Sub-Word Based Language Modeling of Morphologically Rich Languages for LVCSR

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Amr Ibrahim El-Desoky Mousa, MSc. Computer Science

aus Kairo, Ägypten

Berichter:
Professor Dr.–Ing. Hermann Ney
Professor Dr. François Yvon

Tag der mündlichen Prüfung: 18. Juni 2014

Abstract

Speech recognition is the task of decoding an acoustic speech signal into a written text. Large vocabulary continuous speech recognition (LVCSR) systems are able to deal with a large vocabulary of words, typically more than 100k words, pronounced continuously in a fluent manner. Although most of the techniques used in speech recognition are language independent, still different languages are posing different types of challenges. Efficient language modeling is considered one of the hard challenges facing LVCSR of morphologically rich languages. The complex morphology of such languages causes data sparsity and high out-of-vocabulary rates leading to poor language model probability estimates. The traditional m-gram language models estimated over the normal full-words are usually characterized by high perplexities and suffer from the inability to model unseen words that are more likely to occur in open vocabulary speech recognition tasks, like open domain dictation and broadcast news transcription.

This thesis addresses the problem of building efficient language models for morphologically rich languages. Alternative language modeling approaches are developed to handle the complex morphology of such languages. This work extensively investigates the use of sub-word based language models using different types of sub-words, like morphemes and syllables, and shows how to carefully optimize their performance to minimize word error rate. In addition, the pronunciation model is combined with the language model through the use of sub-words combined with their context dependent pronunciations forming a set of joint units called graphones. Moreover, a novel approach is examined using extended hybrid language models comprising multiple types of units in one flat model.

Although the sub-word based language models are successful in handling unseen words, still they suffer from the lack of generalization with regard to unseen word sequences. To overcome this problem, morphology-based classes are incorporated into the modeling process to support the probability estimation for sparse m-grams. Examples of such models are the stream-based and class-based language models, as well as the factored language models. A novel methodology is proposed, which uses morphology-based classes derived on the level of morphemes rather than the level of full-words to build the language model. Thereby, the benefits of both sub-word based language models and morphology-based classes are retained.

Moreover, the aforementioned approaches are combined with the efficient state-of-the-art language modeling techniques, like the hierarchical Pitman-Yor language model which is a type of Bayesian language model based on the Pitman-Yor process that has been shown to improve both perplexity and word error rate over the conventional modified Kneser-Ney smoothed m-gram models. In this thesis, hierarchical Pitman-Yor models are used to estimate class-based language models with sub-word level classes.

Recently, continuous space language models have shown significant performance improvements in LVCSR tasks. The continuous nature of such models allows for better levels of generalization due to the inherent smoothing capabilities in continuous space. One of the successful continuous models used in pattern recognition tasks is the feed-forward deep neural network with multiple hidden layers. This model can capture higher-level and abstract information about the input features. Recently, feed-forward deep neural networks have shown improved performance compared to shallow neural networks in many pattern recognition tasks. In this work, the use of feed-forward deep neural networks is explored to estimate sub-word based language models. In addition, word and sub-word level classes are used as inputs to the neural networks in order to improve probability estimation in cases of morphological richness.

The methods applied in this work are tested on Arabic, German and Polish as good examples of languages having rich morphology. Experiments are conducted using the state-of-the-art LVCSR systems used by RWTH Aachen in GALE, Quaero, and BOLT research projects. The methods developed in this thesis reduce the word error rate by up to 7% relative compared to heavily optimized traditional approaches applied on very large vocabulary sizes, typically up to one million words.

Zusammenfassung

Spracherkennung bezieht sich auf die Umwandlung eines akustischen Sprachsignals in einen geschriebenen Text. Spracherkennungssysteme sind heute in der Lage, kontinuierliche Sprache mit einem großen Wortschatz von in der Regel mehr als 100k Wörtern zu erkennen. Obwohl die meisten der in der Spracherkennung verwendeten Techniken unabhängig von der Sprache sind, stellen verschiedene Sprachen in der Regel dennoch weitere Herausforderungen. Insbesondere morphologisch reiche Sprachen stellen harte Herausforderungen an eine effektive Sprachmodellierung. Eine komplexe Morphologie führt in der Regel zu sehr großen Vokabularien bzw. zu einem erhöhten Maß nur selten oder gar nicht gesehener Wörter, sog. *out-of-vocabulary* (OOV) Wörter und damit zu nicht robust schätzbaren Sprachmodellwahrscheinlichkeiten. Die traditionellen m-gram Sprachmodelle über einem festgelegten Vokabular ganzer Wörter sind in der Regel durch hohe Perplexitäten gekennzeichnet, und sind nicht in der Lage, ungesehene Wörter vorherzusagen, wie es viele Spracherkennungsanwendungen, wie Diktiersysteme oder Transkription von Nachrichtensendungen erfordern.

