PADERBORN UNIVERSITY
The University for the Information Society

FAKULTÄT FÜR
ELEKTROTECHNIK,
INFORMATIK UND
MATHEMATIK

Discovering structure in speech recordings: Unsupervised learning of word and phoneme like units for automatic speech recognition

Von der Fakultät für Elektrotechnik, Informatik
und Mathematik der Universität Paderborn

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

genehmigte Dissertation

von

Dipl.-Ing. Oliver Walter

Erster Gutachter:	Prof. Dr.-Ing. Reinhold Häb-Umbach
Zweiter Gutachter:	apl. Prof. Dr. Frank Kurth

Tag der mündlichen Prüfung: 01.12.2021

Paderborn 2021

Diss. EIM-E/362

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig und unter Verwendung der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.
Die Arbeit wurde bisher noch keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Teile der Ergebnisse der Arbeit wurden in Konferenzen, Journalen und technsichen Reports veröffentlicht: [Wal+13a; Wal+13b; Wal+14; Wal+15a; WH16; Hey+13; Hey+14; DWH15; Gla+17; DWH18; WSH13; Wal+15b]

Paderborn, den _____ _____

(Datum) (Unterschrift)

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

FAKULTÄT FÜR
ELEKTROTECHNIK,
INFORMATIK UND
MATHEMATIK

Zusammenfassung der Dissertation:

Discovering structure in speech recordings: Unsupervised learning of word and phoneme like units for automatic speech recognition

des Herrn Oliver Walter

Spracherkennung spielt eine immer wichtigere Rolle in unserem täglichen Leben. Sie ist in vielen Geräten und Anwendungen vorhanden, welche wir häufig verwenden. Spracherkennungssysteme müssen für die Sprache, die sie erkennen sollen, trainiert werden. Dieser Trainingsprozess erfordert normalerweise annotierte Trainingsdaten in Form von Sprachaufzeichnungen und dazugehörigen Transkriptionen. Die Menge an Trainingsdaten für Systeme mit kleinem Vokabular reicht von weniger als 10 Stunden bis zu mehren 100 Stunden. Moderne Systeme mit großem Vokabular werden auf mehreren 1000 Stunden und mehr transkribierten Sprachaufzeichnungen trainiert. Während Sprachaufzeichnungen einfach erstellt werden können, kann die Transkription dieser Aufzeichnungen sehr teuer und zeitaufwendig sein. Daher können Methoden zum automatischen Erstellen solcher Transkriptionen für nicht annotierte Daten dabei helfen, das Training von Spracherkennern für Sprachen zu vereinfachen, für die wenige oder keine annotierten Trainingsdaten verfügbar sind.

Diese Arbeit untersucht und stellt Methoden zum automatischen Lernen von Transkriptionen allein aus Audioaufzeichnungen vor. Dabei werden Algorithmen zum Erlernen von Phonemen, den kleinsten Einheiten der Sprache, und Worten, vorgestellt. Diese Methoden können zum automatischen Training eines Spracherkenners aus nicht annotierten Daten verwendet werden. Diese Arbeit untersucht die Methoden zum Erlernen von Phonemen und Worten jeweils separat. Der Hauptfokus dieser Arbeit liegt auf dem unüberwachten Lernen von Worten in hierarchischen Modellen, bestehend aus Phonem- und Worttranskriptionen.

Drei oft verwendete Verfahren werden untersucht, zum einen heuristische Methoden und zum anderen zwei Varianten statistischer modellbasierter Verfahren. Die erste Variante basiert auf einem probabilistischen Aussprachelexikon, während das zweite Verfahren auf der Segmentierung von Wortgittern beruht. Schließlich wird ein vollständig unüberwachtes System aus einer Kombination von unüberwachtem Phonemlernen und unüberwachter Wortsegmentierung präsentiert.

Diese Arbeit schließt mit der Integration des unüberwachten Phonem- und unüberwachten Wortlernens in eine semantische Inferenz ab, um die Verwendbarkeit von unüberwacht gelernten Phonemen und Worten in einem übergeordneten System sowie ihre Fähigkeit, die Erkennungsergebnisse zu verbessern, zu demonstrieren.

Diese Arbeit zeigt, dass das unüberwachte Lernen von Phonemen und Worten im Prinzip möglich ist, wenn auch mit einer niedrigeren Qualität als überwacht trainierte Modelle.

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

FAKULTÄT FÜR
ELEKTROTECHNIK,
INFORMATIK UND
MATHEMATIK

Abstract of the thesis:

Discovering structure in speech recordings: Unsupervised learning of word and phoneme like units for automatic speech recognition

by Mr. Oliver Walter

Speech recognition plays an increasing role in our daily life. It is present in many devices and applications we frequently use. Speech recognition systems need to be trained for the speech and language they are supposed to recognize. This training process usually requires labeled training data in the form of speech recordings and corresponding transcriptions. The amount of required training data for systems with small vocabulary ranges from less than 10 hours to several 100 hours. Modern Large Vocabulary Continuous Speech Recognition systems are usually trained on several 1000 and more hours of transcribed speech recordings. While speech recordings are easy to obtain, the transcription of those recordings can be very costly and time-consuming. Therefore, automatic methods to derive such transcriptions from unlabeled data can help simplifying the training of speech recognizers in languages where little to no labeled training data is available.

This thesis investigates and introduces methods to automatically learn transcriptions from audio recordings only. Algorithms for the unsupervised learning of phonemes, the smallest units in speech, and words are presented. These methods can then be used for the automatic training of a speech recognizer from unlabeled data. This thesis investigates these unsupervised learning methods separately for the learning of phonemes and words. The main focus of this thesis is laid on the unsupervised learning of words in hierarchical models consisting of phoneme and word transcriptions.

Three main approaches are investigated. Firstly, heuristic methods. Secondly, two variants of statistical model-based approaches. The first variant is based on a probabilistic pronunciation lexicon while the second approach is based on word segmentation over lattices, instead of a single best sequence. Finally, a fully unsupervised system with unsupervised phoneme discovery and unsupervised word segmentation combined, is presented. The thesis concludes by integrating the unsupervised phoneme and word discovery into a semantic inference task in the setting of a simple command and control interface to demonstrate the usability of unsupervised learned phonemes and words in upstream tasks and their ability to improve their performance over purely supervised methods.

This thesis shows that unsupervised learning of phonemes and words is in principle possible, although still with lower quality than supervised models.

Acknowlegdement

I would like to express my gratitude to my supervisor Prof. Reinhold Häb-Umbach for his valuable guidance in this work and constant encouragement. Working with him at the department of communications engineering of the university Paderborn has been a pleasant, enriching and memorable experience.

I would also like to thank my colleagues and friends at the department of communications engineering. In particular Sven Bevermeier, Aleksej Chiaev, Volker Leutnant, Tran Vu Dang Hai, Florian Jacob, Jahn Heymann, Lukas Drude, Jörg Schmalenströer, Thomas Glarner, Jörg Ulmann, Peter Schütte, Anita Hüser Thank you for the numerous discussions on and around my work and the good times in and around office.

Thank you to the Deutsche Forschungsgemeinschaft for partly funding this research work under the contract number Ha 3455/9-1 within the Priority Program SPP1527 "Autonomous Learning". Thank you for the opportunity to visit the Carnegie Mellon University and the support by Prof. Bhiksha Raj, Rita Shingh and Sourish Chaudhuri. Thank you to Dr. Joachim Köhler and Dr. Christoph Schmidt at the Fraunhofer IAIS for their trust in and support of me. Thank you to Michael Gref and all my new colleagues at the Fraunhofer IAIS. Thank you to all the students, colleagues and friends not mentioned in here.

A great thanks goes to my parents and family for their ongoing support and encouragement.

Thank you all.

Contents

1 Introduction

Speech recognition plays an increasing roll in our daily life. It is present in many items and applications we frequently use, for example in smartphones, home appliances, cars and telephone hotlines. A reason for this increasing use is the fact that speech is the most natural form of interaction among humans and it is therefore also used for many communication tasks between humans and machines. Speech recognition can also be used in dictation systems or to transcribe and archive recordings of speech in multimedia archives like the "ARD Mediathek" and therefore make them searchable based on the spoken words in the recordings. Personal assistant devices usually use speech recognition in a first step to convert spoken commands into text and to interpret them in a next step by analyzing this text. Prominent examples of personal assistants and home automation systems are Amazon's Alexa, Google's Assistant, Apple's Siri and Microsoft's Cortana, to name a few.

Speech recognition systems need to be trained for the speech and language they should recognize. This training process usually requires labeled training data in the form of speech recordings and corresponding transcriptions. The amount of training data for typical small systems ranges from less than 10 hours to several 100 hours. Modern Large Vocabulary Continuous Speech Recognition (LVCSR) systems can be trained on several 1000 and more hours of transcribed speech recordings. While speech recordings are easy to obtain, the transcription of those recordings can be very costly and time consuming. Therefore, automatic methods to derive such transcriptions from unlabeled data can help simplifying the training of speech recognizers in languages where little to no labeled training data is available.

This thesis will investigate and introduce methods to automatically learn transcriptions from audio recordings only. Algorithms for the unsupervised learning of phonemes, the smallest units in speech, and words will be presented. These methods can then be used for the automatic training of a speech recognizer from unlabeled data. This thesis will investigate these unsupervised learning methods separately for the learning of phonemes and words. First for heuristic methods and finally for statistical model based approaches are investigated. Afterwards a fully unsupervised system with unsupervised phoneme discovery and unsupervised word segmentation will be presented. The thesis concludes by integrating the unsupervised phoneme and word discovery into a semantic inference task in the setting of a simple command and control interface.

2 State of the Art

This work deals with the topic of unsupervised learning for speech recognition. In particular, it deals with two main areas of speech recognition, acoustic model learning and word modeling. In the first area this work deals with unsupervised representation learning for the acoustics of a speech recording. Representations are learned in an unsupervised way on the phonetic and word level. In the second area this work is concerned with the unsupervised learning of word models. Specifically the unsupervised learning of words as sequences of phonemes from unsegmented phoneme sequences, called unsupervised word segmentation, is considered. The phoneme sequences for this second area can be delivered by an already trained speech recognizer or by an unsupervised learning algorithm for acoustic representation learning. On both levels, but especially on the word level, this work additionally includes the unsupervised learning of language models which model the probabilities of phoneme or word sequences. Heuristic approaches as well as model based approaches are considered in this work. The basis of the model based approaches presented in this work lies in classical speech recognition models. The following describes the state of the art for automatic speech recognition and unsupervised learning.

2.1 Automatic speech recognition

The training of automatic speech recognition systems on unlabeled data delivers the basic motivation for this work. Therefore automatic speech recognition is the basis for the model based approaches to unsupervised learning for automatic speech recognition. The model based algorithms for unsupervised learning in this work are mainly based on the traditional Gaussian Mixture Model based Hidden Markov Models (GMM-HMMs) as well as n-gram language models. Aside from the GMM-HMM models, Deep learning based models are applied for speech recognition purposes only. Beside model based approaches, heuristic approaches, as introduces later, are also used for the purpose of unsupervised learning and speech recognition.

Algorithms for continuous speech recognition using Dynamic Time Warping approaches and Hidden Markov Models have been developed a long time ago. Early descriptions, even though not the first mentionings, can be found in [Jel76] with the description of statistical models for Hidden Markov Model based continuous speech recognition. Further [MRR80] investigates Dynamic Time warping based approaches to speech recognition. With [Rab89] (Rabiner HMM) being one of the most popular publications on Hidden Markov Model

based speech recognition. All these mentioned publications have in common that either labeled examples of speech recordings have to be used for comparison (Dynamic Time Warping) or used for the training of parameters for statistical models (Hidden Markov Models). This work applies existing methods for the unsupervised learning of model parameters for the aforementioned models, adapts them to the requirements of speech recognition and also introduces new methods. In most cases, traditional GMM-HMM based automatic speech recognition systems still use the same structures as mentioned above. Many further developments were done to improve quality or increase efficiency in implementation. Developments to improve quality are model and feature transform as described in [Gal98] and [Gop98] Maximum likelihood linear transform (MLLT adaption) and [Ana+96] and [Gal99] feature-space maximum likelihood linear regression (fMLLR adaptation). These methods also mostly rely on labeled training data as well. In this work, Speaker adaptive training and feature transformation are applied to improve the robustness of unsupervised representation learning in an unsupervised way. More specific implementations to optimize speech recognition are described in [All+07] by using finite state transducer based algorithms (OpenFST) for sequence representation and decoding. This is specifically realized in [Pov+11] within the KALDI speech recognition toolkit. Beside the classical GMM-HMM based approach a Hybrid modeling approach with neural network based acoustic models as described in [Zha+14], specifically p-norm ASR with KALDI are used for speech recognition only. Other approaches to speech recognition have been developed, for example in [Van08] an approach based on a Histogram of Acoustic Co-occurrences (HACs) based representations for variable length segments. This specific approach is used later in this work for representations in a semantic inference task. Neural network approaches were not considered in the unsupervised learning setting for this work due to their requirements with respect to the amounts of training data and also vast computational complexity. In the last chapter of this work, a small overview of more current methods is given which came up toward the end of this work or after finalizing most experiments.

2.2 Unsupervised learning for speech

Unsupervised learning for speech has been of interest for a while due to the vast amount of unlabeled audio data available in “high resource” languages and especially also for languages with no labeled speech, called “Zero resource speech recognition”. For the former, labeled training data is required for the training of speech recognition algorithms while for the latter such labels are not always available. Therefore unsupervised learning for speech recognition can help on both sides. Firstly unlabeled training data can be leveraged and secondly labels can be automatically derived for languages where no such labeling is available as might be the case for rarely spoken languages.

Some examples for audio data with only limited amount of labeled data used in this work are the Waimaa language data [Bel+06] and the Wooi corpus [Kir+15]. Both languages belong to the Austronesian language family. The corpora contain recordings of conversations in every day situations including background noises. The corpora were

recorded for linguistic purpose to analyze and preserve the languages.

Additionally, audio data was taken from the Zero resource speech challenge 2015 [Ver+15]. The Zero speech challenge 2015 especially tackles the problem of unsupervised learning and the comparison of algorithms for unsupervised learning by providing a specified dataset and baselines to compare the algorithms to. The challenge provides the Buckeye Corpus [Pit+07] containing recordings of interview conversations in English. The Challenge additionally provides the Xitsonga corpus [De +14], with speech recordings in the Xitsonga language, a south African language. The corpus was collected via Smartphone.

The results of the Zero resource speech challenge were published in [Dup16].

In contrast to the aforementioned corpora traditional transcribed corpora are for example the Wall Street Journal corpora [Fra+94] (WSJCAM0 Transcribed corpus) and [PB92] (WSJ Corpus). Both corpora were used in this work as simulated zero resource corpora to benchmark the unsupervised learning algorithms on languages with transcribed resources.

As mentioned previously two areas of unsupervised learning have been addressed in particular. Firstly the unsupervised learning from audio recordings, and secondly the unsupervised learning of representations from text, in particular the learning of words. Especially two approaches have been used, Heuristic approaches and model based approaches as introduced in the following.

2.3 Heuristic approaches to unsupervised learning

Heuristic approaches to unsupervised learning for speech recognition look at the problem of unsupervised learning and pattern discovery from a rather intuitive perspective, leaving aside statistical models. Heuristic approaches often directly work on the speech signals, e.g. by comparing segments of speech signals to one another.

One such approach is the Unbounded Dynamic time warping approach presented in [AMO10] (UDTW). This approach is based on the previously mentioned dynamic time warping approach for speech recognition. Additionally to clustering similar speech segments, several algorithms are applied. Namely the k-means++ algorithm [AV07] and similar approaches. In [SBH11] the k-means++ and k-means algorithm were used for acoustic event clustering.

In [PG08] the dynamic time warping approach was used for unsupervised pattern discovery. Here a spectral clustering method by Newman [NG04] is applied, which performs unsupervised pattern discovery by clustering a similarity graph derived by first apply a segmental dynamic time warping approach similar to UDTW. Further, unsupervised pattern discovery on posteriorgrams was performed, using segmental dynamic time warping based algorithms [ZG09]. Additional spectral clustering methods were evaluated in [LSJ15] during the zero resource speech challenge 2015.

2.4 Model based approaches to unsupervised learning

In contrast to the previously described heuristic approaches, model based approaches mainly make use of statistical models. With model based approaches, the parameters of these statistical models are learned, giving them more flexibility to adapt to the data and for example better consider statistical variations in the data. Usually two approaches are used: Generative models describe the process that generates observed data. Discriminative models on the other hand only focus on what discriminates the different classes, for example words, the data belongs to.

This work will mainly make use of generative models to express observed data and in this way learn models which represent relevant parts of the data in an unsupervised way. Several methods for model optimization exists, namely Expectation-Maximization (EM) [DLR77], Variational Bayes, Maximization likelihood and others. In this work extensive use is made of sampling methods for model learning. Namely Gibbs Sampling [CG92] based which in turn is a variant of Metropolis Monte Carlo Sampling [Met+53].

Another model based approach, also used in unsupervised learning from speech, is based on Non-Negative matrix factorization. In [SH12] Non-Negative matrix factorization is used for pattern discovery for ASR Vocabulary acquisition and in [GV13] NMF-Based Keyword Learning from scarce data. Both approaches rely on some grounding labels, however. Although those can be more coarse: For example grounding labels for a whole expression or word, with the goal to learn smaller sub units in form of a dictionary matrix in NMF.

NMF based approaches are applied in [Gem+12] for a self learning Voice User Interface (VUI), in [Gem+13] for the ALADDIN Project and in [Loo+12] for supervised grammar induction aiming for unsupervised learning. The NMF based methods and the VUI will be applied later in this work for semantic inference tasks. In preliminary studies for this work, NMF was also studied for acoustic model learning and word discover but not found to perform with a reasonable performance and therefore not followed up further.

2.4.1 Unsupervised acoustic modeling

Unsupervised acoustic modeling is been used to learn representations of the acoustic observations, for example in terms of acoustic units, similar to phoneme models.

Several statistic models, mainly Hidden Markov or Gaussian Mixture Model based models have been used for unsupervised acoustic model discovery, mainly based on the assumption that speech can be represented by such models as it is done in tradition speech recognition systems.

In [CHR11] so called Acoustic Unit Descriptors (AUDs) are learned as traditional HMMs in a completely unsupervised manner in an iterative way for acoustic event detection. This work will use a similar algorithm for acoustic model learning for unsupervised word discovery. In [LG12] a non-parametric approach to unsupervised model discovery based on a Dirichlet process prior is taken to allow a potentially infinite number of classes. Although in here again some grounding in the from of a word level transcription was assumed. In [Siu+13] so called HMM based self organized units were used for topic

classification and keyword discovery. These are similar to the AUDs mentioned above. In [JH13] it was shown that weak top down constraint for unsupervised acoustic modeling can improve the model discovery performance. Later in this work a similar approach is taken to improve the word discovery performance by passing information from one word discovery step (DTW) to another (model based). In [HSN16], supervised acoustic modeling was used to improve Dirichlet Process Gaussian Mixture Model Clustering (DPGMM Clustering), where the supervision information, namely acoustic labels, are taken from a first unsupervised learning step. A similar approach will be used in this work to improve the Bayesian phoneme language model supported acoustic unit discovery described in [Ond+17]. For the evaluation of unsupervised models, special evaluation algorithms are presented. This work will use the approach in [Car+11], called the same difference task.

Unsupervised acoustic modeling is of broad interest in several fields. One obvious use is the acoustic model discovery in case of unlabeled or underresourced data. Beside of this it is also of interest for acoustic modeling in case speech disorders, since the acoustic representations are different to standard models. Several publications deal with the challenges of ASR in those cases, for example for dysarthric speech. Later in this work, unsupervised acoustic modeling will be used for dysarthric speech recognition in voice user interfaces. In the following works, the influence of dysarthric speech on performance is investigated and mainly supervised methods have been used for adaptation: [Chr+12], [CT13], [Gre+03], [Has+06], [NF92] [Rud11], [RY00], [San+02], [SH10], [SPK12].

Speech recognition is for example used in voice controlled user interfaces: [Haw+07], [Haw+13], [Ons+13]

2.4.2 Unsupervised word modeling

In addition to unsupervised acoustic modeling, the unsupervised word modeling is mainly concerned with the unsupervised discovery of words as sequences of acoustic units. Most previous works perform word segmentation on character or phoneme sequences as input. In [Nag96] a count based language model combined with a so called generalized Forward-Backward search was used for the extraction of new words from Japanese texts. One challenge of unsupervised word discovery is to cope with an unknown number of words. To deal with a possible unknown number of words, Dirichlet Process (DP) based language models are used. In [Teh06b] the hierarchical Pitman-Yor model (HYPLM) was proposed, based on the Pitman-Yor process. The model represents a hierarchical n-gram structure over possibly infinitely many words, and is therefore able to deal with unknown words. In [Teh06a] it was show that the HPYLM is a generalized form of an interpolated Kneser-Ney Language model. One weakness of the HPYLM is the missing ability to calculate different probabilities for different unknown words. This was solved in [MYU09] by nesting two HPYLMs within each other, forming a nested HPYLM (NHPYLM) and applying it to the problem of word segmentation. The outer language model calculates word probabilities and the inner language model calculate probabilities of words based on character sequence probabilities. All of these approaches have in common that they work on character sequences. In [Neu+12] this was extended to lattice input to learn a language model in

an unsupervised way. The main focus of this work was on the language modeling, rather than the word discovery. Therefore it made some approximations to aid the language model learning. In this work a similar approach without the approximation and with other extensions to exactly calculate word probabilities is used for word segmentation. In [CR12] a slightly different approach compared to the previous ones was taken by modeling words as sequence of characters instead with an n-gram language model, with an HMM for length modeling with bag of characters emission distributions. This was applied to acoustic event discovery. In this work the approach will be extended to unsupervised word discovery and using state dependent emission distributions instead. Additionally this was combined with acoustic unit learning to form a fully unsupervised hierarchical modeling. A similar approach will be taken in this work by combining the NHPYLM based word segmentation with acoustic unit discovery. A rather simple hierarchical segmentation approach was proposed in [RDF15] by combining a syllable segmentation on the acoustic level with word segmentation by longest sequence/most frequent count. In [Kam17] several approaches for unsupervised word discovery were investigated. Most notably the NHPYLM based word segmentation was combined with GMM based clustering of sequence embeddings.

2.5 Further approaches

The area of unsupervised learning is in constant change. Especially the huge and rapid progress in deep learning based algorithms for speech recognition after beginning this work, has sparked interest in unsupervised learning for speech recognition with deep learning based models. Some approaches were proposed and evaluated in the Zerospeech challenge [Dup16] but could be not feasibly combined into a complete system for word discovery, partly also because of their huge demand in computing power. Also most approaches usually focused only on a single aspect, for example the acoustic model learning. Even though a couple of algorithms have been briefly investigated during this work, especially Bootleneck features and Autoencoder structures [Gre+07] and word-to-vec [Mik+13], their relative novelty, lack of sufficient data, high requirements in computing resources and therefore little promising results within reasonable time resulted in them not being considered in depth for this work. After many of the experiments in this work were done, new deep learning based unsupervised learning algorithms came up and have evolved. An overview of these algorithms and further developments in unsupervised learning with deep learning based methods will be given at the end of this work in Chapter 9.

3 Contribution and Organization

In this work, models and algorithms to automatically discover the structure of speech and language from untranscribed speech recordings are introduced, extended and investigated. A major focus is laid on the hierarchical modeling of speech, first discovering phonetic units as the smallest units in speech and then words as sequences of phonetic units. The unsupervised learning will be related to established supervised modeling approaches by treating the unsupervised learning as a classical learning problem with hidden variables. Roughly speaking, the transcriptions usually available for supervised learning will be considered as a hidden variable and to be discovered by the unsupervised learning algorithms and modeling approaches.

A first focus is therefore laid on the unsupervised learning of a phonetic inventory and transcription in terms of the phonetic inventory for given speech recordings, in analogy to phonetic transcriptions determined by phonologists and linguists. The automatically discovered units of this inventory will be called Acoustic Units (AUs). The second and major focus of this work is laid on the unsupervised learning of larger meaningful units, similar to words, usually represented as sequences of AUs, from the sequences of discovered or recognized AUs. These meaningful units will be called words in this work, since they are analogous to words as they are usually used in speech. Next, the learning of a phonetic transcription and the learning of bigger meaningful units are combined in a sequential learning algorithm to automatically discover a hierarchical representation of the speech recordings. Finally the learned representations are used in a semantic inference task to demonstrate its usability for natural language understanding tasks.

First a heuristic approach solely based on comparing segments to each other and then further clustering similar segments into words is presented. Then model based approaches, where assumptions about the composition of AUs in terms of acoustic feature vectors, the words in terms of AUs and the relationship among AUs and words themselves as well as between each other are captured in terms of statistical models, are presented. The problem of unsupervised learning is then formulated in terms of discovering latent variables, where the transcription of the speech recordings in terms of AUs and words and their numbers as well as the model parameters of the aforementioned models are the latent variables.

This work is structured into four chapters, with each chapter describing different approaches. Starting with a heuristic approach in Chapter 4, continuing with statistical model based approaches in Chapter 5 and Chapter 6, and finally concluding with a semantic inference task employing automatically learned acoustic units in Chapter 7.

Chapter 4 introduces and describes the principles of unsupervised language acquisition from speech recordings and sets them into the context of language acquisition and speech recognition in general. To give an intuitive introduction into the discovery of clusters of similar repeating patterns in speech recordings, Section 4.2 first reviews a heuristic Dynamic Time Warping (DTW) based approach, where each possible speech segment is compared to each other possible speech segment and then clustered into similar segments using a graph clustering algorithm. An initialization method based on local minima search is introduced to better select potential correct segment pairs to be compared and to reduce computational cost by avoiding potential false pairs. In Section 4.2.4 the results of the heuristic dynamic time warping based approach on a small vocabulary task and unsupervised speech recognizer training by employing the discovered words are presented.

In Chapter 5, unsupervised language acquisition is formulated in terms of discovering hidden variables in the framework of traditional ASR algorithms and models. Therefore, Section 5.3 gives an overview of the traditional ASR models, where phonemes are modeled as sequences of acoustic feature vectors, employing a Gaussian Mixture Model (GMM) HMM and words are modeled as sequences of phonemes, employing a deterministic or probabilistic pronunciation lexicon. Additionally, n-gram language models capture the probabilities of word and phoneme sequences.

Therefore in Section 5.4 unsupervised learning in the context of traditional GMM-HMM ASR algorithms and models in introduced. The section is divided in subsections for acoustic unit learning and probabilistic pronunciation lexicon learning. In Section 5.4.1 the transcription of the speech recordings and the inventory of AUs, together with the corresponding model parameters, are considered as hidden variables. An approach, consisting of the three steps: segmentation of the speech recordings, clustering of the segments and HMM-GMM learning for the clusters, is presented and applied for the unsupervised learning of AUs. In Section 5.4.2 a probabilistic pronunciation lexicon, mapping phoneme sequences to words, and the transcription of the speech recordings in terms of words are considered as hidden variables. Each word is modeled as an HMM with multinomial emission distributions over AUs. Originating from a Bag-of-words approach with position independent emission distributions used for acoustic event discovery, in this work, the model is extended by position dependent emission distributions to capture the sequential nature of words. In Section 5.4.4, the discovered AUs are evaluated in a same-different task to determine their ability to discriminate between different words and group similar words. Next, the discovered words from the discovered AUs, using the probabilistic pronunciation lexicon, are evaluated on a small vocabulary task. Eventually the chapter again concludes with an unsupervised speech recognizer training.

In Chapter 6, the NHPYLM is presented, a language model suitable for learning words from unsegmented character sequences. The NHPYLMs represents a set of n-gram models, being non-parametric by assuming an infinite number of possible words, similar to an infinite number of classes in a Dirichlet process. The NHPYLM jointly models sequence probabilities of characters and words and couples them together via a deterministic lexicon, where the probabilities of unknown words are calculated as the sequence probabilities of their consisting characters. This modeling of unknown words makes it particularly

useful for unsupervised language acquisition. On the character and the word level, probabilities are modeled as n-gram probabilities, each by an HPYLM together resembling the NHPYLM. In this work, instead of characters, phonemes are used as input. The phoneme sequence probabilities and the lexicon as well as the transcription of the speech recordings in terms of words and the word sequence probabilities are considered as latent variables and learned using a Gibbs sampling based learning algorithm. In Section 6.3, an extended learning algorithm over multiple hypotheses of phoneme sequences for a speech recording in terms of lattices as input is presented. A phoneme lattice is a probabilistic representation of a speech recording, in contrast to a single best sequence, and therefore captures more information of the speech recording. In this work, the learning algorithm is extended to sequential processing of input lattices, first extracting a best sequence employing a phoneme HPYLM and then the NHPYLM on the best sequence. Finally the unsupervised word learning algorithm is applied to automatically discovered AU lattices to achieve a fully unsupervised learning of a hierarchical representation of the speech recordings.

The word segmentation algorithm is realized using WFSTs, since they are also used in traditional supervised speech recognition approaches, and efficient algorithms for sequence transformation and decoding exist. In Section 6.4 the implementation of the aforementioned NHPYLM based word learning algorithm and models is presented. Parts of the learning and inference steps are implemented as WFSTs. Following traditional ASR algorithms and implementations, Section 6.4.3 describes the implementation of the lexicon as a WFST, transducing from input character sequences to segmented output characters and word sequences. In addition to a traditional lexicon, modeling a list of known words, the lexicon presented here jointly models known words and additionally performs a segmentation of the input sequence into all possible words, effectively modeling out-of-vocabulary words. Section 6.4.3 presents the WFST based representation of the NHPYLM, weighing the character and word sequences according to the NHPYLM. Section 6.4.3 concludes with a forward filtering, backward sampling algorithm on the composition result of input, lexicon and language model WFST graph, used in the Gibbs sampling based learning algorithm.

In Section 6.6 several setups for the unsupervised word segmentation using the NHPYLM are presented and evaluated. Section 6.6.1 starts with word segmentation on error free text to demonstrate the feasibility of the unsupervised word segmentation algorithm for unsupervised language acquisition. Section 6.6.2 presents results on lattices from pre-trained speech recognizers using a monophone GMM-HMM recognizer on the same language to simulate AU discovery results with high unit error rate. In Section 6.6.3 a triphone Deep Neural Network (DNN)-HMM recognizer is used to simulate a mismatched model training, in this case on another dialect and language, resulting in a high phoneme error rate, moving towards a cross language approach where knowledge from one language is transfered to another similar one. Section 6.5 concludes with a complete system, first learning AUs in an unsupervised way and then processing the resulting AU lattices by the word learning algorithm. This results in a completely unsupervised hierarchical AU and word discovery from speech recordings.

Chapter 7 presents an algorithm for semantic inference on automatically learned

acoustic units. This task is especially interesting because it demonstrates the utility of automatically learned acoustic units, especially in the case of distorted speech where standard speech recognition systems and usual models would be of little use. The semantic inference algorithm performs intent recognition and slot filling using automatically discovered acoustic units as input, while the semantic inference is trained in a supervised way. Section 7.5 describes the experiments and results for a semantic inference task on automatically learned acoustic units.

The thesis eventually concludes with Chapter 8, where the new contributions to each setup and experimental results for each chapter are summarized.

4 Unsupervised Speech and Language Acquisition: A Heuristic Approach

Unsupervised speech and language acquisition means the automatic learning of structure of speech and language from unlabeled and in general unsegmented speech. In this work, the speech is given as audio recordings. Speech acquisition focuses on speech perception and speech production, whereas language acquisition focuses on grammatical and syntactical learning. This chapter introduces the ideas to unsupervised learning and presents an intuitive algorithm that aims to learn speech and language similarly to a child, without prior knowledge and with only few assumptions about the structure of speech and language. The following chapters will present more sophisticated statistical model based approaches.

In this work an algorithm, e.g. a speech recognizer supported by unsupervised learning algorithms, takes the role of the child learning a language. Instead of transcribed speech recordings which are generally used for the supervised training of a speech recognizer, unsupervised training procedures are used, where the transcription of the speech recordings is automatically learned during the training process. This can be helpful in domain specific tasks for which no speech recognition system exists yet, from training to recognize a small vocabulary in a command and control task with uncommon speech and language to training an LVCSR system for languages that only have little to no transcribed data available or do not even have a written form. Those setups are usually called low resource or zero resource setups. The latter is especially interesting, since the presented algorithms can assist in building speech recognizers for those languages or help to analyze speech and language that is rarely spoken or being on the edge of vanishing or which has already disappeared and is only present in terms of speech recordings.

In contrast to a child though, which is usually exposed to multi-modal input, for example visual input or the input of a parent or teacher teaching the child, the presented algorithms are constrained to speech recordings only. The reason for this constraint lies in the fact that it is easier to record speech only, for example by a small recording device, while recording synchronized speech, video and other modalities is usually more difficult. Most available speech databases therefore contain only speech recordings. The speech recordings are generally unsegmented in terms of words, although often segmented in terms of sentences or utterance fragments, also called intonation units,

with the segment borders often defined by shorter or longer silences. Therefore the learning algorithms have to be capable to automatically segment and transcribe the speech recordings simultaneously. In the following sections and chapters, the idea of unsupervised language acquisition and the computational models used to solve the task are presented.

4.1 An Example for and Approaches to Unsupervised Learning

To get an intuition for the problem one might first consider the following example: Imagine you are in a situation where you do not know the language and the only thing you can do is hearing. You listen to someone speaking the sounds "ekdoteendochaardoek...". Listening to the sequence of sounds you will probably first notice that the sound "do" appears quite often in several distinct contexts. If so, congratulations, this is the first word you discovered in the unknown language. Listening more closely, you will probably also notice, that the sound "ek" appeared in the beginning and the end of what you heard. This is a second word you discovered. Finally you might notice the remaining sounds "teen" and "chaar" in between the previously discovered words and indeed these are two more words that you just learned from the unknown language. If you were to "ask" someone about the meaning of those sounds, he or she will probably happily explain to you in gestures or visual examples their meaning, which in this case are the numbers 1 (ek), 2 (do), 3 (teen) and 4 (chaar) in the Hindi language. You might have used one of two ways to discover the reoccurring sounds, which will be presented in the following sections.

What the above example is meant to show: search for repeated patterns. This first intuitive approach, based on searching for repeated patterns by comparing all possible segments of each speech recording with all other possible segments of each speech recording and clustering similar segments, is presented in this chapter in Section 4.2. In the following chapter, statistical models used for speech recognition are introduced and in Section 5.4, a second iterative statistical model based approach, based on these statistical ASR models and algorithms, is presented. In chapter Chapter 6 another alternative approach to unsupervised language model learning for word segmentation is presented, including a description of the explicit implementation based on WFSTs.

4.2 Heuristic Dynamic Time Warping-based Approach

A first intuitive approach to unsupervised learning is based on the dynamic time warping algorithm and will be called "Heuristic dynamic time warping-based approach". This approach will be explained in this section. The approach is solely based on the direct comparison of audio segments with each other and does not rely on learning any parameters of a model other than the alignment of similar audio segments. The dynamic

time warping algorithm only makes the assumption that similar sequences have a similar spectral representation and a similar temporal structure.

4.2.1 An Intuitive Example: Language Acquisition

Intuitively the approach works as follows: You take what you heard and compare a small segment of it with other segments of what you heard. You then try to find a similar sounding small segment in what you heard. If you have found such a pair of segments, you try to extend this pair to longer segments by increasing the size of both the segments until the pair becomes too dissimilar. This pair of segments of audio is then memorized. You repeat this step for all other remaining small segments of what you heard and memorize all other found pairs as well. Finally you compare all discovered pairs with each other and group similar pairs together. This approach resembles a rather intuitive way for a human to discover words in an unknown language. Due to its simplicity and assumptions but practicality it will be called heuristic.

As an example, Figure 4.1 shows two utterances in the time domain and the spectral domain. For visualization, the spoken words are labeled above the figures. Usually this labeling would not be present. The digits “three” (3), “two” (2), “zero” (z), “seven” (7) and “six” (6) are spoken in the first sequence. In the second sequence the digits “three” (3), “oh” (o), “two” (2), “seven” (7), “four” (4), “six” (6) and “nine” (9) are spoken. It can be seen that the digits 3, 2, 7 and 6 are appearing in both utterances. If the labels were not given, one has to compare the time or spectral domain representations of both utterances to find similar sequences. Especially in the spectral domain, similar temporal structure can be recognized for matching digits more easily. But additionally, it can also be seen, that the representations have different lengths due to different rates of speaking.

4.2.2 Dynamic Time Warping: Algorithm

Algorithmically the dynamic time warping-based approach works as follows:

- Calculate the distance of each sample to each other sample, with an appropriate definition of distance, resulting in a distance matrix.
- Find similar subsequences by identifying paths along low distances in the distance matrix.

A distance matrix for the sequences of Figure 4.1 with discovered similar sequences is shown in Figure 4.2. In this example the digits 3, 2 and 7 are discovered by the dynamic time warping-based approach, while the digit 6 was not discovered due to a higher distance between both sequences. To compare and match these subsequences of different lengths a modified version of the Dynamic Time Warping algorithm is used, called Unbounded Dynamic Time Warping (UDTW). The UDTW algorithm will be introduced in the following sections.

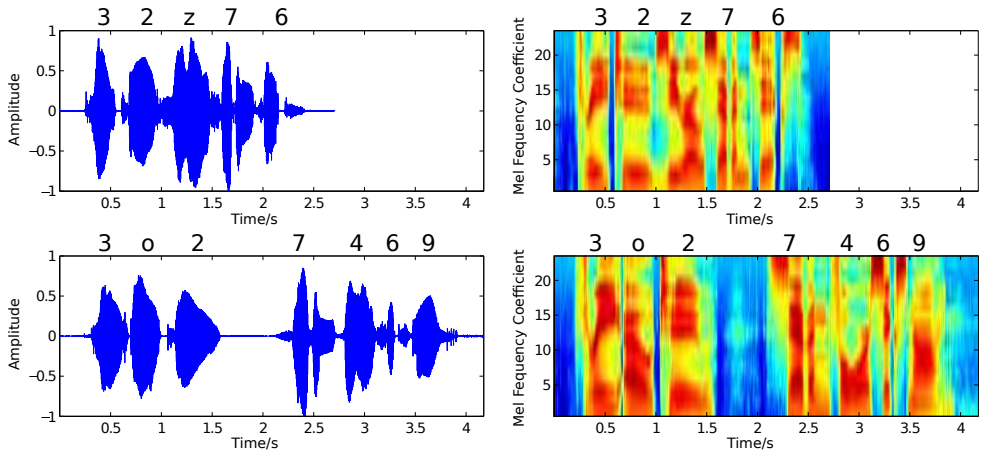


Figure 4.1: Example Sequences: Time domain representation (left) and spectral domain representation (right) of two audio recordings with the digit sequences "three (3) two (2) zero (z) seven (7) six (6)" and "three (3) oh (o) two (2) seven (7) four (4) six (6) nine (9)". The digits are written on top of each sequence at the time they are spoken. The time domain representation shows the audio signal sample values. The frequency domain representation shows the logarithmic mel spetrogram with 23 mel frequency bins. The color shows the energy (logarithmic scale) in each bin. Red indicates higher energy, blue indicates lower energy.

Distance Matrix

The distance matrix in Figure 4.2 is calculated as follows: Let $\boldsymbol{D}_{a,b}$ be the distance matrix computed from two sequences $\boldsymbol{o}^a_{1:L^a} = \{\boldsymbol{o}^a_1, \ldots, \boldsymbol{o}^a_{L^a}\}$ and $\boldsymbol{o}^b_{1:L^b} = \{\boldsymbol{o}^b_1, \ldots, \boldsymbol{o}^b_{L^b}\}$ of feature vectors, in this case mel frequency cepstral coefficients (MFCCs), of length L^a and L^b, respectively. The distance matrix has entries $[\boldsymbol{D}_{a,b}]_{i,j} = d(\boldsymbol{o}^a_i, \boldsymbol{o}^b_j)$, where $d(\cdot,\cdot)$ is an appropriately chosen distance measure. A commonly used distance measure is the cosine distance between the two feature vectors.

$$d(\boldsymbol{o}^a_i, \boldsymbol{o}^b_j) = \frac{1}{2}\left(1 - \frac{\boldsymbol{o}^{a\mathrm{T}}_i \boldsymbol{o}^b_j}{|\boldsymbol{o}^a_i||\boldsymbol{o}^b_j|}\right) \tag{4.1}$$

Sequence Alignment

The similar subsequences shown in Figure 4.2 are found by applying the UDTW algorithm, described in the next section, to the distance matrix. In general, the objective of the dynamic time warping algorithm is to find an optimal sequence of mappings, i.e. an alignment path $\Phi_k = (i_k, j_k)$ between $\boldsymbol{o}^a_{1:L^a}$ and $\boldsymbol{o}^b_{1:L^b}$, where k is the index of a mapping between a pair of feature vectors at the time instances i and j. The sequence of mappings is found by minimizing the cumulative distance

$$\overline{D}\left(\boldsymbol{o}^a_{1:L^a}, \boldsymbol{o}^b_{1:L^b}\right) = \sum_{k=1}^{K} d\left(\boldsymbol{o}^a_{i_k}, \boldsymbol{o}^b_{j_k}\right) \tag{4.2}$$

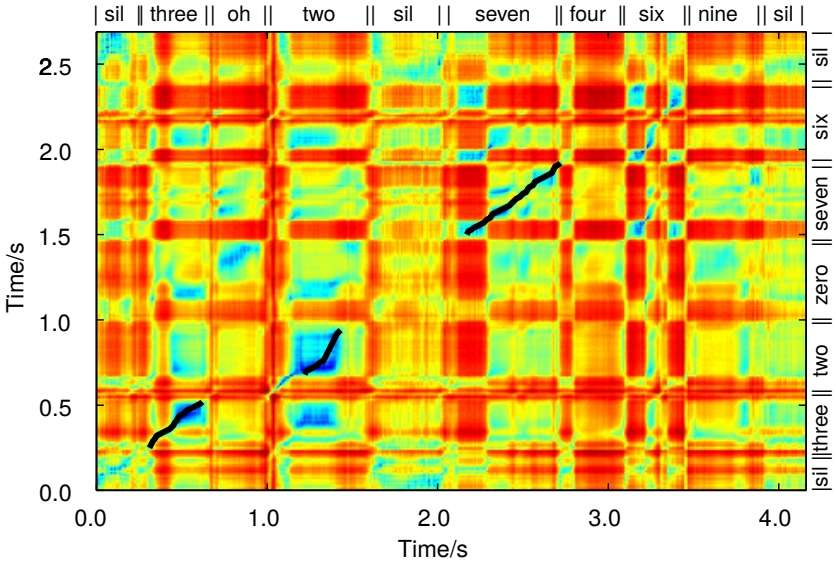


Figure 4.2: Example Sequences: Distance matrix with discovered alignment paths. Low distances are blue and high distances are red. Paths of discovered sequences along low distance are marked as black lines. The two sequences correspond to the sequences in Figure 4.1. The digits are written in the form they are spoken out and aligned to the spectrum. Vertical bars roughly indicate the beginning and end of a digit. The sequence "sil" represents a silence.

with respect to certain constraints on the allowed transitions from (i_k, j_k) to (i_{k+1}, j_{k+1}). Here, K is the length of the alignment path.

Unbounded Dynamic Time Warping Algorithm (UDTW)

Since the traditional dynamic time warping algorithm delivers only a single alignment path over the complete length of the two sequences, UDTW [AMO10] is used here to discover multiple similar subsequences within two longer sequences. To achieve this, $\boldsymbol{o}^a_{1:L^a}$ and $\boldsymbol{o}^b_{1:L^b}$ are replaced by subsequences within these two longer sequences. In principle all possible subsequences would have to be aligned with all other possible subsequences. The UDTW algorithm constrains the possible subsequences by restricting the transitions taken in Equation (4.2) and presents an algorithm to solve a modified version of the optimization problem in Equation (4.2).

The UDTW algorithm optimizes the objective function for a Length Constrained Minimum Average (LCMA) path, i.e., the subsequence on the alignment path for which the accumulated average distance along the path

$$\overline{D}^{\mathrm{LCMA}}\left(\boldsymbol{o}^a_{1:L^a}, \boldsymbol{o}^b_{1:L^b}\right) = \frac{1}{k_e - k_s + 1} \sum_{k=k_s}^{k_e} d\left(\boldsymbol{o}^a_{i_k}, \boldsymbol{o}^b_{j_k}\right) \tag{4.3}$$

is minimal, under the constraint that the LCMA length $k_e - k_s$ is larger than a given

minimum length $L^{\min}$, as proposed in [PG08]. The LCMA distance is used because it is independent of the path length.

The alignment is achieved with UDTW by extending multiple alignment paths, starting from multiple synchronization points within one distance matrix in forward and backward direction alternatingly. The following constraints are applied on the taken steps in Equation (4.3):

$$(i_{k+1}, j_{k+1}) \in \begin{cases} (i_k + 1, j_k + 2) \\ (i_k + 1, j_k + 1), \\ (i_k + 2, j_k + 1) \end{cases} \quad (i_{k-1}, j_{k-1}) \in \begin{cases} (i_k - 1, j_k - 2) \\ (i_k - 1, j_k - 1). \\ (i_k - 2, j_k - 1) \end{cases} \tag{4.4}$$

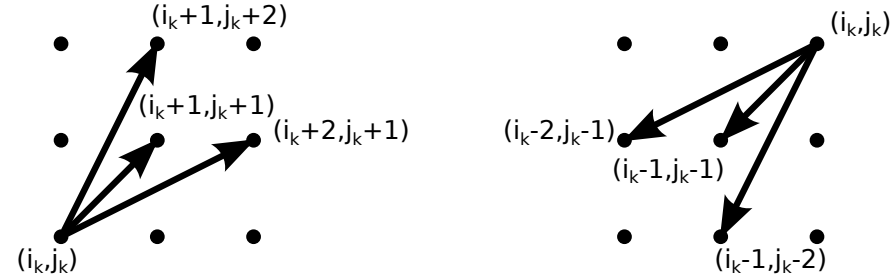


Figure 4.3: Step constraints for UDTW (forward (left), backward (right)).

In one of the sequences one step forward or backward has to be taken for the extension of the alignment path in forward or backward direction respectively. In the other sequence, one step forward or two steps forward (skipping one sample) has to be taken. The skipping allows for the compensation of different speaking rates while the need of one step forward ensures a continuous mapping between two subsequences. This results in diagonal alignment paths within the distance matrix with a maximum slope of two samples. Paths are only extended if the resulting average distance is below a maximum average distance $\overline{D}^{\max}$.

Alignment paths from different synchronization points are joined once they connect to each other, only keeping the longest alignment path. In order to reduce the amount of false matching pairs, only segments with at least a minimum length $L^{\min}$ are kept.

Two Step Pattern Clustering

The pattern discovery step delivers pairwise distances of segments found during the comparison of two feature vector sequences. The next step is to cluster these sequence pairs. There exist several algorithms to solve this task. One approach would be to (again) calculate the pairwise distortion between every segment found in the first clustering step and then use k-means to cluster these segments as described in [SBH11] and [AMO10]. This is computationally expensive due to the high number of comparisons. In this work, a similar two step clustering algorithm to [PG08] is applied.

Step 1: Iterative Node Search

The graph clustering algorithm of [NG04] is used to cluster similar sequences. Therefore a graph has to be built, where sequences are represented by nodes and two nodes connected by edges represent pairs found by the UDTW algorithm. In a first step, the start and end positions of the found segments in each sequence are used to form a similarity profile by simply counting how many segments overlap in each time step. In [PG08] the LCMA distance of each pair is used to weight the found segments instead, while here it turned out that simply counting without weighing gives a better similarity profile in a small vocabulary task.

