed that it is possible to find an unsupervised phoneme set solely from speech, i.e. independent of the language. Our algorithm for blind speech segmentation is comparable to other approaches, while having a shorter runtime. Using our segmentation, we can build a phoneme set through clustering and automatically transcribe speech in this unit representation. In our German corpus, we analysed some common unit n-grams by listening manually to them and observed similar sounds and words. Furthermore, we proposed an EM-algorithm with DTW to align our unit representation automatically with text transcriptions, if they are available for the utterances. We evaluated our alignment based on word alignments and compared it to a reference alignments in German, English, Polish and Hungarian. Our evaluation suggests that German and Hungarian speech can be aligned with good quality with our method, while Polish and English showed inferior segmentations. Using more speech data improves alignments considerably. Different parameters, as chosen in the generation of units, have similar effect profiles in all languages that were analysed.

Some of our processing steps still pose limitations. For instance, the segmentation accuracy is currently limited to the sampling rate of speech features. Also, it operates locally and is not context sensitive, which could be beneficial to segmentation quality. The ability to align longer utterances than currently possible with our approach, e.g. a whole audio book, would also be very desirable. An obvious extension is to also use letter n-grams instead of single letters in the EM-alignment, to model language specific pronunciation rules beyond single letters. Using larger corpora with our method, e.g. using 1000+ hours of speech would allow to have a better look at the effect that adding more data has to unit generation and alignment quality. Measurements with our smaller corpora indicate that all our processing steps scale linearly and that processing 1000+ hours of speech should be feasible in reasonable time. We have not yet evaluated our approach on corpora with multiple speakers and it might be necessary to implement speaker-adoption techniques in this case.

Promising are use cases for ASR systems and TTS systems. More work is needed to adapt our unsupervised unit representation to ASR systems, so that they could replace phoneme based acoustic models. Particularly the dictionary generation should be investigated further. If our alignment poses problems due to inaccuracies, this could be investigated by using the reference word segmentation instead of our own unsupervised method. Nonetheless, our first results are very promising and encourage further research.

Our units could also potentially be useful for unit selection synthesis, a popular TTS approach. Similar to ASR, a unit-dictionary would be created from speech with available transcriptions. The database containing units could be computed from the transcribed corpus and additionally a much larger corpus of untranscriped speech recordings, which would vastly improve the amount of available data. Untranscriped speech is much easier to obtain and record than transcribed speech.

To prove feasibility of building a TTS in such a way, it might be possible to use our units representation by modulating someone's voice into a pre-recorded voice of someone else. We would identify units in a new live-recording and would select the best matching units in our database of pre-recorded units to best match the new recording and to minimize distortions. New speech would than be synthesized based on the characteristics of the input recording. Conceptually, this would mean doing ASR and TTS at the same time, but bypassing the transcription level. Not only could it be used to show feasibility of doing TTS with our unit representation, but it could also be used in data gathering. A web-service that could change someone's voice into another voice could attract a wide audience. A lot of speech recordings could be gathered this way, even with transcriptions, by requiring a text passage to be read before usage. A large corpus, even if imperfect and with varying recording situations, could help enormously

to facilitate further research. In this light, incorporating unsupervised learning into speech processing could tremendously be of benefit to the speech community in one way or another.

8 Acknowledgement

I acknowledge gratefully Prof. Chris Biemann and Dr. Dirk Schnelle-Walka for their excellent supervision of my thesis and many good remarks and comments. I particularly enjoyed the freedom they gave me in my research and that I could work on a topic that I proposed myself. Their guidance also helped considerably to shape this work and also led to a not yet published but submitted research paper. I also thank family and friends for their support, particularly Michelle Sandbrink and Niklas Büscher for proofreading and advice.

Bibliography

[1] D. Arthur and S. Vassilvitskii. How Slow is the k -Means Method ? *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 144–153, 2006.

[2] D. Arthur and S. Vassilvitskii. k-means ++ : The Advantages of Careful Seeding. In *ACM-SIAM symposium on Discrete algorithms*, volume 8, pages 1027–1035, 2007.