Diese Arbeit behandelt das Problem des Aufbaus effizienter Sprachmodelle für morphologisch reiche Sprachen. Alternative Konzepte zur Sprachmodellierung werden entwickelt, um die Sprachen mit komplexer Morphologie behandeln zu können. Diese Arbeit beinhaltet eine umfassende Untersuchung zur Sprachmodellierung auf Basis von Teilwörtern, wie Morphemen oder Silben, und zeigt auf, wie mit Hilfe dieser Ansätze die Fehlerraten bestehender Systeme verbessert werden können. Auch das Aussprachmodell wird hier mitberücksichtigt, indem die Aussprachen von Teilwörtern kontextabhängig modelliert werden. Der hybride Ansatz zur Sprachmodellierung wird zudem durch die Kombination unterschiedlicher Teilworttypen erweitert.

Obwohl Teilwort-basierte Sprachmodelle nicht explizit im Vokabular enthaltene (OOV) Wörter erfolgreich behandeln, besteht immer noch ein Problem in Bezug auf Ihre Generalisierung auf im Training nicht gesehene Wortfolgen. Zur Behandlung dieses Problems werden morphologisch motivierte Klassen für die Sprachmodellierung herangezogen, um die Schätzung selten gesehener m-gramme zu verbessern. Beispiele solcher Modelle sind Klassen- und *Stream*-Sprachmodelle, sowie faktorisierte Sprachmodelle. Ein neuer Ansatz zur Verwendung morphologisch orientierter Klassen zur Modellierung auf Morphem-Basis, anstatt auf Wortbasis wird vorgestellt. Dies erlaubt, die Vorteile von Teilwort-basierten Sprachmodellen und morphologisch orientierten Klassen auszunutzen.

Die genannten Anstze werden zusätzlich mit aktuellen Ansätzen zur Sprachmodellierung kombiniert. Dies beinhaltet hierarchische Pitman-Yor Sprachmodelle, einen Typus Bayes'scher Sprachmodelle auf Basis des Pitman-Yor Prozesses, für die Verbesserungen in Perplexitt und Wortfehlerrate im Vergleich zum konventionellen modifizierten Kneser-Ney Modell berichtet werden. In dieser Arbeit werden Pitman-Yor Modelle zur Schätzung von Teilwortsprachmodellen herangezogen.

Seit einiger Zeit zeigen sog. *continuous space* Sprachmodelle signifikante Verbesserungen in der kontinuierlichen Spracherkennung bei großem Vokabular. Die kontinuierliche Natur dieser Sprachmodelle lässt eine bessere Generalisierung aufgrund des kontinuierlichen Räumen eigenen Glättungsverhaltens erwarten. Einen erfolgreichen Ansatz stellen hier die aufgeschaltete tiefe neuronale Mehrschichtennetze mit mehreren verborgenen Schichten dar. Dieses Modell erlaubt die Erfassung von übergeordeter Informationen bzw. eine Abstraktion von den Eingabemerkmalen. Tiefe Netzwerke zeigen dabei seit kurzem in vielen Mustererkennungsansätzen deutliche Verbesserungen gegenüber flacheren Netzwerken. In dieser Arbeit werden aufgeschaltete tiefe neuronale Netze zur Sprachmodellierung auf Teilwortebene untersucht. Dies beinhaltet ebenfalls die Verwendung von Wort- und Teilwortklassen als Eingabemerkmale dieser neuronalen Netze, um eine verbesserte Wahrscheinlichkeitsschätzung für morphologisch reiche Sprachen zu erreichen.

Die in dieser Arbeit verwendeten Methoden wurden für Spracherkennungsaufgaben in arabischer, deutscher und polnischer Sprache, als gute Beispiele morphologisch reicher Sprachen, getestet. Die Experimente wurden mit Spracherkennungssystemen der RWTH Aachen durchgeführt, die dem aktuellen Stand der

Technik entsprechen, und in Forschungsprojekten wie GALE, Quaero oder BOLT verwendet wurden. Die in dieser Arbeit entwickelten Methoden reduzieren die Wortfehlerrate um bis zu 7% relativ im Vergleich zu stark optimierten traditionellen Ansätzen bei sehr großem Wortschatz, d.h. in der Regel einem Vokabular von bis zu einer Million Wörter.

Acknowledgement

First of all, I would like to thank my doctoral adviser, Prof. Dr.-Ing. Hermann Ney, head of the Chair of Human Language Technology and Pattern Recognition, Lehrstuhl für Informatik 6, at the RWTH Aachen University, for his support and his interest. He introduced me to his respectful research group in 2007 when I started my studies as a PhD student and he has since then given me the opportunity and the freedom to pursue my ideas.

Special thanks and deep gratitude are due to Dr. Ralf Schlüter for his continuous support and encouragement. His valuable advices helped me to make my decisions and to define my research goals.

I would also like to thank Prof. Dr. François Yvon for agreeing to review this thesis and for the interest in my work.

This work would not have been possible without the never ending support and encouragement of my lovely wife Hala Hassan and sons Zeyad, Ahmed and Yahia, all my deep love to them.