Subsequently, the similarity profile is used to form a first set of clusters by iteratively selecting the time frame where the highest number of segments overlap as a node and then removing all overlapping segments from the profile. All segments overlapping this time frame are then represented by a single node. This is iteratively repeated until all segments are removed and therefore assigned to a certain node. This simple assignment proved to be effective in a small vocabulary task as the start and end points of the found segments did not vary much for a certain segment and mostly only one time frame was selected as a node per expected segment. See Figure 4.4 for an example similarity profile and selected time frames.

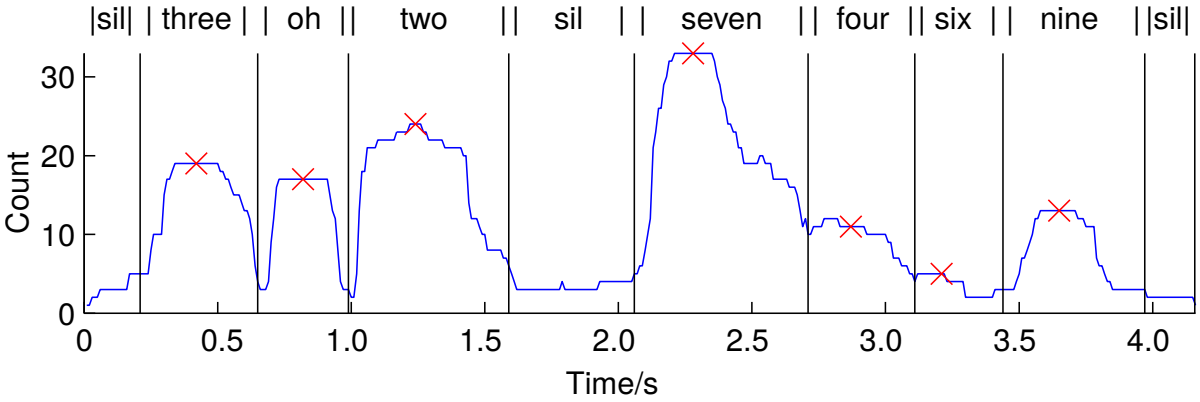


Figure 4.4: Similarity profile with counts per time step i.e., number of segments overlapping at this time step (blue line), selected nodes (red cross) and ground truth segment borders (black lines). The spoken digits are written on the top of the graph.

Note that this first step can also be interpreted as a segmentation of the feature vector sequences, since the minima of the similarity profiles correlate well with the true start and end points of the patterns.

For a more complex vocabulary, this approach has to be modified to consider cases where a longer sequence consists of smaller subwords and therefore might correspond to multiple nodes. The algorithm should then assign the longer sequences and subsequences to their corresponding nodes. This case is not considered in this chapter. Later in this work, model based algorithms are used for such vocabularies instead.

Step 2: Graph Clustering

The second step clusters the found segments to form the final pattern clusters. For this, the graph clustering algorithm of [NG04] is applied which was also used in [PG08]. The idea is to form a graph/adjacency matrix by identifying each selected time step with a node in the graph and each pairwise similarity found in the pattern discovery step with an edge between two nodes.

Weights are assigned to the edges proportional to the LCMA distance according to $c = \frac{\overline{D}^{\max} - \overline{D}^{\mathrm{LCMA}}}{\overline{D}^{\max}}$. The motivation behind this is that similar segments do share more edges with higher weights than dissimilar segments. This becomes obvious if the optimal case is assumed where the pattern discovery step only returns correct pairs. In that case there would only be edges between similar nodes.

The graph clustering algorithm is then used to partition the constructed graph in clusters of similar nodes by maximizing the modularity of the graph. As a result, several clusters of similar patterns which correspond to similar words are obtained. In the example case in Figure 4.1 they correspond to single digits. Note that there is no need to set the number of clusters explicitly as it is automatically determined by the maximum modularity.

4.2.3 Contribution

The main contribution of this work in this chapter consists of an improved local minima based synchronization point search, the application of iterative unsupervised training for a speech recognizer based on discovered patterns and finally the experimental results. The aforementioned contributions are described in this and the following subsections.

Local Minima based Synchronization Point search

In contrast to the original initialization method for synchronization points, in this work, the UDTW Algorithm of [AMO10] is modified by initializing the start points for possible path candidates using a local minima search. For the minimum search, the distance matrix is smoothed by sweeping a 2-dimensional hexagonally shaped smoothing kernel $\boldsymbol{K}$ over the logarithmic distance matrix:

$$\left[\widetilde{\boldsymbol{D}}_{a,b}\right]_{i,j} = \sum_{n=-\kappa}^{\kappa} \sum_{m=-\kappa}^{\kappa} \log\left\{\left[\boldsymbol{D}_{a,b}\right]_{i+n,j+m}\right\} \cdot \left[\boldsymbol{K}\right]_{n,m} \tag{4.5}$$

where $\kappa = 2$, and where the kernel is given by

$$\boldsymbol{K} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 2 & 2 & 1 \\ 1 & 2 & 3 & 2 & 1 \\ 1 & 2 & 2 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{pmatrix} \tag{4.6}$$

To avoid finding adjacent minima which correspond to the same alignment path, exclusion areas around selected minima are introduced within which all other local minima are discarded. The UDTW alignment is carried out on the unsmoothed distance matrix.

Unsupervised Training of a speech recognizer

For the training of an automatic speech recognizer, the clusters derived during the clustering step are used for labeling the training data with dummy labels (lacking any meaning in terms of word identity). As input to the clustering step the pattern matching pairs of the Min+UDTW (local minimum search + UDTW) approach were used as they had the best performance in terms of the defined performance measures. Some a priori knowledge about the amount of clusters was assumed to be available and thus the focus was laid on the 11 biggest clusters, as this is the expected number of digits. If the clustering process ends up with more clusters, the remaining ones are dropped. Admittedly this is a small limitation of the aspect "unsupervised", however, it allows the performance comparison of this approach with a speech recognizer trained in a supervised manner.

The cluster labels now form the transcriptions to train HMMs for a speech recognizer. Note that only the sequence of patterns is used for the training of alls HMMs together and that no isolated training of separate HMMs was performed, since start or stop indices from the found segments were not used. The word boundaries are rather found as a side product of the training procedure. This training procedure is also called embedded training [You+06] or Baum-Welch-Algorithm [RJ86].

It is important to note that the cluster labels serve only as initial transcriptions. Exploiting the power of statistical models, the trained acoustic model was used to decode the training data and thus to determine an updated transcription, which in turn is input to the next iteration of the acoustic model training. Overall, the iterative training is governed by the following two equations:

$$\Lambda^{r+1} = \text{argmax}_{\Lambda} \prod P\left(\boldsymbol{X}|T^{r};\Lambda\right) \tag{4.7}$$

$$T^{r+1} = \text{argmax}_{T}\, P\left(T|\boldsymbol{X};\Lambda^{r+1}\right) \tag{4.8}$$

where r is the iteration counter. At first the HMMs Λ are trained using the transcriptions T from the clustering process and the audio features $\boldsymbol{X}$ by maximizing the likelihood of the observation with Equation (4.7). Subsequently, the trained HMMs are employed to decode the training data and derive new transcriptions. The new transcriptions are then used in the next training iteration. The iterations could be extended to also train a language model. For the task considered here (connected digit recognition), a language model, however, was not necessary.

This iterative scheme resembles the acoustic unit training presented in [CHR11], where, however, the goal was to discover acoustic events rather than recognize speech.

4.2.4 Experimental Results

Preliminary experiments were performed on the TIDIGIT Database using the whole subset of training speakers consisting of 112 speakers and 77 digit sequences per speaker. Each speaker was processed separately in this evaluation, assuming speaker labels are known or can be derived using other algorithms for speaker diarization. Later in Section 5.4.2 all speakers are processed together in a speaker independent setup. For comparison, the proposed initialization was applied to both the SDTW-based pattern discovery approach of [PG08], by setting the centers of the diagonal search regions at the found minima instead of using fixed bands and limiting the length of a diagonal search region to a maximum length L^{Band} and to UDTW, as described above.

The ETSI standard front-end was used to extract the first 13 Mel-Frequency Cepstral Coefficients (MFCCs) from the audio data and additionally the first and second order derivatives, resulting in a 39 dimensional feature vector per 10 ms frame. As parameters, $L^{\text{LCMA}} = 20$ and $R^{\text{Band}} = 20$ for the original SDTW and $R^{\text{Band}} = 10$ and $L^{\text{Band}} = 60$ in case of SDTW with minimum based initialization were used. For the UDTW approach $L^{\text{LCMA}} = 15$ was used. L^{LCMA} is the minimal sequence length and R^{Band} the with of the diagonal band covered for the DTW alignment. The parameters were determined by several experiments to yield the best results.

Pattern Discovery and Clustering

As a performance measure for the pattern discovery step the false alarm rate (FalseRate) FR and the missed hit rate (MissingRate) MR defined as follows are used:

$$\text{FalseRate} = \frac{N^f - N^r}{N^f}, \ \text{MissingRate} = \frac{N - N^f}{N}, \tag{4.9}$$

where N^f is the total number of found alignment paths/pairs, N^r the number of correctly found pairs and N the number of correct pairs in the database. The resulting values of the performance measures are displayed in a Receiver Operating Characteristic (ROC)-like curve. Different tradeoffs between the FalseRate and the MissingRate are found by varying the parameter $\overline{D}^{\max}$ in the pattern clustering stage, see Section 4.2.2.

The experimental results from the TIDIGIT database are depicted in Figure 4.5. The proposed initialization approach improves the effectiveness of both, SDTW ("Min+SDTW") and UDTW ("Min+UDTW"), in terms of the evaluated performance measures within a region up to 35% FalseRate. At higher FalseRates, the "Min+SDTW" approach seems to approach the performance of the "Min+UDTW" and might even surpass it. Performance at higher FalseRates is not further investigated, since a lower FalseRate is more desired than a low MissingRate to avoid false pattern matches. Especially for a fixed false alarm rate, the missing rate is lower if the new starting point selection is used, see Table 4.1 for an example with 10% FalseRate.

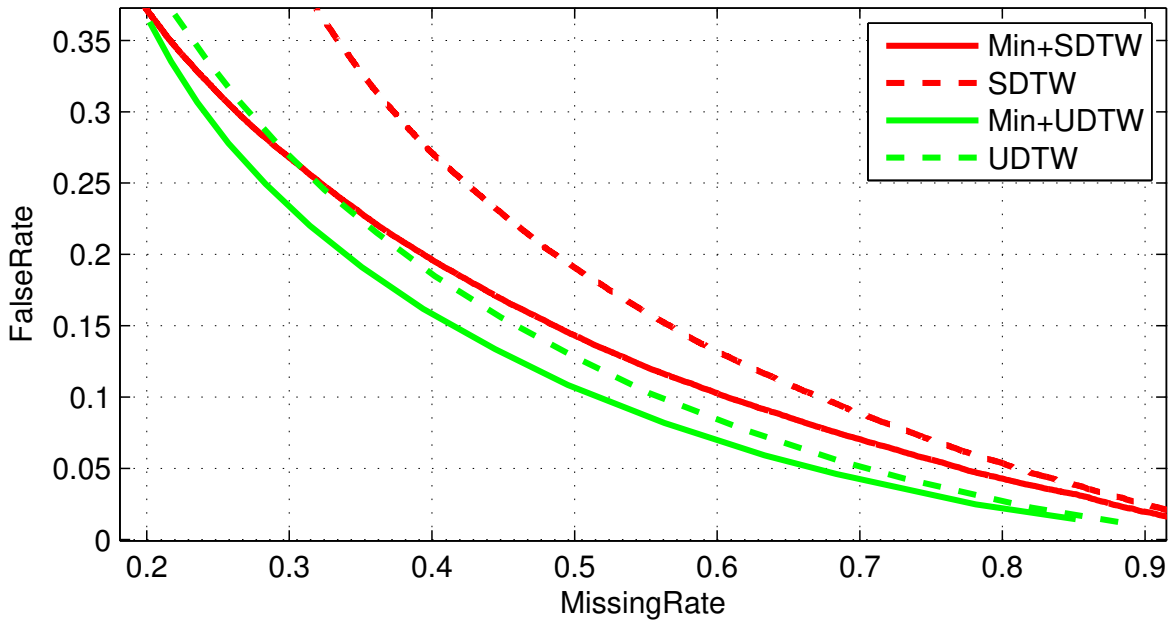


Figure 4.5: ROC of proposed initialization for SDTW ("Min+SDTW") and for UDTW ("Min+UDTW"), compared to original segmental DTW (SDTW) and unbounded DTW (UDTW)

Table 4.1: Missed hit rate at 10% false alarm rate

min+SDTW	SDTW	min+UDTW	UDTW
60.65%	67.19%	51.69%	55.95%

Unsupervised speech recognizer training

First the clustering was applied and then the derived clusters are used to train a speech recognizer in an unsupervised way. Labeling each cluster with the most likely label allows to calculate a word error rate. The word error rate (WER) is calculated by aligning the reference and hypothesis transcription according to the minimal Levenshtein distance as follows:

$$\mathrm{WER} = \frac{\mathrm{SUB} + \mathrm{DEL} + \mathrm{INS}}{N_{\mathrm{ref}}}, \tag{4.10}$$

where INS is the number of insertions (extra words) in the hypothesis, DEL the number of deletions (missing words), SUB the number of substitutions (exchanged words) and N_{ref} the total number of words in the reference.

Additionally the transcriptions can be used to train a speech recognizer. As described in the previous section an iterative training is used. Table 4.2 shows the word error rates of the clustering step and the word error rates when training a GMM HMM based speech recognizer on these transcriptions and decoding the same data with this speech recognizer. Repeating the decoding step and training with the transcriptions from the previous

decoding step several times reduces the word error rate further. After 5 iterations, the word error rate did not change significantly anymore.

Table 4.2: Speech recognition results in % on the TIDIGIT database using iterative unsupervised learning of patterns in speech.

Step	SUB	DEL	INS	WER
Clustering	9.98	9.13	5.51	24.62
Speech Recognizer (Trained on clustering)	9.06	4.97	0.98	15.01
Speech Recognizer (Iteration 5)	8.92	3.15	0.83	12.89

4.2.5 Conclusion

This chapter introduced the concept of unsupervised learning for speech recognition from speech recordings and gave an intuitive example to unsupervised learning. It then presented a heuristic DTW based approach approach to unsupervised learning. In this thesis, the heuristic DTW based approach was improved by introducing a local minima based synchronization point initialization in combination with the UDTW based pattern matching algorithm. This local minima based initialization reduced the missed hits rate of similar sequences from 67.2% to 60.7% for the sDTW based approach and from 56% to 51.7% for UDTW based approach. This is a significant reduction in missed hits rate by almost 30% relative when comparing UDTW and local minima based initialization to the traditional sDTW approach.

The result of this unsupervised learning approach was then clustered into similar sequences using a graph based representation and clustering algorithm. The clusters are then used for an unsupervised speech recognizer training. While the clustering yields a Word Error Rate (WER) of 24.62%, subsequent speech recognizer training, decoding and retraining yielded a WER of 15.01% and after 5 decoding and retraining iteration a WER of 12.89% respectively. This demonstrates the concept of unsupervised learning and the feasibility of unsupervised speech recognizer training.

Overall this first step yields results in unsupervised speech recognizer training. However, the approach only considers a flat hierarchy from audio recoding directly to segments, without intermediate acoustic units and was applied in a speaker dependent manner. Additionally no statistical modeling was used, but only heuristic comparisons between segments. Therefore in the next chapter a hierarchical statistical model based approach is introduced and applied. The approach works in a speaker independent manner as well, significantly improving the results from this chapter.

5 Model based approach to Unsupervised Learning for Automatic Speech Recognition

A second approach to unsupervised learning is the more sophisticated statistical model based approach. It is explained in this chapter. Instead of comparing audio segments to each other, parameters of statistical models, representing audio segments, are learned. The probabilities of audio segments are computed, given these statistical models and their parameters. This can make the learning approach more robust, since similar patterns are commonly represented by a statistical model which ideally also models the variance in these patterns. The computational demand can also be reduced since audio segments only have to be compared to the statistical models instead of being compared to each other.

Here, automatically learned transcriptions are used to learn an acoustic and a language model, which represent the acoustic features as well as the grammar and syntax for the structure of spoken words in the speech recordings. Then the acoustic and language model are used to produce better transcriptions. This results in an iterative process, alternating between acoustic and language model update and decoding of speech recordings. In the first iteration, this second approach can be initialized with a random set of transcriptions or model parameters or with transcriptions discovered using the first approach.

5.1 An example for and approaches to unsupervised learning

Intuitively the approach can be described as follows: Imagine having several utterances, where all utterances are built from a limited inventory of patterns but you don't know this inventory. You then try to describe what you hear with a limited inventory of patterns, while simultaneously trying to learn this inventory from the utterances. While learning, you try to keep the number and size of patterns in this inventory small though. This is similar to learning a dictionary of sounds or words of a language. At the same time you also try to keep the number of times you have to use one of the patterns from the dictionary, to describe what you hear, as small as possible. This is similar to building

a sentence with as little words or sounds as useful but using representative words or sounds from the lexicon.

Imagine the following two corner cases: You could take each utterance as one pattern in your inventory and just refer to the pattern in the inventory to describe the utterance you hear. In this first case you need a dictionary that consists of the utterances themselves. On the other hand you could build your inventory of patterns from very small parts of the utterances you hear. In this second case you would need to use a high number of patterns from the dictionary for each utterance. Both cases are not optimal, while an optimal solution lies somewhere in the middle, with the patterns in the inventory being big enough to capture whole repeating or unique patterns, while not being too big or too specialized. This can also be seen as an information theoretic approach in finding an optimal code under a certain objective, where the code is represented by the statistical models and their parameters.

For speech and language, a hierarchy of models representing increasing segment lengths and decreased granularity are used. The model assumptions and an unsupervised learning algorithm for these models are presented in the following sections.

5.2 Hierarchical structure of Speech and Language

Speech is often represented in a hierarchical way, where the smallest units are phonemes, followed by syllables, words and sentences, as depicted in Figure 5.1. The phonemes are often modeled as three state HMMs with left-to-right structure and some emission distribution for the observed speech features. This work will focus on the learning of these phoneme like units and the words. The statistical model based approach to unsupervised speech and language discovery described in this thesis is based on this hierarchy and the widely used ASR models and algorithms. To better understand the concept and structure of the statistical models, traditional ASR models are therefore first explained and then extended to unsupervised learning.

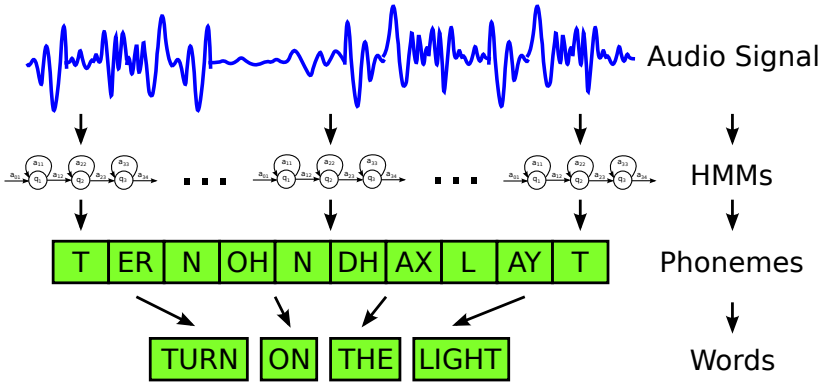


Figure 5.1: Hierarchical representation of speech

In ASR, speech is usually modeled in a hierarchical way, as depicted in Figure 5.1. First, an inventory of phonemes is modeled as HMMs. Phonemes are the basic building

blocks of the speech which is produced by concatenating these building blocks. Second, another inventory is modeled, consisting of words represented as sequences of phonemes, the meaningful bigger units.

For unsupervised learning, the same hierarchical model is used, learning AUs on the first level and words on the second level. Before describing the unsupervised learning from speech recordings, the standard supervised ASR models and approaches will be explained in the following section. From there on, the algorithms are extended towards unsupervised learning by taking away more and more supervision and instead introducing generative models or prior distributions over latent variables to arrive at a Maximum Likelihood (ML) or Bayesian learning approach.

5.3 Automatic Speech Recognition

In the ASR framework, recognizing a word sequence $w_{1:L} = w_1, \ldots, w_L$ of length L from a given sequence of observations $\boldsymbol{o}_{1:T} = \boldsymbol{o}_1, \ldots, \boldsymbol{o}_T$ of length T, is usually described as a Maximum a Posteriori (MAP) decision problem:

$$\hat{w}_{1:L} = \underset{w_{1:L}}{\operatorname{argmax}} \Pr\left(w_{1:L}|\boldsymbol{o}_{1:T}\right) \tag{5.1}$$

$$= \underset{w_{1:L}}{\operatorname{argmax}} \frac{p\left(\boldsymbol{o}_{1:T}|w_{1:L}\right) \Pr\left(w_{1:L}\right)}{p\left(\boldsymbol{o}_{1:T}\right)} \tag{5.2}$$

$$= \underset{w_{1:L}}{\operatorname{argmax}}\, p\left(\boldsymbol{o}_{1:T}|w_{1:L}\right) \Pr\left(w_{1:L}\right), \tag{5.3}$$

where the maximization is carried out over all word sequences $w_{1:L}$ of length L. A word is modeled as a discrete class, while an observation usually consists of a continuous valued vector. Since the length of the word sequence can range form zero to infinity, every possible length has to be considered. Ideally the decision rule of Equation (5.1) would be applied. Since an analytical expression for the posterior probability is often unknown and difficult to derive, the decision rule is reformulated into Equation (5.3). The conditional Probability Density Function (PDF) $p\left(\boldsymbol{o}_{1:T}|w_{1:L}\right)$ is called the acoustic model. It relates the word sequence to its acoustic realization, the sequence of observations. The Probability Mass Function (PMF) $\Pr\left(w_{1:L}\right)$ is called the language model and is used to calculate the a priori probability of the word sequence.

Speech is usually modeled according to the hierarchical representation depicted in Figure 5.1, with words consisting of phoneme sequences and phonemes again consisting of HMM state sequences. Equation (5.3) can therefore be rewritten to include a phoneme sequence $x_{1:N} = x_1, \ldots, x_N$ of length N and an HMM state sequence $q_{1:T} = q_1, \ldots, q_T$ of length T:

$$\hat{w}_{1:L} = \underset{w_{1:L}}{\operatorname{argmax}} \sum_{N=1}^{\infty} \sum_{\{x_{1:N}\}} \sum_{\{q_{1:T}\}} p\left(\boldsymbol{o}_{1:T}, q_{1:T}, x_{1:N}|w_{1:L}\right) \Pr\left(w_{1:L}\right) \tag{5.4}$$

$$= \underset{w_{1:L}}{\operatorname{argmax}} \sum_{N=1}^{\infty} \sum_{\{x_{1:N}\}} \sum_{\{q_{1:T}\}} p\left(\boldsymbol{o}_{1:T}, q_{1:T}|x_{1:N}, w_{1:L}\right) \Pr\left(x_{1:N}|w_{1:L}\right) \Pr\left(w_{1:L}\right). \tag{5.5}$$

Here, the length of the HMM state sequence equals the number of observations, since each observation is assumed to be emitted by a specific state. The length of the phoneme sequence, similar to the length of the word sequence, can range from zero to possibly infinity. To limit the possible lengths of the phoneme sequences and the word sequences, a phoneme is often modeled by a sequence of at least three distinct HMM states and a word by a sequence of at least one phoneme. Marginalization is carried out over all possible HMM state sequences of length T and all possible phoneme sequences of length N. Phonemes and HMM states are modeled as discrete classes, similar to words. The newly introduced conditional PMF $\Pr(x_{1:N}|w_{1:L})$ in Equation (5.5) is called probabilistic pronunciation lexicon, relating the word sequences to their phonemic realization. The acoustic model is extended to calculate the probability of the observation sequence given a certain phoneme and word sequence and considering the HMM state sequences, by marginalizing over the HMM state sequences.

The acoustic model, the pronunciation lexicon and the language model are explained in the following sections.

5.3.1 Acoustic Model

In general, the acoustic model relates a particular phoneme sequence and a particular word sequence to an observation sequence, considering the corresponding underlying HMM state sequences. Applying the two basic assumptions of a first order HMM, as used for ASR, conditional independence of an observation given the corresponding state and the assumption of a first oder Markov chain, the acoustic model can be simplified. Applying Bayes' theorem to the extended acoustic model as given by the first PDF and the sum over all HMM states in Equation (5.5), the acoustic model on can be reformulated:

$$p(\boldsymbol{o}_{1:T}|x_{1:N}, w_{1:L}) = \sum_{\{q_{1:T}\}} p(\boldsymbol{o}_{1:T}, q_{1:T}|x_{1:N}, w_{1:L}) \tag{5.6}$$

$$= \sum_{\{q_{1:T}\}} p(\boldsymbol{o}_1, q_1|x_{1:N}, w_{1:L}) \prod_{t=2}^{T} p(\boldsymbol{o}_t, q_t|\boldsymbol{o}_{1:t-1}, q_{1:t-1}, x_{1:N}, w_{1:L}) \tag{5.7}$$

$$= \sum_{\{q_{1:T}\}} p(\boldsymbol{o}_1|q_1, x_{1:N}, w_{1:L}) \Pr(q_1|x_{1:N}, w_{1:L}) \prod_{t=2}^{T} p(\boldsymbol{o}_t|\boldsymbol{o}_{1:t-1}, q_{1:t}, x_{1:N}, w_{1:L}) \Pr(q_t|\boldsymbol{o}_{1:t-1}, q_{1:t-1}, x_{1:N}, w_{1:L}) . \tag{5.8}$$

For HMM-based speech recognition, the two assumptions described above are applied: According to the conditional independence assumption, the observation $\boldsymbol{o}_t$, if conditioned on the current state q_t, is independent of all other observations and states as well as the phoneme and word sequences:

$$p(\boldsymbol{o}_1|q_1, x_{1:N}, w_{1:L}) \approx p(\boldsymbol{o}_1|q_1) \text{ and } p(\boldsymbol{o}_t|\boldsymbol{o}_{1:t-1}, q_{1:t}, x_{1:N}, w_{1:L}) \approx p(\boldsymbol{o}_t|q_t) . \tag{5.9}$$

According to the first-order Markov assumption, the current state q_t is only dependent on the previous state q_{t-1}:

$$\Pr(q_1|x_{1:N}, w_{1:L}) \approx \Pr(q_1) \text{ and } \Pr(q_t|\boldsymbol{o}_{1:t-1}, q_{1:t-1}, x_{1:N}, w_{1:L}) \approx \Pr(q_t|q_{t-1}). \quad (5.10)$$

This results in the simplified expression for the acoustic model:

$$p(\boldsymbol{o}_{1:T}|x_{1:N}, w_{1:L}) \approx \sum_{q_{1:T} \in \mathcal{B}'(x_{1:N}, w_{1:L})} p(\boldsymbol{o}_1|q_1) \Pr(q_1) \prod_{t=2}^{T} p(\boldsymbol{o}_t|q_t) \Pr(q_t|q_{t-1}). \quad (5.11)$$

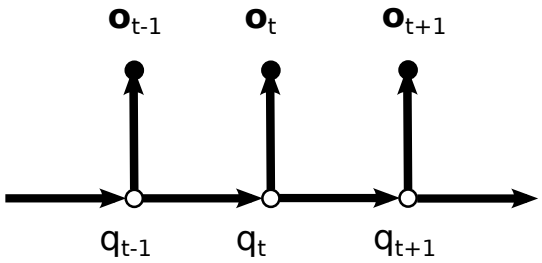


Figure 5.2: Graphical model of acoustic model structure with hidden states and observations.

The PMF $\Pr(q_1)$ is the a priori probability of the first state and $\Pr(q_t|q_{t-1})$ is the transition probability from the current state to the next state. Both probabilities are modeled as categorical distributions. The observation PDF $p(\boldsymbol{o}_t|q_t)$ of an observation being emitted by the given state is usually modeled as a GMM.

Since the first-order Markov assumption also removes the dependence on the phoneme and word sequence, a deterministic mapping function $\mathcal{B}'(x_{1:N}, w_{1:L})$ is introduced. This function maps from a particular phoneme and word sequence to a set of HMM state sequences representing this phoneme and word sequence and constrains the sum over all possible state sequences to a sum over the set of sequences that is given by this function. For the acoustic model to differentiate between different word and phoneme sequence pairs, based on observations, each pair therefore has to map to a$\mathcal{B}'(x_{1:N}, w_{1:L})$ unique set of state sequences. In practice, the mapping function is constrained even further, by dropping the dependence on the word sequence. The mapping from word sequences to phoneme sequences is left to the probabilistic pronunciation lexicon. This results in the mapping function $\mathcal{B}(x_{1:N})$, leaving the acoustic model to only model the mapping between observation sequences and phoneme sequences:

$$p(\boldsymbol{o}_{1:T}|x_{1:N}, w_{1:L}) \approx p(\boldsymbol{o}_{1:T}|x_{1:N}) \approx \sum_{q_{1:T} \in \mathcal{B}(x_{1:N})} p(\boldsymbol{o}_1|q_1) \Pr(q_1) \prod_{t=2}^{T} p(\boldsymbol{o}_t|q_t) \Pr(q_t|q_{t-1}). \quad (5.12)$$

Typical mapping functions result in monophone based acoustic models, where each phoneme is independently mapped to its corresponding HMM state sequence and context

dependent triphone based acoustic models, where a phoneme is mapped to a HMM state sequence, depending on the previous and following phoneme. In the latter case, decision tree based clustering techniques are applied during training to the HMM states to cluster similar HMMs states and to reduce the amount of actual states and therefore the number of parameters. The resulting models are called context dependent tied-state triphone models. A monophone or a triphone is usually modeled as a three state left to right HMM, resulting in an HMM sequence of at least three HMM states per monophone or triphone, where each state has to occur in consecutive order at least once but can also be repeated multiple times before changing to the next state. One pass through the state sequence of one monophone or triphone HMM is corresponding to one single monophone or triphone. The HMM structure is shown in Figure 5.3

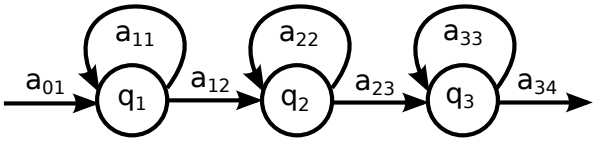


Figure 5.3: Three state Bakis left-to-right HMM.

The observation PDF $p(\boldsymbol{o}_t|q_t)$ is usually realized as a GMM with B_{q_t} mixture components for the corresponding state q_t:

$$p(\boldsymbol{o}_t|q_t) = \sum_{b_t=1}^{B_{q_t}} p(\boldsymbol{o}_t|b_t, q_t) \Pr(b_t|q_t), \tag{5.13}$$

where

$$p(\boldsymbol{o}_t|b_t, q_t) = \mathcal{N}\left(\boldsymbol{o}_t; \boldsymbol{\mu}_{\boldsymbol{o}|b_t,q_t}, \boldsymbol{\Sigma}_{\boldsymbol{o}|b_t,q_t}\right) \tag{5.14}$$

$$= \frac{1}{\sqrt{(2\pi)^D |\boldsymbol{\Sigma}_{\boldsymbol{o}|b_t,q_t}|}} \mathrm{e}^{-\frac{1}{2}\left(\boldsymbol{o}_t-\boldsymbol{\mu}_{\boldsymbol{o}|b_t,q_t}\right)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1}_{\boldsymbol{o}|b_t,q_t}\left(\boldsymbol{o}_t-\boldsymbol{\mu}_{\boldsymbol{o}|b_t,q_t}\right)}, \tag{5.15}$$

with the mean vector $\boldsymbol{\mu}_{\boldsymbol{o}|b_t,q_t}$ and the covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{o}|b_t,q_t}$. The mixture probabilities, also called mixture weights, $\Pr(b_t|q_t)$ are modeled as categorical distributions.

5.3.2 Pronunciation Lexicon

The mapping between phoneme sequences and word sequences is established by a probabilistic pronunciation lexicon $\Pr(x_{1:N}|w_{1:L})$. In the most common case, the pronunciation lexicon independently maps each word in the word sequence to a single or multiple phoneme sequences. The resulting phoneme sequence is the concatenation of the phoneme sequences of the words in the word sequence:

$$\Pr(x_{1:N}|w_{1:L}) = \prod_{l=1}^{L} \Pr\left(x_{n_{l-1}+1:n_l}|w_l\right), \tag{5.16}$$

with $n_0 = 0$, $n_L = N$ and $n_{l-1} < n_l$.

In the case of a non-probabilistic pronunciation lexicon, each word in the lexicon maps to one single phoneme sequence, resulting in $\Pr\left(x_{n_{l-1}+1:n_l}|w_l\right) = 1$, if the phoneme sequence corresponds to the word and otherwise in $\Pr\left(x_{n_{l-1}+1:n_l}|w_l\right) = 0$. There might be multiple words mapping to the same phoneme sequence though. In this non probabilistic case, the above probability results in an indicator function which equals to one if the phoneme sequence corresponds to the concatenation of the phoneme sequences of the words in the word sequence and otherwise to zero. Such a model is employed in the word discovery algorithm presented in Section 6, with the further restriction that each word in the lexicon maps to a unique phoneme sequence, making it a deterministic pronunciation lexicon, therefore no two distinct words can result in the same phoneme sequence.

For a so-called probabilistic pronunciation lexicon, each word in the lexicon can map to multiple phoneme sequences with their corresponding pronunciation probabilities, therefore a single word sequence can result in multiple phoneme sequences with their corresponding probabilities according to Equation (5.16).

A further generalization of this case, where each word is mapped to multiple sequences of phonemes in a probabilistic way, similar to the acoustic HMM, but with phoneme sequences as observations, is presented in Section 5.4.2. In this case $\Pr\left(x_{n_{l-1}+1:n_l}|w_l\right)$ is calculated by an HMM with categorical emission distributions.

5.3.3 Language Model

Finally the language model $\Pr(w_{1:L})$ is used to calculate the a priori probability of a given word sequence. The language model is used to emphasize more common word sequences and can therefore enforce grammatical structure and disambiguate similar sounding words with the same phoneme sequences by considering past and present words in the word sequence. Applying Bayes' theorem, the language model is often reformulated into a product of conditional probabilities:

$$\Pr(w_{1:L}) = \Pr(w_1) \prod_{l=2}^{L} \Pr(w_l|w_{1:l-1}) \tag{5.17}$$

$$\approx \Pr(w_1) \prod_{l=2}^{L} \Pr(w_l|w_{l-n+1:l-1}), \tag{5.18}$$

where the last approximation is called the n-gram approximation, leading to so-called n-gram language models. An n-gram language model approximates the probability of a word, given the $n-1$ preceding words. The probabilities are usually modeled as a categorical distribution. In general, hierarchical models with several n-gram contexts are used to apply smoothing of probabilities and backing off to shorter contexts in case a word has not been seen in a given context during training. In the word discovery algorithm in Section 6 the so called NHPYLM is used as a language model, applying smoothing and back off to shorter contexts and phoneme sequence based word probabilities.

5.4 Unsupervised learning for ASR

As a model based pattern discovery algorithm for words and phonemes, the algorithm proposed in [CR12] is adopted and applied to speech. This algorithm is a two level hierarchical algorithm, originally applied to acoustic event detection and scene classification, aiming to learn AUs representing basic sound events and sequences of AUs, which then represent acoustic scenes in an unsupervised manner. Instead of basic sound events and acoustic scenes, the aim of this work is to learn phone like units and words. In this work AUs will be used synonymously for phones.

The algorithm in [CR12] resembles an unsupervised learning of the ASR models described in the previous section. On the first level, the HMM resembling the acoustic model, modeling the relation between AUs and acoustic features, is learned in an unsupervised way from the continuous audio recordings, automatically discovering a transcription in terms of acoustic units which could resemble phones. The sequence of acoustic units from the first level is used as input to the second level of the unsupervised learning algorithm to discover recurring sequences of acoustic units, assumed to represent words. The second level resembles the unsupervised learning of the pronunciation lexicon and language model described in the previous section, automatically discovering words and their probabilities.

In the following, the unsupervised AU learning algorithm and the learning of a pronunciation lexicon and language model are described. The adoptions to the characteristics of speech and language will be described as well.

5.4.1 Acoustic Unit Learning

The goal of the first level is the unsupervised learning of AUs from continuous speech. The unsupervised learning uses the models described earlier. However in contrast to supervised learning, unsupervised learning does not have labeled training data for the model training and the labels have to be discovered automatically. The AU learning therefore resembles the unsupervised learning of the acoustic model described previously. The unsupervised learning is realized as an iterative learning algorithm, consisting of a model estimation step given a label sequence, and a decoding step to derive a new label sequence.

In this work, the continuous speech is represented as a vector sequence of Mel Frequency Cepstral Coefficients (MFCCs), normalized to zero mean and unit variance. The task now is to discover reoccurring patterns in this vector sequence.

Since an initial model or an initial label sequence is required to initialize the learning algorithm, the following two steps are performed for unsupervised learning of AUs:

- Segmentation and clustering for initialization.
- Iterative HMM training consisting of decoding and learning steps.

Since the learning algorithm requires a label sequence to learn the models, the first step will deliver a transcription to initialize the model training. In the second step the

model parameters, including the transcription, are refined by iterating between training and decoding steps. The decoding step delivers a new label sequence which is used in the next training step. In contrast, during supervised learning the label sequence would be kept constant. The training and decoding step resemble the same steps applied in the case of supervised learning but without fixed labels. The unsupervised learning is therefore achieved by allowing the system to determine its own transcription and update its models respectively.

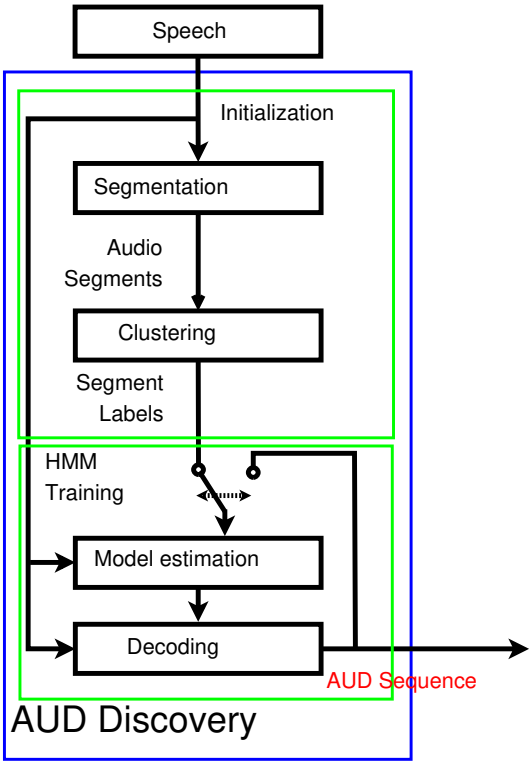


Figure 5.4: Unsupervised AU learning algorithm. The two steps "initialization" and "iterative training" are depicted in green boxes. Both steps are combined in the blue box to the AU discovery, with speech as input and an AU sequence as output. The switch is flipped over after the initialization to use the estimated model in subsequent iterations.

Figure 5.4 depicts a block diagram for the unsupervised AU learning algorithm. The block "Initialization" depicts the initialization phase while the block "HMM Training" depicts the iterative HMM training of the acoustic model. The input to the algorithm is the continuous speech and the output a sequence of AUs. The steps are further explained in the following sections and related to the speech models described above.

Segmentation and Clustering

To bootstrap the learning algorithm, an initial label sequence is derived by unsupervised segmentation of the audio and clustering of similar segments. The speech input is first

segmented into chunks according to a local (cosine) distance measure between the mean representative of the current segment and the next feature vector using a constraint on the minimum length of a segment. MFCC feature vectors were used as input.

The segmentation is applied as follows: Let $k-1$ indicate the last frame of the previous segment ($k-1=0$ at sentence start). If the average of the similarity between the MFCC feature vectors $\boldsymbol{o}_k, \ldots, \boldsymbol{o}_i$ and $\boldsymbol{o}_{i+1}$ is smaller than some predefined threshold $\delta^{\boldsymbol{o}}$ and if the segment length exceeds Δ frames:

$$\frac{1}{i-k}\sum_{j=k}^{i} d(\boldsymbol{o}_j, \boldsymbol{o}_{i+1}) < \delta^{\boldsymbol{o}} \quad \text{AND} \quad |i-k| > \Delta \tag{5.19}$$

a segment boundary is placed after time frame i, i.e., a segment is formed by $S = (\boldsymbol{o}_k, \boldsymbol{o}_{k+1}, \ldots, \boldsymbol{o}_i)$. The threshold $\delta^{\boldsymbol{o}}$ and the minimum segment length Δ are chosen such that the average segment length corresponds to the expected length of a phoneme. As the distance measure $d(\cdot,\cdot)$ the cosine distance as described in the previous chapter is employed.

Next the clustering step is applied. The goal of clustering is to group the obtained segments according to acoustic similarity. As a result of this clustering, a sequence of cluster labels can be assigned to each utterance. This sequence of cluster labels will serve as the initial label sequence for the iterative training of the AUD models. The algorithm used for clustering is similar to the algorithm used for the heuristic dynamic time warping-based approach described in the previous chapter, consisting of score calculation via sequence alignment followed by graph clustering as following:

Let $\mathcal{S} = \{S^1, \ldots, S^{N_s}\}$ be the set of segments obtained from the segmentation step. As a similarity measure between segments the (length normalized) DTW distance $d_{a,b}$ between two segments S^a and S^b is employed, using the cosine distance. This is similar to the heuristic DTW algorithm described before, with the length of the segment already given by the segmentation. Therefore only a single DTW distance between two segments has to be calculated.

The previously described algorithm was extended to avoid the huge computational effort from computing all pairwise distances. The clustering is carried out on a representative subset of the segments instead. To find such a subset the approach of [SBH11] is adopted, which applies the k-means++ algorithm [AV07] to determine K elements of the set of segments $\mathcal{S}$ such that the variance in the data is well represented:

1. Set $k=1$. Choose the first segment S^k uniformly at random from the set $\mathcal{S}$.
2. Compute the DTW distances $d(S^k, S^i)$ between the chosen segment S^k and all other $N-1$ segments in $\mathcal{S}$, and store the distances in the vector $\mathbf{d}_{\min}$.
3. Increment k and sample the next seed value $S^k \in \mathcal{S}$ with probability proportional to its distance in $\mathbf{d}_{\min}$.
4. Compute the DTW distances between S^k and all other segments and replace an entry in the minimum distance vector $\mathbf{d}_{\min}$ if the computed distance is smaller than the stored value.

5. Go to 3. until K representatives are found.

The idea behind this kind of seed selection is to prevent elements of $\mathcal{S}$ from being drawn which are very close to the set of already drawn segments. On the other hand, although insignificant outliers in $\mathcal{S}$ may have a great distance to the set of previously drawn elements, the probability to draw one of them is small, since the overall number of outliers is by definition small.

Clustering is now carried out on a sparse distance matrix containing only the distances between the chosen K segments and all other segments. As a clustering algorithm the graph clustering algorithm by Newman is again used, which iteratively maximizes the modularity of the clustered graph, the ratio of the edges connecting vertices within a cluster to the edges connecting vertices of different clusters [NG04]. The adjacency matrix is computed from the distance matrix using $e^{-d_{a,b}}$, with diagonal elements and elements with no distance assigned set to zero. Finally a set of $\hat{K}$ clusters is obtained at the maximum modularity, where $\hat{K} \ll K$.

Thus, through the choice of K representatives, the number of distance computations is reduced from the order of $\mathcal{O}(N^2)$ to $\mathcal{O}(K \cdot N)$. Moreover, the graph clustering algorithm runs significantly faster on the resulting sparse distance matrix.

Iterative AU HMM Training

In this second step, the actual acoustic model is trained in an unsupervised way. The training is achieved by iterating between training and decoding steps for the acoustic model. In the first iteration, the cluster labels are interpreted to be AU labels, and the cluster labels obtained in the previous step are used as an initial label sequence. The label sequence is denoted as transcription $T^{d,0}$, $d = 1, \ldots, D$. Here, d is the index of the training utterance and D is the total number of training utterances. For each AU $A \in \mathcal{A}$ an HMM λ^A is defined and the set of all AU models is referred to as $\Lambda^{\mathcal{A}}$. Every model is a 3-state left-to-right HMM with Gaussian mixture emission distributions as described previously for the acoustic model.

Let $T^{d,i}$ denote the transcription of the d-th training utterance in the i-th iteration, and let $\boldsymbol{o}^d_{1:\tau_d} = (\boldsymbol{o}^d_1, \ldots \boldsymbol{o}^d_{\tau_d})$ denote the MFCC feature vector sequence of the d-th utterance. The maximum likelihood estimation of the HMM parameters $\Lambda^{\mathcal{A}}$ and the transcriptions is done with the following iterative Expectation Maximization (EM) algorithm, which alternates between reestimation of the AU parameters, Equation (5.20), and decoding, i.e., reestimation of the label sequence, Equation (5.21) [CR12; Siu+13]:

$$\Lambda^{\mathcal{A},i+1} = \underset{\Lambda^{\mathcal{A}}}{\operatorname{argmax}} \prod_{d=1}^{D} p(\boldsymbol{o}^d_{1:\tau_d} | T^{d,i}; \Lambda^{\mathcal{A}}) \tag{5.20}$$

$$T^{d,i+1} = \underset{T}{\operatorname{argmax}}\, Pr(T | \boldsymbol{o}^d_{1:\tau_d}; \Lambda^{\mathcal{A},i+1}). \tag{5.21}$$

Here the Viterbi approximation is used instead of employing the Forward-Backward algorithm to simplify the learning process. After convergence, when the label sequence does not change further with each new iteration, the obtained label sequence is returned. The trained acoustic model can also be used to decode new data.

5.4.2 Probabilistic Pronunciation Lexicon and Language Model learning

The next step is the learning of words from unsegmented sequences of AUs delivered by the AU learning algorithm. The term "word" refers to a consistent sequence of AUs which in the unsupervised case need not be identical with the linguistic notion of a word. Here it is assumed that a word can map to a single sequence of AUs or several similar sequences of AUs. A particular AU sequence is always mapped to one single word. This one-to-many mapping is called a probabilistic pronunciation lexicon.

The probabilistic pronunciation lexicon is then used to determine the possible words in a sequence of AUs and hereby segment the sequence of AU into possible sequences of words. Using a language model, these sequences are weighted according to their word sequence probabilities. The probabilistic pronunciation lexicon and language model, and therefore the possible sequences of words, given the AU sequences, are learned in an unsupervised way directly from the unsegmented AU sequences.

Revisiting the models used for ASR previously described in Section 5.3, the Bayesian decision rule applied to ASR aims at finding that word sequence $\hat{w}_{1:L}$ which maximizes the posterior probability given the acoustic evidence $\boldsymbol{o}_{1:T}$ or, equivalently,

$$\hat{w}_{1:L} = \underset{w_{1:L}}{\operatorname{argmax}}\, p\left(\boldsymbol{o}_{1:T}|w_{1:L}\right) \Pr\left(w_{1:L}\right) \tag{5.22}$$

with the acoustic model $p\left(\boldsymbol{o}_{1:T}|w_{1:L}\right)$ and the language model $\Pr\left(w_{1:L}\right)$.