[3] G. Aversano, A. Esposito, and G. Chollet. A JAVA interface for speech analysis and segmentation. *ITRW on Non-Linear Speech Processing*, pages 4–7, 2003.

[4] M. Bacchiani and M. Ostendorf. Joint Lexcion, Acoustic Unit Inventory and Model Design. *Speech Communication*, pages 99–114, 1999.

[5] P. Cosi, D. Falavigna, and M. Omologo. A preliminary statistical evaluation of manual and automatic segmentation discrepancies. *EUROSPEECH-1991*, pages 693–696, 1991.

[6] S. B. Davis, P. Mermelstein, D. F. Cooper, and P. Nye. Comparison Of Parametric Representations For Monosyllabic Word Recognition In Continuously Spoken Sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 28(4):357–366, 1980.

[7] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal statistical Society*, 39(1):1–38, 1977.

[8] S. Dusan and L. Rabiner. On the relation between maximum spectral transition positions and phone boundaries. In *INTERSPEECH*, pages 645–648, 2006.

[9] Y. Estevan, V. Wan, and O. Scharenborg. Finding Maximum Margin Segments In Speech. *Acoustics, Speech and Signal Processing, 2007*, pages 937–940, 2007.

[10] M. Fék, P. Pesti, and G. Németh. Corpus-based unit selection TTS for Hungarian. *Text, Speech and Dialogue*, pages 367–373, 2006.

[11] J. S. Garofolo. TIMIT Acoustic-Phonetic Continuous Speech Corpus. *Linguistic Data Consortium Philadelphia*, 1993.

[12] J. Glass. Towards unsupervised speech processing. *IEEE Transactions on Audio, Speech and Language Processing*, pages 1–4, July 2012.

[13] S. Har-peled. How Fast is the k-means Method ? *Algorithmica*, pages 185–202, 2010.

[14] T. Hazen. Automatic Alignment and Error Correction of Human Generated Transcripts for Long Speech Recordings. *INTERSPEECH*, pages 1606–1609, 2006.

[15] A. J. Hunt and W. Black. Unit Selection In A Concatenative Speech Synthesis System. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 373–376, 1996.

[16] A. Jansen, K. Church, and H. Hermansky. Towards spoken term discovery at scale with zero resources. *INTERSPEECH*, pages 1676–1679, 2010.

[17] F. Jelinek. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, pages 532–556, 1976.

[18] B. Juang and L. Rabiner. Automatic speech recognition–a brief history of the technology development. Technical report, 2005.

[19] S. Kim, S. Park, and W. Chu. An Index-Based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases. In *Data Engineering*, pages 607–614, 2001.

[20] T. Kisler, F. Schiel, and H. Sloetjes. Signal processing via web services : the use case WebMAUS. *Proceedings Digital Humanities*, pages 30–34, 2012.

[21] C.-Y. Lee, Y. Zhang, and J. Glass. Joint Learning of Phonetic Units and Word Pronunciations for ASR. In *EMNLP*, pages 182–192, 2013.

[22] K.-F. Lee. *Automatic Speech Recognition: The Development of the Sphinx Recognition System*. Springer, 1989.

[23] S. Levinson. An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Autmatic Speech Recognition. *The Bell System Technical Journal*, 1983.

[24] P. Moreno, C. Joerg, J. V. Thong, and O. Glickman. A Recursive Algorithm For The Forced Alignment Of Very Long Audio Segments. In *ICSLP*, page 0068 (paper number), 1998.

[25] A. Muscariello. Towards Robust Word Discovery By Self-Similarity Matrix Comparison. *Acoustics, Speech and Signal Processing (ICASSP)*, pages 5640–5643, 2011.

[26] A. S. Park, J. R. Glass, and S. Member. Unsupervised Pattern Discovery in Speech. *IEEE Transactions on Audio, Speech and Language Processing*, 16(1):186–197, 2008.

[27] L. Pevzner and M. Hearst. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, pages 19–37, 2002.