I would like to thank all my colleagues in the speech recognition group for the great teamwork in performing evaluations, discussing ideas, and developing solutions. In no particular order, this includes Mahaboob Ali Basha Shaik, Mahdi Hamdani, Martin Sundermeyer, Markus Nussbaum, Björn Hoffmeister, Stefan Hahn, Christian Plahl, and David Rybach.

For the good times and the memorable moments I had at the Lehrstuhl für Informatik 6, I would like to thank all my former and current colleagues including Muhammad Ali Tahir, Saab Mansour, Tamer Alkhouli, Zoltán Tüske, Patrick Lehnen, Simon Wiesler, Saša Hasan, Georg Heigold, and Jonas Lööf.

Also, my thanks go to our system administration team, and our secretariat for their always available help and their excellent support and patience.

During my time at the Lehrstuhl für Informatik 6, I worked together with other people whom I would like to thank for the fruitful collaborations. Especially; Lidia Mangu, Jeff Kuo, Hagen Soltau, Brian Kingsbury, Ebru Arisoy, and Abdel-rahman Mohamed for the great teamwork during my internship at IBM Watson research center in New York, USA.

Finally, I would like to thank my parents, brothers, and all my family members in Egypt for their understanding and encouragements during the years of my doctoral studies.

This work was realized as part of the following programmes: the Defense Advanced Research Projects Agency (DARPA) BOLT Programme under Contract No. HR0011-12-C-0015, Quaero Programme funded by OSEO French State agency for innovation, and DARPA GALE Programme under Contract No. HR001-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of those agencies.

Contents

1 Introduction **1**
- 1.1 Challenges in ASR . . . 2
- 1.2 Statistical Speech Recognition . . . 2
- 1.3 Feature Extraction . . . 3
- 1.4 Acoustic Model . . . 4
- 1.5 Language Model . . . 7
- 1.6 Search . . . 9
- 1.7 Multi-Pass Search . . . 10
  - 1.7.1 N-Best Lists and Lattices . . . 10
  - 1.7.2 Confusion Networks . . . 11
  - 1.7.3 Speaker Adaptation . . . 12
- 1.8 Model and System Combination . . . 13
  - 1.8.1 Log-linear Model Combination . . . 13
  - 1.8.2 System Combination . . . 13
  - 1.8.3 Cross-Adaptation . . . 14
- 1.9 Morphologically Rich Languages . . . 14
  - 1.9.1 Arabic . . . 14
  - 1.9.2 German . . . 15
  - 1.9.3 Polish . . . 15
- 1.10 Language Modeling: the State of the Art . . . 15
  - 1.10.1 Sub-Word Based LMs . . . 16
  - 1.10.2 Word Classes in LMs . . . 17
  - 1.10.3 Novel LM Estimation . . . 18

2 Scientific Goals **21**

3 Sub-Word Based Language Models **25**
- 3.1 Sub-Word Based m-gram Models . . . 25
- 3.2 Sub-Word Units . . . 27
  - 3.2.1 Morphemes . . . 27
  - 3.2.2 Syllables . . . 27
  - 3.2.3 Graphones . . . 28
  - 3.2.4 Arabic Diacritized Sub-Words . . . 28
- 3.3 Word Decomposition . . . 28
  - 3.3.1 Supervised Morphological Decomposition . . . 29
  - 3.3.2 Unsupervised Morphological Decomposition . . . 30
  - 3.3.3 Syllabification . . . 31
- 3.4 Sub-Word Units Combined with Pronunciations . . . 32
  - 3.4.1 Grapheme-to-Phoneme Conversion . . . 32
  - 3.4.2 Graphones as Recognition Units . . . 33
  - 3.4.3 Letter-Phoneme Sequence Alignment . . . 33
- 3.5 Experimental Results . . . 34
  - 3.5.1 Experiments on Arabic . . . 34
  - 3.5.2 Experiments on German . . . 37
  - 3.5.3 Experiments on Polish . . . 40
  - 3.5.4 Overview of Experimental Results . . . 42
- 3.6 External Evaluations . . . 46
  - 3.6.1 Quaero German ASR Evaluation 2010 . . . 46

3.6.2 Quaero German ASR Evaluation 2011 . . . 46
3.6.3 Quaero German and Polish ASR Evaluation 2012 . . . 46
3.6.4 Quaero German ASR Evaluation 2013 . . . 47
3.6.5 IWSLT German ASR Evaluation 2013 . . . 47
3.6.6 OpenHaRT Arabic Handwriting Recognition Evaluation 2013 . . . 48
3.7 Summary . . . 48

4 Language Modeling with Morphology-Based Classes **51**
4.1 Generating Classes . . . 52
4.1.1 Morphology-Based Classes for Arabic . . . 52
4.1.2 Morphology-Based Classes for German . . . 53
4.1.3 Data-Driven Word Clustering . . . 53
4.2 Stream-Based m-gram Models . . . 54
4.3 Class-Based m-gram Models . . . 55
4.4 Factored Language Models . . . 56
4.5 Hierarchical Pitman-Yor Language Models . . . 58
4.6 Combining Multiple Language Models . . . 60
4.6.1 Linear Interpolation . . . 60
4.6.2 Score Combination . . . 60
4.7 Experimental Results . . . 61
4.7.1 Optimization of Factored Language Models . . . 61
4.7.2 Experiments on Arabic . . . 64
4.7.3 Experiments on German . . . 70
4.7.4 Overview of Experimental Results . . . 73
4.8 Summary . . . 76