Including the pronunciation lexicon $\Pr\left(x_{1:N}|w_{1:L}\right)$, the acoustic model was rewritten as

$$\hat{w}_{1:L} = \underset{w_{1:L}}{\operatorname{argmax}} \sum_{N=1}^{\infty} \sum_{\{x_{1:N}\}} p\left(\boldsymbol{o}_{1:T}|x_{1:N}, w_{1:L}\right) \Pr\left(x_{1:N}|w_{1:L}\right) \Pr\left(w_{1:L}\right). \tag{5.23}$$

In this work, a probabilistic pronunciation lexicon is assumed, mapping a word to multiple pronunciation variants. For the probabilistic pronunciation lexicon, an HMM with discrete emission distributions for each word is employed.

The lexicon learning algorithm presented here is also based on the algorithm for semantic analysis of audio by Chaudhuri and Raj [CR12]. The algorithm, however, is modified in one key aspect: The original algorithm used a Bag-of-AUs model, where a word is described by the set of AUs occurring within the word, irrespective of the order of AUs. Is this work, this rather simple model is replaced by state dependent emission probabilities over AUs, where each state in the HMM can have a different emission distribution over AUs. Here, a linear HMM is used, only allowing transitions in the forward direction or to the final state but no self loops or backward transitions. This allows for better capturing the temporal succession of AU within a word and results in better error rates, as will be demonstrated in the experimental section later. This structure of the HMM was chosen so that the transition probabilities from one state to another result in a length distribution over the possible lengths for the word and therefore additionally allowing to probabilistically model the length of a word.

According to [CR12], a word is modeled as an HMM emitting an AU sequence. In principle, the same iterative algorithm as in the previous section for training the HMM

parameters of the AUs is applied. However, the input data are now of categorical nature: The AU token sequences T^d, $d = 1, \ldots, D$, delivered by the first stage. In the following the model and its training procedure are first described.

Secondly the important issue of initialization is described in Section 5.4.3. Similar to the unsupervised HMM learning, the algorithm is an iterative EM algorithm, iterating between model estimation and update. In this work, the learning algorithm is extended by an initialization step, based on the patterns discovered by the heuristic DTW approach.

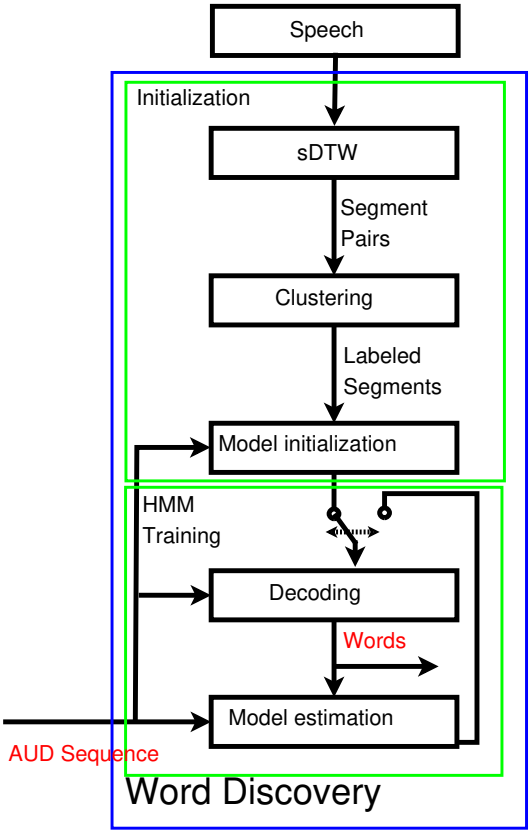


Figure 5.5: Unsupervised word learning algorithm. The two steps "initialization" and "iterative training" are depicted in green boxes. Both steps are combined in the blue box to the word discovery, with AUs as input and word sequences as output. The switch is flipped over after the initialization to use the estimated model in subsequent iterations.

The algorithm is depicted in Figure 5.5 and similar to the unsupervised HMM learning algorithm, consists of two steps:

- DTW and clustering based model initialization (Initialization)
- Iterative decoding and model estimation steps (HMM Training)

HMM Topology

Instead of the Bakis left-to-right HMM topology, which is commonly used in (supervised) ASR for the acoustic model, a more constrained topology is used for the pronunciation dictionary. The HMM topology is chosen, such that the transition probabilities are determined by a length distribution over the length of words. To achieve this, self loops are removed and only forward transitions are allowed. Additionally a strong length constraint is used, resulting in the HMM structure depicted in Figure 5.6.

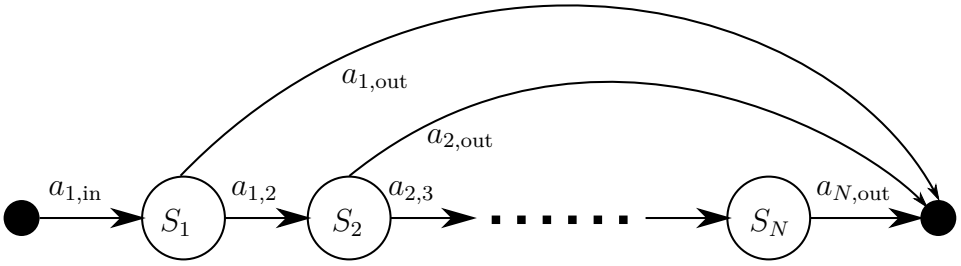


Figure 5.6: HMM topology of word model

Conveniently this model collapses to a deterministic pronunciation dictionary, if the length distribution and the emission distribution for each word have a probability of one for only a single length and a single AU sequence.

Here, the length n of a realization of the word w^k, in number of AUDs, is modeled to be a draw from the Negative Binomial (NB) distribution with word-specific parameters (r_k, p_k), where the Negative Binomial distribution is a generalization of the Poisson distribution, which is often used to model lengths:

$$n \sim P_{\text{NB}}(n; r_k, p_k), \quad \text{where} \tag{5.24}$$

$$P_{\text{NB}}(n; r_k, p_k) = \binom{n + r_k - 1}{n} p_k^n (1 - p_k)^{r_k}. \tag{5.25}$$

The transition probabilities can be readily computed from the NB distribution:

$$a_{1,\text{out}} = \text{NB}(1; r, p) \tag{5.26}$$

$$a_{1,2} = 1 - \text{NB}(1; r, p) \tag{5.27}$$

$$a_{2,\text{out}} = \frac{\text{NB}(2; r, p)}{a_{1,2}} \tag{5.28}$$

$$a_{2,3} = \frac{1 - \text{NB}(2; r, p)}{a_{1,2}} \tag{5.29}$$

$$\dots$$

$$a_{n,\text{out}} = \frac{\text{NB}(n; r, p)}{\prod_{i=2}^{n} a_{i-1,i}} \tag{5.30}$$

$$a_{n,n+1} = \frac{1 - \text{NB}(n; r, p)}{\prod_{i=2}^{n} a_{i-1,i}} \tag{5.31}$$

(5.32)

The probability $a_{1,\text{in}}$ is determined by the language model, giving the probability of entering the HMM and therefore the probability of the occurrence of the word, assuming a unigram language model.

The emission probabilities at each state for each word are modeled as multinomial probabilities, where $\Phi_{k,l,m}$, with $k = 1, \ldots, N$ and $l = 1, \ldots, L_k$, $m = 1, \ldots, M$, is the probability that AUD m is emitted by state l of word w_k.

Language Model

In the unsupervised setting the words are not known in advance however, we can still employ a priori knowledge about them: We assume that the unigram word probabilities adhere to a power law distribution, the Zipf law, which holds ubiquitously across many languages [MS99].

Let w^k, $k = 1, \ldots, N$ denote the words, where k denotes the rank of the word in a probability table ordered according to descending frequency of occurrence, and where N is the lexicon size. According to Zipf's law the unigram probabilities are given by

$$P(w^k; s) = \frac{1/k^s}{\sum_{i=1}^{N} 1/i^s}. \tag{5.33}$$

The exponent s will be estimated on the data. Here we assume that either the lexicon size N is known or the number N of different words to be discovered is fixed in advance, whereas the latter assumption actually does not pose a huge constraint, since the number of words could be set to a rather high number (higher than the actual number of words). This would result in several HMMs without observations which could be removed during the iterative learning.

Iterative Training

The model parameters are updated using iterative training and estimated using the Baum-Welch algorithm.

Let $T^d = (T_1^d, \ldots, T_t^d, \ldots, T_{N^d}^d)$ be the observed AUD sequence of the d-th utterance of length N^d AUDs. Here, $T_t^d \in \mathcal{A} = \{A^1, \ldots, A^M\}$ will be called the t-th observation, with $\mathcal{A}$ being a set of M AUDs.

In the E-step, the posterior probability $\gamma_t(q_{k,l})$ of being in the l-th HMM state of word w^k for the t-th observation, given the whole observed sequence T^d, is computed by the forward-backward algorithm:

$$\gamma_t(q_{k,l}) = \frac{\alpha_t(q_{k,l}) \cdot \beta_t(q_{k,l})}{\sum_{m,n} \alpha_t(q_{m,n}) \cdot \beta_t(q_{m,n})}, \tag{5.34}$$

where $\alpha_t(q_{k,l})$ is the forward probability, the probability of being in state $q_{k,l}$ at time t and having observed $(T_1^d, \ldots, T_t^d)$, and $\beta_t(q_{k,l})$ denotes the backward probability of being in state $q_{k,l}$ at time t given the future observations $(T_{t+1}^d, \ldots, T_{N^d}^d)$.

In the M-step the multinomial emission probabilities are reestimated as follows

$$\Phi_{k,l,m} = \frac{\sum_{d=1}^{D}\sum_{t=1}^{N^d}\gamma_t(q_{k,l})\delta(T_t^d = A^m)}{\sum_{m'=1}^{M}\sum_{d=1}^{D}\sum_{t=1}^{N^d}\gamma_t(q_{k,l})\delta(T_t^d = A^{m'})}, \tag{5.35}$$

for all $k = 1, \ldots, N$, $l = 1, \ldots, L^k$, $m = 1, \ldots, M$. Here $\delta(\cdot)$ denotes the Kronecker delta symbol which takes the value one if the argument is true and zero otherwise.

The NB parameters (r_k, p_k) for each word w^k are estimated by maximizing the word dependent likelihood

$$L^k = \prod_{i=1}^{m^k} P_{\mathrm{NB}}(n^i; r_k, p_k); \quad k = 1, \ldots, N. \tag{5.36}$$

Here, m^k denotes the number of occurrences of word w^k in the training corpus, and n^i the length of the i-th occurrence. Maximizing Equation (5.36) leads to the following update formulas [CR12]:

$$p_k = \frac{\sum_{i=1}^{m^k} n^i}{m^k r_k + \sum_{i=1}^{m^k} n^i}. \tag{5.37}$$

$$0 = m^k \ln\left(\frac{r_k}{r_k + \sum_{i=1}^{m^k} n^i/m^k}\right) - m^k\Psi(r_k) + \sum_{i=1}^{m^k}\Psi(n^i + r_k). \tag{5.38}$$

In Equation (5.38), $\Psi(\cdot)$ denotes the digamma function. Since no closed-form solution exists for r_k, Equation (5.38) has to be solved iteratively to find the ML estimate for r.

The iterative training is carried out by alternating between the E-step, Equation (5.34) and the M-step, Equations (5.35), (5.37) and (5.38).

Finally, the parameter s of the Zipf distribution is estimated as the slope of the best fit line between the log-expected frequencies $\mathbf{y} = (\ln E_{w_1}, \ldots, \ln E_{w_N})^T$, where $E_{w_k} = \frac{m^k}{\sum_j m^j}$, and its log-rank $\mathbf{x}$ using linear regression:

$$s = -\boldsymbol{x}^+\boldsymbol{y}. \tag{5.39}$$

Bag-of-AUDs Model

In Equation (5.35) word and state dependent multinomial distributions are estimated. As an alternative one could use tied states, where all HMM states of a word model share the same emission probability. This results in the Bag-of-AUDs model that was proposed in [CR12]. To achieve this, Equation (5.34) is replaced by

$$\gamma_t(k) = \frac{\sum_l \alpha_t(q_{k,l}) \cdot \beta_t(q_{k,l})}{\sum_{m,n} \alpha_t(q_{m,n}) \cdot \beta_t(q_{m,n})} \tag{5.40}$$

where the additional summation is over all HMM states of word w_k. The reestimation formula for the multinomial emission distribution then becomes

$$\Phi_{k,m} = \frac{\sum_{d=1}^{D}\sum_{t=1}^{N^d}\gamma_t(k)\delta(T_t^d = A^m)}{\sum_{m'=1}^{M}\sum_{d=1}^{D}\sum_{t=1}^{N^d}\gamma_t(k)\delta(T_t^d = A^{m'})}, \tag{5.41}$$

where $\Phi_{k,m}$ is the probability of observing AUD A^m in word w^k.

5.4.3 Contribution

The main contribution of this work in this chapter consists of the application of the aforementioned algorithms to speech recognition. The k-means++ based clustering scheme for AUD learning was proposed. Especially the HMM topology of state dependent emission probabilities for the probabilistic pronunciation lexicon and the application to speech recognition were introduced in this work. A major contribution is the initialization by pattern discovery, as described in the following. Additionally the overall system for unsupervised training of a speech recognizer with iterative retraining as well as the experimental results are a final major contributions in this chapter, as described in the following.

Initialization by Pattern Discovery

In the unsupervised setting neither the pronunciation dictionary nor the transcription of an utterance in terms of a word sequence is given. To obtain an appropriate initialization, the dynamic time warping (DTW) approach described in the previous chapter is employed. In [JH13] it has been argued that word-level patterns are fairly stable across speakers. Thus acoustic pattern discovery techniques, such as DTW, are suitable to discover word or phrase sized patterns even if uttered by different speakers.

Here, the DTW algorithm described earlier was employed. It delivered a collection of segment pairs and a similarity measure for each pair. Then, the found segment pairs were clustered using the same graph clustering algorithm as described earlier and a predefined number of the N biggest clusters was returned.

DTW, however, is usually unable to find all occurrences of a word in a corpus. Thus there will be parts of the acoustic signal for which no similar pattern has been found and which will therefore not be assigned a cluster label. Also, the speech may contain more words/phrases than one is able to discover by DTW.

Thus the output of the clustering is unable to deliver a complete transcription of the audio data. Instead of initializing by a label sequence, the HMM training is initialized with an initial model: For each segment found by DTW, the AUD sequence within this segment was extracted, and a multinomial emission distribution over AUDs was estimated from the AUDs in the AUD sequences of all segments belonging to the same cluster. Each distribution of one state dependent word HMM will then be set to the same initial distribution while the transition probabilities are unchanged. If more word HMMs are assumed than are discovered by the DTW algorithm, the remaining word HMM will be left unchanged. By this, an initial model is partly initialized and the iterative training can start.

5.4.4 Experimental Results on Unsupervised Learning for ASR

Experiments were performed on the TIDIGITs database, downsampled to 16 kHz, both in a speaker-dependent and speaker-independent setting. The training set comprises 112 speakers and 77 digit sequences per speaker. The dataset contains sequences of 11

distinct words, the numbers “oh” and “zero” to “nine”. The ETSI standard front-end was used to extract 13 Mel-frequency cepstral coefficients (MFCC) from the audio data and additionally the first and second order derivatives, resulting in a 39 dimensional feature vector per 10 ms frame. Finally cepstral mean and variance normalization (CMVN) was applied on the full feature vector.

AUD Discovery Performance

First, the performance of the AUD extraction algorithm was evaluated by calculating the average precision and precision-recall break-even as proposed by [Car+11] on the AUDs and, for comparison, on the MFCCs extracted from the TIDIGITS database. The average precision is a measure of how well a representation is able to capture the (desired) variability between different subword units while being insensitive to variations due to different realisations of the same subword unit.

It has been shown that average precision has a high correlation with the phoneme recognition rate [Car+11]. The average precision and precision-recall break-even are calculated in a same-different task [Car+11] which tests whether a given speech representation can correctly classify two speech segments as having the same word type or not. For each word pair in a pre-defined set $\mathcal{S}$ the DTW cost between the AUD or acoustic feature vectors under the given representation is computed. Two segments are then considered a match if the cost is below a threshold. Precision P and recall R at a given threshold τ are defined as

$$P(\tau) = \frac{N^r(\tau)}{N^f(\tau)},\ R(\tau) = \frac{N^r(\tau)}{N}, \tag{5.42}$$

where N^f is the number of discovered pairs, N^r is the number of correctly discovered pairs with the same word type and N the actual number of pairs with the same word type in $\mathcal{S}$. By varying the threshold τ a precision-recall curve can be computed, where the final evaluation metric is the average precision (AP) or the area under that curve. The precision-recall break-even is the point where precision and recall are equal.

Table 5.1 shows the results for the average precision and precision-recall break-even for the speaker dependent and speaker independent setups. $\hat{K} = 128$ AUDs were extracted and $K = 1024$ seed values in the clustering step were used.

Table 5.1: Average precision and precision-recall break-even in speaker dependent and speaker independent case for AUDs and MFCCs extracted form TIDIGITs database in %

	Average precision		Precision-recall break-even	
Setup	AUD	MFCC	AUD	MFCC
Speaker dependent	94.7	92.6	83.3	85.9
Speaker independent	64.6	61.7	60.0	57.5

The AUD sequences for each word were extracted from the transcriptions delivered by the AUD training using the ground truth (GT) transcriptions that were estimated

doing a forced alignment on the MFCCs. For this the discovered AUD sequences were compared with the ground truth word transcriptions that were obtained from a forced alignment on the MFCCs. An AUD was assigned to the AUD sequence forming a word if at least half of the AUD was covered by the segment given by the ground truth labels. This was done because an AUD naturally can have a longer duration than an MFCC vector so that AUD borders might not align with the segment borders derived from the MFCCs and to assign an AUD to that word with the biggest overlap. As the distance measure between AUD sequences the length normalized edit distance was used.

As a reference, Table 5.1 also shows the average precision and precision-recall break-even, when using MFCCs and the length normalized cosine distance. It can be seen that the average precision and precision-recall break-even of the AUDs is comparable to the MFCCs and perform slightly better in most of the performance measures. The results show that the AUD extraction algorithm in fact delivers meaningful features which allow to discriminate between words, similar to MFCCs. It also shows that the AUD representation can be slightly more robust, especially in the speaker independent case. It can also be seen that the average precision and precision-recall break-even in the speaker dependent case are higher than in the speaker independent case because of the lower variability in words uttered by the same speaker.

Table 5.2 shows the average duration of the AUDs extracted in the speaker dependent and speaker independent case. For comparison, the average duration of a phoneme as given by the ground truth labels is shown as well. It can be seen that the average length of the AUDs is about half as long as the average length of a phoneme. This indicates that AUDs may not be directly equal to phonemes. They rather model subparts of a phoneme. This can for example result from the context dependency of phonemes which are uttered slightly different depending of the previous or following phoneme.

Table 5.2: Average AUD length in speaker dependent and speaker independent case compared to average phoneme length on ground truth labels extracted from TIDIGITs database

Speaker dependent	Speaker independent	Ground truth
74 ms	62 ms	126 ms

Word Discovery Performance

Next the performance of the word discovery method presented in Section 5.4.2 was evaluated. This was done by comparing the decoding result with the ground-truth word transcription, where the HMMs were assigned that word label which led to the overall smallest word error rate. The algorithm was set to extract $N = 11$ words. The Zipf parameter s was estimated to be $s \approx 0.5$ by the algorithm. For the digit recognition task under investigation, with equal probability of all words, however the true value is $s = 0$. Fixing $s = 0$ did not change the discovery performance considerably, however in

this case, indicating that the difference in parameters in this simple case did not seem to have a significant influence on the result.

Table 5.3 lists the word discovery results for three different HMM setups: The first two columns show the word accuracy (ACC = 1−WER) for the HMM topology of Figure 5.6 with $N_s = 7$ states in the speaker dependent case and $N_s = 11$ states in the speaker independent case. The column heading "Bag-of-AUDs" indicates that all states of a HMM share the same multinomial emission probability, while "State dependent" corresponds to the training of different emission probabilities for each HMM state. As a further comparison, a Bakis left-to-right HMM topology with $N_s = 7$ states in the speaker dependent case and $N_S = 8$ states in the speaker independent case and "State dependent" emission probabilities was also tested.

Table 5.3: Word accuracy (in %) for speaker dependent and speaker independent case and for different setups

Setup	Bag-of-AUDs	State dependent	Bakis left-to-right
Speaker dependent	74.5	62.5	12.5
Speaker independent	57.3	67.9	15.4

Several conclusions can be drawn from these results. First it can be observed that the use of state dependent emission probabilities increases the performance in the speaker independent case from 57.3% to 67.9%, while the performance in the speaker-dependent case suffers compared to using the Bag-of-AUDs model. Not surprisingly, the temporal order of subword units is an important characteristic of a word, which is lost when using a Bag-of-AUDs model. This information can only be taken advantage of if the models can be trained reliably, as the reduced accuracy in the speaker dependent case is most likely attributed to the lack of sufficient training data. This precludes the reliable estimation of state-dependent probabilities in the speaker dependent case.

Second we can observe that a simple Bakis left-to-right HMM model does not give useful results. This indicates that the modeling strength of the Bakis left-to-right model is weak with respect to pattern discovery. The proposed models do incorporate more stringent word length constraints which helps in guiding the training process and discriminating between different words.

Third we can see in the difference between speaker dependent and speaker independent performance that the performance in the speaker dependent case is significantly better than the speaker independent performance. This is expected due to the nature of speech varying from person to person. As the speaker independent case is the most interesting case, because normally data of several different speakers which are not separated will be available, it can be deduced that the modification of the original algorithm performs best in the setup of unsupervised word discovery. Therefore a hierarchical algorithm is presented that is able to perform unsupervised word discovery in the domain of speaker independent input data.

DTW Initialization

The initialization of the parameters in an EM learning algorithm is known to have significant impact on the quality of the learned models. The simplest option is to initialize the parameters uniformly at random, as it was done for the previous experiments reported in Table 5.3.

In the following experiments, with results presented in Table 5.4, the word pronunciation training was initialized with the help of a DTW based pattern discovery algorithm as described in Section 5.4.3. However here the DTW algorithm was used in a speaker independent manner by applying it on the recordings of all speakers at once. In the following, only the results for the speaker independent case are presented, using state-dependent emission probabilities, as this is the most relevant setup.

For DTW, only a subset of the whole database was employed, which contained all 112 speakers but only 7 randomly selected utterances per speaker. This was done to reduce computational time and to demonstrate that only part of the data is already sufficient for initialization. On this dataset DTW was run to find similar segment pairs. A high acceptance threshold was chosen to make sure that the segments found indeed showed high similarity. Doing this the selected segments made up only 3.5% of the whole dataset.

For each segment in a cluster, its start and end points were used to extract an AUD label sequence which was then used for the initial estimation of the emission probabilities of the word model corresponding to the cluster.

The assignment of a word label to a DTW cluster is necessary to carry out an evaluation with respect to word error rate. A cluster was given the word label of that word that was most often represented in the segments of the cluster. This resulted in a high cluster purity of 98%. Note that this was only done for the evaluation. If used in the learning process, this would also resembles a weakly supervised setup where the segments and clusters are discovered in an unsupervised manner and only the class labels have to be assigned using some other knowledge source.

This delivers a label file with a word accuracy of 52%, 1% insertions, 2% substitution and 46% deletions. This is worse then the speaker dependent setup in the previous chapter however in here the cluster purity is more relevant. It can be seen that the discovered clusters/classes contain only little substitutions and insertions which results in the high cluster purity of 98%.

There is, however, an issue with this assignment. The number of clusters obtained from DTW has been set to equal the number of words N for which models are to be developed. However, it turned out that no clusters corresponding to the digits "two", "eight" and "oh" had been found for initialization. Because of the three missing classes only $N - 3 = 8$ clusters were initialized according to the extracted AUD sequences while the remaining three clusters were still initialized using random values.

Table 5.4 shows that the DTW-based initialization improved the word accuracy from 67.9% to 81.9% compared to a random initialization, a reduction in error rate by more than 40%. The word discovery algorithm was also able to discover 11 distinct words in both cases and therefore find additional uninitialized words. As a comparison, the word accuracy is given for perfect initialization, i.e. given the correct transcription in

terms of word labels and the correct word boundaries for the whole database (denoted "ground truth"). It can be seen that the DTW initialization comes close to the ground truth even though it uses much less segments for initialization and even though only 8 of 11 clusters were initialized using discovered segments. The algorithm was also able to successfully recover the 3 remaining randomly initialized words. The Zipf parameter s was estimated to be $s \approx 0.3$ when using DTW initialization and $s \approx 0.1$ when using the ground truth initialization which is closer to the true value $s = 0$ compared to the case without initialization.

Table 5.4: Word accuracy (in %) for different initialization strategies

Random	DTW	Ground truth
67.9	81.9	88.1

The following conclusions can be drawn from these results. First it can be seen that the initialization with ground truth labels increases the word accuracy from 67.9% to 88.1%. This shows that the model can be used to recognize words and that the AU sequences allow discrimination between different words.

Second it can be seen that the initialization with only a couple of segments discovered by the DTW based algorithm, and only 8 of 11 models being initialized, still drastically increases the pattern discovery performance from 67.9% to 88.1%.

Automatic Speech Recognizer Training

In a last step the discovered transcriptions of the word discovery algorithm were used as initialization for a automatic speech recognizer training. Whole-word HMMs with Gaussian Mixture emission probabilities on MFCC input features for each discovered cluster were trained. The discovered label sequence of the word discovery algorithm was used as the transcription for the acoustic model training and the training was again alternated between decoding and model estimation, similar to the AUD training. In each iteration, the decoding result of the previous iteration was used as the new transcription.

Table 5.5 shows the results of the iterative training in terms of word accuracy for the different iterations using random and DTW initialization. Iteration 0 is the result delivered by the word discovery algorithm. Further iterations are the results of the iterative training.

Table 5.5: Word accuracy (in %) for iterative speech recognizer training over iterations and for different initialization strategies on training set.

Iteration	0	1	3	5	7
Random	67.9	80.8	82.9	84.4	84.7
DTW	81.9	96.6	98.4	98.5	98.5

The results show that the discovered transcriptions can in fact be used for an automatic speech recognizer training. The iterative training improves the accuracy from iteration

to iteration. The improvement from the 0th to the 1st iteration is a result of cleaning the transcriptions and especially removing insertions in the recognition result. For the completely unsupervised transcriptions an accuracy of 84.7% is achieved after 7 iterations. For the transcriptions discovered using the DTW based initialization, an accuracy of 98.5% is achieved. This comes close to the accuracy of 99.4% using a completely supervised training.

To test if the trained acoustic models generalize on unseen test data, the acoustic models were used to decode the unseen test set of the TIDIGITS database. Table 5.6 shows the results.

Table 5.6: Word accuracy (in %) using trained acoustic models on test set.

Random	DTW
84.3	98.3

It can be seen that the automatic speech recognizer delivers almost the same results on the test data set as during the discovery step using the training data.

5.4.5 Conclusion

This chapter presented a first statistical model based approach to unsupervised learning. Here, speech is model in a hierarchical manner consisting of acoustic units (phonemes) and then words as sequences of acoustic units (phonemes). First the traditional GMM/HMM based statistic model based approach to speech recognition is presented, followed by the unsupervised learning algorithm, relating it to the model based approach to speech recognition by basically defining the transcriptions of audio segments as additional hidden variables to be discovered.

In a first step, acoustic units are discovered by basically learning GMM/HMM models in an unsupervised way and defining the learned GMM/HMMs as units. To speed up the initialization of the GMM/HMM model learning, a fist unsupervised segmentation and clustering step is performed on the audio segments. The clustering is similar to the graph based clustering used in the previous chapter, but modified by using a kMeans++ based approach and only calculating distances between seed values and segments. This reduces the computational demand and reduced the time required for the clustering step. Again, unsupervised speech recognizer training is performed on discovered acoustic units.

Evaluation of Average Precision shows an improved Average Precision on acoustic units compared to MFCCs only, meaning they are more discriminative. In the speaker dependent setup, the Average Precision is 94.7% for acoustic units versus 92.6% on MFCCs. In the speaker independent setup the values are 64.6% versus 61.7%. This still shows a big gap between speaker dependent and speaker independent modeling. But the evaluation shows that acoustic units can be learned in an unsupervised way such that they are more discriminate than MFCCs.

In the next step, a probabilistic pronunciation lexicon for words and a language model over words are learned. This step resembles the unsupervised word discovery.

In here as the probabilistic pronunciation lexicon an HMM over the discrete symbol sequence of acoustic units is used. Different HMM structures are evaluated, where an HMM with state dependent emission distributions delivers the best performance in the speaker independent case, with 67.9% word accuracy vs. 57.3% word accuracy with state independent emission distributions. For a speaker dependent setup, the state independent emission distributions result in a higher word accuracy of 74.5% vs. 62.5% with state dependent emission distributions. For the unsupervised setting, the speaker independent setup is more relevant though, since speaker identities are often unknown as well. Overall these results demonstrate the suitability of the presented modeling approach. Additionally they show that the proposed state dependent modeling significantly improves results.

Finally, the proposed DTW based initialization within the speaker independent setup demonstrates that combining both approaches significantly improves the results. While both approaches separately have a lower Word accuracy of 67.9% for the model based approach and 52% for the DTW based approach, the combined approach achieves a word accuracy of 81.9%. While it is known that EM algorithms can suffer from bad initialization, the proposed DTW based initialization delivers a very useful initialization for the EM based word discovery step. The DTW based approach is able to discover longer sequences more consistently by directly taking the speech signal into account but leaves out many not too similar looking sequences. This complements the probabilistic pronunciation lexicon approach since it is more flexibly in modeling variations.

Again an iterative speech recognizer training was performed, boosting the unsupervised discovery performance from 81.9% word accuracy to further 98.5%. This is very close to a supervised training.

Overall this chapter introduced a first model based approach and the evaluations demonstrated its suitability in a small vocabulary task in a speaker independent setting. A large vocabulary case is considered in the next chapter.

6 Unsupervised word segmentation with n-gram Language Models

As an alternative to the approach presented in the previous section, which used a probabilistic pronunciation lexicon on the single best unit sequence, word segmentation can also be applied to lattices of unit or phoneme sequences using a deterministic pronunciation lexicon and n-gram language models to leverage contextual information in terms of word and phoneme context. Assuming a deterministic pronunciation lexicon means that each distinct phoneme sequence is assigned to a separate word and no word can have more than one pronunciation.

A deterministic pronunciation lexicon might seem restrictive and in fact it has the drawback of not being able to cope with pronunciation variants, spelling errors and recognition errors. On the other hand this reduces the computational costs, compared to the previously described algorithm. Due to the probabilistic pronunciation lexicon in the previous algorithm, the number of possible pronunciation variants for a single word can be large. This is avoided by having a deterministic pronunciation lexicon. Instead the algorithm described in this chapter uses multiple phone sequence hypotheses per speech utterance as input and tries to simultaneously find the best phone sequence hypotheses and a segmentation of it into words, while previously only the single best AU sequence hypotheses was used.

If the input to the segmentation algorithm includes more than one phoneme sequence hypothesis per speech recording and if among these hypotheses the correct one is present, a deterministic pronunciation lexicon is a reasonable assumption. In this case minor spelling errors and recognition errors could be corrected by having the correct pronunciation present in one of the phoneme sequence hypotheses and extracting it due to a higher probability after applying a language model, like the nested hierarchical Pitman-Yor language model used in the algorithm described in this chapter. This also enables to extract a better phoneme sequence hypothesis as compared to a purely acoustic model based scoring, from the input lattice, by applying the learned phoneme and word language models as well as the learned lexicon.

On the other hand, pronunciation variants and spelling errors where no “correct” pronunciation exists amongst the phoneme sequence hypotheses will still be mapped to different words. These could later be grouped together by some clustering algorithm. If different words map to the same pronunciation, they will be indifferentiable by their acoustics though. They could only be disambiguated by their word context. However

clustering and disambiguation by context will not be considered in this chapter. Here, a major focus is laid on grouping together similar sounding speech segments.

The algorithm described in the following resembles this unsupervised language model and pronunciation lexicon learning. It was first introduced in [MYU09] for the segmentation of error free character or symbol sequences. The algorithm described in this chapter is based on [Neu+12], which mainly aims at unsupervised learning of the nested hierarchical Pitman-Yor language model for improving decoding results, when a language model is unknown. This algorithm was further extended in [Hey+14] to a sequential algorithm, first extracting the single best phoneme sequence using a phoneme based hierarchical Pitman-Yor language model with word end symbols and then applying word segmentation with the nested hierarchical Pitman-Yor language model, where both language models are learned on the word segmentation result, loosely coupling them.

The implementation described in this chapter brings the ideas in [Neu+12] and [Hey+14] back together, using only a single nested hierarchical Pitman-Yor language model for word segmentation over lattices. While the first only uses an approximate WFST realization, resulting in an oversegmentation, the second requires the learning of two language models and performs word segmentation only on the single best sequence extracted using the learned phoneme language model.

The main focus of this chapter is on the unsupervised word segmentation. For unsupervised acoustic unit or phoneme learning, an existing state of the art algorithm [Ond+17] will be used. It is a variant of the algorithm described in the previous chapter and also learns HMMs as AUs however it follows a more integrated learning approach and is therefore more suitable for larger corpora. In addition this algorithm will be improved in this work by introducing unsupervised feature vector transformation for more robust feature vectors, further improving the AU discovery performance.

6.1 Unsupervised word segmentation from symbol sequences

To give an intuitive introduction, in this section it is first assumed that each spoken utterance is represented as one single phoneme sequence. The phoneme sequence can be the result of decoding a speech recording into the first best phoneme sequence or when having character or symbol sequences without spaces, as in the Chinese writing systems. In the following section the algorithm is then extended to work with multiple phoneme sequence hypotheses.

The NHPYLM [MYU09], which will be described in the next section, is applied as an n-gram language model to the given phoneme sequences and calculates the probabilities of word sequences according to Equation (5.18). The task given here is as follows: given a corpus of unsegmented phoneme sequences, segment the input into words. This will deliver a word list for that corpus, ultimately resulting in unsupervised language acquisition. When doing so in an unsupervised setup, the algorithm is faced with a chicken-and-egg problem: it needs a word list and a language model (i.e., probabilities of

words and word sequences) to do the segmentation. However, to estimate the language model, it needs to have a segmentation. Therefore the optimization problem is to jointly optimize the word sequence and the language model. In this model, the hidden variables to be discovered are the language model, the words and the deterministic pronunciation lexicon.

The optimization problem can be formulated analog to Equation (5.1) by including the language model into the decision rule and replacing the observations with the phoneme sequence. Assuming one phoneme sequence per speech recording results in the following decision rule:

$$\hat{w}_{1:L} = \underset{w_{1:L}}{\operatorname{argmax}} \Pr\left(w_{1:L}|x_{1:N}\right) \tag{6.1}$$

$$= \underset{w_{1:L}}{\operatorname{argmax}} \sum_{\{G\}} p\left(w_{1:L}, G|x_{1:N}\right). \tag{6.2}$$

Since the marginalization in Equation (6.2) over the language model is in general intractable and since the most probable language model is also of interest, the decision rule can be modified further to include the decision for the most probable language model as well:

$$\hat{w}_{1:L}, \hat{G} = \underset{w_{1:L}, G}{\operatorname{argmax}} p\left(w_{1:L}, G|x_{1:N}\right). \tag{6.3}$$

Instead of a single speech recording and therefore a single phoneme sequence, the optimization is carried out over a set of M phoneme sequences $\boldsymbol{X} = \left\{x^1_{1:N^1}, \ldots, x^M_{1:N^M}\right\}$ resulting in a set of M sentences $\boldsymbol{W} = \left\{w^1_{1:L^1}, \ldots, w^M_{1:L^M}\right\}$, where the superscript denotes the sequence or sentence index, resulting in:

$$\hat{\boldsymbol{W}}, \hat{G} = \underset{\boldsymbol{W}, G}{\operatorname{argmax}} p\left(\boldsymbol{W}, G|\boldsymbol{X}\right). \tag{6.4}$$

Determining the joint PDF $p\left(\boldsymbol{W}, G|\boldsymbol{X}\right)$ is still computationally intractable, especially for hierarchical n-gram language models like the NHPYLM. Therefore Gibbs sampling is applied to generate samples from this distribution and get an approximation for the PDF.

In the following, first the NHPYLM language model, which is able to cope with an unknown number of words, is described. Next the Gibbs sampling based optimization algorithm for the problem stated above, followed by a description of the forward-filtering backward-sampling algorithm is introduced. Finally the derivation of the forward and backward variables for the forward-filtering backward-sampling algorithm to sample a word segmentation from a phoneme sequence or phoneme lattice is described.

6.1.1 NHPYLM

The statistical language model employed to solve the task of unsupervised word segmentation must be able to assign a probability to unseen words based on their phoneme

sequence, to handle an a priori unknown number of words (e.g. be nonparametric). Additionally it assigns a probability to already seen words based on their word identity and frequency of occurrence. The model should have three properties, imposed as a prior on the language model:

- n-gram structure applies for the words in the word sequences and the phonemes in the phoneme sequences forming a word.
- The occurrence of words and the phonemes in the phoneme sequences forming a word follow a power law distribution for each n-gram context.
- The probability of an unseen word is proportional to the sequence probability of the phonemes in the phoneme sequence forming the word.

The fist property has been shown to be effective in modeling language data and relies on the fact that predictability is helpful to discover structure in data. It is imposed as a prior on the language model by choosing an n-gram structure for the language model. The second property, known as Zipf's law has been shown to be a reasonable property for natural languages. It is imposed as a prior on the language model by choosing the Pitman-Yor (PY) process to model the probabilities within each n-gram context. The third property is crucial in modeling unseen words and therefore to enable the discovery of new words.

An example for such a nested hierarchical language model is shown in Figure 6.1. The probability of an unseen word is calculated analog to Equation (5.18), using the phoneme sequence forming the word, assuming the first phoneme to be the beginning of the sequence, including a word end symbol and a phoneme language model. The word probability calculated in this way can be interpreted as a word unigram probability. Therefore a word segmentation can be performed solely by using this language model. To extend this model, a word based n-gram language model is stacked on top of the phoneme sequence likelihood, using the phoneme sequence likelihood as a fall back option in case the probability of an unseen word is to be calculated.

This combination conveniently combines the advantages of both modeling approaches. It allows to calculate the probability of unseen words and reinforces already seen words in given contexts. A model which meets all of these requirements is the NHPYLM, first introduced in [MYU09]. This model incorporates two HPYLMs, one for the discovered words and one for the phonemes [Teh06b].

The Pitman-Yor process

The HPYLM is based on the PY process, which is similar to the Dirichlet process, but governed by two parameters, a discount parameter d and a strength parameter θ, as well as a base distribution G_0, which is defined over a probability space $\mathcal{X}$ of symbols (words or phonemes).

Intuitively, the PY process is similar to the Dirichlet process, which again is very similar to the Dirichlet distribution. Instead of a limited number of categories, the

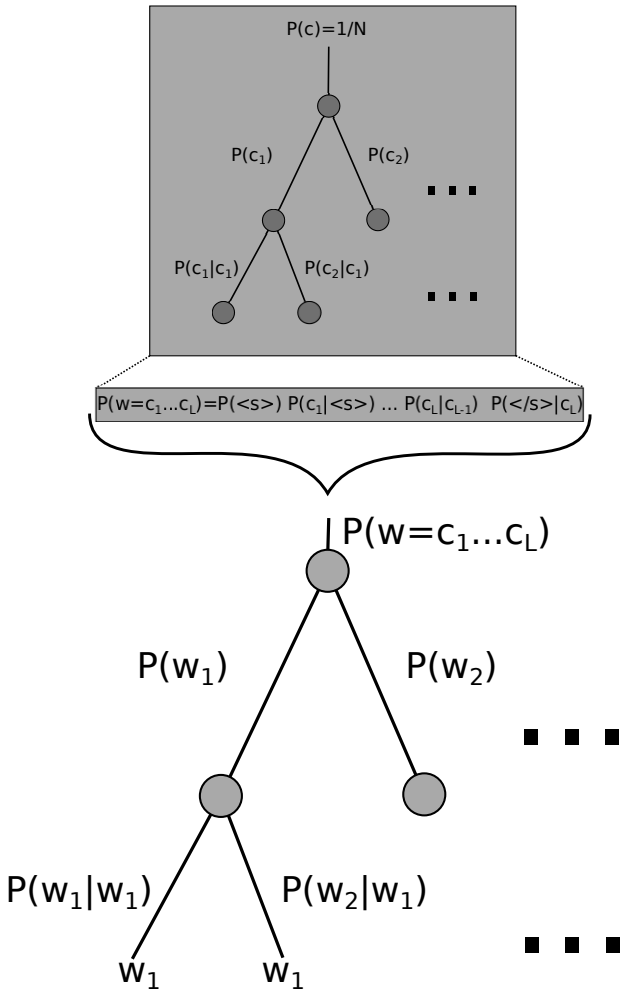


Figure 6.1: Nested hierarchical language model, modeling up to the bigram context. The character language model (gray box) is nested in the word language model. It is used to calculate the word probability as the sequence probability of the characters in the word, with start and end markers added. The base probability of the character model is a uniform distribution.

Dirichlet process models an infinite number of categories. In contrast to the Dirichlet distribution, which is defined over a fixed number of classes, the Dirichlet process and respectively the PY process therefore models a possibly infinite number of classes [Teh10]. The probabilities according to the Dirichlet process can be derived by taking the limit to infinitely many categories in the Dirichlet distribution [Teh10].

This property of having a possible infinite number of classes is used in the NHPYLM to model an a priori unknown number of words. A draw from the PY process G is an infinite discrete probability distribution:

$$G \sim \mathrm{PY}\left(d, \theta, G_0\right). \tag{6.5}$$

The parameter θ controls how similar this drawn distribution is to the base distribution which itself can be seen as a mean distribution of the drawn ones. The parameter d influences the tail behavior of the PY process. While the Dirichlet process has an exponential tail, the PY process has a more power-law like tail, making it more useful for language modeling tasks.

The Hierarchical Pitman-Yor Language Model

The n-gram structure is captured by embedding the above process in a hierarchical structure, as shown in Figure 6.1. At each n-gram level a separate PY process is instantiated for every $(n-1)$-word or $(n-1)$-phoneme context. The base distribution for each PY process at the n-gram level is a $(n-2)$-context specific PY process. One can view the entire structure as a tree, where the root node gives the unigram probability, its children the bigram probability and so forth. So instead of one, there are several distributions, one for each context:

$$G\left(\boldsymbol{u}\right) \sim \mathrm{PY}\left(d_{|\boldsymbol{u}|}, \theta_{|\boldsymbol{u}|}, G\left(\pi\left(\boldsymbol{u}\right)\right)\right). \tag{6.6}$$

The notation $\pi\left(\boldsymbol{u}\right)$ describes the shorter context, e.g. if $\boldsymbol{u} = \left(w_{l-n+1:l-1}\right)$ then $\pi\left(\boldsymbol{u}\right) = \left(w_{l-n+2:l-1}\right)$. The length of the context is denoted by $|\boldsymbol{u}|$ and corresponds to $n-1$ in this example.

The Nested Hierarchical Pitman-Yor Language Model

To cope with unseen words, a phoneme HPYLM is built for the phonemes in the phoneme sequences forming the words and then used to calculate the likelihood of the phoneme sequences forming the words according to Equation (5.18). It serves as the base distribution for the word model, resulting in the aforementioned NHPYLM. Finally the base distribution for the phoneme HPYLM is a uniform distribution over all phonemes.

The NHPYLM has underlying sufficient statistics Σ, which are also called "seating arrangement" in a Chinese-restaurant analogy. The Chinese-restaurant analogy is shown in Figure 6.2. In the Chinese-restaurant analogy, classes are represented as "tables" and samples (draws) belonging to the classes are represented as "customers" sitting at the

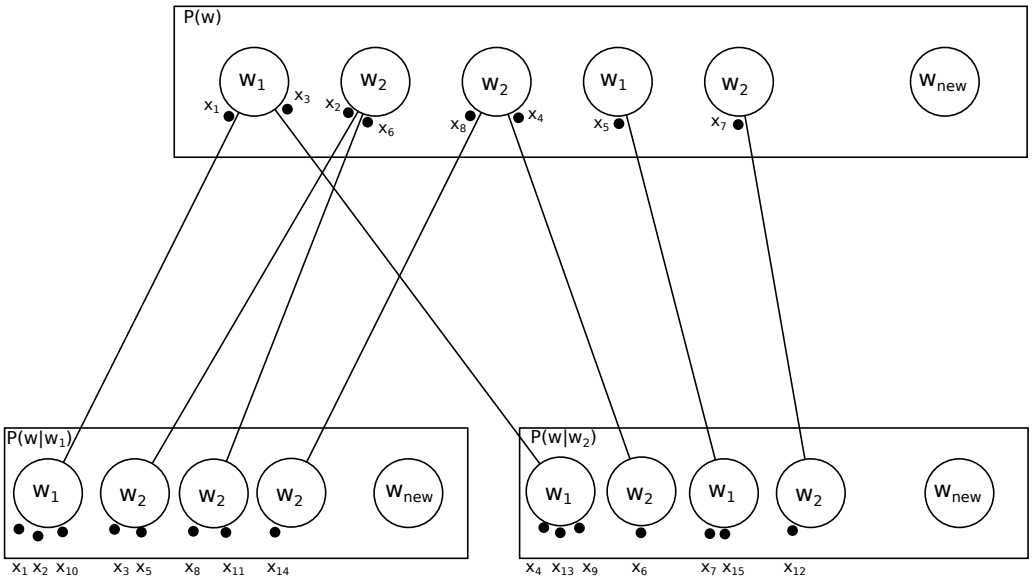


Figure 6.2: Chinese restaurant representation of the HPYLM up to the bigram context. Squares are restaurants, circles are tables, black dots are customers. The top level represents the unigram (where the customers are the tables of the second level). The second level represents the bigram (where the customers are the words). Each table of the bigram level sits as a customer at one corresponding table in the unigram level. This dependence is depicted by the black lines. The word present at each table is written within the table. Each restaurant belongs to a specific context, depicted in the top left corner of each restaurant. Each restaurant computes a probability for the word w_1 and w_2 in its given context. The empty tables with w_{new} depict possible new tables which are assigned with new or existing words when used. The probability of assigning a word to a table is given by the parent restaurant according to Equation (6.7). The probability of assigning a customer to a table is given by Equations (A.8) (A.9). The figure represents an example configuration after seeing/generating the word sequence $w_1w_1w_2w_1w_2w_2w_1w_2w_1w_1w_2w_2w_1w_2w_1$ at the bigram level. The variables x_1 (first word) to x_{15} (15-th word) represent the position of the word in the sequence at the bigram level and unigram level. The variables are depicted in the figure above at the corresponding customers (black dots). For example: The first (x_1) word w_1 is seated at the first table, the third (x_3) word w_2 is seated at the second table and so forth. For simplification, the same notation and visulization is used at the unigram level. This results in the "word" sequence $w_1w_2w_1w_2w_1w_2w_2w_2$ at the unigram level.

corresponding table. The underlying sufficient statistics Σ are used to calculate the predictive probability of a word w given its context $\boldsymbol{u}$:

$$p\left(w|\boldsymbol{u}\right) = \frac{c_{\boldsymbol{u}w} - d_{|\boldsymbol{u}|}t_{\boldsymbol{u}w}}{\theta_{|\boldsymbol{u}|} + c_{\boldsymbol{u}}} + \frac{\theta_{|\boldsymbol{u}|} + d_{|\boldsymbol{u}|}t_{\boldsymbol{u}}}{\theta_{|\boldsymbol{u}|} + c_{\boldsymbol{u}}} p\left(w|\pi\left(\boldsymbol{u}\right)\right), \tag{6.7}$$

with the parameters $\Sigma = \left\{c_{\boldsymbol{u}wk}, t_{\boldsymbol{u}w}, d_{|\boldsymbol{u}|}, \theta_{|\boldsymbol{u}|}\right\}$ for each word w, context $\boldsymbol{u}$, context length $|\boldsymbol{u}|$ and table k. In case of $\pi\left(\boldsymbol{u}\right) = \emptyset$, the empty context, the likelihood of the phoneme sequences forming the words, according to Equation (5.18) is used. For the phoneme language model, the same expression as in Equation (6.7) is used, this time with phones and phoneme contexts but with a uniform distribution over phonemes as the base distribution in case of $\pi\left(\boldsymbol{u}\right) = \emptyset$.