[28] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, pages 267–296, 1989.

[29] L. Rabiner and B. Juang. *Fundamentals of speech recognition*. 1993.

[30] O. Rasanen, U. Laine, and T. Altosaar. Blind segmentation of speech using non-linear filtering methods. In *Speech Technologies*, page Chapter 5. 2011.

[31] C. A. Ratanamahatana. On Clustering Multimedia Time Series Data Using K-Means and Dynamic Time Warping. *International Conference on Multimedia and Ubiquitous Engineering*, pages 733 – 738, 2007.

[32] S. Renals. HMM Speech Recognition: Search and Decoding. *ASR course, University of Edinburgh*, page 3, 2012.

[33] V. Rokhlin, A. Szlam, and M. Tygert. A randomized algorithm for principal component analysis. *SIAM Journal on Matrix Analysis and Applications*, pages 1100–1124, 2009.

[34] F. Schiel. MAUS Goes Iterative. *EUROSPEECH*, pages 785–788, 2003.

[35] F. Schiel, C. Draxler, and J. Harrington. Phonemic segmentation and labelling using the MAUS technique. *New Tools and Methods for Very-Large-Scale Phonetics Research*, 2011.

[36] J. Schroeter. Basic Principles of Speech Synthesis. In *Springer Handbook of Speech Processing*, pages 413–428. 2008.

[37] A. Stan, O. Watts, and Y. Mamiya. TUNDRA: A Multilingual Corpus of Found Data for TTS Research Created with Light Supervision. *INTERSPEECH*, pages 2331–2335, 2013.

[38] S. Stevens. A scale for the measurement of a psychological magnitude: loudness. *Psychological Review*, 1936.

[39] T.K.Vintsyuk. Speech discrimination by dynamic programming. *Kibernetika*, 4(1):81–88, 1968.

[40] H. Wang, T. Lee, C. Leung, B. Ma, and H. Li. Unsupervised Mining of Acoustic Subword Units with Segment-Level Gaussian Posteriorgrams. *INTERSPEECH*, pages 2297–2301, 2013.

[41] O. Watts, A. Stan, and R. Clark. Unsupervised and lightly-supervised learning for rapid construction of TTS systems in multiple languages from 'found' data: evaluation and analysis. In *Proc. of 8th ISCA Workshop on Speech Synthesis*, pages 101–106, 2013.

[42] J. D. Williams, I. D. Melamed, T. Alonso, B. Hollister, and J. Wilpon. Crowd-sourcing for difficult transcription of speech. *2011 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 535–540, Dec. 2011.

[43] K. Zhang, I. W. Tsang, and J. T. Kwok. Maximum margin clustering made practical. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 20(4):583–96, Apr. 2009.

[44] Y. Zhang and J. R. Glass. Unsupervised spoken keyword spotting via segmental DTW on Gaussian posteriorgrams. *2009 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 398–403, Dec. 2009.

[45] G. Zipf. The psycho-biology of language. 1936.

9 Appendix

In the course of this master thesis, audio files have been generated to inspect units and common unit n-grams for all corpora. These sound files are on the attached DVD. The sound format is OGG.

The basic folder structure is the same for every corpus:

```
<corpus>/iter-figs
<corpus>/units
<corpus>/unit-n-grams
```

`<corpus>/iter-figs` contains correlation costs in pdf and python numpy matrix formats for all iterations of the alignment process, matching the units in the `units` folder.

The file format in `<corpus>/units` is `<cluster number>.ogg`: These unit sound files in the units folder contains all occurrences of a particular unit, with small pauses between them. All occurrences in the corpus are sorted by distance to the cluster center, so that the most representative units are at the beginning of each audio file.

The file format in `<corpus>/unit-n-grams` is

`<n>gram-occs<no of occurences>-<unit1>_<unit2>_...<unit_n>`: Similarly, these audo files contain all occurrences of a particular unit n-gram. For example `4gram-occs69-12_5_2_12.ogg` has all 69 occurrences, separated by a small pause, of the unit 4-gram `12 5 2 12`.