5 Deep Neural Network Language Models **77**
5.1 Continuous Space Language Models: An Overview . . . 78
5.2 Feed-Forward Neural Network Language Models . . . 80
5.2.1 Shallow Neural Network Language Model . . . 80
5.2.2 Deep Neural Network Language Model . . . 82
5.2.3 Deep Neural Network Language Model with Classes . . . 82
5.2.4 Lattice Rescoring . . . 83
5.3 Back-Propagation Training Algorithm . . . 85
5.3.1 Weight Decay Regularization . . . 86
5.3.2 Stochastic Back-Propagation . . . 86
5.3.3 Computational Complexity . . . 87
5.4 Pre-Training Strategies . . . 87
5.5 Speeding Up Techniques . . . 89
5.5.1 Lattice or N-best Rescoring . . . 89
5.5.2 Regrouping Probability Requests . . . 89
5.5.3 Vocabulary Truncation . . . 89
5.5.4 Bunch Mode . . . 90
5.5.5 Resampling the Training Data . . . 90
5.6 Generating Morphemes and Classes for Egyptian Arabic . . . 90
5.6.1 Word Decomposition . . . 91
5.6.2 Class Derivation . . . 91
5.7 Experimental Results . . . 91
5.8 Summary . . . 96

6 Scientific Contributions **97**

7 Outlook **101**

A Corpora and Systems **103**

A.1 Development and Evaluation Corpora . . . 103
A.2 Modern Standard Arabic Testing System . . . 104
A.3 German Testing System . . . 104
A.4 Polish Testing System . . . 105
A.5 Egyptian Colloquial Arabic Testing system . . . 105

B Symbols and Acronyms **107**
B.1 Mathematical Symbols . . . 107
B.2 Acronyms . . . 109

List of Figures **113**

List of Tables **115**

Bibliography **119**

Chapter 1

Introduction

Speech is considered as the most natural way of communication among humans. This makes an automatic speech recognition (ASR) system a natural choice for a human-machine interaction. In recent years, a huge amount of audio and video data became available on the Internet. Most of this material use speech as the natural form of communication. Therefore, ASR has been a goal of research for more than six decades as it is very useful in many applications and environments. For example, in industrial applications, mobile applications, telephone and communication applications, aviation and space applications and also in personal computer applications. In fact, ASR is the first step to make the information contained in the speech data available for machine processing.

The speech recognition problem is defined as the task of decoding an acoustic speech signal into a written text (the recognized word sequence). The automatic speech recognizer serves as a human-machine interface or it passes the input for further processing like machine translation. According to the given task, an ASR system has to fulfill certain requirements, e.g. an ASR system which serves as a human-machine interface has to work in real-time. The ASR systems considered in this thesis are large vocabulary continuous speech recognition (LVCSR) systems. This means that, according to nowadays standards, the system vocabulary consists of more than 100k words, and recognition is performed on complete natural utterances (in contrast to the isolated or connected word recognition), while real-time is not required. In modern LVCSR systems, a preprocessing step is performed by applying signal analysis to convert the speech signal into a sequence of feature vectors. Then, a statistical approach is used to find the highest probable sequence of words given the acoustic features. The standard evaluation measure for LVCSR systems is the word error rate (WER). The goal of the ASR system is to minimize the WER measured on the decoded output.

In recent years, the research on developing LVCSR systems has been extended to include an increasingly wide range of languages. Different types of problems associated with the development of LVCSR systems have received prominent research attention such as acoustic modeling, and language modeling. Although the statistical approach to speech recognition is mainly language independent, still some language specific properties are highlighting specific modeling challenges. Currently, there is a growing interest in language modeling approaches that are suitable for morphologically rich languages. Languages falling in this category are characterized by a huge lexical variety as a large number of distinct lexical forms can be generated using various morphological processes like derivation, inflection and compounding. At the same time, in many LVCSR tasks, systems are required to operate over open and constantly changing vocabularies; like in the open domain dictation, broadcast news transcription, political debates translation, etc. This means that the number of recognizable words is supposed to be unlimited. For languages with a fairly poor morphology, language models (LMs) built on the word level are proven successful. Applying the same approach to languages with rich morphology leads to high out-of-vocabulary (OOV) rates, and poor LM probability estimates due to data sparsity. Hence, the LM fails to perform reliable generalization or to model unseen words. Moreover, the LVCSR systems suffer from high resource requirements such as CPU time and memory as a result of the huge lexicons and LMs that are usually required in such cases to achieve a reasonable lexical coverage.