The probability in Equation (6.7) consists of the contributions of each table in Figure 6.2 at a particular n-gram level. The contribution of existing tables, for each table k, is:

$$p\left(w|\boldsymbol{u}, k\right) = \frac{c_{\boldsymbol{u}wk} - d_{|\boldsymbol{u}|}}{\theta_{|\boldsymbol{u}|} + c_{\boldsymbol{u}}} \tag{6.8}$$

and the contribution for each new table k_{new} is:

$$p\left(w|\boldsymbol{u}, k_{\text{new}}\right) = \frac{\theta_{|\boldsymbol{u}|} + d_{|\boldsymbol{u}|}t_{\boldsymbol{u}}}{\theta_{|\boldsymbol{u}|} + c_{\boldsymbol{u}}} p\left(w|\pi\left(\boldsymbol{u}\right)\right). \tag{6.9}$$

Note that for Equation (6.8) a particular table can only be assigned one word type. Therefore k determines the word w. The count for the other words at this table will always be 0.

The Chinese Restaurant Analogy

A Chinese-restaurant analogy may be used to interpret Equation (6.7), because for a fixed context this equation resembles the generative Chinese Restaurant Process (CRP) representation of the PY process, as shown in Figure 6.2: At any time the process has a number of "tables", which can grow infinitely large, and each of these tables has a symbol (word or phoneme) from $\mathcal{X}$ associated with it. A new draw (a "customer") is either "seated" at an existing table and assigned the symbol associated with it, or assigned to a new table. When a new table is selected, the symbol assigned to it is drawn from G_0. This can be interpreted as back-off and smoothing, since a certain amount of the probability mass is moved to the base distribution and the probability of unseen symbols can be calculated from the probability of choosing a new table and the base distribution. The degree of smoothing is controlled by the discount parameter d. If the base distribution is a categorical distribution, multiple tables can be assigned the same label. The probability of "seating" a "customer" at an existing table is proportional to the number of "customers" already sitting at the table. Therefore this process exhibits a "big gets bigger" property and the draws from the distribution G obey the power law and hence incorporate the prior knowledge of Zipf's law. In the limit of infinitely many

samples, the overall process results in draws from the distribution G. The actual sampling equations for the process of drawing a language model are described in Section A.1.3.

The variable $c_{\boldsymbol{u}w} = \sum_{\{k\}} c_{\boldsymbol{u}wk}$ describes the counts of word w in the context $\boldsymbol{u}$. The variable $t_{\boldsymbol{u}w}$ describes how many "tables" are occupied by this word. The parameters $d_{|\boldsymbol{u}|}$ and $\theta_{|\boldsymbol{u}|}$ are the discount and strength parameters of the NHPYLM at the context length $|\boldsymbol{u}|$, which are shared by all contexts of the same length. The variable $c_{\boldsymbol{u}} = \sum_{\{w\}} c_{\boldsymbol{u}w}$ is the count of all words and the variable $t_{\boldsymbol{u}} = \sum_{\{w\}} t_{\boldsymbol{u}w}$ is the number of all tables with the context $\boldsymbol{u}$. In the generative perspective of the Chinese-restaurant analogy, the fraction in the second summand is the probability for assigning a new table for a new draw while the fraction in the first summand is the overall probability of choosing one of the existing tables with word w for a new draw. The term $p\left(w|\pi\left(\boldsymbol{u}\right)\right)$ is the base distribution to draw a new word from. Note: Since the base distributions over words (phoneme sequences) and phonemes are categorical distributions, multiple tables can be assigned the same symbol.

Gibbs Sampling Equations

To estimate the parameters $\left\{c_{\boldsymbol{u}wk}, t_{\boldsymbol{u}w}, d_{|\boldsymbol{u}|}, \theta_{|\boldsymbol{u}|}\right\}$, given a word sequence, Gibbs sampling based optimization is used. The principle of gibbs Sampling is explained in the follow section. Gibs sampling usually consists of resampling from a conditional distribution for a parameter to be sampled given all other parameters. This requires removing and adding words as described in the folowing. Other optimization techniques, for example Bayesian optimization, are usually not tractable for a HPYLM. During the Gibbs sampling based optimization algorithm, the variables $c_{\boldsymbol{u}wk}$ and respectively $t_{\boldsymbol{u}}$ have to be sampled, by sampling the corresponding table to add a given word to and also to remove a given word from. The sampling equations are similar to Equations (6.8) and (6.9) and were first described in [Teh06a]. To add a word, an existing table k is sampled from:

$$p\left(k|\boldsymbol{u}, w\right) = \frac{c_{\boldsymbol{u}wk} - d_{|\boldsymbol{u}|}}{\theta_{|\boldsymbol{u}|} + c_{\boldsymbol{u}w}} \tag{6.10}$$

and a new table k_{new} is sampled from:

$$p\left(k_{\text{new}}|\boldsymbol{u}, w\right) = \frac{\theta_{|\boldsymbol{u}|} + d_{|\boldsymbol{u}|} t_{\boldsymbol{u}}}{\theta_{|\boldsymbol{u}|} + c_{\boldsymbol{u}w}} p\left(w|\pi\left(\boldsymbol{u}\right)\right). \tag{6.11}$$

If a new table is sampled the table is "created" and the word is also recursively added to the parent contexts according to the same equations. This is recursively repeated until the word is added to an existing table or the base distribution is reached. If the base distribution of the word language model is reached, the character sequence of the word is added to the character language model, with word start and end markers added. The same sampling equations are used for the character language model.

Since in Gibbs sampling, a prameter is resampled conditioned on all other parameters, the influence of the current word has to be removed. In this case, removing a word is also done by sampling. Altough one could keep track which word is added to which table

and then simply remove the word, this is not done in here as it would require additional storage. To remove a word, an existing table k is sampled from:

$$p\left(k|\boldsymbol{u}, w\right) = \frac{c_{\boldsymbol{u}wk}}{c_{\boldsymbol{u}w}}. \tag{6.12}$$

If a table is becoming empty, it is removed and the corresponding word is also recursively removed from the parent distribution until the base distribution is reached. If the base distribution of the word language model is reached, the character sequence belonging to the word is also removed from the character language model using the same sampling equation.

The parameters $\theta_{|\boldsymbol{u}|}$ and $d_{|\boldsymbol{u}|}$ are resampled according to the description in Section A.1.1.

Additional length modeling

It has been observed that an n-gram model at the character level results in inadequately low probabilities assigned to long words, because the model has a largely exponential distribution over length [Nag96]. To correct this, a Poisson distribution of the word length is imposed by dividing the likelihood of a character sequence of length l, $P(c_1, \cdots, c_l)$ at the base distribution, by the probability of a word length l according to the language model and multiplying it with the probability of a word length l according to the Poisson distribution [MYU09]:

$$P(c_1, \cdots, c_l) \leftarrow P(c_1, \cdots, c_l)\frac{P_p(l;\lambda)}{P(l)}, \tag{6.13}$$

where the Poisson distribution is given by

$$P_p(l;\lambda) = e^{-\lambda}\,\frac{\lambda^k}{l!}. \tag{6.14}$$

The distribution $P(l)$ is estimated using sampled words from the language model. The Poisson parameter λ is estimated after each iteration of the word segmentation algorithm on the segmentation result according to A.1.2.

6.1.2 Gibbs Sampling based Optimization Algorithm

Gibbs sampling [Met+53], [CG92] is an iterative process to generate samples from a joint distribution of variables by sampling from the marginal distribution of each variable given all other variables. Gibbs sampling is applied in a sequential manner, starting by removing the contribution of one variable and sampling a new value for this one variable given all the others, then replacing the value of this variable by the newly sampled value and continuing with the next variable given all other variables including the previously resampled ones. This is repeated until all variables are resampled. The value of all variables at that point are then taken as one sample from the joint distribution. The process is repeated, until enough samples are obtained to approximate the joint

distribution. Gibbs sampling is therefore a Markov Chain Monte Carlo (MCMC) method, where the above sampling steps construct a Markov chain that has the joint distribution as its equilibrium distribution.

A variant of the traditional Gibbs sampling is called blocked Gibbs sampling, meaning that a whole set of variables is resampled instead of just a single variable. Note that the blocked Gibbs sampling also remotely resembles a batch training as it is often used in Stochastic Gradient Descent (SGD) based optimization algorithms. After applying the model to input data, the model is updated with the result. In contrast to the SGD based optimization, in Gibbs sampling the contribution of the variables to be resampled are removed from the model and the Gibbs sampling is also moving towards a maximum, but generates samples from the joint distribution, where samples from around the maximum are naturally drawn more often.

An intuitive example for Gibbs sampling on a bivariate correlated Gaussian distribution can be found in Section A.2.

While the example of the bivariate Gaussian is simple and intuitive, Gibbs sampling is often applied in cases where the distributions become more complex and when the joint distribution becomes complicated to calculate or is computational intractable. This for example is the case with the unsupervised word segmentation algorithm.

Gibbs sampling for word segmentation and language model estimation

For the word segmentation, a blocked Gibbs Sampling approach is chosen. The set of variables to be resampled together is the segmentation of a whole sentence. The segmentation for a whole sentence is therefore resampled before turning to the remaining variables. The following conditional PMF and conditional PDF are used to generate samples from the joint distribution $p\left(\boldsymbol{W}, G|\boldsymbol{X}\right)$:

$$\Pr\left(w_{1:L^m}^m|G, \boldsymbol{X}, \boldsymbol{W} \setminus w_{1:L^m}^m\right) \approx \Pr\left(w_{1:L^m}^m|G^{-w_{1:L^m}^m}, x_{1:N^m}^m\right), \tag{6.15}$$

$$p\left(G^{w_{1:L^m}^m}|G^{-w_{1:L^m}^m}, \boldsymbol{X}, \boldsymbol{W}\right) \approx p\left(G^{w_{1:L^m}^m}|G^{-w_{1:L^m}^m}, w_{1:L^m}^m\right). \tag{6.16}$$

The notation $G^{-w_{1:L^m}^m}$ means "language model with the contribution of the variables in the superscript removed", while $G^{w_{1:L^m}^m}$ means "only the language model part influenced by the variables in the superscript". In case of the language model these are the variables defining the seating arrangement of the words in the language model, as explained later.

In Equation (6.15) the m-th word sequence is resampled and therefore removed from the conditioning, indicated as $\boldsymbol{W} \setminus w_{1:L^m}^m$. The dependence on the remaining word sequences is also removed because when conditioned on the language model, the probability of the m-th word sequence is independent of the remaining word sequences. Removing the m-th word sequence from the set of word sequences is equivalent to first removing its contribution from the language model and then applying the updated language model, therefore the language model is modified by removing the contribution of the m-th word sequence from it, as described later. Also, the m-th word sequence only depends on the m-th phoneme sequence, because it is assumed that the sentences, given the language

model, are statistically independent of each other and all other phoneme sequences. Therefore the dependence on all other phoneme sequences is removed as well.

Equation (6.15) is similar to the standard decoding rule to decode an observation sequence, in this case the phoneme sequence, into a word sequence. Therefore it seems as if a forward-backward algorithm or Viterbi algorithm could be used to update the language model or estimate the word sequence. Since in the problem at hand, the language model needs to be learned simultaneously, the forward-backward algorithm and the Viterbi algorithm tend to get stuck in local minima, especially if the language model does not yet represent the probabilities of the desired word distribution correctly. Therefore, here, Gibbs sampling is used to sample a word sequence. More specifically, a forward-filtering backward-sampling approach, as described later, will be applied.

In Equation (6.16) the language model is resampled. More precisely, those variables in the language model which correspond to the previously resampled sentence, given the part of the language model which has the contribution of this sentence before the resampling removed, are resampled. The dependence on the phoneme sequences and the remaining sentences is removed, since the variables of the language model which are to be resampled only depend on the language model, with the contribution of the current sentence, before its resampling, removed and only the current sentence. In here Gibbs sampling is used again to sample the variables defining the language model since for the NHPYLM no tractable analytical solution for the parameter update is known. The CRP based representation of the language model delivers convenient equations to resample the corresponding variables, as described later.

The algorithm described here resembles an EM algorithm. Equation (6.15) is similar to an E-step, where a hard decision for a particular word sequence is taken. Equation (6.16) is similar to an M-step, where the model parameters, in this case the language model, are updated. This can also be visualized in an intuitive way: the language model is trained on all sentences, except the one to be segmented. Then the sentence is segmented and the language model updated with the result. For the next iteration another sentence is chosen and segmented. The whole process is repeated until convergence is reached, meaning that the segmentation and language model do not change anymore.

Since here the goal is to segment a set of phoneme sequences into word sequences and at the same time learn the language model, the actual learning algorithm can be described as follows: Given an unsegmented phoneme sequence at the input, a segmentation into words is carried out by the iterative Gibbs sampling based learning process described above, alternating between drawing a segmentation for the phoneme sequence using the estimated NHPYLM to calculate the segmentation likelihoods and reestimating the NHPYLM parameters given the drawn segmentation. Starting with an "empty" NHPYLM, a phoneme sequence of the corpus is chosen at random and its contribution from the NHPYLM is removed if its contribution has been added to the NHPYLM before. Then this NHPYLM is employed to draw a new segmentation for that phoneme sequence. Finally the contribution of the newly sampled segmentation is added to the NHPYLM again. This process is repeated until a stable segmentation is found or enough samples are obtained.

Finally the segmentation and language model with the highest joint probability is of

interest. Since combining different samples of sentences and language models is not a straight forward task and only those samples with the highest probability are of interest to solve the maximization problem, only a single sample for the set of sentences and the language model is used to approximate the maximum. This can either be done by taking the sample with the highest probability after dropping the samples of the first iterations, a so called burn-in period, or by taking the final sample after a fixed number of iterations. In the actual implementation a small number of final Viterbi decoding iterations is used to derive the most probable segmentation, while the language model is always updated using Gibbs sampling.

6.1.3 Forward-Filtering Backward-Sampling Algorithm

The sampling of a segmentation according to Equation (6.15) is achieved by the forward-filtering backward-sampling algorithm [Frü94], [CK96]. In contrast to the forward-backward algorithm, which aims at calculating state posterior probabilities and the Viterbi algorithm, which aims at finding the sequence with the highest probability, the forward-filtering backward-sampling algorithm aims at generating a sample from the posterior distribution of sequences, given the observations. Note that here the term "state q_t" is used as an abstract term which will later be replaced by the length of a word or the word itself, depending on the actual setup. The state "state q_t" will be separately described for different modelling approached in the following sections. In this section a generic description of the forward-filtering backward-sampling algorithm will be given.

The forward-filtering backward-sampling algorithm works as follows: A state sequence is sampled, starting with the final state and then iteratively going backwards, sampling the current state according to the posterior probabilities of the current states, given the already sampled states and all observations up to the current state. This is repeated until the beginning of the sequence is reached.

For the forward-filtering backward sampling algorithm, first, the forward variable $\alpha[t][q_t]$, the probability of being in the current state q_t and all observations $\boldsymbol{o}_{1:t}$ up to the current observation are calculated:

$$\alpha[t][q_t] = p\left(\boldsymbol{o}_{1:t}, q_t\right). \tag{6.17}$$

Second, a backward variable $\beta[t][q_t]$, the probability of the future states, given the current state is calculated:

$$\beta[t][q_t] = \Pr\left(q_{t+1:T}|q_t\right). \tag{6.18}$$

Third, the posterior probability $\gamma[t][q_t]$ of the current state, given the already sampled future states and all observations up to the current state is calculated:

$$\gamma[t][q_t] = \Pr\left(q_t|\boldsymbol{o}_{1:t}, q_{t+1:T}\right) = \frac{\alpha[t][q_t]\beta[t][q_t]}{\sum_{\{q'_t\}} \alpha[t][q'_t]\beta[t][q'_t]}. \tag{6.19}$$

This differs from the forward-backward algorithm in the sense that the backward variable considers future states instead of future observations. For simplicity, the future state

sequence is not included in the indexes of the backward variable, since for this description it is assumed that the backward variable is only calculated for an already sampled future state sequence. Finally the current state q_t is sampled according to the posterior probabilities $\gamma[t][q_t]$:

$$q_t \sim \gamma[t][q_t]. \tag{6.20}$$

The actual definition of the forward and backward variables as well as the observations and states depends on the explicit realization of the underlying distributions. In the following sections, different variants of realizations are described, starting with a simple model where a deterministic observation model is given and the state only describes the length of the word. In later sections the realization is extended to a probabilistic observation model where the state describes the underlying representation in terms of HMM state sequences or phoneme sequences and the length of the word in terms of the number of observations.

6.2 Unsupervised Word Segmentation of Phoneme Sequences

With the help of Equation (5.18) and Equation (6.7), a probability can be assigned to any word sequence, derived by segmenting a phoneme sequence. The sampling of a segmentation can be efficiently accomplished with the forward-filtering backward-sampling algorithm, according to [MYU09], as described in the previous section.

6.2.1 Contribution

Although the general algorithm for word segmentation of error free character sequences has already been proposed in [MYU09], this work revisits the derivation of the forward-filtering backward-sampling equations in a more formal way compared to the one found in [MYU09]. In the following subsections Section 6.2.2 and Section 6.2.3 the derivations are revisited for a bigram and unigram word language model.

6.2.2 Bigram Word language model

The forward-filtering backward-sampling algorithm previously described is applied for a bigram word language model as follows: To sample a word from a phoneme sequence, first the probability of this phoneme sequence with all possible word segmentations is calculated (forward-filtering step). Then, going backwards, the final word is first assumed to be the sentence end symbol "</s>" and the previous word, given the final word, is sampled (backward-sampling step). Then the newly sampled word is taken as the final word and the previous word is again sampled. This is repeated until the beginning of the phoneme sequence is reached. Since a bigram language model is assumed, the

probability of the final word given the previous word is used as the backward variable and the previous word is drawn from the probability

$$x_{t-k-j+1:t-k} \sim \underbrace{\Pr\left(x_{t-k+1:t}|x_{t-k-j+1:t-k}\right)}_{\beta[t-k][j]} \underbrace{\Pr\left(x_{1:t-k-j}, x_{t-k-j+1:t-k}\right)}_{\alpha[t-k][j]}. \tag{6.21}$$

Here, t is the length of the phoneme sequence including the final word. The length of the final word is denoted by k and the length of the word to be sampled (the previous word) is denoted by j. The lengths t and k are fixed in this case and the length j is sampled. Note that in this case with the length j the state q_t in the forward-filtering backward-sampling algorithm is determined by the previous word $x_{t-k-j+1:t-k}$. For simplicity the indices t, k and j are used in here and since a bigram model is assumed, Equation (6.18) is defined as $\Pr\left(x_{t-k+1:t}|x_{t-k-j+1:t-k}\right)$ written as $\beta[t-k][j]$. This first probability mass function denotes the bigram probability and is calculated using the NHPYLM, therefore giving the backward variable. The second probability mass function is the probability of the phoneme sequence of length $t-k$ over all segmentations with the final j characters being a word, giving the forward variable.

The forward probability $\Pr\left(x_{t-k-j+1:t-k}, x_{1:t-k-j}\right)$ can be efficiently calculated using a forward algorithm. Let $\alpha[t][k]$ denote the forward probability of the string $x_{1:t}$, with the last k phonemes being a word. The forward recursion can be developed as follows:

$$\begin{aligned}
\alpha[t][k] &= \Pr\left(x_{1:t-k}, x_{t-k+1:t}\right) && (6.22)\\
&= \sum_{j=1}^{t-k} \Pr\left(x_{1:t-k-j}, x_{t-k-j+1:t-k}, x_{t-k+1:t}\right) && (6.23)\\
&= \sum_{j=1}^{t-k} \Pr\left(x_{t-k+1:t}|x_{1:t-k-j}, x_{t-k-j+1:t-k}\right) \Pr\left(x_{1:t-k-j}, x_{t-k-j+1:t-k}\right) && (6.24)\\
&\approx \sum_{j=1}^{t-k} \Pr\left(x_{t-k+1:t}|x_{t-k-j+1:t-k}\right) \Pr\left(x_{1:t-k-j}, x_{t-k-j+1:t-k}\right) && (6.25)\\
&= \sum_{j=1}^{t-k} \Pr\left(x_{t-k+1:t}|x_{t-k-j+1:t-k}\right) \alpha[t-k][j]. && (6.26)
\end{aligned}$$

For $k = t$ the recursion is initialized with $\alpha[t][t] = \Pr\left(x_{1:t}|\text{<s>}\right)$, where "<s>" denotes the sentence start symbol. In Equation (6.22) $x_{1:t-k}$ denotes the phoneme sequence of length $t-k$ and all its possible segmentations, while $x_{t-k+1:t}$ denotes the phoneme sequence of length k being a word. The summation in Equation (6.23) marginalizes over all previous words, therefore $x_{t-k+1:t}$ is a word as well, while the remaining variables are used as in Equation (6.22). For Equation (6.25) Bayes rule and the bigram assumption are applied. Due to the bigram assumption, the conditional PMF only depends on the previous word and not the remaining segmentations of the phoneme sequence. The phoneme sequences in Equation (6.26) are mapped to words by a deterministic pronunciation lexicon. Therefore each distinct phoneme sequence is mapped to a separate word. For the PMF in Equation (6.26) the predictive probability according to the NHPYLM from

Equation (6.7) is used to calculate the probability of this word in the given context, while the word and context are assumed to be composed of the phoneme sequence representing it.

6.2.3 Unigram Word language model

A similar forward-filtering backward-sampling algorithm can be derived for the unigram case. In the backward-sampling step the previous word of length k is sampled from:

$$x_{t-k+1:t} \sim \underbrace{\Pr\left(x_{t-k+1:t}\right) \Pr\left(x_{1:t-k}\right)}_{\alpha[t][k]}. \tag{6.27}$$

Note that in this case the length k corresponds to the state q_t in the forward-filtering backward-sampling algorithm. Here, the first PMF denotes the unigram probability and is calculated using the NHPYLM. The second PMF is the probability of the phoneme sequence of length $t-k$ over all segmentations. The product of both probabilities results in the forward variable. Due to the unigram assumption, the backward variable is independent of the current word and is therefore omitted. The forward probability $\Pr\left(x_{1:t-k}\right)$ can again be efficiently calculated using a forward algorithm. Let $\alpha[t]$ denote the forward probability of the string $x_{1:t}$. The forward recursion can be developed as follows:

$$\begin{aligned}
\alpha[t] &= \Pr\left(x_{1:t}\right) && (6.28)\\
&= \sum_{k=1}^{t} \Pr\left(x_{1:t-k}, x_{t-k+1:t}\right) && (6.29)\\
&= \sum_{k=1}^{t} \Pr\left(x_{t-k+1:t} | x_{1:t-k}\right) \Pr\left(x_{1:t-k}\right) && (6.30)\\
&\approx \sum_{k=1}^{t} \Pr\left(x_{t-k+1:t}\right) \Pr\left(x_{1:t-k}\right) && (6.31)\\
&= \sum_{k=1}^{t} \Pr\left(x_{t-k+1:t}\right) \alpha[t-k]. && (6.32)
\end{aligned}$$

Here, for $t = 0$ the recursion is initialized with $\alpha[0] = 1$. For Equation (6.31) Bayes rule and the unigram assumption are applied. Due to the unigram assumption, the conditional PMF does not depend on any previous word and become a simple PMF. The meaning of the variables and the forward-filtering backward-sampling procedure are the same as for the bigram, with the difference that no sentence start and sentence end symbol are needed due to the unigram assumption. Equation (6.32) can be nicely interpreted to see the meaning of "probability over all segmentations": Up to the considered sequence of length t, all possible previous words are taken and multiplied with the corresponding forward variable. Expanding the recursion and sum, it can be seen that every possible segmentation of the phoneme sequence is considered, where the probability of each segmented phoneme sequence is the product of the individual word probabilities and therefore a sum over all possible segmentations of the phoneme sequence is calculated.

6.3 Word Segmentation from Speech Recordings

For the given problem, unsupervised learning needs to be performed from acoustic feature vectors, the observations derived from speech recordings. In the previous section, the segmentation algorithm is derived for one single phoneme sequence. With the help of Equation (5.12) a probability can be assigned to any observation sequence, given a phoneme sequence. Combining this with Equation (5.18) and Equation (6.7), a probability for each word sequence, considering the observations, can be calculated. The main difference to the previous optimization problem over a single phoneme sequence is that instead of one single phoneme sequence, multiple different phoneme sequences are possible for one observation sequence. Therefore the segmentation algorithm has to consider all possible phoneme sequences and find the most probable phoneme sequence, considering the acoustic model and the language model.

Starting from Equation (5.1), considering the underlying phoneme and HMM state sequence as well as the language model, in analogy to Equation (5.4), the optimization problem can be formulated as follows:

$$\hat{w}_{1:L} = \underset{w_{1:L}}{\operatorname{argmax}} \Pr\left(w_{1:L}|\boldsymbol{o}_{1:T}\right) \tag{6.33}$$

$$= \underset{w_{1:L}}{\operatorname{argmax}} \sum_{N=1}^{\infty} \sum_{\{x_{1:N}\}} \sum_{\{q_{1:T}\}} \sum_{\{G\}} \Pr\left(q_{1:T}, x_{1:N}, w_{1:L}, G|\boldsymbol{o}_{1:T}\right). \tag{6.34}$$

To simplify the optimization problem, the language model and the HMM state sequence or the phoneme sequence are included in the maximization. This follows the same idea described in the previous chapter. Instead of marginalizing out the hidden variables, they are included into the optimization problem, since their values are also relevant for the problem at hand. Moreover finding an analytical solutions, especially with the language model marginalized out, is often intractable. This results in the following two optimization problems:

In the first case, the deterministic mapping from the HMM state sequence to the phoneme sequence is exploited, allowing to apply the language model directly to the HMM state sequence:

$$\hat{w}_{1:L}, \hat{q}_{1:T}, \hat{G} = \underset{w_{1:L}, q_{1:T}, G}{\operatorname{argmax}} \Pr\left(q_{1:T}, w_{1:L}, G|\boldsymbol{o}_{1:T}\right). \tag{6.35}$$

Here the fact that multiple HMM state sequences can map to the same phoneme sequences is neglected, and it is assumed that the most probable HMM state sequence corresponds to the most probable phoneme sequence, delivering the most probable word sequence. This case can be used if the phoneme sequence is of interest, for example to update the acoustic model after a segmentation and the HMM state sequence are sampled.

In the second case, the fact that multiple HMM state sequences can map to the same phoneme sequences is taken into account by applying the maximization over the phoneme sequence instead of the HMM state sequence:

$$\hat{w}_{1:L}, \hat{x}_{1:N}, \hat{G} = \underset{w_{1:L}, x_{1:N}, G}{\operatorname{argmax}} \Pr\left(x_{1:N}, w_{1:L}, G|\boldsymbol{o}_{1:T}\right). \tag{6.36}$$

This results in a more complex optimization problem, needing to optimize the phoneme sequence as well as summing over all HMM state sequences during the maximization.

A third case would jointly optimize the phoneme sequence and the HMM state sequence:

$$\hat{w}_{1:L}, \hat{x}_{1:N}, \hat{q}_{1:T}, \hat{G} = \underset{w_{1:L}, x_{1:N}, q_{1:T}, G}{\operatorname{argmax}} \Pr\left(q_{1:T}, x_{1:N}, w_{1:L}, G | \boldsymbol{o}_{1:T}\right). \tag{6.37}$$

This is an extension to the second case and therefore not discussed further. After finding the most probable phoneme sequence, the most probable HMM state sequence can be derived from it in a separate optimization step, given the phoneme sequence and the observations. One option is a so called HMM state alignment derived by a Viterbi decoding applied with the acoustic model of Equation (5.12) or another forward-filtering backward-sampling step over the HMM state sequence.

In this section, it is assumed that a phoneme based acoustic model is already trained and the acoustic model will be kept fixed and not updated. The training can be either done on the target language, another language or even multiple languages. Alternatively the acoustic model of an acoustic unit discovery step can be used. All of these models assume that some basic knowledge about the phonetic structure is given, can be transfered from another phonetically similar language or learned in an unsupervised way.

The main task here is to learn a more coarse structure, the words, for the HMM state sequence or phoneme sequences delivered by the acoustic model. The learning algorithm could also be extended to update the acoustic model by taking the sampled phoneme sequence or HMM state sequences and updating the transition probabilities and emission distributions of the acoustic HMM. Since this increases the computational effort significantly beyond the capacity available for this work, the acoustic model is learned separately from the word segmentation and kept constant.

In the following, first, the learning of a word segmentation directly from HMM state sequences, given the observations, and second, a word segmentation over phoneme sequences, given the observations, is derived.

6.3.1 Contribution

As mentioned in the previous section, the algorithm for error-free character sequences was already proposed in [MYU09]. An algorithm for lattice based segmentation was proposed in [Neu+12] and extended in [Hey+14]. As a contribution, this work revisits a more formal derivation of the forward-filtering backward-sampling equations in the subsections Section 6.3.2 and Section 6.3.3 for segmentation on HMM state sequences and phoneme lattices.

6.3.2 Unsupervised Word Segmentation on HMM States

To give an intuitive description of the word segmentation algorithm from speech recordings, the segmentation is first performed on the HMM state sequence instead of the phoneme sequence. In the following $\boldsymbol{O} = \{\boldsymbol{o}^1_{1:T^1}, \ldots, \boldsymbol{o}^m_{1:T^m}\}$ denotes the set of M observation sequences and $\boldsymbol{Q} = \{q^1_{1:T^1}, \ldots, q^m_{1:T^m}\}$ denotes the set of M HMM state sequences.

Assuming a blocked Gibbs sampling procedure, the word and HMM state sequence for the m-th observation sequence can be sampled from:

$$\Pr\left(w_{1:L^m}^m, q_{1:T^m}^m | G, \boldsymbol{O}, \boldsymbol{Q} \setminus q_{1:T^m}^m, \boldsymbol{W} \setminus w_{1:L^m}^m\right) \approx \Pr\left(w_{1:L^m}^m, q_{1:T^m}^m | G^{-w_{1:L^m}^m}, \boldsymbol{o}_{1:T^m}^m\right). \tag{6.38}$$

Note, that in contrast to Equation (6.15) which considered character or phoneme sequences, here the HMM state sequence sequence is used and the observation sequence is included therefore resulting in the given definition. Here, the number of states and the number of observations is equal. Similar to Equation (6.15), the dependence on the remaining word sequences is removed and the modified language model is applied. Also the m-th HMM state sequence is independent of all the other HMM state sequences and the other observations, given the acoustic model. Therefore the dependence on all the other HMM state sequences and all the other observation sequences, except the m-th observation sequence is removed.

In the backward-sampling step, the length of the previous word and its representation in terms of a HMM state sequence is drawn from:

$$q_{t-k-j+1:t-k} \sim \underbrace{\Pr\left(q_{t-k+1:t} | q_{t-k-j+1:t-k}\right)}_{\beta[t-k][q_{t-k-j+1:t-k}]} \underbrace{p\left(\boldsymbol{o}_{1:t-k}, q_{t-k-j+1:t-k}\right)}_{\alpha[t-k][q_{t-k-j+1:t-k}]}. \tag{6.39}$$

Here, t is the length of the observation sequence including the final word. The length of the final word is denoted by k and its corresponding HMM state sequence as $q_{t-k+1:t}$. The length of the word to be sampled is denoted as j and its corresponding HMM state sequence as $q_{t-k-j+1:t-k}$. The lengths t and k are fixed in this case. In contrast to Equation (6.21) the length j of the previous word and its representation in terms of the HMM state sequence $q_{t-k-j+1:t-k}$ are sampled jointly. Note that in this case the HMM state sequence $q_{t-k-j+1:t-k}$ corresponds to the state q_t in the forward-filtering backward-sampling algorithm. The first PMF denotes the bigram probability and is calculated using the NHPYLM, where the HMM state sequences are mapped to their corresponding phoneme sequence which is then put into Equation (6.7), giving the backward variable. The second PDF is the probability of the observation sequence of length $t-k$ over all segmentations and all HMM state sequences with the final j observations being a word represented by the HMM state sequence $q_{t-k-j+1:t-k}$, giving the forward variable. The backward sampling is initialized by assuming the last word to be the sentence end symbol "</s>" and then iteratively carried out setting the previously sampled word as the final word.

The forward probability $p\left(\boldsymbol{o}_{1:t-k}, q_{t-k-j+1:t-k}\right)$ can be efficiently calculated using a forward algorithm. Let $\alpha[t][q_{t-k+1:t}]$ denote the forward probability of the observation sequence $\boldsymbol{o}_{1:t}$ with the last k observations being a word represented by the HMM state sequence $q_{t-k+1:t}$. The forward recursion can be developed as follows:

$$\alpha[t][q_{t-k+1:t}] = p\left(\boldsymbol{o}_{1:t}, q_{t-k+1:t}\right) \tag{6.40}$$

$$= \sum_{\{q_{1:t-k}\}} p\left(\boldsymbol{o}_{1:t}, q_{1:t-k}, q_{t-k+1:t}\right) \tag{6.41}$$

$$= \sum_{j=1}^{t-k} \sum_{\{q_{1:t-k}\}} p\left(\boldsymbol{o}_{1:t}, q_{1:t-k-j}, q_{t-k-j+1:t-k}, q_{t-k+1:t}\right) \tag{6.42}$$

$$= \sum_{j=1}^{t-k} \sum_{\{q_{1:t-k}\}} p\left(\boldsymbol{o}_{1:t} | q_{1:t-k-j}, q_{t-k-j+1:t-k}, q_{t-k+1:t}\right) \Pr\left(q_{t-k+1:t} | q_{1:t-k-j}, q_{t-k-j+1:t-k}\right) \Pr\left(q_{1:t-k-j}, q_{t-k-j+1:t-k}\right) \tag{6.43}$$

$$\approx \sum_{j=1}^{t-k} \sum_{\{q_{1:t-k}\}} \prod_{\tau=1}^{t} p\left(\boldsymbol{o}_\tau | q_\tau\right) \Pr\left(q_{t-k+1:t} | q_{t-k-j+1:t-k}\right) \Pr\left(q_{1:t-k-j}, q_{t-k-j+1:t-k}\right) \tag{6.44}$$

$$= \prod_{\tau=t-k+1}^{t} p\left(\boldsymbol{o}_\tau | q_\tau\right) \sum_{j=1}^{t-k} \sum_{\left\{q_{t-k-j+1:t-k}\right\}} \Pr\left(q_{t-k+1:t} | q_{t-k-j+1:t-k}\right) \sum_{\left\{q_{1:t-k-j}\right\}} p\left(\boldsymbol{o}_{1:t-k}, q_{1:t-k-j}, q_{t-k-j+1:t-k}\right) \tag{6.45}$$

$$= \prod_{\tau=t-k+1}^{t} p\left(\boldsymbol{o}_\tau | q_\tau\right) \sum_{j=1}^{t-k} \sum_{\left\{q_{t-k-j+1:t-k}\right\}} \Pr\left(q_{t-k+1:t} | q_{t-k-j+1:t-k}\right) \alpha[t-k][q_{t-k-j+1:t-k}]. \tag{6.46}$$

For $k = t$ the recursion is initialized with $\alpha[t][q_{1:t}] = \prod_{\tau=1}^{t} p\left(\boldsymbol{o}_\tau | q_\tau\right) \Pr\left(q_{1:t} | \text{<s>}\right)$, where "<s>" denotes the sentence start symbol. For Equation (6.44) the bigram assumption for the word dependence and the conditional independence assumption for the observations emitted by the HMM states is applied. In Equation (6.45) the product is split into the product over the observation sequence of the current word and the previous observation sequence from the beginning up to the current word. The sum over all possible state sequences is also split into a sum over all possible state sequences up to the previous words and all possible state sequences for the previous word. This allows to rearrange the equation to move the product over the observation sequence of the current word to the front, since it is independent of the previous segmentation and the previous state sequences. Also the product over the observation sequence probabilities up to the current word can be recombined with the last PDF. Moving the sum over all possible state sequences up to the previous word in front of the last PDF finally delivers the forward variable including the previous word. Finally the conditional PMF $\Pr\left(q_{t-k+1:t} | q_{t-k-j+1:t-k}\right)$ is calculated using the NHPYLM, the HMM transition probabilities and applying the first-order Markov assumption:

$$\Pr\left(q_{t-k+1:t} | q_{t-k-j+1:t-k}\right) \approx \Pr\left(q_{t-k+1} | q_{t-k+2:t}, q_{t-k-j+1:t-k}\right) \prod_{\tau=t-k+2}^{t} \Pr\left(q_\tau | q_{\tau-1}\right). \tag{6.47}$$

The first probability is the initial state probability. In this case the initial state probability is the probability of the current word formed by the HMM state sequence given the previous word and is calculated by the NHPYLM. The probability is therefore calculated by mapping the current word $q_{t-k+1:t}$ and the previous word $q_{t-k-j+1:t-k}$ to its corresponding phoneme sequence and evaluating the word probability using Equation (6.7).

A similar recursion can be derived for the unigram case, resulting in:

$$q_{t-k+1:t} \sim \underbrace{\prod_{\tau=t-k+1}^{t} p\left(\boldsymbol{o}_\tau|q_\tau\right) \Pr\left(q_{t-k+1:t}\right) p\left(\boldsymbol{o}_{1:t-k}\right)}_{\alpha[t][q_{t-k+1:t}]}, \tag{6.48}$$

$$\alpha[t] = \sum_{k=1}^{t} \sum_{\{q_{t-k+1:t}\}} \prod_{\tau=t-k+1}^{t} p\left(\boldsymbol{o}_\tau|q_\tau\right) \Pr\left(q_{t-k+1:t}\right) \alpha[t-k], \tag{6.49}$$

where the forward variable in Equation (6.48) is calculated using the HMM observation and transition distributions similar to Equation (6.47), while dropping the dependence on the previous word. Due to the unigram assumption, the backward variable is independent of the current word and is therefore omitted. The same equations are used in Equation (6.49). The recursion is initialized with $\alpha[0] = 1$. Note that in this case the HMM state sequence $q_{t-k+1:t}$ corresponds to the state q_t in the forward-filtering backward-sampling algorithm.

6.3.3 Unsupervised Word Segmentation on Phoneme Lattices

Instead of segmenting the HMM state sequence, the segmentation can also be formulated in terms of segmenting a phoneme sequence, given the observation sequence. Segmentation of the HMM state sequence in terms of words, assuming a phoneme based word language model has the disadvantage that many different HMM state sequences map to the same phoneme sequence and the same word. Therefore a segmentation based on the phoneme sequence summarizes all HMM state sequences belonging to a particular phoneme into one single phoneme. Assuming a blocked Gibbs sampling procedure, the word and phoneme sequence for the m-th observation sequence can be sampled from:

$$\Pr\left(w_{1:L^m}^m, x_{1:N^m}^m|G, \boldsymbol{O}, \boldsymbol{X} \setminus x_{1:N^m}^m, \boldsymbol{W} \setminus w_{1:L^m}^m\right) \approx \Pr\left(w_{1:L^m}^m, x_{1:N^m}^m|G^{-w_{1:L^m}^m}, \boldsymbol{o}_{1:T^m}^m\right). \tag{6.50}$$

Note, that in contrast to Equation (6.15) which considered character or phoneme sequences as given, here the observation sequence is included and the phoneme sequence is also sampled, therefore resulting in the given definition. In contrast to the HMM states, where it is assumed that each observation is assigned one state, meaning that the length of the observation sequence equals the length of the HMM state sequence, here, the length of the phoneme sequence does not equal the length of the observation sequence. In Equation (6.50) this is indicated by an observation sequence length of T and a phoneme sequence length of N. Again, the same independence assumptions as in

the previous sections are made, only this time for the phoneme sequences instead of the HMM state sequences.

In the following, the notation $x_{[j:k]}$ is used to indicate the phoneme sequence spanning the time interval, and therefore the observation sequence, from j to k. The phoneme sequence $x_{[j:k]}$ can have an arbitrary length, meaning an arbitrary number of phonemes, from 1 to possibly infinity. It is assumed that the phoneme sequences are non overlapping and that consecutive phoneme sequences follow each other without a gap. Therefore in the backward sampling step, the previous word is drawn from:

$$x_{[t-k-j+1:t-k]} \sim \underbrace{\Pr\left(x_{[t-k+1:t]}|x_{[t-k-j+1:t-k]}\right)}_{\beta[t-k][j][x_{[t-k-j+1:t-k]}]} \underbrace{p\left(\boldsymbol{o}_{1:t-k}, x_{[t-k-j+1:t-k]}\right)}_{\alpha[t-k][j][x_{[t-k-j+1:t-k]}]}. \tag{6.51}$$

In contrast to the segmentation of HMM state sequences in the previous section, the forward and backward variables also have to include the length of the word to be sampled in terms of observations, since this length is no longer equivalent to the number of states. Here, the length j of the word to be sampled and its representation in terms of the phoneme sequence $x_{[t-k-j+1:t-k]}$ are jointly sampled. The remaining variables are used in the same way as in the previous section. Note that in this case the phoneme sequence $x_{[t-k-j+1:t-k]}$ corresponds to the state q_t in the forward-filtering backward-sampling algorithm. The first PMF denotes the bigram probability and is calculated using the NHPYLM for the given phoneme sequence, resulting in the backward variable. The second PMF is the probability of the observation sequence of length $t-k$ over all segmentations and phoneme sequences with the final j observations being a word represented by the phoneme sequence spanning the time interval $t-k-j+1$ to $t-k$, giving the forward variable. The backward sampling is again initialized by assuming the last word to be the sentence end symbol "</s>" and then iteratively carried out, setting the previously sampled word as the final word.

The forward probability $p\left(\boldsymbol{o}_{1:t-k}, x_{[t-k-j+1:t-k]}\right)$ can be efficiently calculated using a forward algorithm. Let $\alpha[t][k][x_{[t-k+1:t]}]$ denote the forward probability of the observation sequence $\boldsymbol{o}_{1:t}$ with the last k observations being a word represented by the phoneme sequence $x_{[t-k+1:t]}$. The forward recursion can be developed as follows:

$$\begin{aligned}
\alpha[t][k][x_{[t-k+1:t]}] &= p\left(\boldsymbol{o}_{1:t}, x_{[t-k+1:t]}\right) && (6.52)\\
&= \sum_{\left\{x_{[1:t-k]}\right\}} p\left(\boldsymbol{o}_{1:t}, x_{[1:t-k]}, x_{[t-k+1:t]}\right) && (6.53)\\
&= \sum_{j=1}^{t-k} \sum_{\left\{x_{[1:t-k]}\right\}} p\left(\boldsymbol{o}_{1:t}, x_{[1:t-k-j]}, x_{[t-k-j+1:t-k]}, x_{[t-k+1:t]}\right) && (6.54)\\
&= \sum_{j=1}^{t-k} \sum_{\left\{x_{[1:t-k]}\right\}} p\left(\boldsymbol{o}_{1:t}|x_{[1:t-k-j]}, x_{[t-k-j+1:t-k]}, x_{[t-k+1:t]}\right)\\
&\qquad \Pr\left(x_{[t-k+1:t]}|x_{[1:t-k-j]}, x_{[t-k-j+1:t-k]}\right)
\end{aligned}$$

$$\Pr\left(x_{[1:t-k-j]}, x_{[t-k-j+1:t-k]}\right) \tag{6.55}$$

$$\approx \sum_{j=1}^{t-k} \sum_{\left\{x_{[1:t-k]}\right\}} p\left(\boldsymbol{o}_{1:t-k}|x_{[1:t-k-j]}, x_{[t-k-j+1:t-k]}\right) p\left(\boldsymbol{o}_{t-k+1:t}|x_{[t-k+1:t]}\right)$$
$$\Pr\left(x_{[t-k+1:t]}|x_{[t-k-j+1:t-k]}\right) \Pr\left(x_{[1:t-k-j]}, x_{[t-k-j+1:t-k]}\right) \tag{6.56}$$

$$= p\left(\boldsymbol{o}_{t-k+1:t}|x_{[t-k+1:t]}\right) \sum_{j=1}^{t-k} \sum_{\left\{x_{[t-k-j+1:t-k]}\right\}} \Pr\left(x_{[t-k+1:t]}|x_{[t-k-j+1:t-k]}\right)$$
$$\sum_{\left\{x_{[1:t-k-j]}\right\}} p\left(\boldsymbol{o}_{1:t-k}, x_{[1:t-k-j]}, x_{[t-k-j+1:t-k]}\right) \tag{6.57}$$

$$= p\left(\boldsymbol{o}_{t-k+1:t}|x_{[t-k+1:t]}\right) \sum_{j=1}^{t-k} \sum_{\left\{x_{[t-k-j+1:t-k]}\right\}} \Pr\left(x_{[t-k+1:t]}|x_{[t-k-j+1:t-k]}\right)$$
$$\alpha[t-k][j][x_{[t-k-j+1:t-k]}]. \tag{6.58}$$

For $k = t$ the recursion is initialized with $\alpha[t][t][x_{[1:t]}] = p\left(\boldsymbol{o}_{1:t}|x_{[1:t]}\right) \Pr\left(x_{[1:t]}|\text{<s>}\right)$, where "<s>" denotes the sentence start symbol. For Equation (6.56) the bigram assumption for the word dependence and the conditional independence assumption for the observations emitted by the acoustic model over words is applied. It is assumed that an observation sequence spanning a certain time range only depends on the corresponding phoneme sequence spanning the same time range. Therefore the probabilities and sums can be split up and rearranged to arrive at the final recursion for the backward variable. For the second probability mass function in Equation (6.58), the predictive probability according to the NHPYLM from Equation (6.7) is used to calculate the probability of this word in the given context, while the word and the context are assumed to be composed of the phoneme sequence representing it. For the first probability mass function in Equation (6.58) the acoustic model from Equation (5.12) is employed.

A similar recursion can be derived for the unigram case, resulting in:

$$x_{[t-k+1:t]} \sim \underbrace{p\left(\boldsymbol{o}_{t-k+1:t}|x_{[t-k+1:t]}\right) \Pr\left(x_{[t-k+1:t]}\right) p\left(\boldsymbol{o}_{1:t-k}\right)}_{\alpha[t][k][x_{[t-k+1:t]}]}, \tag{6.59}$$

$$\alpha[t] = \sum_{k=1}^{t} \sum_{\left\{x_{[t-k+1:t]}\right\}} p\left(\boldsymbol{o}_{t-k+1:t}|x_{[t-k+1:t]}\right) \Pr\left(x_{[t-k+1:t]}\right) \alpha[t-k], \tag{6.60}$$

where the forward variable in Equation (6.59) is calculated using the acoustic model given by Equation (5.12) and the NHPYLM. Due to the unigram assumption, the backward variable is independent of the current word and is therefore omitted. The same equations are used in Equation (6.60). The recursion is initialized with $\alpha[0] = 1$. Note that in this case the phoneme sequence $x_{[t-k+1:t]}$ corresponds to the state q_t in the forward-filtering backward-sampling algorithm.