In this thesis, a major attention has been paid to develop improved approaches to deal with the problems related to morphologically rich languages. The use of sub-lexical LMs has been extensively investigated based on different types of sub-word units, like morphemes and syllables. At the same time, it has been shown how to optimize the performance of such LMs to minimize the WER. In addition, novel types of sub-lexical units are efficiently incorporated into LMs, where sub-words are combined with their context dependent pronunciations to form a set of joint units. This can be viewed as a combination of language

and pronunciation model, in which the context dependent pronunciations of the underlying sub-words are taken into account. Moreover, a novel approach has been developed that uses extended hybrid LMs comprising multiple types of units in a single flat model. In addition, a novel technique is proposed that incorporates various morphology-based classes derived on the morpheme level into the LM estimation. Thereby, the benefits of sub-lexical language modeling along with the advantages of morphology-based modeling are retained together. To push the performance to the highest level, the above approaches have been combined with the recent state-of-the-art language modeling techniques, like the hierarchical Bayesian LMs based on the Pitman-Yor process and the continuous space LMs based on feed-forward neural networks (NNs). Particularly, feed-forward deep neural networks (DNNs) have been explored to estimate sub-lexical and morphology-based LMs. Our experiments are conducted using the state-of-the-art LVCSR systems for Arabic, German and Polish as good examples of languages having rich morphology.

1.1 Challenges in ASR

The ultimate long-term goal of an ASR system is to act as a "hearing machine" in the sense that, for any spoken utterance, it converts the acoustic signal into a sequence of written words. The major challenges in unrestricted, continuous speech recognition are:

- In the acoustic signal, there is no indication of the word or sub-word boundaries.
- There is a large degree of variation in the speaking rates in continuous speech.
- words are pronounced less carefully in fluent speech specially the word endings.
- There is a great deal of inter-/intra-speaker variability due to gender, physiological and psychological conditions.
- The quality of the speech signal is affected by environmental noise or the transmission system such as microphone or telephone.
- The task-inherent syntactic/semantic constraints of the language should be exploited by the recognition system in a way similar to human-to-human communication.

1.2 Statistical Speech Recognition

The statistical approach to ASR takes a sequence of acoustic feature vectors x_1^T as input and searches for the optimum word sequence w_1^N that maximizes the posterior probability. This approach applies the Bayes decision rule [Bayes 1763]:

$$\begin{aligned} \left[w_1^N\right]_{opt} &:= \operatorname*{argmax}_{w_1^N} p(w_1^N|x_1^T) \\ &= \operatorname*{argmax}_{w_1^N} p(x_1^T|w_1^N)p(w_1^N) \end{aligned} \tag{1.1}$$

The result is referred to as the maximum a-posterior (MAP) hypothesis. The equation defines two stochastic models, the acoustic model $p(x_1^T|w_1^N)$ and the language model $p(w_1^N)$. The acoustic model computes the probability of observing the feature sequence x_1^T given the word sequence w_1^N. The language model denotes the a-priori probability of the word sequence w_1^N.

The extraction of the feature sequence x_1^T from the continuous speech signal happens in a preprocessing step using signal analysis techniques. Usually, the feature extraction models are based on models of the human auditory system. The resulting features are further processed by data-driven approaches in order to provide the sequence of feature vectors x_1^T.

Figure 1.1 summarizes the interaction between feature extraction, acoustic model, and language model during the ASR search. The search algorithm aims at finding the word sequences that fulfills Equation (1.1). The search space for a LVCSR system consists of all possible word sequences over a (finite) vocabulary. This is normally a huge search space which makes the complete exploration prohibitive

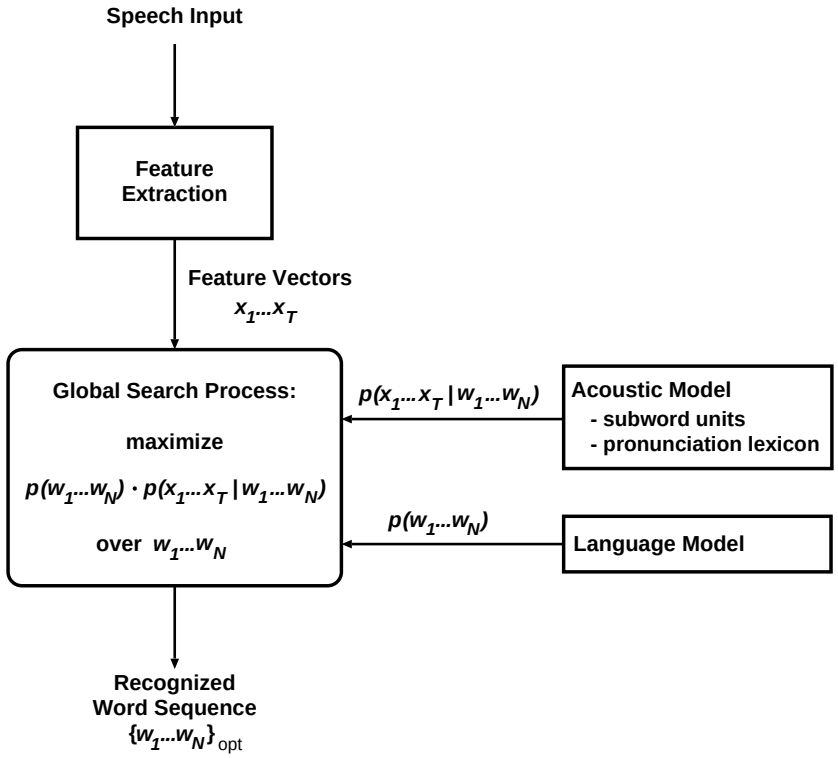


Figure 1.1. Basic architecture of a statistical automatic speech recognition system according to [Ney 1990].

and thus pruning techniques are used to restrict the effective number of hypotheses. The subset of the search space considered during the search process can be stored in a form of lattices or N-best lists and used for applying sophisticated methods that are too complex to be applied to the full search space.

1.3 Feature Extraction

The feature extraction module of the ASR system is based on signal analysis and provides the statistical model with a sequence of observations or acoustic vectors. The objective of this module is to keep only the information from the speech signal that is relevant for finding the correct word sequence. Discarding all the irrelevant information makes the acoustic model robust e.g. to the intensity of the speech, to background noise and to speaker gender and identity. The state-of-the-art feature extraction used for LVCSR systems works in the following three steps:

1. A first set of features is extracted from the speech signal based on models of the human auditory system.
2. The features are transformed, augmented, and/or reduced by parametric models, where the model parameters are estimated on the acoustic training data.
3. Speaker normalization steps are applied either to the feature space or to the acoustic model parameters in order to achieve speaker independence; usually the free parameters are estimated based on the results of a preceding unadapted recognition run.

The most common signal analysis applied in the first step is based on a short term spectral analysis, usually a fast Fourier transform (FFT) [Rabiner & Schafer 1979]. Widely used procedures for further processing the FFT results yield the Mel-Frequency Cepstral Coefficients (MFCCs) [Davis & Mermelstein 1980] or the perceptual linear predictive (PLP) coefficients [Hermansky 1990]. Another commonly used features by RWTH Aachen are the features based on Gammatone (GT) filter, which works in the time domain [Aertsen & Johannesma$^+$ 1980; Schlüter & Bezrukov$^+$ 2007]. The recognition performance can

be significantly improved by concatenating articulatory motivated acoustic features to the short-term FFT-based features [Kocharov & Zolnay+ 2005; Zolnay & Schlüter+ 2005].

An alternative recent approach is the usage of phone posterior probability estimates as acoustic features. In this approach, features from the first step are feed into a classifier, usually a multilayer perceptron (MLP) neural network, which provides the posterior estimates at the output layer [Chen & Zhu+ 2004; Hermansky & Ellis+ 2000; Valente & Vepa+ 2007]. The parameters of the classifier are estimated on the training data.

Dynamic information can be included by augmenting the feature vector with the first and second derivatives. A more general approach is to apply linear discriminant analysis (LDA) [Fisher 1936] or heteroscedastic LDA (HLDA) [Kumar & Andreou 1998] to a window of usually 9 or 11 of the original feature vectors. The result is a linear transformation which projects the original features into a lower dimensional feature space such that the class separability is maximized, assuming that the data given a class follows a normal distribution. The LDA/HLDA are also successfully used to combine acoustic features from several feature extraction procedures, e.g. several short-term FFT features [Schlüter & Zolnay+ 2006].

The third step intends to focus on the gender and speaker independence of the acoustic features which is hard to meet and usually not well achieved by the feature extraction procedures mentioned above. For example, the MFCC and PLP features are affected by the gender of the speaker, and in fact they are used for gender detection [Stolcke & Bratt+ 2000] or even for speaker identification [Doddington & Przybocki+ 2000]. Several methods have been developed to reduce the speaker dependency of the acoustic features. Two wide-spread approaches are the vocal tract length normalization (VTLN) and the maximum likelihood linear regression (MLLR) transformation [Gales & Woodland 1996; Lee & Rose 1996; Leggetter & Woodland 1995]. The MLLR approach consists of a speaker-dependent linear transformation of the model parameters. A version of MLLR that can be applied to the feature space is called constrained MLLR (CMLLR) or equivalently feature space MLLR (fMLLR). More specifically, CMLLR is a feature adaptation technique that estimates a set of linear transforms for the features which reduces the mismatch between an initial model set and the adaptation data. The effect of these transformations is to shift the feature vectors in the initial system so that each state in the HMM system is more likely to generate the adaptation data. A comprehensive comparison of speaker normalization and adaptation methods is given in [Pitz 2005].