Equation (6.58) and Equation (6.60) can be nicely interpreted to see the different contributions of the acoustic model and the language model as well as the sum over all previous segmentations and their possible representations. The acoustic model delivers the probability of the observation sequence given the phoneme sequence forming the current word and therefore can be regarded as a weight for the phoneme sequence to emphasize more probable phoneme sequences. For HMM based acoustic models, conditional independence of a sequence of observations given a phoneme spanning the corresponding sequence is assumed. In general, in the acoustic model, independence of consecutive phonemes is assumed as well. This leads to the effect that the acoustic model contains no information about the sequence of phonemes to weigh more probable phoneme sequences higher. Since the NHPYLM considers the sequential nature of the phonemes, it weighs more probable phoneme sequences, forming a word, higher but in turn does not consider the observations.

Since the acoustic model is assumed to be given and not updated during the learning process, the acoustic model can be applied to all speech recordings at once and the result kept constant for all iterations of the learning process. Therefore the first PDF in Equation (6.58) is computed using a speech recognizer. The result can be conveniently stored and represented as a WFST, transforming from observation sequences to phoneme sequences. The WFST based representation and segmentation algorithm is described in the next chapter.

6.4 WFST based implementation of word segmentation

For the implementation of the word segmentation algorithm described in Chapter 6, Section 6.2 and Section 6.3, a WFST based implementation is chosen. The WFST based implementation simplifies the calculation of the forward variable and subsequently the backward variable. It uses a graph based representation of the observation sequence, pronunciation lexicon and language model. Determining all possible segmentations, mapping them to known and unknown words and weighing them according to the language model is achieved by a composition operation as explained in the following.

A WFST in an automaton with a finite set of states and a finite set of weighted transitions from one of the states to another state. By specifying a discrete input symbol and a discrete output symbol for each transition, a weighted mapping from a sequence of input symbols to a sequence of output symbols is realized. In the case of unsupervised word segmentation, the mapping from phoneme sequences to word sequences and their weighting according to the language model is realized. Additionally, states can be defined as initial and final states, restricting the beginning and end of possible sequences.

Additional weights can be provided for the initial and final states to allow initialization of the weight of a sequence beginning in a certain state and to weight the finalization of a sequence at a certain state.

The WFST is a realization of a Mealy machine [Moh97], where transitions are taken according to the input symbols and outputs are generated according to the output symbols at the taken transitions. A WFST is usually represented as a weighted directed graph, where the states are represented as nodes, also called vertices, and the transitions as arcs, also called edges, with their corresponding input and output symbols and weights.

The use of a WFST based implementation allows for the efficient use of existing libraries like the OpenFST library [All+07] and algorithms for composing of two WFSTs and finding the shortest path in a single WFST. The libraries and algorithms make use of efficient implementations to speed up these algorithms and simplify the implementation. This allows to focus on building the model structure instead of reimplementing existing algorithms. Especially the calculation of the forward variables is optimized by the use of WFSTs since the WFST allows to efficiently generate possible segmentations and corresponding weighting. In the following a simple example for the use of WFSTs in speech recognition is given, followed by the WFST based implementation of the word segmentation algorithm.

6.4.1 WFSTs for speech recognition

For speech recognition, WFSTs are usually used to realize the weighted mapping from a phoneme sequence to a word sequence according to Equation (5.16) and Equation (5.18) or the weighted mapping from HMM state sequences to phoneme sequences as in Equation (5.12). Both can also be combined to realize a mapping from HMM sequences to word sequences. In the following, an example for mapping from phoneme sequences to weighted word sequences is given, since this will also be the application for word segmentation.

Input representation

First the observation sequence, called input sequence here, because it is the input to another WFST, has to be represented as a WFST as well. An explicit input sequence is usually modeled as a Weighted Finite State Automaton (WFSA), with the corresponding symbols of the input sequence at the input and output of its transitions, as depicted in Figure 6.3.

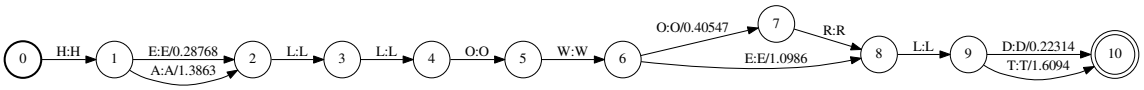


Figure 6.3: Representation for an input sequence with multiple alternative representations

Since the sequence follows a temporal direction forward in time and is of finite length, the representation resembles a Directed Acyclic Graph (DAG). Possible sequences in this

example are “HELLOWORLD”, “HELLOWELT”, “HALLOWELT”, “HALLOWORLD”, “HELLOWELD”, “HALLOWELD”, “HELLOWORLT” and “HALLOWORLT”.

For automatic speech recognition, the WFSA in Figure 6.3 is interpreted as follows: Each node represents a fixed point in time, a time stamp. Nodes don’t need to be uniformly spaced in time. Each arc represents the phoneme emitted between the time stamps spanned by the arc. Multiple possible phonemes between two time stamps are represented by multiple arcs between two nodes. An arc can also span multiple time stamps. A floating point number at an arc represents the weight, usually the negative log likelihood of the corresponding arc.

The input sequence could be the output of a speech recognizer, outputting possible phonemes and the likelihood of the feature sequence between the time stamps, given the phoneme according to Equation (5.12). To reduce the complexity of the resulting graph, unlikely transitions and phoneme sequences are often removed from the graph. The removal of unlikely transitions and phoneme sequences is called pruning.

Lexicon and Language model: Sequence to Sequence Mapping and Weighting

The mapping from phoneme sequences to words according to the pronunciation dictionary and weighting according to Equation (5.16) is realized by a lexicon WFST. This is a mapping from one symbol sequence to another and is usually realized by another WFST, with a different set of symbols at the input and the output, as depicted in Figure 6.4.

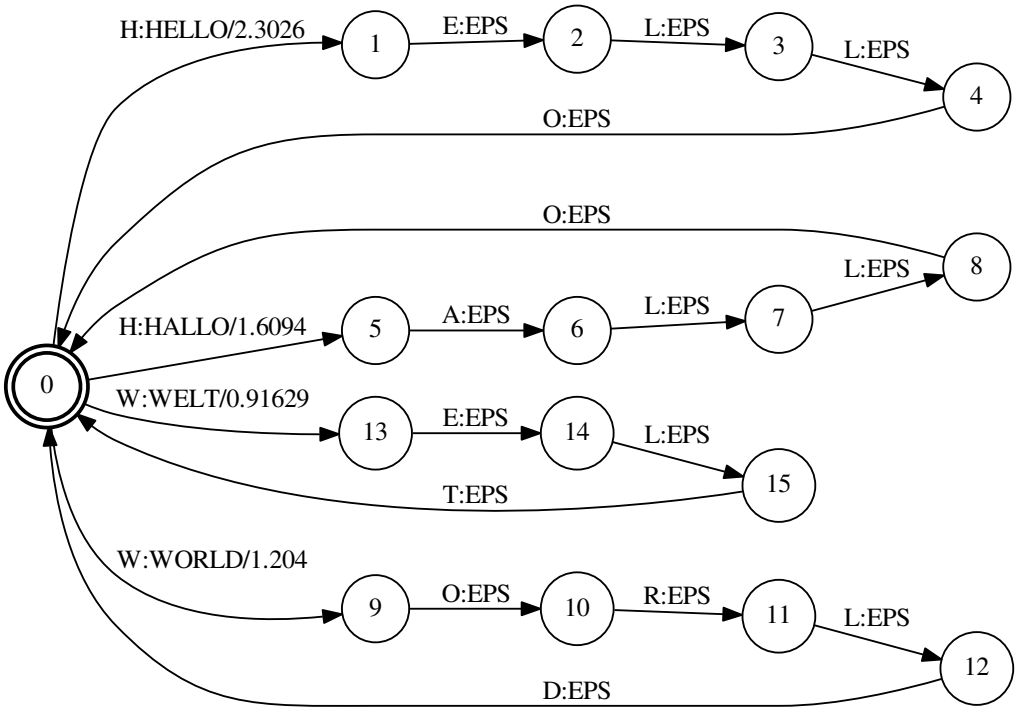


Figure 6.4: Representation of a WFST mapping from input sequences to words

Since an input symbol sequence of arbitrary length needs to be translated by the

WFST, the depicted WFST resembles a Directed Cyclic Graph (DCG). The lexicon rewrites an input sequence of phonemes into an output sequence of words.

Here, the input sequence “H E L L O” is rewritten into the word “HELLO”, “H A L L O” into “HALLO”, “W O R L D” into “WORLD” and “W E L T” into “WELT”. Other sequences cannot be represented by this lexicon and are ignored. The output symbol “EPS” represents a so called “epsilon symbol”, a symbol which stands for “no symbol”. It is used if no symbol is to be emitted when transitioning the arc. This is usually the case when translating a longer low level sequence, like phonemes, into a shorter high level sequence, like words.

This example realizes a deterministic lexicon. For automatic speech recognition, the WFST in Figure 6.4 can also be interpreted as a combination of lexicon and unigram language model. The unigram probability of a word is added to the pronunciation probability or used as the word probability. As the weight, the negative log probability of the word or pronunciation is used in this case.

A language model according to Equation (5.18) can be realized as another WFSA. In case of a unigram language model this only results in adding an additional weight to the pronunciation probabilities. The language model assigns weights to words and word sequences according to Equation (5.18). Since a unigram language model is assumed in this example, the lexicon and the language model are conveniently combined into one WFST in this example. The realization of a higher order language model will be presented in Section 6.4.3.

Composition of Input WFSA and Lexicon WFST

The composition operation is used to combine the input sequence WFSA and the lexicon WFSTs into one WFST. The resulting WFST represents a direct mapping and weighting from the input phoneme sequences to possible output word sequences, as depicted in Figure 6.5.

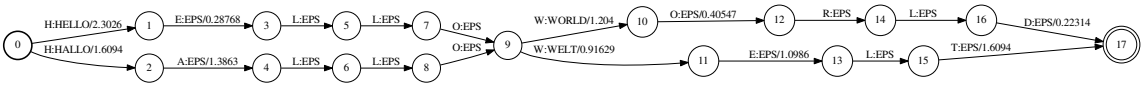


Figure 6.5: Composition result of input WFSA and lexicon WFST

Since the lexicon restricts the possible sequences to only contain the words in the lexicon, the output WFST also only contains the words “HELLO”, “HALLO”, “WORLD” and “WELT”. The weights of the input sequence WFSA arcs and the weights of the lexicon WFST are summed up to get the new arc weight, since logarithmic weights are used.

Here, the composition results in a WFST represented as a DAG since the input WFST was a DAG. In general, all kinds of WFSTs and WFSAs can be composed, as long as their output and input alphabets contain common symbols. The most common composition is the composition of a lexicon WFST and a language model WFSA, resulting in another WFST represented as a DCG.

Decoding: Determining the most probable sequence

After composing the input WFSA with the lexicon and language model WFST, decoding is performed by applying an algorithm to find the shortest path. Usually the Viterbi algorithm is used, delivering the shortest Viterbi path, as depicted in Figure 6.6.

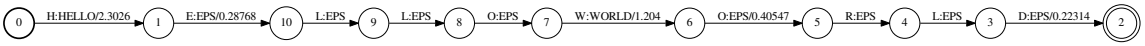


Figure 6.6: Shortest path in composition result of input WFSA and lexicon WFST

This realizes the calculation of word sequence probabilities according to Equation (5.18) and the decoding rule Equation (5.3). Since the negative log probability is used as the score, the shortest path is the path with the lowest score and therefore the highest probability. In this case the shortest path is "HELLO WORLD".

6.4.2 Contribution

A WFSTs based representation for word segmentation from lattices was already proposed in [Neu+12]. This work makes major contributions to the lexicon representation and the realization of the language model, as described in the following. The contribution especially consists of an exact handling of unknown words while [Neu+12] only uses an approximation for the calculation of their probabilities (e.g. not excluding known words from the unknown words in the lexikon and calculating probabilities of known words including their unknown forms instead of excluding them. Both issues are solved with the proposed lexicon and language model structure. In contrast to [Hey+14], where segmentation could only performed on the single best phoneme sequence due to a special lexicon construction for each input sequence, the lexicon proposed in this work is independent of the input sequence and solely represents all known words and corresponding unknown words in a single lattice which only has to be updated as new words are discovered or known words removed. The language model representation is mostly kept as in [Hey+14], correcting the approximation used in [Neu+12]. Forward filtering backward sampling is performed as described in [Neu+12] and also used in [Hey+14].

6.4.3 WFSTs for word segmentation

The word segmentation algorithm can also be realized with WFSTs. In principle the steps are the same as for the WFST based approach to speech recognition. For the word segmentation, an input WFSA, a lexicon WFST and a language model WFSA are used. The WFSTs are used to efficiently calculate the probability of possible segmentations, simplifying the calculation of the forward variable for the forward-filtering backward sampling algorithm. Instead of explicitly implementing the calculation of Equations (6.26), (6.32), (6.58) or (6.60) the WFST based representation and composition operations are used. This also enables simple use of higher order language models, other than the bigram and unigram used for the derivations in the previous sections. Additionally the

calculation of the forward and backward variable has to be implemented only once for a generic acyclic graph representation instead of having specific implementations for each setup, e.g language model order.

In the following, the explicit realizations of the WFSTs and WFSAs for the word segmentation algorithm are explained, beginning with the realization of the input sequence and followed by the lexicon and the language model. A character based example is used to intuitively visualize the different WFSTs and steps. The same structure of the WFSTs can be used for phoneme lattices and even for the segmentation from HMM states. The use of WFSTs allows for an easy extension of the model. One example is the addition of another WFST representing an HMM, mapping from state dependent observation probabilities to phoneme sequences. The HMM WFST can either be predefined, as in a speech recognizer or learned as well. The latter will not be covered in this work though.

All WFSTs have to be modified to consider character sequences alongside word sequences and apply word segmentation to the character sequences. Finally the sampling with the forward-filtering backward-sampling algorithm over the resulting graph is described.

Input Sequence

The input sequence for the word segmentation algorithm is represented as a WFSA, similar to the input sequence for the WFST based approach to speech recognition. Figure 6.7 shows an example WFSA representing the two possible sequence "_I _a _m _EOS" and "_I _m _EOS".

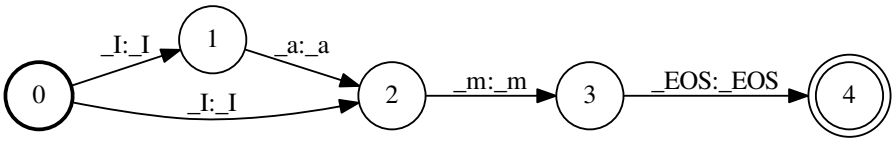


Figure 6.7: Input WFSA for sequence "_I _a _m _EOS" and "_I _m _EOS"

The input consists of phoneme (or character) sequences either output by a speech recognizer according to Equation (5.12) or constructed from given phoneme (or character) sequences. To distinguish the input symbols from the later word symbols, the input symbols are prefixed with an underline "_". As will be seen later, the output symbols will consist of the input symbols plus another set of output symbols, the word symbols. The input symbol "_EOS" plays a special role by marking the end of a sequence.

The input sequence is segmented to words by hypothesizing an optional word end symbol after each input symbol. The word end is marked by the "_EOW" symbol. The word end symbol is added with a self loop after the input symbol to enable 0 or more word end symbols. The maximum number of word end symbols will be limited to 1 by the lexicon in the next step. Figure 6.8 shows an example WFSA representing all possible word segmentations.

Some possible segmentations are "_I _EOW _a _m _EOW _EOS _EOW" and "_I _EOW _m _EOW _EOS _EOW" but also "_I _a _m _EOW _EOS _EOW".

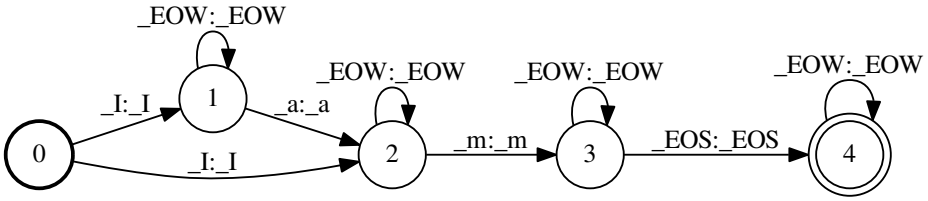


Figure 6.8: Input WFSA with all possible segmentations for sequence "_I _a _m _EOS" and "_I _m _EOS" with end of word symbols "_EOW"

Multiple repetitions of "_EOW" symbols are restricted by the lexicon and the sequence "_EOS _EOW" is mapped to an end of word word "EOW" by the lexicon later.

Lexicon

A deterministic pronunciation lexicon is used to map known input sequences into known words. This is similar to the WFST based approach to speech recognition. In addition to the previously introduced lexicon, this lexicon also has to pass through all input sequences which are not mapped to words. Figure 6.9 shows a lexicon WFST with the known words "I", "am" and "EOS".

The WFST consists of two parts. The lower part maps the input sequences "_I _EOW", "_a _m _EOW" and "_EOS _EOW" to the word "I", "am" and "EOS". The upper part passes all input sequences with unknown words through. This is realized by removing the "_EOW" transition after an input sequence which already represents a word. All input symbols are directly output for unknown words, while known words are replaced by an "_EPS" symbol and the word. Therefore the input alphabet and the word alphabet have to be kept separately. The mapping from known input sequences to known words is required for the NHPYLM language model to enable the correct scoring of known words as will be see later in the language model WFST.

Composing the input sequence containing all possible segmentations of Figure 6.8 with the lexicon WFST, results in another WFST containing known input sequences mapped to known words and unknown input sequences passed through as unknown words. Figure 6.10 shows the resulting WFST.

The WFST based representation results in a compact representation of all possible segmentations with known words replaced. It can be seen that the sequence "_I _EOW" is mapped to "I" and the sequence "_a _m _EOW" is mapped to "am" while the sequence "_I _a _m _EOW" is left unchanged, as are the remaining sequences with unknown words.

Language Model

The language model has to score the resulting segmentation WFST of Figure 6.10 according to the NHPYLM of Equation (6.7). Therefore the language model is realized as a fall-back n-gram language model. The lexicon is realized with "_PHI" transitions to only use the fall-back if no other transition is available. Each transition containing

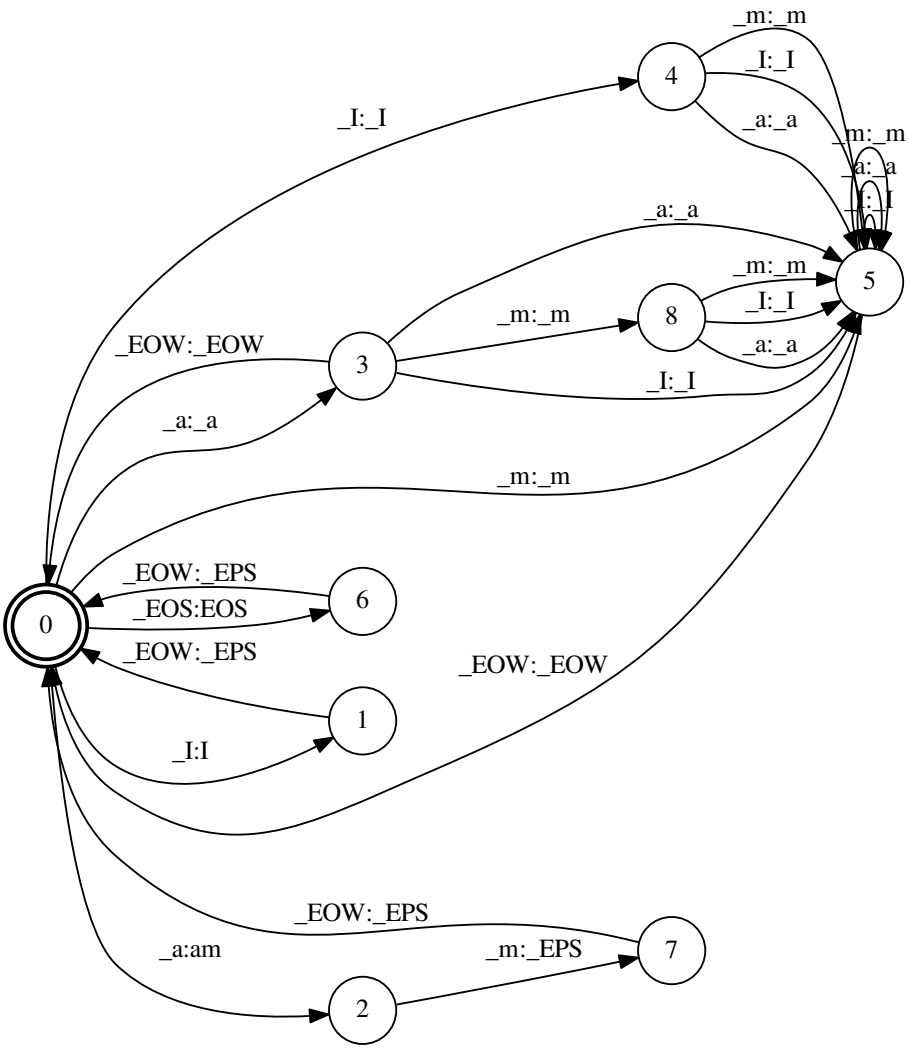


Figure 6.9: Lexicon WFST to generate all possible word and phoneme sequences

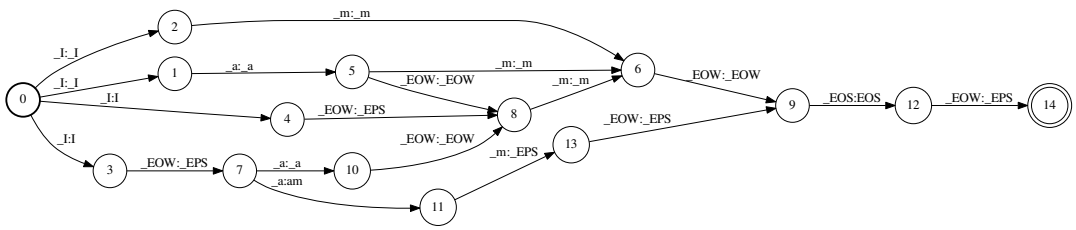


Figure 6.10: WFST with all possible word and phoneme sequences

a word or character gets the probability assigned according to Equation (6.7). Each fall-back transition only gets the fall-back probability of Equation (6.7), excluding the word probability of the shorter context as this probability will again be given by the transitions out of the shorter context. The language model contains two nested language models. The outer language model computes the known word probabilities while the inner language model computes the probabilities of unknown words based on their input symbol sequence. Figure 6.11 shows an example for a language model WFST with a bigram word language model and a unigram input symbol language model.

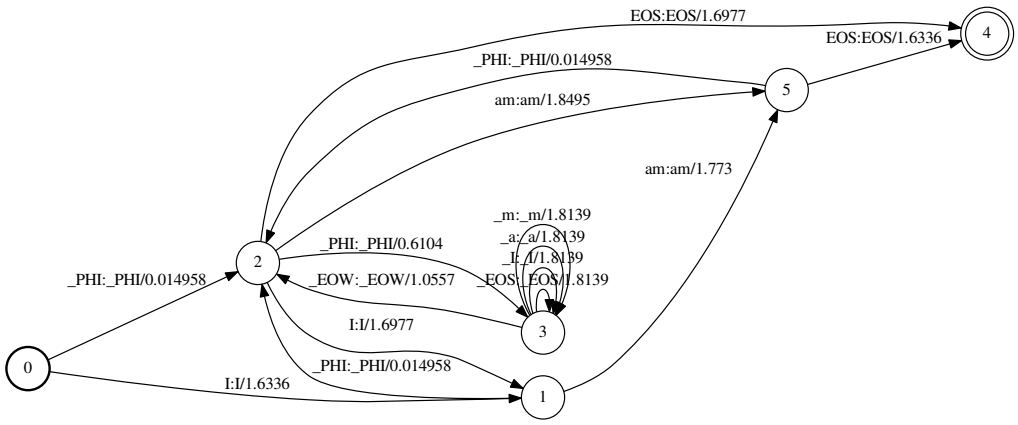


Figure 6.11: Language model WFST with word bigram and character unigram model

Node 0 represents the start of a sequence, while node 4 represents the end of a sequence. Node 1 represents the bigram context for "I". Node 2 represents the fall-back to a word unigram context. Node 3 represents the character unigram context. Node 5 represents the bigram context for "am". The arcs between the nodes represent the probability of a word given the context. The arc from node 1 to node 5 for example represents $\Pr(\text{am}|\text{I})$. The transitions with a "_PHI" symbol are only traversed if the word is not seen in the higher context. The transition from node 3 to node 2 is only possible via an "_EOW" symbol representing the end of an unknown word.

With this structure it is ensured that the inner character language model calculates the character based probability of an unknown word while the outer language model calculates the word probabilities.

Composing the language model WFSA with all possible segmentations and the lexicon results in the WFST in Figure 6.12.

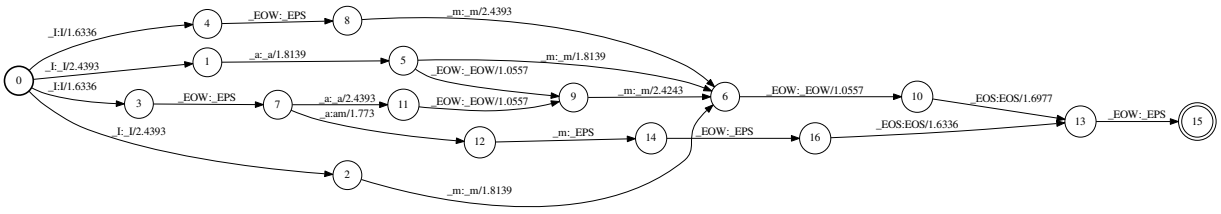


Figure 6.12: WFST with all possible weighted word and phoneme sequences

This WFST is a compact representation of all possible segmentations with known and unknown words and their corresponding weights. In contrast to the WFST just after the lexicon composition, it can be seen that the sequence "I _m" follows a different path since after "I" first the bigram context is visited followed by a fall-back transition. This is visible in the additional weights, which are a sum of two fall-back transitions and the probability for "_m". The sequence "I am" follows a slightly different path for the same reason, traversing from the bigram context "am" to "EOS". This WFST will be used in the next step to sample one possible segmentation using the forward-filter backward sampling algorithm.

Before going to the forward-filtering backward sampling algorithm a look at the shortest path is taken. Applying the Viterbi algorithm results in the WFST in Figure 6.13.

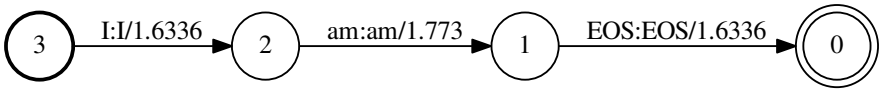


Figure 6.13: Shortest path in composition result of input WFSA, lexicon WFST and language model WFST. States are numbered in reverse due to the Viterbi algorithm outputting the state sequence from the last state (becoming 0) to the first state (becoming 3).

The best Viterbi path in this case is "I am" since the example language model gives the highest probability for this sequence. This example shows the case where some words are already known, e.g after a first Iteration of the word segmentation algorithm.

In general the algorithm is initialized with a lexicon containing no words and a language model with all characters having the same probability and no word probabilities as well. The lexicon and language model are iteratively updated based on the sampled segmentation in the previous step. For the sampling, the forward-filtering backward-sampling algorithm is used as described in the next section.

Forward-filtering backward-sampling on WFSTs

The forward-filering backward-sampling algorithm is used to sample one possible segmentation from the many possible segmentations resulting from the WFST compositions. The same forward-filter backward-sampling algorithm as described in Section 6.1.3 is used for sampling a sequence from a WFST. A generic sampling algorithm for acyclic graphs can be devised. In this case, the graphs resulting from the previously described representations and composition operations are acyclic.

The forward variable $\alpha[k]$ represents the probability of being in node k given all previous nodes. The forward variable is calculated as follows:

$$\alpha[k] = \sum_{i \in S_k} w_{i,k} \alpha[i], \tag{6.61}$$

where S_k represents all predecessor nodes of node k and $w_{i,k}$ is the weight on the arc going from node i to node k. The forward recursion is initialized with $\alpha[S_{\text{start}}] = 1$, where S_{start} is the start state.

For the backward variable $\beta[k][i]$, to sample the arc from node i into node k, the arc weight $w_{i,k}$ is used. This results in the following sampling probabilities:

$$\gamma[k][S_{k,1}] \propto w_{S_{k,1},k}\alpha[S_{k,1}] \tag{6.62}$$

$$\dots$$

$$\gamma[k][S_{k,I}] \propto w_{S_{k,I},k}\alpha[S_{k,I}], \tag{6.63}$$

where I is the number of arcs going into node k. The arc is sampled proportional to $\gamma[k][S_{k,i}]$, similar to Equation (6.20).

Note that in this case the node i corresponds to the state q_t in the forward-filtering backward-sampling algorithm.

Once a segmentation for an input WFST is sampled, the language model and pronunciation lexicon are updated, and the segmentation for the next input WFST is sampled. This is repeated until segmentations for all input WFSTs are sampled. This sampling process is repeated until convergence, meaning the sampled segmentations become more stable and ideally do not change anymore.

For the final iterations, after convergence of the sampling iterations, the Viterbi algorithm is used for an additional number of iterations to determine the most probable segmentations and update the language model and lexicon with the results.

6.5 Full System with Acoustic Unit learning

In this section a fully unsupervised setup is presented, enabling unsupervised learning of phones and words directly from audio recordings. The unsupervised learning setup consists of the following steps:

1. Acoustic unit learning with a Dirichlet process HMM on MFCCs.
2. Supervised acoustic HMM training with maximum likelihood linear transformation (MLLT) and feature-space maximum likelihood linear regression (fMLLR) to estimate a feature transformation for more robust features. This step uses the learned transcriptions of the previous step.
3. Acoustic unit learning with a Dirichlet process HMM on transformed MFCCs using the transformation learned in the previous step.
4. Word segmentation on the lattices derived using the unsupervised acoustic models learned in the previous step.

6.5.1 Setup

With this setup, acoustic units are learned using an unsupervised learning algorithm similar to the algorithm presented in Section 5.4.1. Here, the algorithm presented in

Figure 6.14: Full System Workflow

Section 5.4.1 is replaced by the implementation presented in [Ond+17], which uses a Dirichlet process HMM instead of the three step algorithm presented in Section 5.4.1. The Dirichlet process does not require a fixed number of acoustic units as it is automatically learned. The implementation in [Ond+17] applies Variational inference instead of the three step approach in Section 5.4.1. Additionally, a bigram Hierarchical Pitman Yor language model is learned over the acoustic unit sequence and applied during learning and decoding to improve the unsupervised model learning and extraction of acoustic unit lattices.

6.5.2 Contribution

A major contribution in this chapter is the actual fill system with acoustic unit learning. Additionally, the acoustic unit learning was extended by feature transformation learning performed on discovered acoustic unit sequences rather then GMM states as Originally described in [HSN16].

In this work, the implementation in [Ond+17] was extended by using Maximum likelihood linear transforms (MLLT) [Gop98; Gal99] and feature-space maximum likelihood linear regression (fMLLR) [Ana+96; Gal98], similar to the approach proposed in [HSN16], to generate more robust features for the unsupervised acoustic unit learning. In contrast to [HSN16], here the transcriptions for the supervised model training are derived using the unsupervised Dirichlet process HMM model, naturally delivering a sequence labeling. Originally, in [HSN16], only a Dirichlet process GMM is used, combined with a "heuristic compression", joining adjacent features with the same label into a single sequence. Whereas in here, the temporal dependence of adjacent features belonging to the same phone is modeled explicitly by an HMM, automatically "compressing" adjacent features into the same phone according to a statistical model.

In the following the performed steps are described in more detail.

6.5.3 Step 1: Acoustic unit and feature transformation learning on MFCCs

In the first step, several iterations of the acoustic unit learning algorithm are run to derive acoustic units. Additionally the bigram hierarchical Pitman Yor language model is learned and applied in further iterations to improve the learned acoustic units.

The learned acoustic units are then used as a label sequence for the audio recordings to train a supervised HMM based acoustic model using Kaldi [Pov+11]. In this training step, MLLT and fMLLR feature transformations are learned and used to derive more robust features. Kaldi was used since it had existing implementations of MLLT and

fMLLR estimators. Even though the training in Kaldi is performed as a supervised training, this is still a fully unsupervised process since the HMM training only uses the transcriptions learned by the Dirichlet Process HMM in an unsupervised way. Because of using the learned transcriptions, the learned transformations are also derived in an unsupervised way.

As a result feature transformations are obtained.

6.5.4 Step 2: Acoustic unit learning on transformed MFCCs

Using the learned transformations, adapted features for a second iteration of the unsupervised acoustic unit learning are extracted. This delivers more robust features improving the unsupervised acoustic unit learning, as can be seen in the experimental results further down. This second iteration is similar to step one and only differs by the fact that transformed features are used instead of MFCCs. The unsupervised acoustic unit learning algorithm is applied again on these features to learn a new set of unsupervised acoustic units.

Only one iteration of step 1 unsupervised acoustic unit learning, and step 2, feature transformation and unsupervised acoustic unit learning on transformed features, similar to step 1, is performed, since multiple iterations did not yield further improvements.

As a result of this step, acoustic unit models are obtained which can be used to decode audio recordings into acoustic unit lattices.

6.5.5 Step 3: Unsupervised word segmentation on unsupervised acoustic units

Word segmentation is performed using the unsupervised word segmentation algorithm presented in the beginning of this chapter. The word segmentation is performed on acoustic unit lattices, extracted using the acoustic unit models learned in the previous step on the transformed features.

In addition to the previous word segmentation experiments, an additional character language model for extracting the best sequence from the input lattice is applied. After applying this character language model to the acoustic unit lattices, the first best sequence of acoustic units is extracted from the acoustic unit lattices, using only character n-grams instead of word n-grams. This character language model is iteratively updated on the previous word segmentation result, together with the word language model. However, in contrast to a usual character based language model, this language model includes word end markers and additionally calculates the probability of a word end marker in the character sequence, effectively resulting in a word segmentation on the character level. However, this segmentation is not explicitly used, since only the character sequence without the word end markers is passed to the word segmentation step to segment it again using the Nested Hierarchical Pitman Yor Language Model. Nevertheless including the word end markers in the single best sequence extraction provides additional information about word ends on the character level and potentially improves the single best sequence

and therefore also the overall word segmentation performance. The influence of this additional character language model on the word segmentation results is analyzed in the following.

The use of an additional character language model was proposed in [Hey+14] to accelerate the word segmentation algorithm on lattices by performing word segmentation on the single best sequence. There, the single best sequences are extracted in parallel from each lattice, while additionally reducing the complexity of the word segmentation algorithm, since it has to process a smaller number of hypotheses of segmentations and only consider a single sequence as input instead of a lattice. However, the extended word segmentation on full lattices presented in this work performed equally fast as the parallelized version in [Hey+14] and with better scores on character lattices than presented in [Hey+14]. Therefore until now the additional character language model was not used in the previous described experiments. An improvement of word discovery results was not observed on the cleaner lattices in the previous experiments when using an additional character language model. Therefore no necessity was seen to apply the additional character language model yet and the word segmentation was performed on full lattices in the previous experiments. However, in the following experiments it can be seen that the additional character language model actually improves the word segmentation results on very noisy lattices derived with the unsupervised acoustic unit learning algorithm applied here.

6.6 Experimental results

To evaluate the word segmentation algorithm with different types and quality of input data, with each further setup, more uncertainty is introduced into the input data, going towards a real world scenario with little prior knowledge. The input data quality starts from error free text and then gradually goes to a fully unsupervised system where the acoustic model and the word segmentation are learned in a completely unsupervised manner. The following experimental setups were used:

- Segmentation of error free text to get an upper bound estimate of the word segmentation performance (English, Wooi, Waima'a).

- Segmentation of lattices generated using pre trained speech recognizers to evaluate the performance of word segmentation on lattices containing uncertainty and errors in a setup with an existing acoustic model. Three variants are tested: Same language input (British English model with British English input and American English model with American English input) and cross language input (American English model and Xitsonga input)

- A fully unsupervised system using acoustic unit discovery in a first step and word segmentation on acoustic unit lattices in the second step to evaluate the performance of a fully unsupervised system (Xitsonga).

6.6.1 Segmentation of Text (Character/Phoneme)

Datasets

To first evaluate the unsupervised word segmentation algorithm on error free text, experiments are carried out on three corpora. Two Austronesian languages, Wooi [Kir+15] and Waima'a [Bel+06] and English, to set the results on the two Austronesian languages into context. Wooi is spoken at the western tip of Yapen Island in Indonesia. Approximately 1600 people still speak Wooi. Waima'a is an endangered Austronesian language from Timor Lorosa'e (East Timor). The precise extent of the Waima'a speaking area and population remain to be determined.

Unsupervised word segmentation is performed on text corpora from these languages, employing the Bayesian language modeling based on the Pitman-Yor process as described earlier in this section. The WFST based implementation was used. Experiments with varying language model orders were conducted and the usefulness of word length models investigated. Further, a method to tune lexical discovery results towards high precision to cater the needs of linguists is evaluated. The comparison with an English corpus of similar size, the Wall Street Journal (WSJ) corpus, is done to set the results on the Austronesian corpora into the context of a well known corpus.

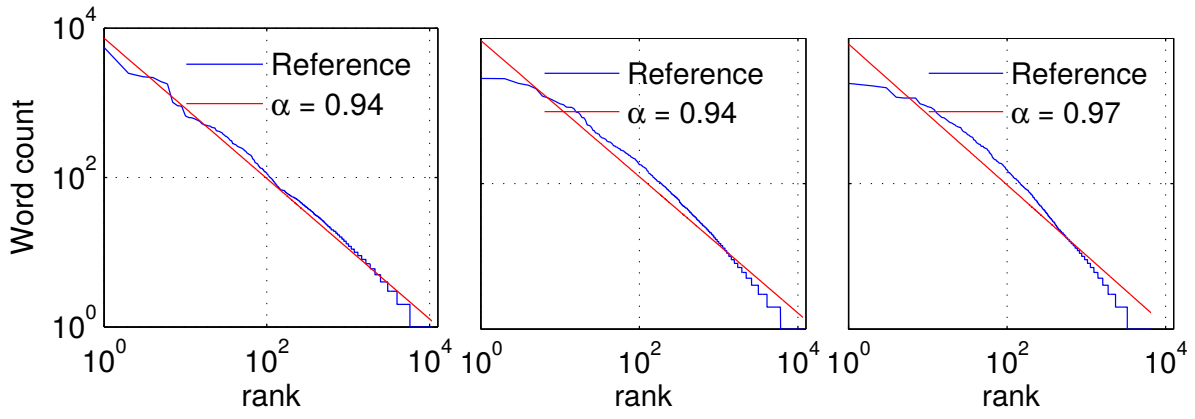
- The first corpus contains the text prompts of the WSJCAM0 training data, consisting of 5612 sentences with 95,442 running words and a lexicon size of 10,658 words. The results on this corpus serve as reference results to set the performance on the following austronesian corpora into comparison to a well-known database.
- The second corpus contains text transcriptions of 743 recording sessions of the Wooi language. The sessions are separated into 37,028 intonation units, with 123,848 running words and a lexicon size of 11,512 words.
- The third corpus contains 391 recording sessions of the Waima'a language. The sessions are separated into 19,888 intonation units, with 88,751 running words and a lexicon size of 6535 words.

The three copora are comparable in terms of the number of words in the running text and the lexicon size. While the WSJ corpus contains text from newspaper articles, the Wooi and Waima'a corpus contain spontaneous speech.

Figure 6.15 displays the word counts on the three corpora as a function of the rank of the word in a list ordered according to word frequency. In the chosen log-log scale, adherence to Zipf's law can be easily checked. If for the probability of the n-th most probable word w_n it holds that

$$\Pr(w_n) \propto \frac{1}{n^\alpha}, \tag{6.64}$$

the distribution of word probabilities will form a straight line with slope $-\alpha$. The WSJ corpus shows a straight line, while the curves for the Wooi and Waima'a corpus slightly deviate from a line for the lower ranks. Since here, the goal was to simply compare the actual distribution to the expected Zipf's law, no further investigations



(a) WSJ (b) Wooi (c) Waima'a

Figure 6.15: Word count as a function of rank. The blue line shows the actual distribution in the corresponding corpus. The red line shows a fitted line according to Zipf's law with the given slope α.

have been performed for the reason of this deviation for the Wooi and Waima'a corpus. For the higher ranks the lines follow the power law property. The maximum likelihood estimates of α are 0.94 for the WSJ corpus, 0.94 for the Wooi corpus and 0.97 for the Waima'a corpus. These results show that the assumption of a power law property by the Pitman-Yor language model holds for English as well as for exotic languages like Wooi and Waima'a.

Word Segmentation Experiments

In a first experiment, the word segmentation algorithm is applied to all three corpora. The quality of the segmentation result for different orders of the character language model and the word language model is evaluated. As performance measures the token F-score and the lexicon F-score are used.

The F-score F is the harmonic mean of the precision P and the recall R. The precision is defined as the ratio of the number of correctly found words N_{correct} and the total number of found words N_{found}, while recall is defined as the ratio of the number of correctly found words N_{correct} and the number of words to be found $N_{\text{reference}}$:

$$F = 2\frac{P \cdot R}{P + R}, \quad P = \frac{N_{\text{correct}}}{N_{\text{found}}}, \quad R = \frac{N_{\text{correct}}}{N_{\text{reference}}}. \tag{6.65}$$

The token F-score is determined by aligning the segmented sequence with the reference sequence, and choosing the alignment with the lowest edit distance, allowing deletions, insertions, substitutions and matches. Matches are then counted as correctly found tokens. For the lexicon F-score a word in the learned lexicon is counted as correctly found if it is present in the reference lexicon.

Figure 6.16 shows the results for the token and lexicon F-score as a function of the character language model order when using a unigram and a bigram word language model. For the unigram word language model, the F-scores reach their peak at around a spelling model order of 3. For the bigram word language model, the F-scores increase with increasing spelling model order, only for the WSJ corpus the F-score decreases for character language model orders greater than 8. Increasing the character language model order further makes no sense since the number of words of length larger than 8 characters is very small.

The bigram word language model performs better than the unigram word language model for the WSJ corpus (72.6 % token and 62.5 % lexicon F-score) and the Waima'a corpus (72.73 % token, 51.82 % lexicon) corpus. For the Wooi corpus, the peak token F-score for the unigram is at 66.99 % and for the bigram at 64.85 %. For the lexicon F-score the bigram word language model performs best, achieving an F-score of 54.01 %. Based on these results a character language model order of 7 and a word model order of 2 is chosen, since this setup gave good results across the three corpora.

The next set of experiments evaluated the explicit word length modeling according to Equation (6.13).

Figure 6.17 shows the distribution of the word length in the reference corpus and in the segmentation result with length modeling activated. It is striking how well the distribution of the word lengths of the segmentation result follows the distributions in the reference corpus. Even the dip at word length 3 in the Wooi corpus is reflected in the automatically found segmentation! The figure also shows a Poisson distribution fitted to the word length distribution measured on the reference text, which resulted in a mean value of 4.9, 4.3 and 3.6 characters per word for WSJ, Wooi and Waima'a, respectively.

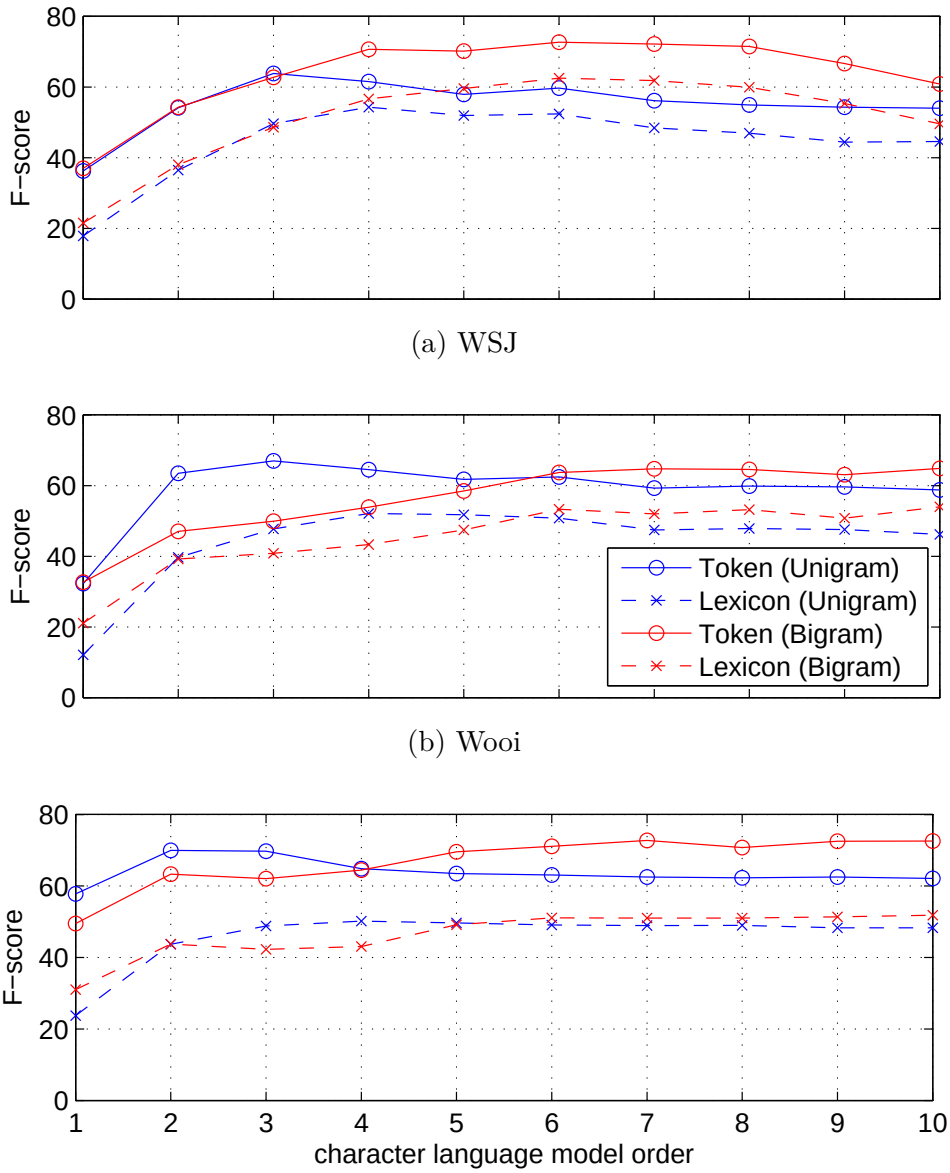
It turned out that the Poisson correction has little influence on the segmentation. This can be seen by the fact that the distribution measured on the segmentation result is different from a Poisson distribution. Indeed, the explicit word length correction of Equation (6.13) did not improve the segmentation performance when comparing to experiments without explicit word segmentation. The results were almost the same as presented. Therefore the explicit length modeling will not be used in further experiments.

Optimization of Lexicon F-Score

To be a valuable tool for linguists, the lexicon discovered by the word segmentation algorithm should have both high precision and high recall. However, a word segmentation algorithm based on n-gram probabilities cannot be expected to perform well for words which occur very rarely in the corpus. Errors will therefore concentrate on infrequent words.

To analyze this effect, word segmentation was carried out with a bigram word model and a 7-gram character model as described. Only word tokens that occurred at least a given count in the segmentation result were kept in a postprocessing step. Table 6.1 shows the effect of discarding infrequent words on the token and lexicon F-score.