1.4 Acoustic Model

The stochastic model that computes the probability of the acoustic features x_1^T given a word sequence w_1^N is called the acoustic model. For LVCSR systems, usually sub-word models like demisyllables, syllables, phonemes, or allophones are used instead of full-word models. The pronunciation model $p(a_1^L|w_1^N)$ assigns probabilities to sequences of sub-word units a_1^L given a sequence of words w_1^N. Most modern LVCSR systems use a finite pronunciation dictionary to store the (weighted) mapping from words to sequences of sub-word units. Assuming independence of the pronunciation of a word from adjacent words yields Equation (1.2) where, l_n is the length of the sequence of sub-word units used for word w_n, $\sum_{n=1}^{N} l_n = L$ and $l_0 := 0$.

$$\begin{aligned} p(x_1^T|w_1^N) &= \sum_{a_1^L} p(x_1^T|a_1^L)p(a_1^L|w_1^N) \\ &= \sum_{a_1^L} p(x_1^T|a_1^L) \prod_{n=1}^{N} p(a_{l_{n-1}+1}^{l_n}|w_n) \end{aligned} \tag{1.2}$$

The advantage of using sub-word units is that they reduce the model complexity, which allows a reliable parameter estimation. Since the set of sub-words is shared among all words, the search vocabulary does not need to be equal to or a subset of the training vocabulary. The acoustic model for a new word with known pronunciation is assembled from the corresponding sequence of sub-word units. Even if a word is not in the pronunciation dictionary, i.e. a new word with unknown pronunciation, there exist algorithms that compute the highest probable corresponding sequence of sub-word units [Bisani & Ney 2003, 2008]. The process of generating such sequence of sub-word units for a given word is usually called

grapheme-to-phoneme (G2P) conversion.

Usually, the modern LVCSR systems use the so-called context-dependent phoneme models, which are models of phonemes with some left and right context. For example, a triphone is a phoneme together with its predecessor and successor. Other examples with larger context are quinphones, septaphones, etc. This is the most common way to model allophones which represent the acoustic realization of a phoneme. The motivation of using context-dependent phonemes is the observation that the articulation of a phoneme highly depends on the adjacent phonemes. Normally, a two-stage mapping is used. First, the pronunciation dictionary is looked-up to provide the weighted mapping from the word to a phoneme sequence. Then, mapping is performed from phonemes to context-dependent phonemes. If the context is considered across word boundaries, then the resulting acoustic model is called an across-word model [Sixtus 2003].

A common approach to cope with the varying acoustic realization of sub-word units at different speaking rates is the hidden Markov model (HMM) [Baker 1975; Rabiner & Juang 1986]. An HMM is a stochastic finite state automaton, where the states represent (hidden) random variables that cannot be observed directly. The output of an HMM is generated according to the probability distributions which depend on the values s_1^T of the hidden variables. The HMM is a generative model which represents an acoustic model that generates feature sequences x_1^T. For a hypothesized word sequence w_1^N, we imagine a super HMM that is obtained by concatenating the corresponding sub-word HMMs using the pronunciation lexicon. By this process, we end up with a large number of copies for each sub-word, these copies should be kept separate during the search in order to satisfy the constraints of the pronunciation lexicon. At sub-word and word boundaries, we have to allow for transitions that link the terminal states of any predecessor HMM to the initial states of any successor HMM. In such a way, we can compute the joint probability of observing the sequence x_1^T of acoustic feature vectors and the state sequence s_1^T through this super HMM. Thus, the acoustic probability for observing x_1^T given word sequence w_1^N is the marginal over all possible state sequences of this joint probability:

$$\begin{aligned} p(x_1^T|w_1^N) &= \sum_{s_1^T} p(x_1^T, s_1^T|w_1^N) \\ &= \sum_{s_1^T} \prod_{t=1}^{T} p(x_t, s_t|x_1^{t-1}, s_1^{t-1}; w_1^N) \\ &= \sum_{s_1^T} \prod_{t=1}^{T} p(x_t|x_1^{t-1}, s_1^t; w_1^N) p(s_t|x_1^{t-1}, s_1^{t-1}; w_1^N) \end{aligned} \tag{1.3}$$

The equation is simplified by applying the first order Markov assumption [Duda & Hart$^+$ 2001], which states that the probability of some observation at time t depends only on the current state; and the probability of the current state depends only on the immediate previous state. Under these assumptions Equation (1.3) simplifies to Equation (1.4):

$$p(x_1^T|w_1^N) = \sum_{s_1^T} \prod_{t=1}^{T} p(x_t|s_t; w_1^N) p(s_t|s_{t-1}; w_1^N) \tag{1.4}$$

Using the so-called Viterbi or maximum approximation [Jelinek 1976], the sum is replaced by the maximum producing Equation (1.5):

$$p(x_1^T|w_1^N) = \max_{s_1^T} \prod_{t=1}^{T} p(x_t|s_t; w_1^N) p(s_t|s_{t-1}; w_1^N) \tag{1.5}$$