If only those words with a count of at least 1, 2 or 3 were kept in the segmentation result and the reference, F-scores at lexicon level rose significantly. Since infrequent words do not

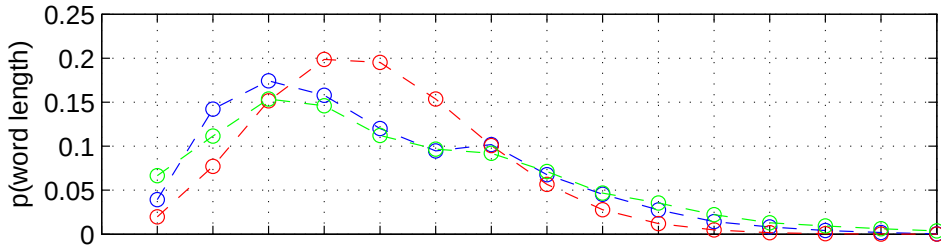


(a) WSJ

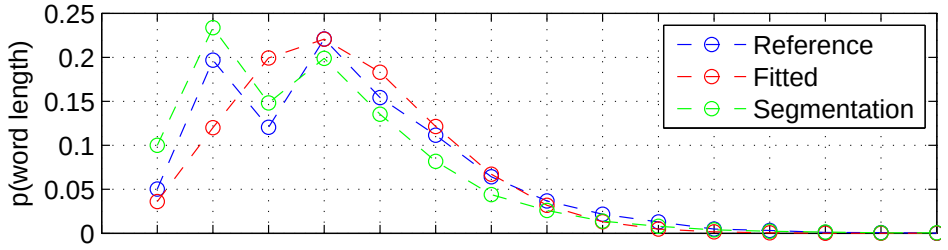
(b) Wooi

(c) Waima'a

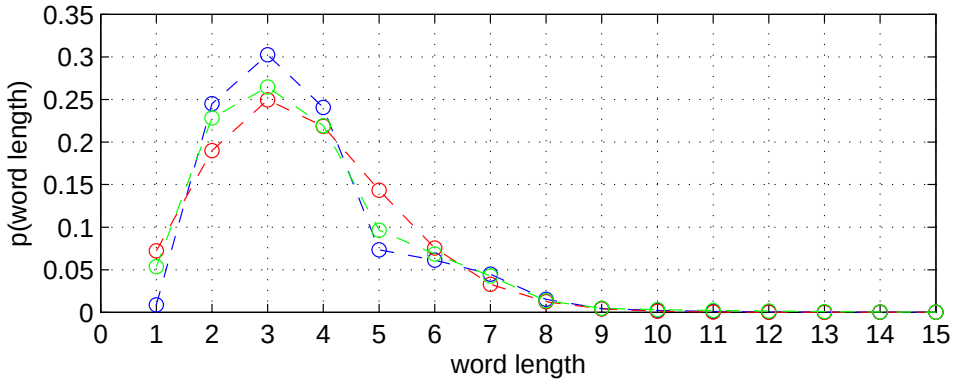
Figure 6.16: Token and lexicon F-score as function of model order. The corresponding language model oder is depicted below the last graph and the same for each graph.



(a) WSJ



(b) Wooi



(c) Waima'a

Figure 6.17: Word length distribution for the observed words (blue), the fitted Poisson distribution (red) and the resulting word length after the word segmentation (green).

Table 6.1: F-scores when only keeping words occurring at least "count" amount of times or a combination of 3 times and minimum length of 2 characters (last entry)

	WSJ		Wooi		Waima'a	
Count	Token	Lex.	Token	Lex.	Token	Lex.
1	69.6	61.8	64.8	52.0	73.6	51.0
2	71.6	69.7	65.6	59.8	74.8	63.9
3	72.3	76.4	66.2	64.4	75.2	68.1
3/2	76.9	76.5	67.8	64.3	76.8	68.1

occur often in the segmentation, the token F-score is not greatly influenced. Discarding words with higher counts did not improve the results significantly but discarded too many words. In the table, the row with the first entry being 3/2 means that only those words with a count of at least 3 and additionally with a length of at least 2 characters were kept. As can be seen, discarding single-character words did slightly improve the segmentation result. Discarding longer words decreased the scores. Table 6.2 shows the resulting precisions as well as the number of found words N_f and remaining words N_r for the last setup.

Table 6.2: Precisions and remaining words of 3/2 setup

	WSJ		Wooi		Waima'a	
	Token	Lex.	Token	Lex.	Token	Lex.
P	81.1	75.7	64.7	69.2	78.1	67.5
N_r	75204	4037	119225	4066	79663	2364
N_f	92561	11070	139718	8641	87927	5783

With such a parameterization the presented word segmentation can serve as a handwork saving preprocessing tool for linguists: while the algorithm takes care of all frequent words and thus most of the text, linguists can concentrate on infrequent words to complete the lexicon.

Conclusions

This first experiment on error free text shows that unsupervised word segmentation based on a nested Pitman-Yor language model can be a valuable tool for linguistic analysis, adding to the set of natural language processing tools to discover, annotate, analyse or combine digital language data. On the Austronesian languages Wooi and Waima'a it achieved segmentation results that are comparable to those obtained on a well-known English corpus, demonstrating the universality of unsupervised learning approaches to language processing. A significantly higher F-score is obtained on frequent compared to infrequent words, leading to the conclusion that the algorithm can be used as a preprocessing tool, which saves, however, not frees linguists from manual effort.

6.6.2 Same Language Lattice Segmentation from pre trained Recognizer

In this next setup, a first experiment for unsupervised word segmentation on lattices is performed. For this first experiment a so called same language setup is chosen where the acoustic units (phoneme models) are trained as a supervised model on the same language as the sentences to be segmented. To simulate the effect of noisy results, only a monophone GMM/HMM acoustic model is trained. This setup and experiment mostly serves as a demonstration to show that word segmentation can be performed on lattices.

Experimental Setup

In this next setup, experimental results for phoneme lattice segmentation were first obtained using the training speech data from the Cambridge version of the Wall Street Journal corpus (WSJCAM0) [Fra+94], comprised of 7861 sentences. The size of the vocabulary is about 10k words. A monophone HMM acoustic model was trained on this training set and decoding was carried out on a subset of the same set, consisting of 5628 sentences, using the trained monophone acoustic models and a zerogram phoneme LM, producing a lattice at the output. The produced lattices were reduced in size by pruning away unlikely paths whose likelihood was $\exp(10)$ worse than the likelihood of the highest scoring path.

This setup serves as an experiment to investigate the influence of an error prone phoneme lattice in comparison to error free text. The same language setup, where the training set is decoded into a lattice was chosen to simulate imperfect lattice without introducing too many errors. For this reason only a monophone HMM model was chosen to simulate results possible with unsupervised acoustic model learning of a simple HMM GMM acoustic model. Yet, the phoneme error rate of the highest scoring path is about 33%, while the lowest possible phoneme error rate in the pruned lattice is about 10.5%, when extracting the path with the lowest edit distance to the reference. These 10.5% only serve as a measure for the lowest possible phoneme error and is naturally not possible to achieve without perfect knowledge about the actual sequence or a very good language model.

The segmentation algorithm was run for $k_{\max} = 100$ iterations, switching the word language model orders from $n = 1$ to $n = 2$ and phoneme language model order from $m = 2$ to $m = 8$ after $k_{\text{sw}} = 25$ Iterations. This setup is similar to the best performing setup in [Hey+14] and uses the same lattices. The difference lies in the implementation of the word segmentation algorithms. [Hey+14] uses a sequential implementation with a character only based language model to first extract single best sequences and then perform segmentation on these sequences, repeating the model updates, best sequence extraction and word segmentation with each iteration. In contrast the implementation presented in this work avoids the extraction of the single best path and performs segmentation directly on the lattices. The sequence of language model order switching and the final language model orders are the same.

Table 6.3: Most often found words with more than 3 characters

Rank	c_w	Phonemes	Characters
1	3489	dh ax	THE
13	447	s eh d	SAID
17	365	m ih s t ax	MISTER
20	327	f ay v	FIVE
23	279	p ax s eh n t	PERCENT
25	268	m ih l ia n	MILLION
26	267	s ah m	SOME
27	263	k ah m p ax n iy	COMPANY
29	259	d oh l ax z	DOLLARS
33	235	hh ae v	HAVE

Results

To visualize the segmentation results, Figure 6.18 shows the 100 most often found phonetic words. The bigger a word is printed, the more often it was found in the data. Some words with more than three characters and their corresponding phoneme

Figure 6.18: 100 most often found phonetic words. See Table 6.3 for some orthographic transcriptions.

based representations are listed in Table 6.3, where c_w is the number of times the word was found. Since the used material contains many business terms, those are primarily discovered. Additionally to the listed words, the numbers one to 10: (“w ah n”, “t uw”, “th r iy”, “f ao r”, “f ay v”, “s ih k s”, “s eh v n”, “ey t”, “n ay n”, “t eh n”), as well as 20 (“t w eh n t iy”), 30 (“th er t iy”), 50 (“f ih f t iy”) and 90 (“n ay n t iy”) are amongst the words. Out of the 100 most often found words, 80 were correct words.

In total 28.6% of the words from the running text were correctly discovered (recall) where 32.2% of all the discovered words are correct (precision), resulting in an F-score of

Table 6.4: Word segmentation results of presented algorithm and of [Hey+14]

	Token			Lexicon				
Algorithm	R	P	F	R	P	F	PER	R@100
Presented	28.6	32.2	30.3	21.8	13.2	16.4	24.5	80
[Hey+14]	21.4	30.4	25.1	19.3	13.6	16.0	25.5	70

30.3%. The phoneme error rate was reduced to 24.5%. The precision for the lexicon is 13.2% with a recall of 21.8%, resulting in a lexicon F-score of 16.4%. Table 6.4 compared the results for the presented algorithm to the results obtained in [Hey+14]. It can be seen that the presented algorithm performs better than the one presented in [Hey+14].

The implementation took a runtime of about 10 hours for the 100 iterations on a single core of an Intel(R) Xeon(R) E5-2640 at 2.50GHz resulting in a realtime factor of 1. This is the same runtime as the implementation in [Hey+14]. It shows that the segmentation of lattices instead of the sequential segmentation presented [Hey+14] does not increase the runtime even though the input to the segmentation is seemingly more complex. Additionally it achieves better scores than a sequential approach.

Conclusion

The experimental results show that lattice word segmentation is indeed possible and delivers reasonable results. It can be seen that consistent phoneme sequences can be discovered and that those sequences actually correspond to real words. It can also be seen that the proposed full lattice segmentation delivers better results than the iterative approach which extracts the first best sequence and performs segmentation on this sequence. In the following sections additional setups are investigated, eventually concluding with a fully unsupervised setup.

6.6.3 Cross Language Lattice Segmentation from pre trained Recognizer

For the cross language setup, the segmentation algorithm is evaluated on datasets provided for the 2015 ZeroSpeech challenge [Ver+15]. The datasets consist of an English dataset containing conversational speech from the Buckeye corpus [Pit+07] and a second dataset containing prompted speech in Xitsonga, a south African Bantu language, from the NCHLT Xitsonga corpus [De +14]. The goal is to demonstrate the possibility of using existing acoustic models from another language to perform the word segmentation. Acoustic models trained on prompted English speech are used to create lattices for both datasets. The English dataset is used to demonstrate the segmentation performance when using acoustic models of the same language and to establish a topline for the acoustic models used in this experiment. In contrast to the previous setup, the input is spontaneous speech, where the previous setup used read speech. The Xitsonga corpus serves as the low resource language. It is assumed that only audio data is available for

the Xitsonga corpus, but no transcriptions.

Here, the tools provided for the 2015 ZeroSpeech challenge are used for the evaluation, to be able to compare the results to previous publications. Instead of all scores, the focus is on the type (same as the lexicon f-score) and token scores, similar to the previous clean text and same language example. The type scores are a measure for the quality of the discovered lexicon and therefore the set of discovered words (same as the lexicon f-score). The token scores are a measure for the quality of the discovered word tokens and therefore the transcription of the speech, also called parsing quality. A detailed description of the evaluation framework and evaluation measures can be found in [Ver+15].

Setup

For the acoustic model a DNN-HMM triphone speech recognizer with p-norm normalization [Zha+14] is used. The speech acoustic model is trained on English speech from the WSJ0+1 corpus [PB92]. The recognizer is built, using the nnet2 p-norm recipe for WSJ provided with the Kaldi [Pov+11] speech recognition toolkit. The recipe was modified to enable phoneme recognition without a word lexicon by building a simple lexicon, mapping each triphone to its middle phoneme.

The recognizer uses LDA transformed 13 dimensional MFCC feature vectors extracted with a frame rate of 10 ms and a context of ± 3 frames at a target dimensionality of 40. FMLLR speaker adaptation of the LDA transformation is performed by a two pass decoding scheme where it is assumed the speaker ID to be known, e.g. from some speaker diarization algorithm.

The recognizer is used to create phoneme lattices for both datasets which are processed by the segmentation algorithm. The word and character language model order are varied in the segmentation algorithm from 1 to 2 (WLM) and 1 to 8 (CLM) to evaluate the performance with different model complexities. Gibbs sampling is performed until iteration 150 to generate the segmentation of a sentence and to update the language model. From iteration 151 Viterbi decoding is performed to generate a segmentation. From iteration 176 the fallback probability to the character model is set to zero to disable the discovery of new words and clean up the language model by removing infrequently, especially uniquely, discovered words. The iterations were chosen so that in each step the algorithm converged.

Results

First, the performance of the segmentation algorithm on the English dataset is evaluated. Since an existing English acoustic model is used, it is expect that the results are superior to purely unsupervised methods which also learn the acoustic model. The English results are therefore not completely comparable and serve more as a baseline in case of a known acoustic model. Figures 6.19 and 6.20 show the type and token F scores with varying language model orders and decoding settings.

In Figure 6.19 it can be seen that the type F score increases with character language model order. Applying Viterbi decoding increases the performance, while deactivating the

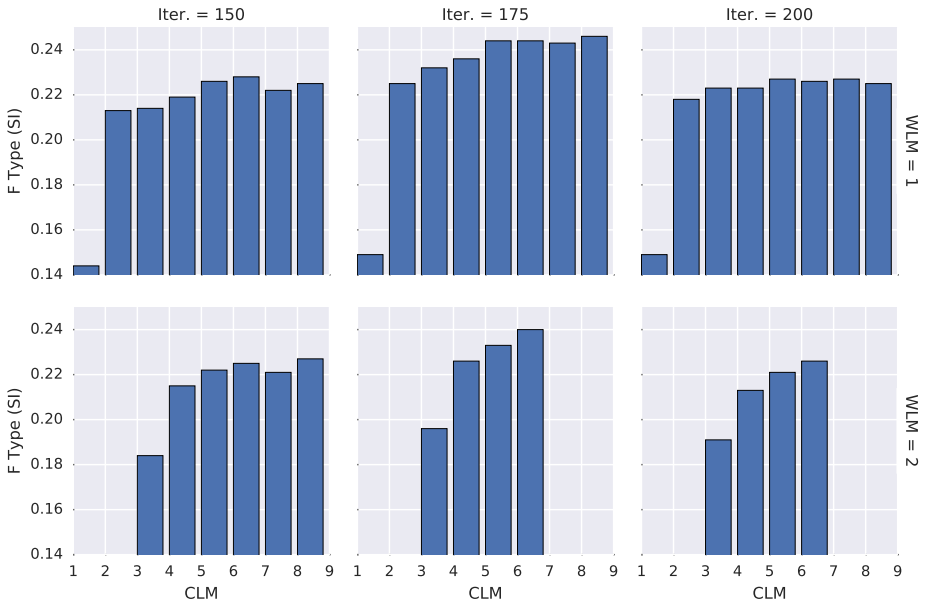


Figure 6.19: Type F-score with varying word and character language model order for English dataset. Iterations: 150 (Gibbs), 175 (Viterbi), 200 (No character model fallback)

character language model seems to reduce the performance. The increased performance when using Viterbi decoding can be explained by the fact that Viterbi decoding delivers the sequence with the highest probability, while Gibbs sampling only delivery samples from the possible distribution. Nevertheless Gibbs sampling is crucial in the learning process to achieve better convergence and avoid local minima in the learning process. The reduced performance without the character model can also be explained with the fact that uniquely and rarely discovered words are removed in favor of similar more frequent ones. Unique and infrequent words usually have a big contribution to the lexicon due to the power law. Increasing the word language model order to two seems to approach similar performance compared to order one. Simulations at character language model order 1 and 2 were skipped due to their low performance. Orders 7 and 8 were not evaluated for higher iteration numbers due to their runtime and similar results to Gibbs sampling only.

In Figure 6.20 it can be seen that the token F score increases when deactivating the character language model, in contrast to the type F score in Figure 6.19. This could be explained by the fact that the infrequent words might lead to wrong segmentations, while removing them promotes the more frequent words. Also removing less frequent words has a lower influence on the number of correct tokens, while placing more frequent words more often in the correct position will increase the token score. On the other hand increasing the word language model order from one to two significantly increases the token F score. This supports the assumption that context helps to fill in the correct words. This is especially the case for similar sounding words with different meanings.

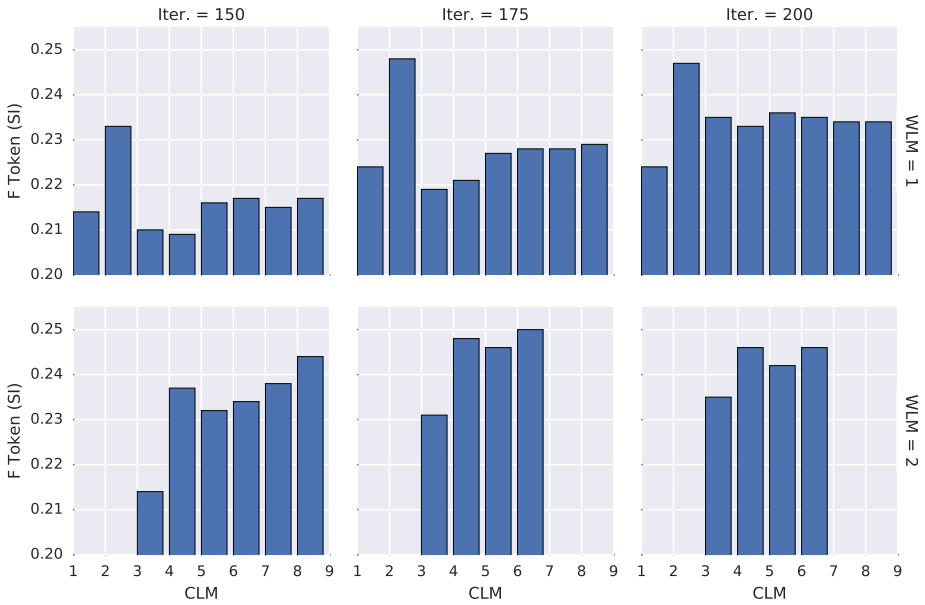


Figure 6.20: Token F-score with varying word and character language model order for English dataset. Iterations: 150 (Gibbs), 175 (Viterbi), 200 (No character model fallback)

Evaluating the performance of the segmentation algorithm on the Xitsonga dataset delivers insight into its usefulness for low resource language processing. The Xitsonga language is treated as a low resource language by assuming that only audio data is available but no transcriptions. It is also assumed that no acoustic model is available and instead the English acoustic model is used to create phoneme lattices for the segmentation algorithm. This concept is also called cross language transfer, where knowledge from one language is transfered to another. One might argue that this setup does not resemble a completely unsupervised setup therefore a completely unsupervised setup will be investigated in the next section. Nevertheless using the acoustic model of an unrelated language could still be seen as almost completely unsupervised setup where the acoustic model merely serves as a generic feature extractor which can be trained on high-resource languages. This is somewhat similar to the assumption of a certain oscillation frequency in the later described Osc. Sec. system for segmentation of speech recordings without learning it from the data.

Figure 6.21 shows the type F scores for different language model orders and decoding settings. It can be seen that the performance increases with increasing character language model order. However the overall scores are fairly low. This is mainly due to the mismatch in acoustic models and the resulting errors and noisiness of the phoneme latices. Viterbi decoding delivers a little lower performance although for higher character language model orders it matches the performance with Gibbs sampling. This might partly be due to the noisy characteristics of the input phoneme lattices. Viterbi decoding is supposed to find the result with the highest probability. Due to the noise this might not be the

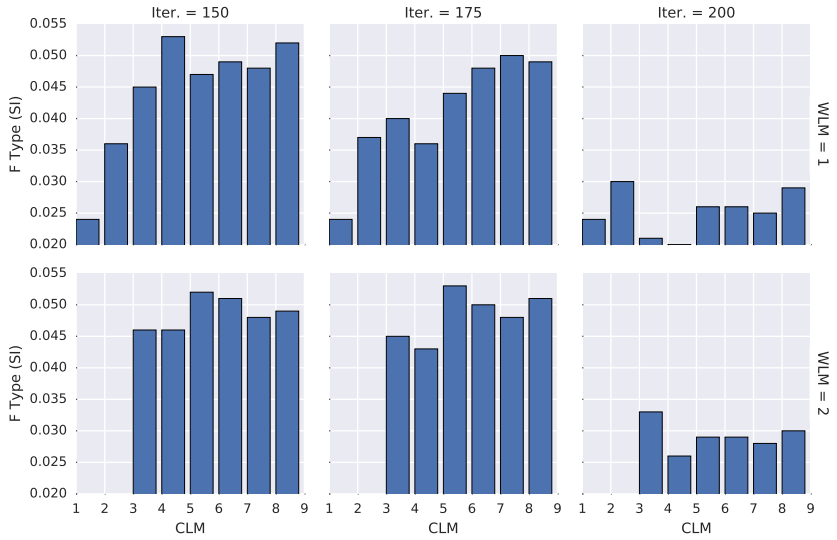


Figure 6.21: Type F-score with varying word and character language model order for Xitsonga dataset. Iterations: 150 (Gibbs), 175 (Viterbi), 200 (No character model fallback)

optimal result. Gibbs sampling delivers samples from the distribution of segmentations and language models seems to result in better performance. Deactivating the character language model deteriorates the results. Most likely the input data is too noisy, resulting in many infrequent words which are being removed in this case. Increasing the word language model order from one to two also does not change the results significantly. The scores are a little higher for the lower order character language models but almost the same for the higher order language models. It seems that word context improves the performance for lower order character language models and noisy input but not for more complex models, contrary to previous results on less noisy data [Hey+14].

Figure 6.22 shows the token F scores. The behavior is similar to the type F score. The result deteriorates with Viterbi decoding and deactivating the character language model. Increasing the word language model order from one to two results in marginally better results. The biggest issue in this low resource setup seems to be the noisy input data, making it difficult to learn appropriate models at higher word language model orders.

Comparison with previous results

In the 2015 ZeroSpeech challenge, two types of systems participated. The two systems can be classified into segmentation systems that segment, cluster and label the complete utterance. The unsupervised lattice word segmentation system also falls into this category. On the other hand, Spoken Term Discovery (STD) based systems discover similar segments and only cluster and label those, leaving segments not discovered as similar to others unlabeled. In Table 6.5, the presented results are compared to the two types of systems. For the 2015 challenge, several results for track 2 (unsupervised word

Figure 6.22: Token F-score with varying word and character language model order for Xitsonga dataset. Iterations: 150 (Gibbs), 175 (Viterbi), 200 (No character model fallback)

discovery) were submitted [Dup16]. Although the results were only submitted by two different authors with a total of two different systems. Therefore the comparison is done to the best setup of each system type as described in the following.

Osc. Seg. (oscilation based segmentation) is based on a simple segmentation algorithm finding minima in a particular oscillation frequency of the speech similar to the theta-rhythm brain oscillations and segment according to these. Fixed length representations of the discovered segments are then clustered, labeled and n-grams of these clusters, sorted in ascending order from longest to shortest, labeled as words [RDF15].

Spoken Term Discovery is a system based on finding similar segments, building a graph with edges connecting those similar segments with weights proportional to their similarity and clustering them using graph clustering algorithms [LSJ15].

For the unsupervised lattice segmentation system presented here, the best setups with word language model order one and highest type F score (NHPYLM 1) and word language model oder two and highest token F score (NHPYLM 2) are compared to the other systems. The system performs best in both settings on the English dataset, since an English acoustic model is used. On the Xitsonga dataset the system performs best on the token precision and F score and second best in all three token performance measures. It also performs better then the Osc. Seg. system. For the type performance the system performs second best in all measures after the STD system. Since the system is a segmentation system it performs better on the token measures while the STD system is able to discover a better lexicon but not label all segments, resulting in higher type measures on the Xitsonga dataset.

Since an English acoustic model is used, the comparison on the English dataset is to be understood as a baseline in case of known and partly matching models. The results

Table 6.5: Precision (P), Recall (R), F-score (F) for Type and Token on English and Xitsonga dataset with different algorithms. Red: best score, blue: second best score.

	English						Xitsonga					
	Type			Token			Type			Token		
System	P	R	F	P	R	F	P	R	F	P	R	F
Osc. Seg.	14.1	12.9	13.5	22.6	6.1	9.6	2.2	6.2	3.3	2.3	3.4	2.7
STD	3.1	9.2	4.6	2.4	3.5	2.8	4.9	18.8	7.8	2.2	12.6	3.7
NHPYLM 1	18.1	38.7	24.6	28.8	19.0	22.9	3.9	8.2	5.3	4	2.7	3.2
NHPYLM 2	17.8	36.7	24.0	24.5	25.5	25.0	3.7	8.5	5.1	4.1	3.4	3.7

are therefore not easily comparable to the other systems and results are merely given for reference.

Conclusion

The evaluation of the unsupervised lattice segmentation system demonstrated a higher performance over a comparable segmentation system while still suffering from noisy input data. Although a better performance than the STD system on the token f-score and precision is achieved, type (lexicon) quality is still behind STD systems. In the next section a fully unsupervised setup is evaluated, also learning the acoustic model in an unsupervised way.

6.6.4 Full System with Acoustic Unit learning

To evaluate the influence and performance of the different steps and applied algorithms, experiments were performed for each step. Each step is evaluated separately. First, the performance of the acoustic unit learning is evaluated. Results for the MFCCs and when using the adapted features in a second iteration of the acoustic unit learning algorithm are given. Additionally the results when using the bigram hierarchical Pitman Yor language model are compared to the results when using the unigram Dirichlet distribution only. The WSJ and Xitsonga datasets are used to perform experiments. In the next step, the word segmentation performance using the unsupervised learned acoustic units is evaluated on the Xitsonga dataset only. For this evaluation, the word segmentation is performed on the acoustic unit lattices.

Experimental Results: Acoustic unit discovery performance (Step 1+2)

To evaluate the performance of the acoustic unit learning algorithm, first the Dirichlet process HMM learning algorithm is applied to learn a first set of acoustic units. An

additional HPYLM based bigram language model over the acoustic units is learned and the first best sequence of acoustic units extracted.

Next these units are used to learn the supervised acoustic HMM and estimate the MLLT and FMLLR feature transform as described before.

Finally a new set of acoustic units is learned using the same algorithm used to learn the first set of acoustic units, but this time on the transformed features. Additionally the HPYLM based bigram language model over the acoustic units transcription is applied again. The HPYLM is learned and applied to further improve the acoustic units transcription, as can be seen in the results in Table 6.6.

The acoustic unit discovery performance is evaluated using the normalized mutual information (NMI) on the WSJ and Xitsonga dataset. To calculate the NMI, the speech recordings are decoded into a acoustic unit sequence $\boldsymbol{X}$ using the learned models. The acoustic unit sequence $\boldsymbol{X}$ is then aligned with the reference phoneme sequence $\boldsymbol{Y}$ and each acoustic unit will be assigned a corresponding phoneme from the reference. The NMI is the mutual information normalized by the entropy of the reference and calculated as

$$NMI = \frac{I(\boldsymbol{Y}, \boldsymbol{X})}{H(\boldsymbol{Y})}, \tag{6.66}$$

where $I(\boldsymbol{Y}, \boldsymbol{X})$ is the mutual information between $\boldsymbol{Y}$ and $\boldsymbol{X}$ and $H(\boldsymbol{Y})$ the entropy of $\boldsymbol{Y}$. The NMI will range between 0 and 1, where 0 means that $\boldsymbol{X}$ carries no information about $\boldsymbol{Y}$ and 1 means $\boldsymbol{X}$ perfectly predicts $\boldsymbol{Y}$.

Table 6.6: NMI scores on WSJ and Xitsonga dataset before and after feature transformation with unigram and bigram unit language model.

	WSJ				Xitsonga			
	MFCC		Adapted		MFCC		Adapted	
unit ngram order	1	2	1	2	1	2	1	2
NMI	30.5	31.0	34.9	34.8	28.7	29.8	36.3	37.4

It can be seen that the feature adaptation significantly increases the NMI, especially for the Xitsonga but also for the WSJ dataset. Using a unit bigram language model further increases the NMI for the Xitsonga dataset, leaving the NMI on the WSJ dataset almost unchanged. The larger improvements in the Xitsonga dataset can possibly by explained by the quality of the audio recordings, containing additional background noise in the Xitsonga dataset as compared to the WSJ dataset. The Xitsonga dataset was recorded in a natural environment while the WSJ dataset was recorded in a studio environment without background noise.

Additionally to the NMI, the ABX score is calculated for the Xitsonga dataset to compare to previous results on the ZeroSpeech 2015 challenge data for Track 1. The ABX score is a measure of discriminability of the given features for an acoustic unit

sequence or word. The ABX score is calculated by comparing the similarity of X to A and B and assigning it the class label of A and B, depending on which one is more similar to X and counting the number of false assignments. Additionally the results are divided into within and across speaker ABX scores. The within score is calculated on segments of the same speaker and the across score is calculated across different speakers. This is done to determine the robustness and generalizability of the given features with respect to different speakers.

This setup uses the adapted features, the learned acoustic unit models and the unit bigram language model applied. The ABX score is calculated on the GMM posteriorgrams, leaving out the influence of the HMM. The HMM did not seem to provide further improvements in the ABX score which might be explained by the more coarse features and smoothing of the posteriorgram over time compared to the GMM posteriorgrams.

Table 6.7: Within and across speaker ABX scores on the Xitsonga dataset with the proposed algorithm. Lower scores are better (the ABX score is also called ABX error. Therefore lower values correspond to less errors).

Algorithm	Within	Across
Our	8.4	11.9
Best published [HSN16]	8.0	12.6

It can be seen that the scores of the presented implementation surpass the best published results on the Xitsonga dataset for unsupervised setups across speakers. This shows the advantage of the presented acoustic unit learning algorithm and improvements in combination with the feature transformation.

However there is still a gap when comparing the within and across ABX scores. Nevertheless it is smaller than the best published result. This also indicates that the proposed algorithm and the feature transformation increase robustness in a multi speaker scenario.

Experimental Results: Word Segmentation Performance (Step 3)

The word segmentation performance is evaluated with several combinations of best sequence extraction character model orders, word segmentation character language model orders, and acoustic model scaling factors. The word segmentation word language model order was fixed to one since a higher order did not improve the results. Language model order switching is also not applied since it did not improve the results as well. This might be due to the noisy nature of the learned acoustic units. The acoustic model scaling factor is varied to find the right balance of acoustic model scores and language model scores for the segmentation.

The following parameters are used:

- Best sequence extraction character language model orders: 0, 2, 4, 6
- Word segmentation character language model orders: 2, 4, 6

- Acoustic model scaling factors: 0.2, 0.5, 0.8

Note: a best sequence extraction character language model order of 0 means that that whole lattice is passed to the word segmentation step. Meaning word segmentation is performed on the whole lattice. In all other cases, word segmentation is only performed on the extracted single best sequence. This setup yields 36 different experiments.

As input to the word segmentation algorithm, acoustic unit lattices are extracted using the previously learned acoustic unit models. These lattices are used as input to the word segmentation algorithm.

As evaluation metrics, the Token F-Score and the Type F-Score from the eval2 measures of the ZeroSpeech Challenge are used to compare to the results obtained in the ZeroSpeech 2015 Challenge [Dup16]. Additionally the Coverage (amount of speech data covered and transcribed by the discovered words), Boundary F-Score (based on precision/recall of discovered word boundaries) and Grouping F-Score (based on Grouping precision/recall of discovered word tokens, equivalent to purity and inverse purity scores used to evaluate clustering) ([Dup16]).

The results are given in Figure 6.23 for the Type F-Score and in Figure 6.24 for the Token F-Score. It can be seen, that the Type F-Score and the Token F-Score for the best results increase with increasing best sequence extraction character language model order. The overall best result is achieved with an acoustic model scaling factor of 0.5, a best sequence extraction character language model order of 6 and a word segmentation character model order of 2. This yields a Type F-Score of 4.5 and a Token F-Score of 2.6. These results compare very well to the best Track 2 ZeroSpeech 2015 challenge results for the word discovery task as shown in Table 6.8. Considering the Type F-Score, the obtained results are in the third place, with the best unsupervised system performing at 7.8. The Token F-Score for the obtained results is just shy of the third place at 2.7.

Table 6.8: For comparison, token and type F-Scores where either of the scores is better than the best setup in this chapter. Additionally the coverage, boundary and grouping scores are given.

Algorithm	Token	Type	Coverage	Boundary	Grouping
Best result (this work)	2.6	4.5	99.3	48	21.3
Räsänen et al. [RDF15]	2.7	3.3	94.7	33.5	5
Lyzinski et al. [LSJ15]	0.8	7.8	30.2	14.2	N/A
Kamper et al. [Kam17]	5.0	4.0	100	40.1	4.1
Kamper et al. [Kam17]	3.9	5.4	100	43.7	4.2

The presented system is similar to the system in [RDF15] which also segments the audio signal into acoustic units and discovers words as sequences of recurring units. The result achieved in this work is a remarkable result since the token F-Score is a measure for the transcription quality of the whole audio recordings. The presented algorithm performs second to the best similar system from [RDF15] in terms of token F-Score, achieving almost the same performance in terms of unsupervised transcription of audio recording.

Notably is the higher Type F-Score, meaning the presented result has an overall better lexicon discovery performance. The presented system simultaneously maintains a high Type F-Score and Token F-Score. This observation additionally shows the quality of the obtained results, having very good scores in Type and Token F-Score together, since many systems only perform good in either of the scores.

For comparison the cov (Coverage of the corpus with transcriptions), Boundary F-Score (Measure how accurate the discovered word boundaries are) and grouping F-Score (cleanliness of discovered word cluster) are also given. For almost all this scores the presented system performs best, with an especially high margin in the Grouping F-Score, indicating much cleaner word clusters. It is remarkable that these scores are achieved at a coverage of 99.3% since such a high coverage often comes with a lover F-Score at some of the presented scores.

Additionally noteworthy are the systems presented in [Kam17] and [LSJ15]. In contrast to the system in this work and the system presented in [RDF15], those systems employ whole word models instead of using a hierarchical representation based on acoustic units. The system presented in [LSJ15] uses a similar approach to the dynamic timewarping presented in Section 4.2 but employs different graph clustering techniques. Therefore the Token F-Score is very low since only about 30% of the audio segments are labeled. On the other hand [Kam17] employs whole word embeddings for each word segment instead of acoustic units, loosing the ability to discovery similar sounds within different words and increasing the acoustic model complexity. However, the overall performance of [Kam17] is remarkable but due the whole word embeddings difficult to compare with the results based on acoustic units in this work.

Beside identifying the best performing setup for this work, the results also indicate a dependence on the best sequence extraction character language model. Using the whole lattice in the word segmentation step delivered the lowest performance in this case with very noisy lattices while in the previous experiments with less noisy settings segmentation performed better on whole lattices. Increasing the sequence extraction character language model and therefore only passing the single best sequence improves the overall performance for an additional character language model order of 2. For the additional character language model orders of 4 and 6 only the performance for the word segmentation character language model order 2 seems to increase. Additionally the acoustic model scaling has an influence, indicating that a lower acoustic model scale, meaning deemphasizing the acoustic model, result in higher performance. Overall the result show complex dependencies between the different parameters as the performance change is not monotonically changing.

Conclusions

In this section a system for unsupervised word discovery from audio recordings was presented. The system especially lays focus on the hierarchical modeling of speech, first discovering acoustic units and then words as recurring sequences of acoustic units. The setup in this chapter combines unsupervised learning techniques on all levels into one complete system for fully unsupervised word discovery from audio recordings. Additionally,

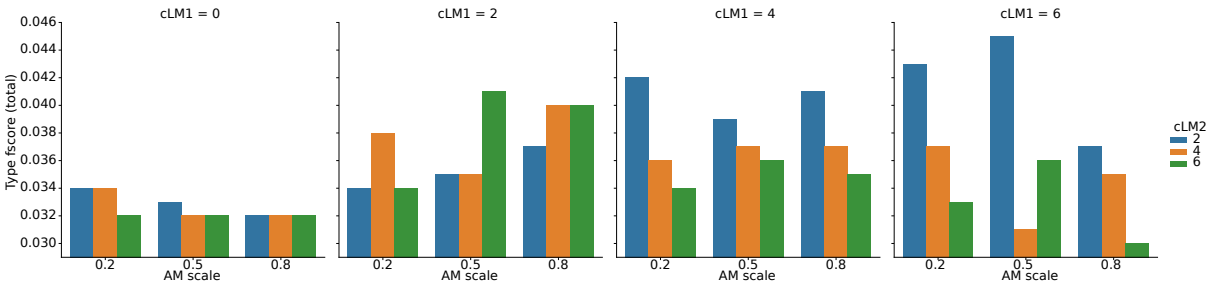


Figure 6.23: Type F-score with varying word segmentation character language model order (cLM2), acoustic model scaling factor and best sequence extraction character language model (cLM1) for Xitsonga dataset.

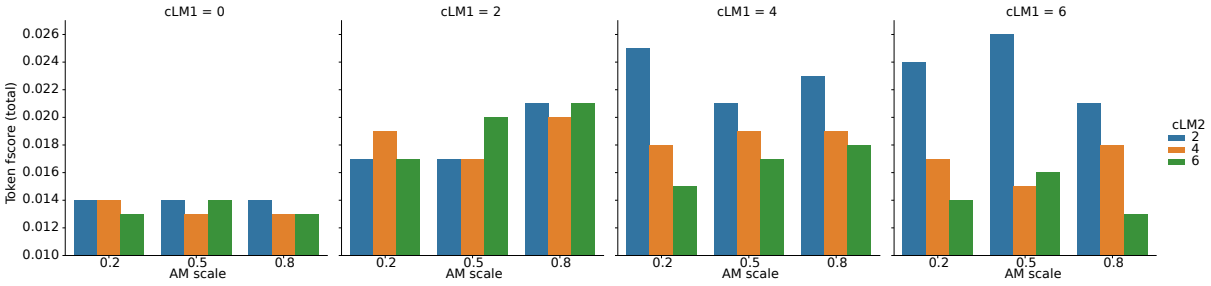


Figure 6.24: Token F-score with varying word segmentation character language model order (cLM2), acoustic model scaling factor and best sequence extraction character language model (cLM1) for Xitsonga dataset.

techniques to derive more robust features against noisy conditions and to increase speaker independence have been applied. The overall system outperforms existing systems in terms of acoustic unit discovery and performance across speakers. On the word discovery level, this systems performs better than other similar systems with a hierarchical modeling approach. The presented system is also one of the very few systems with a hierarchical modeling approach. The system is only outperformed on the word segmentation performance by another system employing word embeddings and whole word models [Kam17]. A combination of both systems would be interesting but was not available at the time the experiments in this section were performed. Similar to other top-down approaches the system in [Kam17] could deliver an initial word discovery result. By using the obtained labeling, the acoustic unit discovery performance could be increased enforcing the same words to have similar unit sequences.

Overall the presented system in this section resembles the classic hierarchical approach employed for speech recognition (acoustic models, lexicon, word language model) and could therefore be more easily combined with a semi supervised learning approach where this algorithm could deliver clusters of words while the semi supervised approach could employ a partial labeling to bootstrap labeling of newly discovered audio segments similar to an existing segment with an label.

7 Unsupervised Acoustic Model Training for Semantic Inference

In this chapter, the unsupervised acoustic model training approach described in Section 5.4.1 is applied to dysarthric-speech recognition for semantic inference in a Vocal User Interface (VUI). In this setup the unsupervised acoustic model training is used learn representations for the audio input, namely an AUD sequence. The AUD sequence is then used in the supervised semantic inference training. Figure 7.1 shows an example for such a vocal user interface, where the user asks to turn on the light.

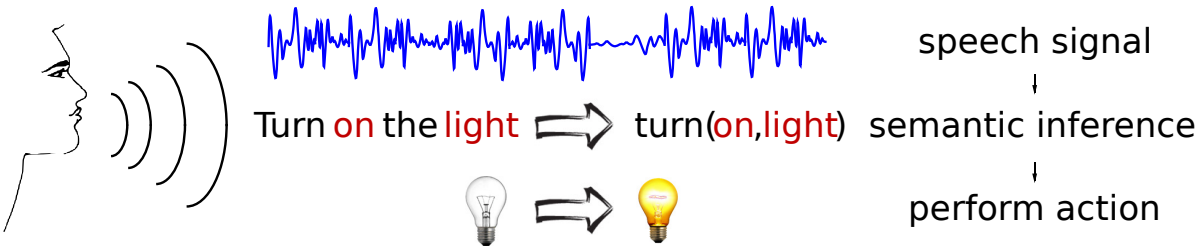


Figure 7.1: User asking the VUI to turn on the light.

This approach adds a next layer of semantic understanding on top of the unsupervised word segmentation and AU learning, connecting AUs and words to meaning. The approach demonstrates that AUs and words learned in an unsupervised way carry semantic information and can be used in further tasks like semantic inference.

Additionally, unsupervised acoustic model training is expected to result in higher performance than standard ASR models, since supervised speech recognizers trained on normal speech are not adapted to the dysarthric speech, demonstrating another use case for unsupervised learning.

For the vocal user interface, the approach described in "A Self Learning Vocal Interface for Speech-impaired Users" [Ons+13] is used. This approach is based on non-negative matrix factorization (NMF) to learn recurring patterns of AUs with minimal supervision, representing semantic concepts.

The experiments were done in cooperation with Bart Ons, Jort F. Gemmeke and Hugo Van hamme from KU Leuven, providing the NMF based vocal user interface implementation [Ons+13] and a speech database, and Vladimir Despotovic, performing follow up experiments with Markov logic networks (MLNs) for semantic inference [DWH15].

The contribution of this thesis is the combination of unsupervised AU learning with the VUI and adapting the VUI to AUs and evaluating different representations of AUs in a semantic inference task. This demonstrates that AUs represent meaningful units which can be used to infer a semantic meaning of a speech utterance while being learned in an unsupervised way.

As representations for the acoustic input, firstly, the Acoustic Units (AUs), which are the hidden Markov models of phone-like units that are trained in an unsupervised fashion as described in Section 5.4.1, and secondly, posteriorgrams over AUs are computed using these AUs models.

As a baseline for an unsupervised representation, frame-based Gaussian posteriorgrams, obtained from Vector Quantization (VQ) are used [Ons+13]. Additionally a speaker-independent phoneme recognizer is used to provide representations, namely phoneme sequence, based on ASR models learned in a supervised way.

Experiments were carried out on a database collected from a home automation task and containing nine speakers, of which seven are considered to utter dysarthric speech. This also delivers a comparison between normal speech and dysarthric speech.

As a result, all unsupervised modeling approaches delivered significantly better recognition rates than a speaker-independent phoneme recognition baseline, showing the suitability of unsupervised acoustic model training for dysarthric speech. While the AU models led to the most compact representation of an utterance for the subsequent semantic inference stage, posteriorgram-based representations resulted in higher recognition rates, with the Gaussian posteriorgram achieving the highest slot filling F-score of 97%.

7.1 Introduction

A speech interface, e.g., to control household devices, can be particularly helpful for physically challenged people [NF92; Haw+13]. Unfortunately, a significant fraction of this group of users also suffers from speech impairments, such as dysarthria, a motor speech disorder, which makes their speech sound quite differently compared to speech uttered by people without speaking impairments.

As a consequence, off-the-shelf automatic speech recognition (ASR) systems exhibit unacceptably high error rates for dysarthric speech [Haw+07]. The deviations from normal speech utterances are usually quite severe and conventional speaker adaptation approaches, such as Maximum-a-posteriori (MAP) or Maximum Likelihood Linear Regression (MLLR) adaption are only able to compensate for these deviations to adapt the acoustic models to some extent to reduce the error rates for impaired speech. A significant amount of research has therefore been devoted to the characterization and recognition of dysarthric speech [RY00; Chr+12; MR11b; Has+06; Gre+03; SH10; Rud11].

An alternative to the adaptation of a speaker-independent system is the training of a speaker-dependent recognizer. This asks for the availability of labeled training data though, i.e., recordings of the user's spoken utterances and the corresponding text files, together with a pronunciation lexicon. In particular in the case of dysarthric speech, pronunciations can be quite different from the standard [San+02; SPK12; MR11a; CT13],

such that the appropriateness of canonical transcriptions is questionable.

In order to avoid the effort for providing an appropriate pronunciation lexicon and transcribing the training data, the ALADIN project follows a different route [Gem+12; Gem+13]. It is concerned with the development of a self-learning vocal interface for a home automation system, where the learning of the acoustic models is done in a "zero-resource" scenario, requiring neither the transcription of the spoken words in the training data nor a pronunciation lexicon. The user still has to follow a training session though, but only to learn the mapping of the user's commands, which he can choose freely, to the action to be carried out in the home automation system. To this end, only weak supervision is required – an action label assigned to an utterance – while no literal transcription of the user's utterance is needed. Thus the system is maximally adapted to the particular artifacts of the user's (dysarthric) speech and to the preferred wording of the user.

This approach, while attractive to the user, poses several challenges, such as unsupervised acoustic model training and the learning of the mapping between the user's utterance and the actions to be performed, using only weak supervision. While the latter has been discussed in [Ons+13], where a non-negative matrix factorization (NMF) based approach was employed to effectively solve the semantic inference problem, this chapter is concerned with the first issue.

In the past years several unsupervised acoustic model training methods have been developed, including Gaussian posteriorgrams [ZG09], hidden Markov model-based self-organising units [Siu+13], and non-parametric Bayesian estimation of HMMs [LG12].

In this chapter, the hierarchical approach described in Section 5.4.1 has been adopted, which has originally been developed for the semantic analysis of the audio track of multimedia data, as described in Section 5.4.1, to perform unsupervised learning of speech representations. On the first hierarchic layer, Acoustic Units (AUs) are learned, phone-like units which are similar to the HMM-based self-organising units in [Siu+13]. The second layer is concerned with the discovery of word-like units, which manifest themselves as recurring sequences of AUs. This approach showed very good performance on the TiDigits corpus with recognition rates coming close to a supervised training [WSH13].

In this chapter, the suitability of unsupervised acoustic learning approaches for dysarthric speech are discussed. This chapter concentrates on two representative approaches, frame based Gaussian posteriorgrams computed from MFCC features as a baseline, and the segment based AUs. Furthermore, posteriorgrams over AUs will also be evaluated.

The chapter is organized as follows: In Section 7.2 a brief overview of the vocal user interface developed in the ALADIN project is given. In Section 7.3, the different feature representations under investigation are introduced. The GMM based posteriorgram is described in Section 7.3.1, while the AU based representation is summarized in Section 7.3.3. The speech database is presented in Section 7.4, followed by Section 7.5 on experimental results. The chapter is finished with a discussion and conclusion in Section 7.5.3.

7.2 Vocal User Interface

This section gives a brief overview of the architecture of the Vocal User Interface (VUI) that has been developed in the framework of the ALADIN project, e.g. for the purpose of controlling a home automation system. The main target group are people who suffer from speech impairment, hence the system should be able to adapt to voice pathologies [Loo+12]. The system is also designed to learn and adapt to unconstrained spoken commands, where the user can formulate a command in the words of his choice.

An action is represented by a semantic frame, a data structure that is composed of slots, which in turn contain values. For example, a semantic frame can contain the slots <device> and <action>, with the corresponding values <television, radio> and <on, off>, respectively.

The mapping between the voice command and the action on the device's user interface is learned during a training phase. The resulting mapping is used in the decoding to recognize the corresponding action for a given voice command. Figure 7.2 shows a block diagram of the training and recognition setup. The training and decoding phase are described in the following. Figure 7.3 shows the NMF based training and decoding algorithm.

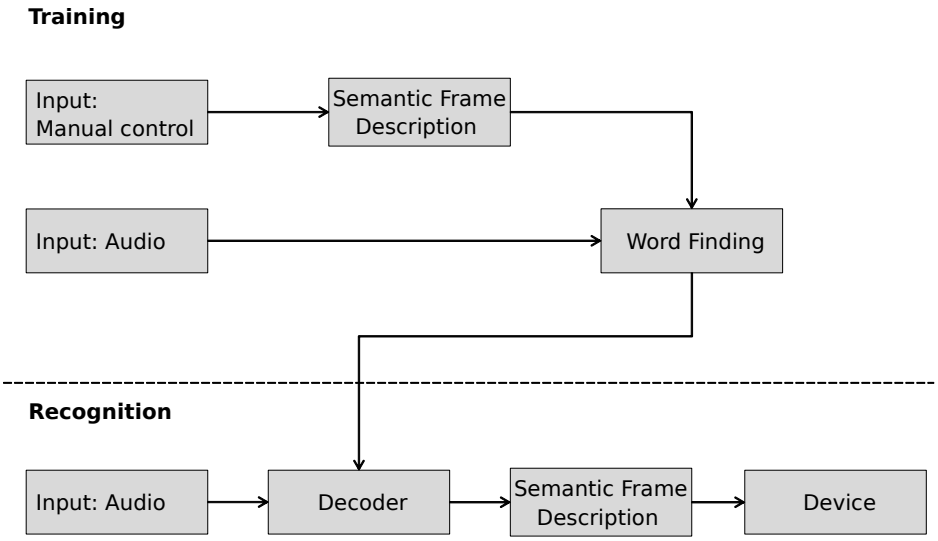


Figure 7.2: Block diagram of VUI training and decoding setup

Training Phase

During the training phase, recurrent acoustic patterns are determined from the spoken commands using NMF [Gem+12] by the word finding module. NMF decomposes a non-negative matrix that represents training data into two lower rank matrices, i.e., a dictionary matrix containing recurrent acoustic patterns $\mathbf{W}$ and a matrix of activations of these patterns $\mathbf{H}$. This is shown in the training step in Figure 7.3). For this the input

audio is represented by corresponding feature vector of fixed size, shown by the light blue part of the $\mathbf{V}$ matrix represented by $\mathbf{V}_1$ in the training step in Figure 7.3. The audio features are described later in Section 7.3. The training process is weakly supervised by augmenting the fixed size vector of representations of the user's utterance with labels indicating the slot values the utterance is referring to, forming a combined supervector. If a slot value is present its value is set to one, if not it is set to zero. The vector of all slot values is added to the fixed sized input audio feature vector, shown by the dark blue part of the $\mathbf{V}$ matrix represented by $\mathbf{V}_0$ in the training step in Figure 7.3). After training, the matrix $\mathbf{W}$ is kept, representing the acoustic model part, while the matrix $\mathbf{H}$ is discarded. The learned matrix $\mathbf{W}$ is further separated into two submatrices, $\mathbf{W}_0$, shown by the dark blue part and representing a dictionary for the slot value labels to reconstruct $\mathbf{V}_0$, and $\mathbf{W}_1$ shown by the light blue part and representing a dictionary for the acoustic feature vectors to reconstruct $\mathbf{V}_1$.

Decoding phase

During the decoding process, only the input audio feature vector describing the user's utterance is decomposed via NMF, represented by the light blue $\mathbf{V}_1$ matrix in the testing step in Figure 7.2), since the labels are unknown. For this only the acoustic model part $\mathbf{W}_1$ obtained in the training step is used. The resulting activation matrix $\mathbf{H}$ is then used to reconstruct the labels $\mathbf{V}_0$, employing the part of the trained dictionary, which included the grounding information $\mathbf{W}_0$ in the recognition step. An estimate of the slot values is obtained by finding the highest values for the labels and the closest match to a known command. This recognized command is then represented by the semantic frame, and finally sent to the target device [GV13].

Summary of NMF based Algorithm

As the mapping of the input utterance to a user command is carried out using NMF, each utterance of the user has to be represented by a vector of fixed size. The compilation and size of these vectors depend on the acoustic representation of the utterance, which will be described in the following section.

Figure 7.3 shows the training and decoding algorithm. The supervision matrix $\mathbf{V}_0$ indicates presence of slot values. It consists of one column per training utterance. The observation matrix $\mathbf{V}_1$ represents utterances as Histograms of Acoustic Cooccurrences (HACs), as described in the following. Again the observation matrix consists of one column per training utterance. In the training step, the "slot value bases" $\mathbf{W}_0$ and the acoustic base vectors $\mathbf{W}_1$ are learned using NMF. The activation matrix $\mathbf{H}^{\text{trn}}$ is discarded after training. The decoding consists of two steps. In the fist step, the activation matrix $\mathbf{H}^{\text{tst}}$ is estimated given the input and the acoustic base vectors $\mathbf{W}_1$. In a next step, the slot values are inferred using the estimated activation matrix and the "slot value bases" $\mathbf{W}_0$. Using the most probable slot values, a semantic frame description can be build and the corresponding action performed. As cost function, the Kullback-Leibler divergence between the observation and reconstruction is used to estimate the matrices

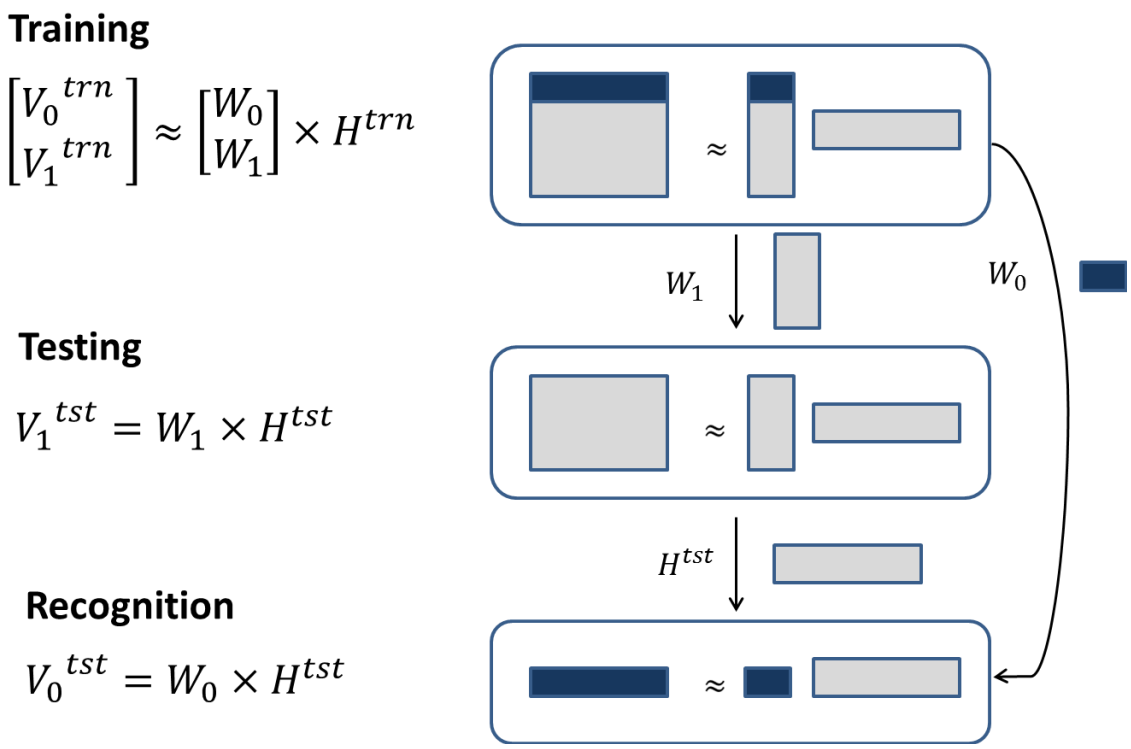


Figure 7.3: VUI training and decoding (testing + recognition) algorithm. $\mathbf{V}_0$ and the dark blue part represent the slot value labels. $\mathbf{V}_1$ and the light blue part represent the acoustic feature vectors. $\mathbf{W}_0$ and $\mathbf{W}_1$ represent the learned bases for slot value labels and acoustic feature vectors respectively. $\mathbf{H}$ represents the activation matrix. $\mathbf{W}_0$, $\mathbf{W}_1$ and $\mathbf{H}$ are estimated using NMF. During training, $\mathbf{V}_0$ and $\mathbf{V}_1$ are known. During decoding only $\mathbf{V}_1$ is known and $\mathbf{V}_0$ is reconstructed using the estimated action matrix $\mathbf{H}$ and $\mathbf{W}_0$ (recognition). The activation matrix $\mathbf{H}$ is estimated from the input $\mathbf{V}_1$ and the bases $\mathbf{W}_1$ (testing).

during training and decoding.

7.3 Acoustic Representations

In this section different representations of the input dysarthric speech are discussed. As the mapping/alignment of the input utterance to a user command is carried out using NMF, see Section 7.2, the utterance will ultimately be mapped to a vector of fixed size, see Section 7.3.1 and Section 7.3.3.

The first step in the audio processing chain is the extraction of Mel Frequency Cepstral Coefficient (MFCC) feature vectors, which are augmented with the log energy and first and second-order temporal difference features to arrive at a 39-dimensional feature vector. Note that cepstral mean and variance normalization is carried out per utterance.

The following different representations of the input dysarthric speech were used.

7.3.1 Gaussian Posteriorgrams

Here, the MFCC feature vector is transformed into a vector of posterior probabilities of Gaussians forming a codebook, using soft vector quantization. To this end, a 100 component full-covariance Gaussian Mixture Model is trained on MFCC vectors. The code book training starts off from a single cluster describing all training data. It is then split along the dominant eigenvector of its covariance matrix, followed by iterations of the Expectation Maximization algorithm. This process is repeated until a desired number of mixture components is obtained.

The posterior probability of each Gaussian mixture component is then computed for each MFCC vector. From the posteriors so-called histograms of acoustic co-occurrences (HACs) are constructed. The HAC is an estimate of the joint posterior probability of two acoustic events happening at a predefined time lag [Van08; SH12].

7.3.2 Contribution

The major contribution in this chapter is the use of acoustic units for semantic inference. The acoustic unit representation was integrated in the voice user interface and corresponding experiments conducted. The goal of this chapter is to apply the learned acoustic units to another upstream task, in this case semantic inference. By this it can be shown that the acoustic units actually carry meaning useful for semantic inference.

7.3.3 AU based representation

Acoustic subword units are meant to capture acoustically consistent phenomena and will be referred to as *acoustic units* (AUs), see Section 5.4.1. They represent similar recurring sequences of feature vectors.

The discovery of AUs is performed in two steps. In the initialization step input speech is segmented and the segments are clustered to generate an initial transcription of the input speech in terms of sequences of segment labels. The second step is the iterative training of hidden Markov models (HMMs) for the discovered clusters. The block diagrams of these steps are depicted in Figure 5.4. For a more detailed description please refer to Section 5.4.1.

Mapping of utterance to fixed-length vector

The AU sequences, which describe the utterance, are not directly usable for NMF. They need to be mapped to a representation of fixed dimension, in which linearity holds, i.e. the utterance-level speech representation is approximately equal to the sum of the speech representations of the acoustic patterns it contains [Van08]. This vector is created by replacing each AU in the recognized sequence of AUs of an utterance by an indicator vector, where the element of the vector representing the AU is set to one and all other

elements to zero. Using all vectors of an utterance a histogram of occurrences and co-occurrences is built and used as input to the NMF.

7.4 DOMOTICA-3 Database

In this work, the DOMOTICA-3 database is employed. It has been collected in the framework of the ALADIN project [Gem+13]. The DOMOTICA-3 database is a collection of recordings of Flemish dysarthric speakers controlling a home automation system. Recordings were collected in two phases. During the first phase users were asked to command 26 distinct actions in a simulated 3D computer animation of a home environment, in order to ensure an unbiased choice of words and grammar by the user. In the second phase speakers were recorded reading these commands to obtain enough repetitions of each spoken command.

In this study only speakers that have uttered at least five repetitions of each command were included. They will be referred to by unique id's 17, 28, 29, 30, 31, 34, 35, 41 and 44. The total number of utterances per speaker was in the range of 151 to 350, with an average of 238. The total size of the database is about 4 hours of speech. Speech intelligibility scores were obtained for all speakers by analysing their recorded speech using an automated tool [Mid12], which led to the conclusion that all except two speakers (id's 17 and 44) were considered to utter dysarthric speech.

7.5 Experiments

The experiments were performed on the DOMOTICA-3 database using the NMF based command recognition framework. The following setups were used (Abbreviations in parenthesis correspond to Table 7.1):

1. Gaussian posteriorgram based representation (GP)
2. AU sequence based representation (AU)
3. AU posteriorgram based representation (AU/HMM and AU/GMM)
4. Phoneme sequences derived using a speaker independent general acoustic phoneme model (Recognizer)

7.5.1 Experimental Setups

For Setup 1 the results were produced by using 100 full-covariance Gaussians trained on all of the speech material available for that speaker as described in Section 7.3.1. From these, the histogram of occurrences and the HACs at four different lags, 2, 5, 9 and 20 frames, were computed. The resulting vector to be forwarded to the NMF-based semantic inference stage was of size $4 \times 100^2 + 100 = 40100$.

For Setup 2, speaker dependent acoustic models of the AUs with an additional silence HMM on the speech material available for that speaker were learned. Each state has one 39-dimensional Gaussian emission density with a diagonal covariance matrix. A zerogram AU language model was used in the first step. The number of AUs per speaker varied between 22 and 98 AUs, depending on the outcome of the unsupervised clustering algorithm described in Section 5.4.1. In a second step, lattices over AUs for each audio recording are produced and the word segmentation algorithm described in Chapter 6 is used to learn a 4-gram language model over the sequence of AUs in an unsupervised way and output a refined sequence of AUs. The word segmentation was discarded, only using the resulting AU sequence, since the resulting words turned out to be too infrequent and variable to learn representative models without overfitting. Then the discovered sequence of AUs, with silence HMMs removed, are used as input to the command recognition algorithm by computing the histogram of occurrences and a HAC with lag 1. The resulting vector was on average of size $50^2 + 50 = 2550$.

For Setup 3 two different posteriorgram representations were derived using the acoustic models of the discovered AUs. The first representation (AU/GMM) was derived by concatenating the Gaussians learned for each state of the HMM to one GMM to again calculate a posteriorgram similar to Setup 1. Each mixture component was assigned the same weight. For the second representation (AU/HMM) the posterior probabilities of being in a certain state of the HMM calculated with the Forward-Backward algorithm are used. In both cases the probabilities of all the states belonging to one HMM are summed to generate an AU based posteriorgram, similar to a phoneme posteriorgram. Vectors in which silence had the highest probability were removed. From the resulting AU based posteriorgrams a histogram of occurrences and HACs at four different lags, 2, 5, 9 and 20 frames, were computed. The resulting vector that was input to the command recognition framework, was on average of size $4 \times 50^2 + 50 = 10050$.

For Setup 4, which served as a baseline for comparison, a pre-trained speaker independent general acoustic model and a zerogram language model were used to decode the audio recordings and produce lattices for each recording. The speaker independent general acoustic model was trained on Dutch speech. A sequence of phonemes was then generated using again the algorithm of Chapter 6 and learning a 4-gram language model in an unsupervised way. The sequence of phonemes was used to compute a HAC-based representation in the same way as was done with the AUs above and then forwarded to the command recognition framework.

7.5.2 Experimental Results

As a performance measure the slot F score of the action recognition was used, which is the harmonic mean of slot precision and slot recall. For its computation a five-fold cross validation procedure was used as described in [Ons+13], with four blocks for training and one for testing.

Figure 7.4 shows the recognized AU sequences for two utterances of the sentence "ALADIN hoofdeinde op stand 1" by speaker 30. Note that the name assigned to an AU is, of course, arbitrary, as no phonetic interpretation can be given to it in an unsupervised

training. The similarity between the two sequences is striking. The AUs can be interpreted as phone-like units and sequences of it as word-like entities. Differences between the two recognized sequences can be viewed as recognition errors or pronunciation variations. Similar observations can be made for all the other speakers as well.

Example 1:
AJ AE AA AC B AF F BJ C H H AH AB AF AC AD BJ C AC F F AD E I AC H AH AB AF F
Example 2:
AJ AE AA AC B AF F BJ C H AH AB AF AC AD E C H BB F AD E I AC H AH AB AF F

Figure 7.4: Recognized AU sequences of two utterances of the same sentence "ALADIN hoofdeinde op stand 1" spoken by speaker 30. Same recognized AU sequences are marked in color.

Figure 7.5 shows the posteriorgrams of the same example utterances obtained by the Forward-Backward algorithm on the AU/HMMs. A certain similarity can again be observed between the two posteriorgrams, indicating that posteriograms on AUs are also a consistent representation of an utterance. Similar observations can again be made for all the other speakers as well.

Table 7.1 shows the slot F-scores of the individual speakers and the average over all speakers (weighted by the relative number of utterances per speaker) for the different setups. Additionally the number of utterances per speaker and the number of discovered acoustic units is shown. The table is ordered so that the left most speaker has the highest intelligibility score while the score decreases when going to the right. Speakers 44 and 17 do not have dysarthric speech.

Table 7.1: F-scores of the different setups; for explanation see text.

Speaker	44	17	34	31	29	28	35	30	41	Average
# Utterances	166	350	335	235	181	214	284	223	151	238
# AUs	98	56	59	38	58	30	53	22	32	50
GP	99.4	99.7	98.8	92.1	99.4	94.0	97.5	93.3	98.0	97.0
AU	95.5	96.9	90.4	79.9	92.7	76.3	94.3	85.3	90.8	89.5
AU/HMM	93.0	96.1	91.3	86.5	95.0	80.0	91.4	88.7	93.5	90.8
AU/GMM	96.3	99.2	97.7	90.5	98.1	89.5	95.7	93.2	94.6	95.3
Recognizer	90.8	87.2	78.70	66.3	84.8	54.2	81.0	56.2	64.8	74.7

7.5.3 Discussion and Conclusion

First of all, the results show that all unsupervised modeling approaches deliver better slot F-scores than the speaker-independent phoneme recognition baseline, showing the

suitability of unsupervised acoustic model training for dysarthric speech. One has to keep in mind that the unsupervised models are learned in a speaker-dependent way, while the speaker-independent phoneme recognizer was not further adapted. The reason for this being twofold: The main goal of this experiment was to show that unsupervised acoustic units indeed carry information suitable for an upstream classification task. On the other hand, adaptation or training of the phoneme recognizer would not be easily possible due to a different language and missing exact transcriptions for the speech commands. Of the unsupervised techniques, the Gaussian posteriorgrams come out first, followed by the posteriorgrams computed from AUs, while the AUs themselves performed clearly worse. It seems that the posteriorgrams are able to capture more information relevant for semantic inference than is available in the mere presence or absence of an AU, since the posteriograms resemble a soft decision.

Note, however, that the Gaussian posteriorgrams are the most expensive description of the utterance in terms of the vector length forwarded to NMF, which was 40100. The AU-based posteriorgrams are coded in a vector of only one quarter of the size, and the AUs in a vector of approximately one fifteenth of the size.

Another interesting observation is that the slot F-score does not monotonously decrease with the intelligibility of the speech. While the speakers are ordered in the table according to decreasing measured speech intelligibility score, the slot F-scores obtained from the various ASR variants are not ordered in the same way. Especially the speakers 29, 35 and 41 achieve higher results than one would expect from their rank according to speech intelligibility, even with the speaker independent recognizer. One reason for this might be that consistency in utterances is more important for the recognition task than intelligibility and that it is not measured in the intelligibility score. However this could not be verified easily due to missing exact transcriptions.

While a definite statement is certainly not possible from this limited dataset, these results nevertheless are encouraging, as they point to the potential of self-learning vocal user interfaces: not only are they superior to off-the-shelf speaker-independent ASR solutions, unsupervised learning approaches have the potential of performing on dysarthric speech as well as on normal speech.

A final interesting observation is that the AU sequences outperform the AU/HMM posteriorgrams for the speakers 44, 17 and 35, while these are also the speakers with a slot F-score above 80% when using the speaker independent phoneme recognizer. Again, this seems to indicate that compact AU sequences can be learned for good speakers more easily that from distorted speech. While for distorted speech the non-HMM models seem to cater better for variability when used in a classification task.

Speakers with less AUs discovered seem to have lower F-Scores on the AUs but also when using the speaker independent model. There seems to be some correlation between the result on the speaker independent model, the number of discovered AUs and the performance on the AUs.

Quantizing in time seems to deteriorate the recognition result, especially restricting the possible transitions by an HMM. This might also be related to the NMF and the input in terms of a histogram (summed over time). So probably the timing information gets summed out.

As a result it can clearly be said that AUs carry semantic information and can be used for semantic inference. Further experiments showing the suitability of AUs in the context of semantic inference were performed using Markov logic networks (MLNs) [DWH15] and further classification approaches [DWH18] by Vladimir Despotovic on provided AU sequences.

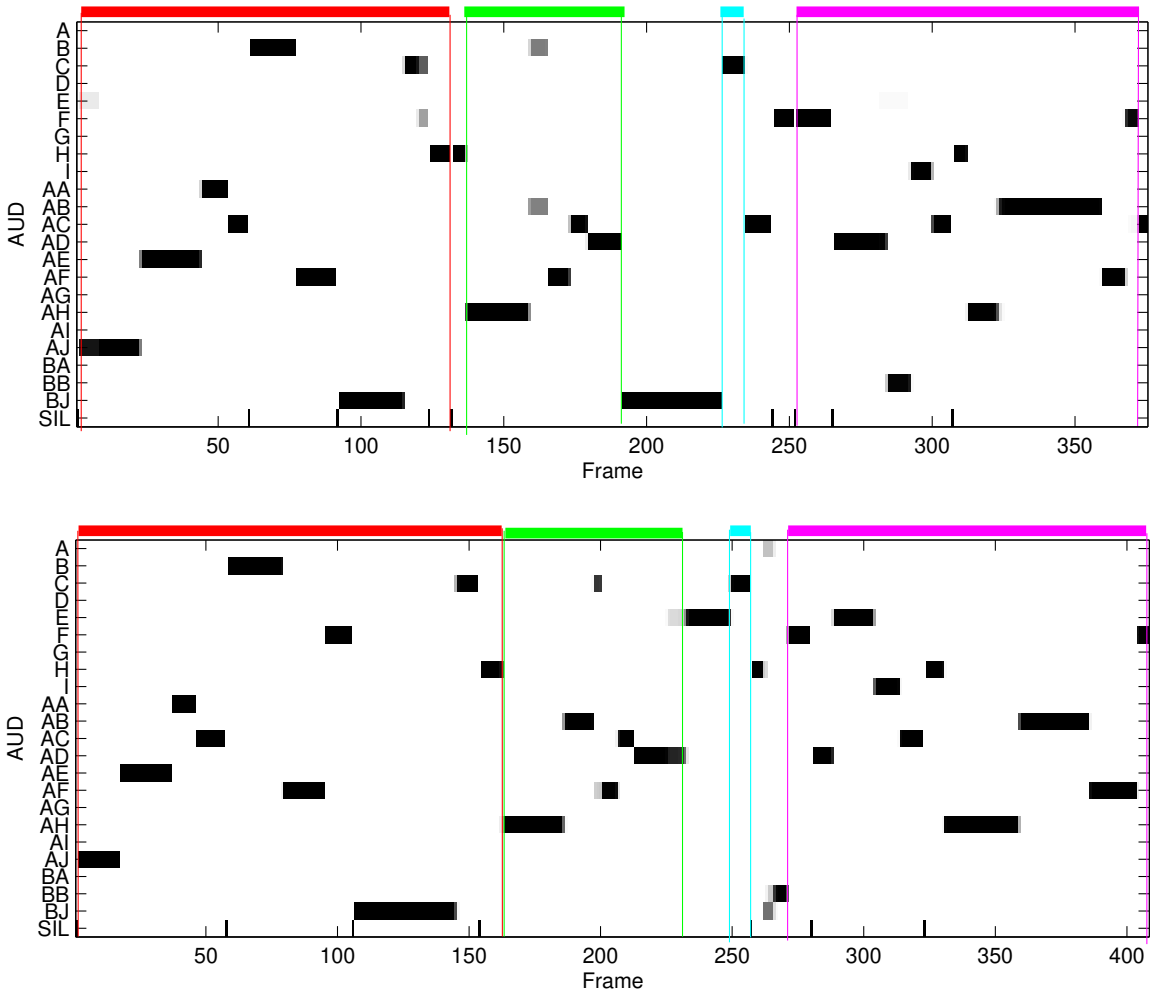


Figure 7.5: AU/HMM: Example 1 (top), Example 2 (bottom). Dark values represent higher values (probabilities). A certain similarity between Example 1 and Example 2 can be seen. The corresponding similar sequences are marked with the same colors as in the example sequences in Figure 7.4. The Y-Axis contains the AU Labels. The X-Axis the frame number. Differences in speaking rate can be observed in the different sequence and utterance lengths. Note that this is only the posteriorgram over the AU/HMM, while Figure 7.4 was extracted after applying the word segmentation with a 4-gram language model. This explains the observation in the first example where in the red sequence the F does not seem to be present and in the magenta sequence the E is very faint. In both cases applying a 4-gram language model seems to boost their probability to result in more consistent sequences over multiple examples.

8 Conclusion

This work presented different approaches to unsupervised language acquisition.

After an intuitive introduction and example to the problem of unsupervised language acquisition in chapter Chapter 4, a heuristic DTW based approach was presented in the same chapter. Building upon state of the art DTW, sDTW and UDTW algorithms, the automatic discovery of similar sequence pairs in different audio segments using UDTW was improved by a local minima based synchronization point search, replacing the originally proposed regular synchronization point pattern with a more explicit initialization based on a local minima search, where a local minimum represents a region of high similarity in the distance matrix of two audio sequences. This resulted in both improving the position of the synchronization points by placing them at a local minimum and ruling out unlikely candidates by not placing any points at arbitrary obviously dissimilar locations originally done by the proposed UDTW synchronization. This local minima based initialization improved the discovery of similar sequences from 56% missed hits to 51.7% for UDTW and 67.2% missed hits to 60.7% for sDTW, all four missed hits rates at a 10% false alarm rate. A small vocabulary task with 11 digits was used in this case.

A two step graph clustering was then applied to the discovered patterns to derive clusters of similar sequences and finally a subsequent unsupervised speech recognizer training, finally demonstrating unsupervised language acquisition and unsupervised speech recognizer training. As a result, the clustering step delivered a word error rate of 24.6%. Subsequent iterative speech recognizer training by first initializing with the clustering results and in the next iterations with the previous decoding results result in a word error rate of 12.9% after 5 iterations of training and decoding.

In chapter Chapter 5, a hierarchical statistical model based approach was introduced and related to traditional automatic speech recognition modeling based on phoneme acoustic models and a lexicon combined with a language model. First another intuitive example was given, followed by an introduction to automatic speech recognition models. Then a hierarchical unsupervised learning algorithm was introduced. Here, another existing algorithm, originally used for unsupervised acoustic event detection, was extended to the statistical modeling used for automatic speech recognition. In a first step, acoustic units are learned, resembling an unsupervised acoustic model training of phoneme models. This step consists of a segmentation step, improved by using a cosine distance to calculate similarity instead of the proposed euclidean distance, as this has already shown better results in the experiments leading to the results in the previous chapter. The segmentation is then again followed by a clustering step, grouping similar units. Again,

the computational demand of the clustering step was reduced by using a more intelligent seed selection based on the k-means++ algorithm and calculating a smaller number of distances only to the selected cluster representatives instead of every sample. Finally, again an iterative acoustic model training consisting of decoding and retraining steps based on the initial clustering result is performed.

The resulting acoustic models were used to decode the audio recordings into acoustic unit sequences. To evaluate the quality of the acoustic units, the average precision and precision-recall break-even was used. The models were trained in a speaker dependent and speaker independent way. The results show that speaker depend modeling naturally delivers higher performance but speaker independent modeling also delivers reasonable performance. The unsupervised acoustic modeling improves performance over plain MFCCs in both cases for the average precision. Especially for the speaker independent task the precision-recall break-even performs better than MFCCs showing the suitability to train speaker independent models. Also acoustic units turned out to be in the same order of length of phonemes, yet about half their length.

In the next step a probabilistic pronunciation modeling approach was used to find similar sequences of acoustic units and cluster them into words. The probabilistic pronunciation model resembles an HMM over discrete acoustic units, where each HMM resembles one word. The original probabilistic pronunciation modeling approach was extended to state dependent emission distributions for the discrete acoustic units, improving the word discovery performance. Additionally, the pattern discovery approach of the previous chapter was used to initialize the HMM model learning, in contrast to the originally proposed random initialization. Additionally an iterative unsupervised speech recognizer training was performed again on the discovered words. This time speaker independent modeling was used instead of speaker dependent models.

As the result, a word accuracy 67.9% for the word discovery step was achieved. Speaker independent modeling and state dependent emission distributions ware used. This is the best result for the speaker independent modeling. For the speaker dependent modeling, the highest results are achieved with Bag-of-AUDs instead. Since the main focus is on speaker independent modeling, this is a good result. The results could be significantly improved by DTW based initialization, resulting in a word accuracy of 81.9%, just shy of the 88.1% achieved with an ideal initialization by ground truth labeling. This shows the remarkable strength of combined acoustic unit based hierarchical modeling with DTW based initialization.

Finally the iterative speech recognizer training results in a remarkable word accuracy of 98.5% for unsupervised acoustic model training, just shy of the 99.4% when using a supervised training. Also the comparison to the 84.7% without DTW initialization demonstrated the strength of combining both unsupervised learning algorithms to achieve good performance. Finally, decoding unseen test data demonstrates that the learned models generalize well with a word accuracy of 98.3% with DTW initialization and 84.3% without initialization DTW initialization.

Leaving the parametric modeling approaches in Chapter 4 and Chapter 5 behind, including the unigram language model, Chapter 6 introduces an n-gram based word segmentation approach based on non parametric character and word n-gram models

connected in a hierarchical way, with the character model being the base of the word model, effectively modeling new (out-of-vocabulary) words based on the character language model. As the non-parametric model, the HPYLM and the resulting NHPYLM are used. This word segmentation approach was first introduced for the segmentation of character sequences into words in written texts in languages which do not use word separators, e.g. whitespaces, like Chinese or Japanese. The word segmentation approach is extended to lattices, based on another existing approach originally only using the learning algorithm to learn a language model to improve decoding results. Here, the algorithm is explicitly used for word segmentation for unsupervised language acquisition on a large vocabulary task. For this the algorithm is extended to exactly model word probabilities based on character sequences, while the original algorithm only used an approximation. Additionally an iterative approach, first extracting the best sequence and then segmenting this sequence, is evaluated.

First, the NHPYLM and the related Pitman-Yor process are introduced. The HPYLM is described in more detail. Finally the Chinese Restaurant analogy is introduced leading to Gibbs sampling equations for the model learning. An additional length modeling for words is introduced, explicitly modeling the word length as a Poisson distribution. The Gibbs sampling based parameter estimation is introduced using a simple example and then followed by the Gibbs sampling equations applied to the problem of word segmentation and language model estimation, also using forward filtering backward sampling to sample a possible word sequence. The equations for different levels of word segmentation, starting from simple phoneme sequence segmentation, continuing with segmentation of audio recordings based on HMM state sequence segmentation and finally unsupervised word segmentation over phoneme lattices.

Next, the implementation of the word segmentation algorithm based on WFSTs is given, first introducing automatic speech recognition with WFSTs. The representation of the input sequence and the lexicon are described concluded by composition operations and best sequence decoding. For word segmentation, the same steps are followed, first introducing the representation of the input sequence, this time including word end markers after each character. A new lexicon will be introduced enabling exact segmentation over lattices in contrast to the original algorithm. Finally a special language model WFST, representing the hierarchy of word HPYLM and character HPYLM forming the NHPYLM is given. Also the language model consists of a novel structure to make it suitable to exactly calculate sequence scores, in contrast to the original implementation which only resembled an approximation. Finally the forward-filtering backward-sampling based on WFSTs is shown. The use of WFSTs allow a very efficient implementation for higher order language models.

Several experimental results are given, beginning with the segmentation of text sequences in English and Austronesian languages. This first experiment investigates different language model orders for the word ans character language model and also investigates the use of length modeling. The results show that word segmentation of text sequences reach very high scores, up to 76% token and lexicon f-score. This experiment also shows the efficiency of pruning infrequent words to increase the token f-score.

In the next experiment word segmentation was performed on lattices generated by a

same language recognizer, demonstrating the influence of noisy recognition. Additionally language model switching was used to demonstrate its effectiveness to improve the segmentation results, first starting with a low order model and then increasing the model complexity to higher order models. Results are given by listing the most frequently discovered words, which are all frequent words in the text as well. Additionally the token and lexicon f-scores are given, together with a recall of the 100 most frequent words. It shows that the presented implementation performs better than an earlier implementation investigated as part of this work by Jahn Heymann. All scores, except of lexicon precision with can be considered equal for both implementations are higher for the implementation in this work, especially the recall at 100 reaches 80 compared to 70. Overall the examples and results show that unsupervised word discovery is well possible to especially discover frequent words with a high performance.

Next, a cross language setup is evaluated employing a pre-trained English speech recognizer. First, an English dataset is evaluated to set a baseline and then the Xitsonga dataset from the Zerospeech 2015 challenge. It can be seen that the presented algorithm performs comparable and partly better than the other algorithms used in the ZeroSpeech 2015 challenge. Also the dependence of the resulting performance on the model orders is evaluated. Overall the performance increases with increasing model order. Additionally a final Viterby decoding step and a step with no character language model are performed, showing that Viterby decoding can additionally improve the performance.

In a last experiment a system with a fully unsupervised learning setup is introduced. The setup uses an existing non-parametric acoustic unit discovery algorithm. The performance of the algorithm is improved in this work by adding automatic and unsupervised feature transformation learning to extract more robust features. The increase in performance is show in the results. Next the word segmentation algorithm is applied and results are shown in comparison to the ZeroSpeech 2015 results. The results show that the presented algorithm delivers a comparable performance to other whole word or heuristic models, with the added advantage of using a hierarchical modeling consisting of acoustic units and words as sequences of units. Additionally for this experiment, the word segmentation algorithm is extended to first perform a best sequence extraction and then perform word segmentation on the best sequence. It can be see that this approach improves the word discovery performance.

The thesis concludes with the application of semantic inference on automatically learned acoustic units in Chapter 7. For this, an existing algorithm is used, but the acoustic representation is replaced by acoustic units to show their suitability for semantic inference. The semantic inference task consists of a simple command and control task and is based on non negative matrix factorization. The experiments show that acoustic units perform comparable to the other representations, but being significantly more compact. A comparison with a standard speech recognizer shows the advantage of unsupervised learning over off-the-shelve models. Often those of-the-shelve models cannot be adapted to given acoustic situations or ways of speaking due to lack of labeled training data. The application of semantic inference shows that unsupervised learning is a suitable way to learn representations for audio recordings which carry meaning.

The contributions and experimental results in this thesis show that unsupervised

learning of phonemes and words is indeed possible. This work could improve over state of the art results by introducing improvements to existing algorithms and transferring algorithms originally used for event detection to the learning of phonemes and words. Extending unsupervised word segmentation algorithms to lattice processing and introducing a modified WFST based representation for the lexicon and language model enabled exact calculation of combined word and character sequence probabilities. This allowed word segmentation on lattices for large vocabulary tasks. Finally using AUs for semantic inference shows that unsupervised learned AUs indeed carry meaning. Results on small vocabulary tasks are already very promising, achieving high word accuracy and low word error rate as shown in Chapter 5 Results on large vocabulary tasks show that word segmentation is indeed possible and can deliver usable results. However there are still improvements to be made before results can be used for high quality speech recognition on large vocabulary tasks. Nevertheless this work consequently followed the path of hierarchical modeling for speech recognition and is one of the first works, if not the first work, presenting a fully unsupervised system learning phonemes and words from unlabeled data in a hierarchical manner.

9 Further Approaches and new Techniques

This thesis mostly documents research work done until April 2017. This chapter will therefore give an overview of further relevant publications and development related to the topic of unsupervised learning for speech recognition.

Firstly, the Zero Resource Speech Challenge should be mentioned here as a series of four challenges in the years 2015 [Ver+15], 2017 [Dun+17], 2019 [Dun+19] and 2020, concerning Zero Ressource Speech Processing. These challenges consist of mainly two tasks in 2015 and 2017: unsupervised subword modeling and spoken term discovery. In contrast, in the 2019, the task of unsupervised subword modeling is the basis for the task "Text to speech without Text", where subword units are learned together with the task of learning a speech synthesizer which performs speech synthesis based on those units, without having any text data or labeling of speech recordings available for training. The main motivation behind this is that to perform speech synthesis, on a compact representation of speech, e.g. acoustic units can be used, similar to the usual character or phoneme sequences in a supervised setup. In contrast to this thesis, the task here differs in such that the units are learned under the condition that they carry information to perform synthesis of speech instead of being used to discovery recurring patterns. The 2020 challenge finally revisits all three previous challenges and asks the participants to submit results on all three tasks and datasets of each challenge. The Zero Resource Speech Challenges have been a constant source of new publications and additional research submitted to various conferences on the given tasks or datasets in the topic of unsupervised learning. Some of the publications in this chapter, but not all, originate from the Zero Resource Speech Challenge.

Research in unsupervised representation learning for speech also began to incorporate algorithms, architectures and ideas from neural networks. Several techniques had been developed and applied in the field of Computer vision but due to computational demand and complexity not been applied to speech.

In [Ebb+17] variational autoencoders in combination with HMMs have been used for Acoustic Unit Discovery. The HMMs were combined with the variational autoencoder to better incorporate temporal dependencies over the latent variables in the variational autoencoder. This setup could improve slightly at about 5-10% relative over an HMM-GMM baseline. In [KLG17] a so called embedded segmental K-means model for unsupervised segmentation and clustering of speech is applied. This model learns

embeddings for variable length segments and at the same time learns a segmentation. In contrast to other models, a segment based embedding is learned. In this publication a very simple embedding was obtained by downsampling segments to a fixed length, while in previous publications several methods have been applied including neural network architectures. The goal of this architecture is to reduce computational demand as much as possible, therefore instead of a fully Bayesian approach, k-means clustering over segment embeddings is used. This is one of the few word discovery systems presented in the Zero Resource Speech Challenges. In [Che+17] multilingual bottle-neck features were learned from untranscribed speech. The bottleneck features are extracted using a neural network. To learn these features, a multiple step approach was developed, first learning phoneme like units with a Dirichlet process GMM and then using those to learn the bottleneck features. [Yua+17] builds on the previous model by extracting bottleneck features and word-like pairs from untranscribed speech for feature representation. In addition to [Che+17] this publication adds a step of finding pairwise similar segments and applies this information to improve the feature learning. This is also known as top down information. In [HSN17] a different approach on feature learning is employed. Here the DPGMM clustering for unsupervised subword modeling is performed in two steps. First labels are learned applying the DPGMM clustering, then improved features are learned using these labels and then another step of DPGMM clusering is performed on the improved features. The fully unsupervised setup described in this thesis was inspired by this approach.

In [Shi+17] so called composite embeddings were extracted by using DNN-HMM and end-to-end speech recognition systems trained in a supervised fashion on high resource corpora and multilingual corpora. Several types of features, PCA-transformed features, bottleneck features and posterior features were extracted employing these systems. A combination of features yielded good performance as representations. In [Ans+17a] acoustic units were learned by first applying a GMM-UBM based clustering and using these cluster labels to learn a DNN based acoustic model. Additionally a second model, where HMM state alignment was applied to derive acoustic units was trained. Finally a multilingual deep bottleneck network was trained to extract bottleneck features for the DNN-HMM training. In [Ans+17b] unsupervised HMM based posteriograms are learned for language independent acoustic modeling in zero resource conditions. This approach is similar to the acoustic unit learning used in this work, with the difference that a hybrid deep neural network acoustic model is applied. In [SRR17] several non-parametric Bayesian Mixture Models for Syllable Clustering were evaluated for Zero-Resource Speech Processing. These models were applied to perform Syllable segmentation. In addition to previous similar models, different model design choices were evaluated. In [PMP17] k-means clustering on was performed to learn feature representations. Also several feature preprocessing methods were applied.

Since 2019 several new neural network based approaches were introduced. Especially noteworthy and widely cited is the approach in [Cho+19], where unsupervised speech representations were learned by training wavenet autoencoders. Wavenet is a neural network based approach for speech synthesis. Using a wavenet approach in an autoencoder therefore directly combines the strength of the wavenet model with unsupervised learning

using an autoencoder. This combination forces the autoencoder-like setup to focus on the most information carrying parts in the input signal and representing those as acoustic units. Therefore, the units also carry more consistent information about the speech signal itself. Several autoencoder setups were evaluated, namely bottleneck networks, variational autoencoders and vector quantized variational autoencoders. Especially the vector quantized autoencoders enable the extraction of discrete acoustic units and force the autoencoder to focus on phonetic information. In [SM19] transcripts were derived from perceptual acoustic units and used for speech synthesis. In [Tja+19] VQVAE (vector quantized variational autoencoders) together with a multi-scale Code2Spec inverter were employed for acoustic unit discovery. [FLP19] combines adversarial training and disentangled speech representation via factorized hierarchical variational autoencoders for robust zero-resource subword modeling. In [Yus+19] temporally-aware acoustic unit discovery using self-organizing maps with special focus on the temporal proximity in addition to the acoustic similarity were applied. Additionally a recurrent sparse autoencoder with clustering on the intermediate softmax layer was used. In [Elo+19] unsupervised acoustic unit discovery for speech synthesis was performed using discrete latent-variable neural networks. Especially convolutional encoding, VQ-VAE discretization, deconvolutional decoding and an FFTNet vocoder were employed. In [LHL19] unsupervised end-to-end learning of discrete linguistic units for voice conversion was performed using multilabel-binary vectors. This especially allowed to separate between linguistic content and speech style, therefore enabling voice conversion to other speakers. In [Gar+19] word discovery in low-resource languages was performed through cross-lingual phonemes combined with more classical DTW approaches to discover word sized units. This is one of the few works tackling the challenge of unsupervised word discovery, similar to this thesis, instead of just representation learning as done in most of the previously cited works. In [MB19] unsupervised acoustic unit discovery was performed with memory-augmented sequence autoencoders.

In [NNK20] vector-quantized neural networks for acoustic unit discovery were trained. Two approaches were investigated: Vector quantized autoencoders and contrastive predictive coding to predict future acoustic units. In [TSN20] transformer VQ-VAE for Unsupervised Unit Discovery and Speech Synthesis was applied together with Transformer-based inverters for the speech synthesis given the extracted codebook. In contrast to most current models, [YO20] uses a Bayesian Subspace HMM for unsupervised acoustic unit discovery. In [MK20] biologically and psychologically motivated neural network modules combined with a variational autoencoder and Dirichlet based clustering were applied for unsupervised unit discovery. In [LEK20] unsupervised feature learning for speech was performed using correspondence autoencoders and triamese networks, as well a a combination of both.

Overall in recent research, publications and challenges the focus is mainly laid on unsupervised acoustic unit learning rather than word discovery as investigated in this thesis. Therefore the unsupervised learning of word-like units, especially in hierarchical models consisting of phone and word models, still is an open field of research. Instead of pure unsupervised learning, the trend nowadays is moving towards transfer learning and self supervised approaches based on unsupervised phoneme models trained on

vast amounts of unlabeled data. A prominent example is wav2vec 2.0 [Bae+20] where pre-training of acoustic units is performed on hundreds of hours of speech and then a fine-tuning step on only several hours of speech is applied. Recently wav2vec-U [Bae+21] has been presented. The approach extends wav2vec 2.0 by additionally leveraging text data from the same language as the audio data to learn a mapping from the acoustic units to phonemes in an unsupervised way.

A Appendix

A.1 NHPYLM Model

A.1.1 Sampling of discount and strength parameter for HPYLM

The parameters $\theta_{|\boldsymbol{u}|}$ and $d_{|\boldsymbol{u}|}$ are resampled according to [Teh06a]. The strength parameter $\theta_{|\boldsymbol{u}|}$ is assumed to have a Gamma distribution $\theta_m \sim \text{Gamma}(\alpha_m, \beta_m)$ as a-priori distribution and the discount parameter $d_{|\boldsymbol{u}|}$ is assumed to have a Beta distribution $d_m \sim \text{Beta}(a_m, b_m)$. Using the auxiliary variables:

$$x_{\boldsymbol{u}} \sim \text{Beta}(\theta_{|\boldsymbol{u}|} + 1, c_{\boldsymbol{u}} - 1) \tag{A.1}$$

$$y_{\boldsymbol{u}i} \sim \text{Bernoulli}\left(\frac{\theta_{|\boldsymbol{u}|}}{\theta_{|\boldsymbol{u}|} + d_{|\boldsymbol{u}|} i}\right) \tag{A.2}$$

$$z_{\boldsymbol{u}wkj} \sim \text{Bernoulli}\left(\frac{j-1}{j - d_{|\boldsymbol{u}|}}\right), \tag{A.3}$$

the parameters $\theta_{|\boldsymbol{u}|}$ and $d_{|\boldsymbol{u}|}$ can be resampled by first sampling values for the auxiliary variables and then resampling them according to

$$d_m \sim \text{Beta}\left(a_m + \sum_{\boldsymbol{u}:|\boldsymbol{u}|=m, t_{\boldsymbol{u}}\geq 2} \sum_{i=1}^{t_{\boldsymbol{u}}-1} (1 - y_{\boldsymbol{u}i}), b_m \sum_{\boldsymbol{u},w,k:|\boldsymbol{u}|=m, c_{\boldsymbol{u}wk}\geq 2} \sum_{j=1}^{c_{\boldsymbol{u}wk}-1} (1 - z_{\boldsymbol{u}wkj})\right) \tag{A.4}$$

$$\theta_m \sim \text{Gamma}\left(\alpha_m + \sum_{\boldsymbol{u}:|\boldsymbol{u}|=m, t_{\boldsymbol{u}}\geq 2} \sum_{i=1}^{t_{\boldsymbol{u}}-1} y_{\boldsymbol{u}i}, \beta_m - \sum_{\boldsymbol{u}:|\boldsymbol{u}|=m, t_{\boldsymbol{u}}\geq 2} \log x_{\boldsymbol{u}}\right) \tag{A.5}$$

A.1.2 Sampling word length

The parameter λ for for the word length distribution is resampled according to [MYU09]. The length distribution parameter λ is assumed to have a Gamma distribution $l \sim \text{Gamma}(a, b)$ as a-priori distribution. The parameter is then sampled from the posterior distribution

$$p(\lambda|\boldsymbol{W}) \propto p(\boldsymbol{W}|\lambda)\, p(\lambda) \tag{A.6}$$

$$= \text{Gamma}(a + \sum_{w\in\boldsymbol{W}} t_w|w|, b + \sum_{w\in\boldsymbol{W}} t_w), \tag{A.7}$$

where $\boldsymbol{W}$ is the set of words and t_w the number of tables holding the word w.

A.1.3 Using of the NHPYLM as a generative model to draw a language model and generate a sample sequence of words

The generative process to draw a language model and to generate a sequence of words works in a similar way as the Gibbs sampling based optimization described in Section 6.1.1. Since the word is not given though, it is split into two steps. First a table is sampled to assign a word to and then the corresponding word. For an existing table the word is determined by the table directly. The table is sampled according to:

$$p\left(k, w|\boldsymbol{u}\right) = \frac{c_{\boldsymbol{u}wk} - d_{|\boldsymbol{u}|}}{\theta_{|\boldsymbol{u}|} + c_{\boldsymbol{u}}}. \tag{A.8}$$

A new table is sampled according to:

$$p\left(k_{\text{new}}|\boldsymbol{u}\right) = \frac{\theta_{|\boldsymbol{u}|} + d_{|\boldsymbol{u}|}t_{\boldsymbol{u}}}{\theta_{|\boldsymbol{u}|} + c_{\boldsymbol{u}}}. \tag{A.9}$$

If a new table is sampled the table is "created" and the corresponding new word is next sampled from the distribution

$$p\left(w|\pi\left(\boldsymbol{u}\right)\right) \tag{A.10}$$

using the same process. This is recursively repeated until the word is generated from an existing or the base distribution. If the base distribution of the word language model is reached, a character sequence is sampled using the same process from the character language model. After sampling, the variables $c_{\boldsymbol{u}wk}$ and respectively $t_{\boldsymbol{u}}$ are adjusted according to the new word. Repeating this process sufficiently long, results in a draw of a language model and additionally one sample sequence of words.

A.2 Example: Gibbs sampling on a bivariate correlated Gaussian distribution

To give an intuitive understanding of Gibbs sampling, the iterative sampling steps of Gibbs sampling are visualized by sampling from a bivariate correlated Gaussian with correlation ρ, zero mean and unit variances and plotting each step in Figure A.1. First an initial sample (green circle) is drawn. Next a new value for x_1, given x_2, is sampled from

$$p\left(x_1|x_2\right) = \mathcal{N}\left(\rho x_2, 1 - \rho^2\right), \tag{A.11}$$

where ρ is the correlation between x_1 and x_2, it was chosen to 0.8 in this example. Sampling this new value for x_1 is depicted as the black line going to the left. Finally a new value for x_2, given x_1, is sampled from

$$p\left(x_2|x_1\right) = \mathcal{N}\left(\rho x_1, 1 - \rho^2\right). \tag{A.12}$$

Sampling this new value is depicted as the black line going up. The new sample from the bivariate Gaussian is depicted as a black circle. This sampling process continues until the desired number of samples is generated. The following 9 samples are depicted with black circles while all 5000 samples generated for this example as depicted in red.

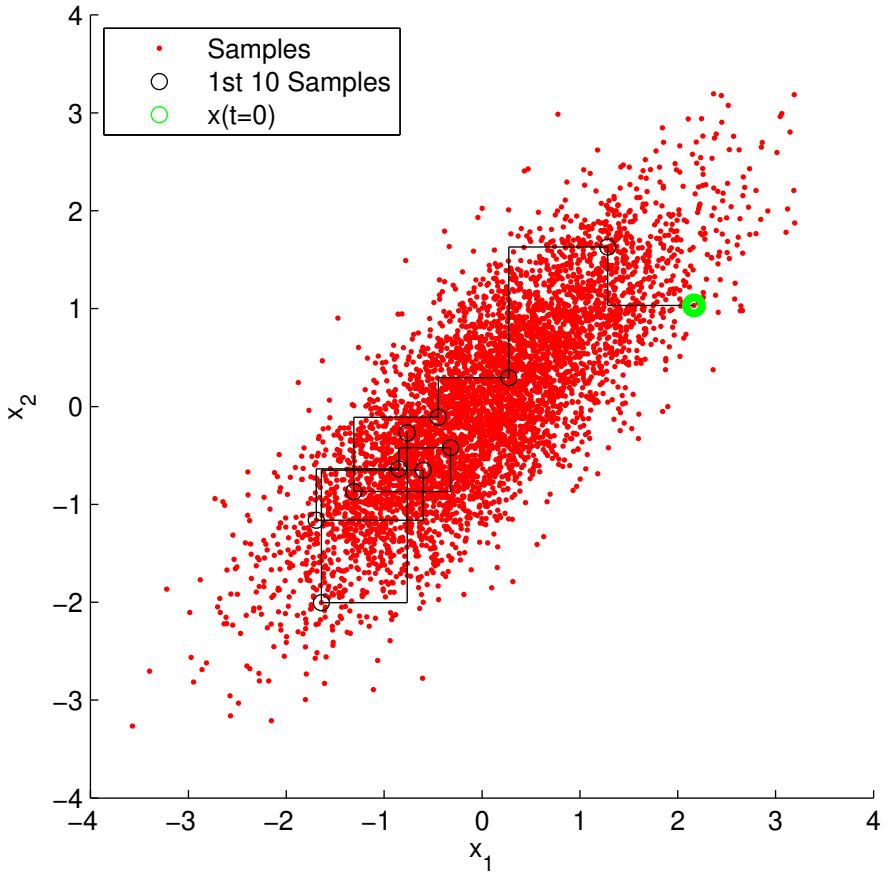


Figure A.1: Drawing samples from a 2D Gaussian distribution using Gibbs sampling. Red: 5000 samples, green: first sample, black: sampling path, black circles: every 2nd step on the sampling path represents a new sample, $\rho = 0.8$

A.3 List of own publications related to this work

A.3.1 First author

O. Walter, T. Korthals, R. Haeb-Umbach, and B. Raj. "A hierarchical system for word discovery exploiting DTW-based initialization". In: *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. Olomouc, Czech Republic, Dec. 2013. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.483.2225&rep=rep1&type=pdf
O. Walter: Algorithmic and experimental design, implementation of and experiments on initialization by pattern discovery and automatic speech recognizer training, main paper editor, scientific supervision of T. Korthals
T. Korthals: Implementation of and experiments on AUD training and word discovery according to baseline system and algorithmic and experimental design, paper editor
R. Haeb-Umbach: Final paper edit, scientific supervision
B. Raj: Final paper edit, scientific supervision
Notes: This paper won a best student paper award

O. Walter, R. Haeb-Umbach, S. Chaudhuri, and B. Raj. "Unsupervised Word Discovery from Phonetic Input Using Nested Pitman-Yor Language Modeling". In: *IEEE International Conference on Robotics and Automation (ICRA 2013)*. Karlsruhe, Germany, 2013. URL: https://groups.uni-paderborn.de/nt/pubs/2013/WaHaChRa2013.pdf
O. Walter: Algorithmic and experimental design, implementation and experiments, main paper editor
R. Haeb-Umbach: Final paper edit, scientific supervision
S. Chaudhuri: initial proposal of general topic
B. Raj: Final paper edit, scientific supervision

O. Walter, V. Despotovic, R. Haeb-Umbach, J. Gemmeke, B. Ons, and H. Van hamme. "An Evaluation of Unsupervised Acoustic Model Training for a Dysarthric Speech Interface". In: *INTERSPEECH 2014*. 2014. URL: https://groups.uni-paderborn.de/nt/pubs/2014/WaDeHaebGeOnVa14.pdf
O. Walter: Algorithmic and experimental design, implementation of and experiments on acoustic unit representation, main paper editor
V. Despotovic: Application of voice user interface on acoustic representations, baseline experiments and inclusion of acoustic unit representations in experiments, paper editor
R. Haeb-Umbach: Final paper edit, scientific supervision
J. Gemmecke: Providing of baseline algorithm and datasets
B. Ons: Providing of baseline algorithm and datasets
H. Van hamme: Providing of baseline algorithm and datasets

O. Walter, R. Haeb-Umbach, B. Mokbel, B. Paassen, and B. Hammer. "Autonomous Learning of Representations". In: *KI - Kuenstliche Intelligenz* (May 2015), pp. 1–13. DOI: http://dx.doi.org/10.1007/s13218-015-0372-1. URL: https://groups.uni-paderborn.de/nt/pubs/2015/WaHaMoPaHa15.pdf
O. Walter: Representation learning (description, algorithm and experiments)

R. Haeb-Umbach: Scientific supervision of representation learning
B. Mokbel: Metric learning (description, algorithm and experiments)
P. Paassen: Metric learning (description, algorithm and experiments)
B. Hammer: Final paper edit, scientific supervision of metric learning
Notes: Journal article

O. Walter and R. Haeb-Umbach. "Unsupervised Word Discovery from Speech using Bayesian Hierarchical Models". In: *38th German Conference on Pattern Recognition (GCPR 2016)*. Sept. 2016. URL: https://groups.uni-paderborn.de/nt/pubs/2016/WaHa16.pdf
O. Walter: Algorithmic and experimental design, implementation and experiments, main paper editor
R. Haeb-Umbach: Final paper edit, scientific supervision

A.3.2 Co-author

J. Heymann, O. Walter, R. Haeb-Umbach, and B. Raj. "Unsupervised Word Segmentation from Noisy Input". In: *Automatic Speech Recognition and Understanding Workshop (ASRU 2013)*. Olomouc, Czech Republic, Dec. 2013. URL: https://groups.uni-paderborn.de/nt/pubs/2013/HeWaHaRa13.pdf
J. Heymann: Algorithmic and experimental design for sequential training, Extension of WFST based Implementation, experiments, main paper editor
O. Walter: Algorithmic and experimental design (baseline system), scientific supervision of J. Heymann, paper editor
R. Haeb-Umbach: Final paper edit, scientific supervision
B. Raj: Final paper edit, scientific supervision of J. Heymann

J. Heymann, O. Walter, R. Haeb-Umbach, and B. Raj. "Iterative Bayesian Word Segmentation for Unspuervised Vocabulary Discovery from Phoneme Lattices". In: *39th International Conference on Acoustics, Speech and Signal Processing (ICASSP 2014)*. May 2014. URL: https://groups.uni-paderborn.de/nt/pubs/2014/HeWaHa2014.pdf
J. Heymann: Algorithmic and experimental design for sequential training, Extension of WFST based Implementation, experiments, paper editor
O. Walter: Algorithmic and experimental design (baseline system), scientific supervision of J. Heymann, main paper editor
R. Haeb-Umbach: Final paper edit, scientific supervision
B. Raj: Scientific supervision of J. Heymann

V. Despotovic, O. Walter, and R. Haeb-Umbach. "Semantic Analysis of Spoken Input using Markov Logic Networks". In: *INTERSPEECH 2015*. Oct. 2015. URL: https://groups.uni-paderborn.de/nt/pubs/2015/DeWaHa.pdf
V. Despotovic: Algorithmic and experimental design for MLN system, implementation and experiments, main paper editor
O. Walter: Acoustic unit representation (Implementation and generation of acoustic

units)
R. Haeb-Umbach: Final paper edit, scientific supervision

T. Glarner, B. Boenninghoff, O. Walter, and R. Haeb-Umbach. "Leveraging Text Data for Word Segmentation for Underresourced Languages". In: *Proc. Interspeech 2017* (2017), pp. 2143–2147. URL: https://www.researchgate.net/profile/Oliver-Walter/publication/319185563_Leveraging_Text_Data_for_Word_Segmentation_for_Underresourced_Languages/links/5b52824845851507a7b6eaa6/Leveraging-Text-Data-for-Word-Segmentation-for-Underresourced-Languages.pdf
T. Glarner: Algorithmic and experimental design, experiments, main paper editor, scientific supervision of B. Boenninghoff
B. Boenninghoff: Implementation of phonetic mapping, experiments, paper editor
O. Walter: Scientific supervision of T. Glarner and B. Boenninghof on WFST based system, Implemented and provided WFST based system
R. Haeb-Umbach: Final paper edit, scientific supervision

V. Despotovic, O. Walter, and R. Haeb-Umbach. "Machine learning techniques for semantic analysis of dysarthric speech: An experimental study". In: *Speech Communication 99 (2018) 242-251 (Elsevier B.V.)* (Apr. 2018). URL: https://groups.uni-paderborn.de/nt/pubs/2018/SpeechCommunication_2018_Walter_Paper.pdf
V. Despotovic: Algorithmic and experimental design, main paper editor
O. Walter: Acoustic unit representation (Implementation and generation of acoustic units)
R. Haeb-Umbach: Final paper edit, scientific supervision
Notes: Journal article

A.3.3 Non-peer-reviewed technical reports

O. Walter, J. Schmalenstroeer, and R. Haeb-Umbach. *A Novel Initialization Method for Unsupervised Learning of Acoustic Patterns in Speech (FGNT-2013-01).* FGNT Technical Report FGNT-2013-01. University of Paderborn Department of Communications Engineering, 2013. URL: https://groups.uni-paderborn.de/nt/pubs/2013/WaScHa2013.pdf
O. Walter: Algorithmic and experimental design, implementation and experiments, main paper editor
J. Schmalenstroeer: baseline system, scientific supervision, paper editor
R. Haeb-Umbach: Final paper edit, scientific supervision

O. Walter, R. Haeb-Umbach, J. Strunk, and N. P. Himmelmann. *Lexicon Discovery for Language Preservation using Unsupervised Word Segmentation with Pitman-Yor Language Models (FGNT-2015-01).* FGNT Technical Report FGNT-2015-01. University of Paderborn Department of Communications Engineering, 2015. URL: https://groups.uni-paderborn.de/nt/pubs/2015/WaHaStHi.pdf
O. Walter: Algorithmic and experimental design, implementation and experiments, main paper editor

R. Haeb-Umbach: Final paper edit, scientific supervision
J. Strunk: Datasets
N.P. Himmelmann: Datasets

Acronyms

ASR Automatic Speech Recognition.

AU Acoustic Unit.

CRP Chinese Restaurant Process.

DAG Directed Acyclic Graph.

DCG Directed Cyclic Graph.

DNN Deep Neural Network.

DTW Dynamic Time Warping.

EM Expectation Maximization.

GMM Gaussian Mixture Model.

HMM Hidden Markov Model.

HPYLM Hierarchical Pitman Yor Language Model.

LCMA Length Constrained Minimum Average.

LVCSR Large Vocabulary Continuous Speech Recognition.

MAP Maximum a Posteriori.

MCMC Markov Chain Monte Carlo.

MFCC Mel-Frequency Cepstral Coefficient.

ML Maximum Likelihood.

NHPYLM Nested Hierarchical Pitman Yor Language Model.

PDF Probability Density Function.

PMF Probability Mass Function.

PY Pitman-Yor.

ROC Receiver Operating Characteristic.

SGD Stochastic Gradient Descent.

UDTW Unbounded Dynamic Time Warping.

WFSA Weighted Finite State Automaton.

WFST Weighted Finite State Transducer.

Symbols

Chapter 4

$\boldsymbol{D}_{a,b}$	Distance matrix for utterances a and b
L^a	Length of feature sequence for utterance a
$\boldsymbol{o}^a_{1:L^a}$	Feature vector sequence of length L^a for utterance a
$\boldsymbol{o}^a_i$	Feature vector i of utterance a
$[\boldsymbol{D}_{a,b}]_{i,j}$	Elements i, j of Distance Matrix
$d(\cdot,\cdot)$	Distance measure between two feature vectors
$\Phi_k = (i_k, j_k)$	Alignment path element k
$\overline{D}(\cdot)$	Cumulative distance
$\overline{D}^{\mathrm{LCMA}}(\cdot)$	LCMA distance
$L^{\min}$	Minimum LCMA path length
$\overline{D}^{\max}$	Maximum distance for path extension
c	Edge weight
$\widetilde{\boldsymbol{D}}_{a,b}$	Smoothed distance matrix
$\boldsymbol{K}$	Smoothing kernel
Λ^{r+1}	Model parameters at iteration $r+1$
T^{r+1}	Set of transcriptions at iteration $r+1$
$\boldsymbol{X}$	Feature sequence set
N^f, N^r, N	Number of found paths, number correctly found paths, number of reference paths

Chapter 5

$w_{1:L}$	Word sequence
L	Word sequence length
w_1	Word
$\boldsymbol{o}_{1:T}$	Observation sequence

T	Observation sequence length
$\boldsymbol{o}_1$	Observation
$x_{1:N}$	Phoneme sequence
x_1	Phoneme
N	Phoneme sequence length
$q_{1:T}$	HMM state sequence
T	HMM state sequence length
q_1	HMM state
$\mathcal{B}'(x_{1:N}, w_{1:L})$	Mapping function from word and phoneme sequences to possible HMM state sequences
a_{01}	HMM state transitions probability
B_{q_t}	Number of Gaussian mixture components
b_t	Gaussien mixture component
$\boldsymbol{\mu}_{\boldsymbol{o}\|b_t,q_t}$	Mean vector
$\boldsymbol{\Sigma}_{\boldsymbol{o}\|b_t,q_t}$	Covariance matrix
$\mathcal{S} = \{S^1, \ldots, S^{N_s}\}$	Set of segments
S^1	Segment
$\Lambda^{\mathcal{A}}$	Set of all AU model parameters
$T^{d,i}$	Transcript of utterance d in iteration i
$\mathcal{A}$	Set of all AUs
$\gamma_t(q_{k,l})$	Posterior probability
$\alpha_t(q_{k,l}$	Forward probability
$\beta_t(q_{k,l})$	Backeard probability

Chapter 6

$w_{1:L}$	Word sequence
$x_{1:N}$	Phoneme sequence
G	Word language model
$\boldsymbol{W}$	Set of sentences
$\boldsymbol{X}$	Set of phoneme sequences
d	Discount parameter
θ	Strength parameter

G_0	Base distribution
$\boldsymbol{u}$	Context
$\pi(\boldsymbol{u})$	Parent word context
$c_{\boldsymbol{u}w}$	Word count
$t_{\boldsymbol{u}w}$	Table count
$\alpha[t][q_t]$	Forward variable
$\beta[t][q_t]$	Backward variable
$\gamma[t][q_t]$	Posterior probability
q_t	HMM State
$\boldsymbol{o}_1$	Observation
$\boldsymbol{O}$	Set of observation sequences
$\boldsymbol{Q}$	Set of HMM sequences

Chapter 7

$\mathbf{W}$	Dictionary matrix
$\mathbf{H}$	Activation Matrix
$\mathbf{V}$	Observation matrix
$\mathbf{W}_0$	Slot values dictionary matrix
$\mathbf{W}_1$	Acoustic feature vector dictionary matrix
$\mathbf{V}_0$	Observe slot values matrix
$\mathbf{V}_1$	Observed acoustic feature vector matrix
$\mathbf{H}^{\text{trn}}$	Training activation matrix
$\mathbf{H}^{\text{tst}}$	Testing/Decoding activation matrix
$\mathbf{V}_0^{\text{trn}}$	Observed training slot values
$\mathbf{V}_0^{\text{tst}}$	Decoded slot values
$\mathbf{V}_1^{\text{trn}}$	Observed acoustic training feature vector matrix
$\mathbf{V}_1^{\text{tst}}$	Observed acoustic test feature vector matrix

List of Figures

List of Tables

Bibliography

[All+07] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri. "OpenFst: A general and efficient weighted finite-state transducer library". In: *International Conference on Implementation and Application of Automata*. Springer. 2007, pp. 11–23.

[AMO10] X. Anguera, R. Macrae, and N. Oliver. "Partial sequence matching using an unbounded dynamic time warping algorithm". In: *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE. 2010, pp. 3582–3585.

[Ana+96] T. Anastasakos, J. McDonough, R. Schwartz, and J. Makhoul. "A compact model for speaker-adaptive training". In: *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP '96*. Vol. 2. Oct. 1996, 1137–1140 vol.2. DOI: 10.1109/ICSLP.1996.607807.

[Ans+17a] T. K. Ansari, R. Kumar, S. Singh, and S. Ganapathy. "Deep learning methods for unsupervised acoustic modeling — Leap submission to ZeroSpeech challenge 2017". In: *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. 2017, pp. 754–761.

[Ans+17b] T. K. Ansari, R. Kumar, S. Singh, S. Ganapathy, and S. Devi. "Unsupervised HMM posteriograms for language independent acoustic modeling in zero resource conditions". In: *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. 2017, pp. 762–768.

[AV07] D. Arthur and S. Vassilvitskii. "k-means++: the advantages of careful seeding". In: *Proc. ACM-SIAM symposium on Discrete algorithms*. 2007, pp. 1027–1035.

[Bae+20] A. Baevski, H. Zhou, A. Mohamed, and M. Auli. *wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations*. 2020. arXiv: 2006.11477 [cs.CL].

[Bae+21] A. Baevski, W.-N. Hsu, A. Conneau, and M. Auli. *Unsupervised Speech Recognition*. 2021. arXiv: 2105.11084 [cs.CL].

[Bel+06] Belo, M. C.A, J. Bowden, J. Hajek, N. P. Himmelmann, and A. V. Tilman. *DoBeS Waima'a Documentation*. MPI Nijmegen. 2002-2006. URL: http://www.mpi.nl/DOBES/.

[Car+11] M. A. Carlin, S. Thomas, A. Jansen, and H. Hermansky. "Rapid Evaluation of Speech Representations for Spoken Term Discovery." In: *INTERSPEECH*. 2011, pp. 821–824.

[CG92] G. Casella and E. I. George. “Explaining the Gibbs sampler”. In: *The American Statistician* 46.3 (1992), pp. 167–174.

[Che+17] 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.

[Cho+19] J. Chorowski, R. J. Weiss, S. Bengio, and A. van den Oord. “Unsupervised speech representation learning using wavenet autoencoders”. In: *IEEE/ACM transactions on audio, speech, and language processing* 27.12 (2019), pp. 2041–2053.

[Chr+12] H. Christensen, S. Cunningham, C. Fox, P. Green, and T. Hain. “A comparative study of adaptive, automatic recognition of disordered speech.” In: *INTERSPEECH*. 2012.

[CHR11] S. Chaudhuri, M. Harvilla, and B. Raj. “Unsupervised Learning of Acoustic Unit Descriptors for Audio Content Representation and Classification”. In: *Proc. INTERSPEECH*. 2011, pp. 2265–2268.

[CK96] C. K. Carter and R. Kohn. “Markov Chain Monte Carlo in Conditionally Gaussian State Space Models”. In: *Biometrika* 83.3 (1996), pp. 589–601. ISSN: 00063444. URL: http://www.jstor.org/stable/2337511.

[CR12] S. Chaudhuri and B. Raj. “Unsupervised Structure Discovery for Semantic Analysis of Audio”. In: *Advances in Neural Information Processing Systems 25*. 2012, pp. 1187–1195.

[CT13] S. O. Caballero-Morales and F. Trujillo-Romero. “Dynamic estimation of phoneme confusion patterns with a genetic algorithm to improve the performance of metamodels for recognition of disordered speech”. In: *Advances in Computational Intelligence*. Springer, 2013, pp. 175–187.

[De +14] N. J. De Vries, M. H. Davel, J. Badenhorst, W. D. Basson, F. De Wet, E. Barnard, and A. De Waal. “A smartphone-based ASR data collection tool for under-resourced languages”. In: *Speech communication* 56 (2014), pp. 119–131.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. “Maximum likelihood from incomplete data via the EM algorithm”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22.

[Dun+17] E. Dunbar, X. N. Cao, J. Benjumea, J. Karadayi, M. Bernard, L. Besacier, X. Anguera, and E. Dupoux. “The zero resource speech challenge 2017”. In: *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. 2017, pp. 323–330.

[Dun+19] E. Dunbar, R. Algayres, J. Karadayi, M. Bernard, J. Benjumea, X.-N. Cao, L. Miskic, C. Dugrain, L. Ondel, A. W. Black, L. Besacier, S. Sakti, and E. Dupoux. *The Zero Resource Speech Challenge 2019: TTS without T*. 2019. arXiv: 1904.11469 [cs.CL].

[Dup16] E. Dupoux. *The Zero Resource Speech 2015 Challenge Results*. 2015 (accessed July 14, 2016). URL: http://www.lscp.net/persons/dupoux/bootphon/zerospeech2014/website/page%5C_5.html.

[DWH15] V. Despotovic, O. Walter, and R. Haeb-Umbach. "Semantic Analysis of Spoken Input using Markov Logic Networks". In: *INTERSPEECH 2015*. Oct. 2015. URL: https://groups.uni-paderborn.de/nt/pubs/2015/DeWaHa.pdf.

[DWH18] V. Despotovic, O. Walter, and R. Haeb-Umbach. "Machine learning techniques for semantic analysis of dysarthric speech: An experimental study". In: *Speech Communication 99 (2018) 242-251 (Elsevier B.V.)* (Apr. 2018). URL: https://groups.uni-paderborn.de/nt/pubs/2018/SpeechCommunication_2018_Walter_Paper.pdf.

[Ebb+17] J. Ebbers, J. Heymann, L. Drude, T. Glarner, R. Haeb-Umbach, and B. Raj. "Hidden Markov Model Variational Autoencoder for Acoustic Unit Discovery." In: *INTERSPEECH*. 2017, pp. 488–492.

[Elo+19] R. Eloff, A. Nortje, B. van Niekerk, A. Govender, L. Nortje, A. Pretorius, E. van Biljon, E. van der Westhuizen, L. van Staden, and H. Kamper. *Unsupervised acoustic unit discovery for speech synthesis using discrete latent-variable neural networks*. 2019. arXiv: 1904.07556 [cs.CL].

[FLP19] S. Feng, T. Lee, and Z. Peng. *Combining Adversarial Training and Disentangled Speech Representation for Robust Zero-Resource Subword Modeling*. 2019. arXiv: 1906.07234 [eess.AS].

[Fra+94] J. Fransen, D. Pye, T. Robinson, P. Woodland, and S. Younge. *WSJCAMO corpus and recording description*. Citeseer, 1994.

[Frü94] S. Frühwirth-Schnatter. "DATA AUGMENTATION AND DYNAMIC LINEAR MODELS". In: *Journal of Time Series Analysis* 15.2 (1994), pp. 183–202. DOI: https://doi.org/10.1111/j.1467-9892.1994.tb00184.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-9892.1994.tb00184.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9892.1994.tb00184.x.

[Gal98] M. Gales. "Maximum likelihood linear transformations for HMM-based speech recognition". In: *Computer Speech & Language* 12.2 (1998), pp. 75–98. ISSN: 0885-2308. DOI: https://doi.org/10.1006/csla.1998.0043. URL: http://www.sciencedirect.com/science/article/pii/S0885230898900432.

[Gal99] M. J. F. Gales. "Semi-tied covariance matrices for hidden Markov models". In: *IEEE Transactions on Speech and Audio Processing* 7.3 (May 1999), pp. 272–281. ISSN: 1063-6676. DOI: 10.1109/89.759034.

[Gar+19] F. García-Granada, E. Sanchis, M. J. Castro-Bleda, J. Á. González, and L.-F. Hurtado. "Word Discovering in Low-Resources Languages Through Cross-Lingual Phonemes". In: *Speech and Computer*. Ed. by A. A. Salah, A. Karpov, and R. Potapova. Cham: Springer International Publishing, 2019, pp. 133–141. ISBN: 978-3-030-26061-3.

[Gem+12] J. F. Gemmeke, J. V. D. Loo, G. D. Pauw, J. Driesen, H. V. hamme, and W. Daelemans. "A self-learning assistive vocal interface based on vocabulary learning and grammar induction". In: *Proc. INTERSPEECH*. 2012, pp. 1–4.

[Gem+13] J. F. Gemmeke, B. Ons, H. Van hamme, J. van de Loo, W. D. G. De Pauw, J. Huyghe, J. Derboven, L. Vugen, B. van Den Broeck, P. Karsmakers, and B. Vanrumste. "Self-taught assistive vocal interfaces : An overview of the ALADIN project". In: *Proc. INTERSPEECH*. 2013, pp. 1–5.

[Gla+17] T. Glarner, B. Boenninghoff, O. Walter, and R. Haeb-Umbach. "Leveraging Text Data for Word Segmentation for Underresourced Languages". In: *Proc. Interspeech 2017* (2017), pp. 2143–2147. URL: https://www.researchgate.net/profile/Oliver-Walter/publication/319185563_Leveraging_Text_Data_for_Word_Segmentation_for_Underresourced_Languages/links/5b52824845851507a7b6eaa6/Leveraging-Text-Data-for-Word-Segmentation-for-Underresourced-Languages.pdf.

[Gop98] R. A. Gopinath. "Maximum likelihood modeling with Gaussian distributions for classification". In: *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181)*. Vol. 2. May 1998, 661–664 vol.2. DOI: 10.1109/ICASSP.1998.675351.

[Gre+03] P. Green, J. Carmichael, A. Hatzis, P. Enderby, M. S. Hawley, and M. Parker. "Automatic speech recognition with sparse training data for dysarthric speakers." In: *INTERSPEECH*. 2003.

[Gre+07] F. Grezl, M. Karafiat, S. Kontar, and J. Cernocky. "Probabilistic and Bottle-Neck Features for LVCSR of Meetings". In: *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*. Vol. 4. 2007, pp. IV-757-IV–760.

[GV13] J. Gemmeke and H. Van hamme. "NMF-Based Keyword Learning from Scarce Data". In: *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. Olomouc, Czech Republic, Dec. 2013.

[Has+06] M. Hasegawa-Johnson, J. Gunderson, A. Penman, and T. Huang. "HMM-based and SVM-based recognition of the speech of talkers with spastic dysarthria". In: *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*. Vol. 3. IEEE. 2006, pp. III–III.

[Haw+07] M. S. Hawley, P. Enderby, P. Green, S. Cunningham, S. Brownsell, J. Carmichael, M. Parker, A. Hatzis, P. O'Neill, and R. Palmer. "A speech-controlled environmental control system for people with severe dysarthria". In: *Medical Engineering & Physics* 29.5 (2007), pp. 586–593.

[Haw+13] M. S. Hawley, S. P. Cunningham, P. D. Green, P. Enderby, R. Palmer, S. Sehgal, and P. O'Neill. "A voice-input voice-output communication aid for people with severe speech impairment". In: *Neural Systems and Rehabilitation Engineering, IEEE Transactions on* 21.1 (2013), pp. 23–31.

[Hey+13] J. Heymann, O. Walter, R. Haeb-Umbach, and B. Raj. "Unsupervised Word Segmentation from Noisy Input". In: *Automatic Speech Recognition and Understanding Workshop (ASRU 2013)*. Olomouc, Czech Republic, Dec. 2013. URL: https://groups.uni-paderborn.de/nt/pubs/2013/HeWaHaRa13.pdf.

[Hey+14] J. Heymann, O. Walter, R. Haeb-Umbach, and B. Raj. "Iterative Bayesian Word Segmentation for Unspuervised Vocabulary Discovery from Phoneme Lattices". In: *39th International Conference on Acoustics, Speech and Signal Processing (ICASSP 2014)*. May 2014. URL: https://groups.uni-paderborn.de/nt/pubs/2014/HeWaHa2014.pdf.

[HSN16] M. Heck, S. Sakti, and S. Nakamura. "Supervised Learning of Acoustic Models in a Zero Resource Setting to Improve DPGMM Clustering". In: *INTERSPEECH*. 2016.

[HSN17] 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)*. 2017, pp. 740–746.

[Jel76] F. Jelinek. "Continuous speech recognition by statistical methods". In: *Proceedings of the IEEE* 64.4 (1976), pp. 532–556.

[JH13] A. Jansen and H. Hermansky. "Weak top-down constraints for unsupervised acoustic model training". In: *Proc. ICASSP*. Vancouver, Ca., May 2013.

[Kam17] H. Kamper. "Unsupervised neural and bayesian models for zero-resource speech processing". In: *arXiv preprint arXiv:1701.00851* (2017).

[Kir+15] Kirihio, J. Karter, V. Unterladstetter, A. Arilaha, F. Morigerowski, A. Loch, Y. Sawaki, and N. P. Himmelmann. *DoBeS Wooi Documentation*. MPI Nijmegen. 2009-2015. URL: http://www.mpi.nl/DOBES/.

[KLG17] H. Kamper, K. Livescu, and S. Goldwater. *An embedded segmental K-means model for unsupervised segmentation and clustering of speech*. 2017. arXiv: 1703.08135 [cs.CL].

[LEK20] P.-J. Last, H. A. Engelbrecht, and H. Kamper. "Unsupervised Feature Learning for Speech Using Correspondence and Siamese Networks". In: *IEEE Signal Processing Letters* 27 (2020), pp. 421–425. ISSN: 1558-2361. DOI: 10.1109/lsp.2020.2973798. URL: http://dx.doi.org/10.1109/LSP.2020.2973798.

[LG12] C.-Y. Lee and J. Glass. "A Nonparametric Bayesian Approach to Acoustic Model Discovery". In: *Proc. of 50th Annual Meeting of the ACL*. Jeju Island, Korea, 2012, pp. 40–49.

[LHL19] A. T. Liu, P.-c. Hsu, and H.-Y. Lee. "Unsupervised End-to-End Learning of Discrete Linguistic Units for Voice Conversion". In: *Interspeech 2019* (Sept. 2019). DOI: 10.21437/interspeech.2019-2048. URL: http://dx.doi.org/10.21437/Interspeech.2019-2048.

[Loo+12] J. V. D. Loo, G. D. Pauw, J. F. Gemmeke, P. Karsmakers, B. Van, D. Broeck, W. Daelemans, and H. V. hamme. "Towards shallow grammar induction for an adaptive assistive vocal interface: a concept tagging approach". In: *Proc. NLP4ITA*. 2012, pp. 27–34.

[LSJ15] V. Lyzinski, G. Sell, and A. Jansen. "An evaluation of graph clustering methods for unsupervised term discovery". In: *Proceedings of Interspeech*. 2015.

[MB19] B. Milde and C. Biemann. "SparseSpeech: Unsupervised Acoustic Unit Discovery with Memory-Augmented Sequence Autoencoders." In: *INTERSPEECH*. 2019, pp. 256–260.

[Met+53] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. "Equation of state calculations by fast computing machines". In: *The journal of chemical physics* 21.6 (1953), pp. 1087–1092.

[Mid12] C. Middag. "Automatic analysis of pathological speech". PhD thesis. Ghent University, Belgium, 2012.

[Mik+13] T. Mikolov, K. Chen, G. Corrado, and J. Dean. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[MK20] T. Morita and H. Koda. *Exploring TTS without T Using Biologically/Psychologically Motivated Neural Network Modules (ZeroSpeech 2020)*. 2020. arXiv: 2005.05487 [cs.CL].

[Moh97] M. Mohri. "Finite-state transducers in language and speech processing". In: *Computational linguistics* 23.2 (1997), pp. 269–311.

[MR11a] K. T. Mengistu and F. Rudzicz. "Adapting acoustic and lexical models to dysarthric speech". In: *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE. 2011, pp. 4924–4927.

[MR11b] K. T. Mengistu and F. Rudzicz. “Comparing humans and automatic speech recognition systems in recognizing dysarthric speech”. In: *Advances in Artificial Intelligence*. Springer, 2011, pp. 291–300.

[MRR80] C. Myers, L. Rabiner, and A. Rosenberg. “An investigation of the use of dynamic time warping for word spotting and connected speech recognition”. In: *ICASSP’80. IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 5. IEEE. 1980, pp. 173–177.

[MS99] C. D. Manning and H. Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

[MYU09] D. Mochihashi, T. Yamada, and N. Ueda. “Bayesian unsupervised word segmentation with nested Pitman-Yor language modeling”. In: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*. Association for Computational Linguistics. 2009, pp. 100–108.

[Nag96] M. Nagata. “Automatic Extraction of New Words from Japanese Texts using Generalized Forward-Backward Search”. In: *Empirical methods in natural language processing*. 1996, pp. 48–59.

[Neu+12] G. Neubig, M. Mimura, S. Mori, and T. Kawahara. “Bayesian Learning of a Language Model from Continuous Speech”. In: *IEICE Transactions on Information and Systems (<A href="http://search.ieice.org/">link)* E95-D.2 (Feb. 2012), pp. 614–625.

[NF92] J. Noyes and C. Frankish. “Speech recognition technology for individuals with disabilities”. In: *Augmentative and Alternative Communication* 8.4 (1992), pp. 297–303.

[NG04] M. E. J. Newman and M. Girvan. “Finding and evaluating community structure in networks”. In: *Physical Review E* 69.2 (Feb. 2004), pp. 026113+. URL: http://dx.doi.org/10.1103/PhysRevE.69.026113.

[NNK20] B. van Niekerk, L. Nortje, and H. Kamper. *Vector-quantized neural networks for acoustic unit discovery in the ZeroSpeech 2020 challenge*. 2020. arXiv: 2005.09409 [eess.AS].

[Ond+17] L. Ondel, L. Burget, J. Černocký, and S. Kesiraju. “Bayesian phonotactic Language Model for Acoustic Unit Discovery”. In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Mar. 2017, pp. 5750–5754. DOI: 10.1109/ICASSP.2017.7953258.

[Ons+13] B. Ons, N. Tessema, J. van de Loo, J. Gemmeke, G. De Pauw, W. Daelemans, and H. Van hamme. “A Self Learning Vocal Interface for Speech-impaired Users”. In: *Proc. Workshop on Speech and Language Processing for Assistive Technologies (SLPAT)*. 2013.

[PB92] D. Paul and J. Baker. “The design for the Wall Street Journal-based CSR corpus”. In: *Proceedings of the workshop on Speech and Natural Language*. 1992.

[PG08] A. Park and J. Glass. “Unsupervised Pattern Discovery in Speech”. In: *Audio, Speech, and Language Processing, IEEE Transactions on* 16.1 (Jan. 2008), pp. 186–197. ISSN: 1558-7916. DOI: 10.1109/TASL.2007.909282.

[Pit+07] M. A. Pitt, L. Dilley, K. Johnson, S. Kiesling, W. Raymond, E. Hume, and E. Fosler-Lussier. “Buckeye corpus of conversational speech (2nd release)”. In: *Columbus, OH: Department of Psychology, Ohio State University* (2007).

[PMP17] T. Pellegrini, C. Manenti, and J. Pinquier. “Technical Report The IRIT-UPS system@ ZeroSpeech 2017 Track1: unsupervised subword modeling”. In: (2017).

[Pov+11] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, et al. “The Kaldi speech recognition toolkit”. In: *IEEE 2011 workshop on automatic speech recognition and understanding*. EPFL-CONF-192584. IEEE Signal Processing Society. 2011.

[Rab89] L. R. Rabiner. “A tutorial on hidden Markov models and selected applications in speech recognition”. In: *Proceedings of the IEEE* 77.2 (1989), pp. 257–286.

[RDF15] O. Räsänen, G. Doyle, and M. C. Frank. “Unsupervised word discovery from speech using automatic segmentation into syllable-like units”. In: *Proc. Interspeech*. 2015.

[RJ86] L. Rabiner and B. Juang. “An introduction to hidden Markov models”. In: *IEEE ASSP Magazine* 3.1 (1986), pp. 4–16. DOI: 10.1109/MASSP.1986.1165342.

[Rud11] F. Rudzicz. “Acoustic transformations to improve the intelligibility of dysarthric speech”. In: *Proceedings of the Second Workshop on Speech and Language Processing for Assistive Technologies*. Association for Computational Linguistics. 2011, pp. 11–21.

[RY00] K. Rosen and S. Yampolsky. “Automatic speech recognition and a review of its functioning with dysarthric speech”. In: *Augmentative and Alternative Communication* 16.1 (2000), pp. 48–60.

[San+02] E. Sanders, M. B. Ruiter, L. Beijer, and H. Strik. “Automatic recognition of dutch dysarthric speech: a pilot study.” In: *INTERSPEECH*. 2002.

[SBH11] J. Schmalenstroeer, M. Bartek, and R. Haeb-Umbach. “Unsupervised learning of acoustic events using dynamic time warping and hierarchical K-means++ clustering”. In: *Interspeech 2011*. 2011. URL: http://nt.uni-paderborn.de/public/pubs/2011/ScBaHa11-2.pdf.

[SH10] H. V. Sharma and M. Hasegawa-Johnson. "State-transition interpolation and MAP adaptation for HMM-based dysarthric speech recognition". In: *Proceedings of the NAACL HLT 2010 Workshop on Speech and Language Processing for Assistive Technologies*. Association for Computational Linguistics. 2010, pp. 72–79.

[SH12] M. Sun and H. V. HAMME. "Coding Methods for the NMF Approach to Speech Recognition and Vocabulary Acquisition." In: *Journal of Systemics, Cybernetics & Informatics* 10.6 (2012).

[Shi+17] H. Shibata, T. Kato, T. Shinozaki, and S. Watanabet. "Composite embedding systems for ZeroSpeech2017 Track1". In: *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. 2017, pp. 747–753.

[Siu+13] M. Siu, H. Gish, A. Chan, W. Belfield, and S. Lowe. "Unsupervised Training of an HMM-Based Self-Organiing Unit Recognizer with Applications to Topic Classification and Keyword Discovery". In: *Comput. Speech Lang.* (2013).

[SM19] K. P. D. S. and H. A. Murthy. "Zero Resource Speech Synthesis Using Transcripts Derived from Perceptual Acoustic Units". In: *Interspeech 2019* (Sept. 2019). DOI: 10.21437/interspeech.2019-2336. URL: http://dx.doi.org/10.21437/Interspeech.2019-2336.

[SPK12] W. K. Seong, J. H. Park, and H. K. Kim. "Multiple pronunciation lexical modeling based on phoneme confusion matrix for dysarthric speech recognition". In: *Advanced Science and Technology Letters* 14 (2012), pp. 57–60.

[SRR17] S. Seshadri, U. Remes, and O. Räsänen. "Comparison of Non-Parametric Bayesian Mixture Models for Syllable Clustering and Zero-Resource Speech Processing". English. In: *INTERSPEECH 2017*. Interspeech. France: ISCA, 2017, pp. 2744–2748. DOI: 10.21437/Interspeech.2017-339.

[Teh06a] Y. W. Teh. "A Bayesian Interpretation of Interpolated Kneser-Ney NUS School of Computing Technical Report TRA2/06". In: *National University of Singapore* (2006).

[Teh06b] Y. W. Teh. "A hierarchical Bayesian language model based on Pitman-Yor processes". In: *Proc. of the 21st Int. Conference on Computational Linguistics and the 44th annual meeting of the ACL*. 2006.

[Teh10] Y. W. Teh. *Dirichlet Process*. 2010.

[Tja+19] A. Tjandra, B. Sisman, M. Zhang, S. Sakti, H. Li, and S. Nakamura. *VQVAE Unsupervised Unit Discovery and Multi-scale Code2Spec Inverter for Zerospeech Challenge 2019*. 2019. arXiv: 1905.11449 [cs.CL].

[TSN20] A. Tjandra, S. Sakti, and S. Nakamura. *Transformer VQ-VAE for Unsupervised Unit Discovery and Speech Synthesis: ZeroSpeech 2020 Challenge*. 2020. arXiv: 2005.11676 [cs.CL].

[Van08] H. Van hamme. “HAC-models: a Novel Approach to Continuous Speech Recognition”. In: *Proc. INTERSPEECH*. 2008.

[Ver+15] M. Versteegh, R. Thiolliere, T. Schatz, X. N. Cao, X. Anguera, A. Jansen, and E. Dupoux. “The zero resource speech challenge 2015”. In: *Proceedings of Interspeech*. 2015.

[Wal+13a] O. Walter, T. Korthals, R. Haeb-Umbach, and B. Raj. “A hierarchical system for word discovery exploiting DTW-based initialization”. In: *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. Olomouc, Czech Republic, Dec. 2013. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.483.2225&rep=rep1&type=pdf.

[Wal+13b] O. Walter, R. Haeb-Umbach, S. Chaudhuri, and B. Raj. “Unsupervised Word Discovery from Phonetic Input Using Nested Pitman-Yor Language Modeling”. In: *IEEE International Conference on Robotics and Automation (ICRA 2013)*. Karlsruhe, Germany, 2013. URL: https://groups.uni-paderborn.de/nt/pubs/2013/WaHaChRa2013.pdf.

[Wal+14] O. Walter, V. Despotovic, R. Haeb-Umbach, J. Gemmeke, B. Ons, and H. Van hamme. “An Evaluation of Unsupervised Acoustic Model Training for a Dysarthric Speech Interface”. In: *INTERSPEECH 2014*. 2014. URL: https://groups.uni-paderborn.de/nt/pubs/2014/WaDeHaebGeOnVa14.pdf.

[Wal+15a] O. Walter, R. Haeb-Umbach, B. Mokbel, B. Paassen, and B. Hammer. “Autonomous Learning of Representations”. In: *KI - Kuenstliche Intelligenz* (May 2015), pp. 1–13. DOI: http://dx.doi.org/10.1007/s13218-015-0372-1. URL: https://groups.uni-paderborn.de/nt/pubs/2015/WaHaMoPaHa15.pdf.

[Wal+15b] O. Walter, R. Haeb-Umbach, J. Strunk, and N. P. Himmelmann. *Lexicon Discovery for Language Preservation using Unsupervised Word Segmentation with Pitman-Yor Language Models (FGNT-2015-01)*. FGNT Technical Report FGNT-2015-01. University of Paderborn Department of Communications Engineering, 2015. URL: https://groups.uni-paderborn.de/nt/pubs/2015/WaHaStHi.pdf.

[WH16] O. Walter and R. Haeb-Umbach. “Unsupervised Word Discovery from Speech using Bayesian Hierarchical Models”. In: *38th German Conference on Pattern Recognition (GCPR 2016)*. Sept. 2016. URL: https://groups.uni-paderborn.de/nt/pubs/2016/WaHa16.pdf.

[WSH13] O. Walter, J. Schmalenstroeer, and R. Haeb-Umbach. *A Novel Initialization Method for Unsupervised Learning of Acoustic Patterns in Speech (FGNT-2013-01)*. FGNT Technical Report FGNT-2013-01. University of Paderborn Department of Communications Engineering, 2013. URL: https://groups.uni-paderborn.de/nt/pubs/2013/WaScHa2013.pdf.

[YO20] B. Yusuf and L. Ondel. *Bayesian Subspace HMM for the Zerospeech 2020 Challenge*. 2020. arXiv: 2005.09282 [eess.AS].

[You+06] S. J. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland. *The HTK Book Version 3.4*. Cambridge University Press, 2006.

[Yua+17] Y. Yuan, C. Leung, L. Xie, H. Chen, B. Ma, and H. Li. "Extracting bottleneck features and word-like pairs from untranscribed speech for feature representation". In: *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. 2017, pp. 734–739.

[Yus+19] B. Yusuf, A. Gök, B. Gundogdu, O. D. Kose, and M. Saraclar. "Temporally-Aware Acoustic Unit Discovery for Zerospeech 2019 Challenge". In: *Proc. Interspeech 2019*. 2019, pp. 1098–1102. DOI: 10.21437/Interspeech.2019-1430. URL: http://dx.doi.org/10.21437/Interspeech.2019-1430.

[ZG09] Y. Zhang and J. Glass. "Unsupervised spoken keyword spotting via segmental DTW on Gaussian posteriorgrams". In: *IEEE Workshop on Automatic Speech Recognition Understanding (ASRU)*. Nov. 2009, pp. 398–403. DOI: 10.1109/ASRU.2009.5372931.

[Zha+14] X. Zhang, J. Trmal, D. Povey, and S. Khudanpur. "Improving deep neural network acoustic models using generalized maxout networks". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. May 2014, pp. 215–219. DOI: 10.1109/ICASSP.2014.6853589.