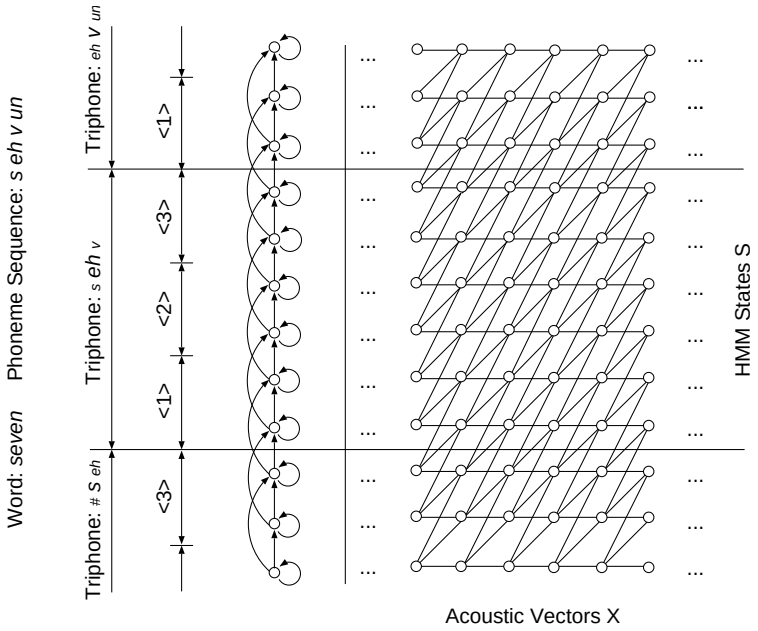


Figure 1.2. 6-state hidden Markov model in Bakis topology for the triphone ${}_{s}eh_{v}$ in the word "seven" and the resulting trellis for a time alignment. The HMM segments are denoted by <1>, <2>, and <3>.

If we substitute Equation (1.5) into the Bayes decision rule of Equation (1.1), we come up with:

$$\begin{aligned}\left[w_1^N\right]_{opt} &:= \operatorname*{argmax}_{w_1^N} p(w_1^N|x_1^T) \\ &= \operatorname*{argmax}_{w_1^N} p(x_1^T|w_1^N)p(w_1^N) \\ &= \operatorname*{argmax}_{w_1^N} \left\{ p(w_1^N) \cdot \max_{s_1^T} \prod_{t=1}^{T} p(x_t|s_t;w_1^N)p(s_t|s_{t-1};w_1^N) \right\} \end{aligned} \tag{1.6}$$

We should note that for the maximum approximation to work, we need only the assumption that the resulting optimal word sequences remain the same, not necessarily that the maximum provides a good approximation to the sum. According to Equation (1.5) two probability distributions are considered: the emission probability $p(x_t|s_t;w_1^N)$ and the transition probability $p(s_t|s_{t-1};w_1^N)$. The emission probability denotes the probability of observing an acoustic feature vector x_t while being in state s_t. The transition probability is the probability of moving from state s_{t-1} to state s_t.

Equation (1.4) and Equation (1.5) are also referred to as the time alignment problem. The result computed for a particular word sequence w_1^N is called the forced acoustic alignment of w_1^N. An efficient algorithm for solving the time alignment problem based on dynamic programming (DP) [Bellman 1957; Ney 1984; Viterbi 1967] is known as the forward-backward algorithm for HMMs [Baum 1972; Rabiner & Juang 1986]. Figure 1.2 shows an example for a time alignment. For a part of the word "seven", the HMM is constructed using the Bakis topology [Bakis 1976], along which a sequence of acoustic feature vectors to be aligned. The Bakis topology uses six state HMMs with skip transitions where, each two successive states are identical. During the time alignment, the HMM is enrolled along the times axis and the resulting graph is referred to as trellis. The trellis visualizes the complete search space for the time alignment. If we used the Viterbi approximation in Equation (1.5), then the solution is the path from the lower left corner to the upper right corner with the highest probability.

The emission probabilities $p(x_t|s_t;w_1^N)$ of the HMM are usually modeled by Gaussian Mixture Models (GMMs). Alternative approaches are discrete probabilities [Jelinek 1976], semi-continuous probabilities [Huang & Jack 1989] or other continuous probability distributions like mixtures of Laplacians [Haeb-Umbach & Aubert$^+$ 1998; Levinson & Rabiner$^+$ 1983]. Usually, GMMs are defined as in Equation (1.7).

$$p(x|s;w_1^N) = \sum_{l=1}^{L_s} c_{sl}\,\mathcal{N}(x|\mu_{sl},\Sigma_{sl};w_1^N) \tag{1.7}$$

The emission probability for state s is described by a GMM of L_s Gaussian densities $\mathcal{N}(x|\mu_{sl},\Sigma_{sl};w_1^N)$ with mean vector μ_{sl} and covariance matrix Σ_{sl} and non-negative mixture weights c_{sl}, where the mixture weights are subject to the constraint $\sum_{l=1}^{L_s} c_{sl} = 1$.

At RWTH, triphone models are usually used in acoustic modeling. A triphone is usually modeled by a linear HMM with three to six states. The possible transitions are the loop transition going from a state back to itself, the forward transition connecting a state to the next one, and the skip transition, which skips the next state and goes to the next to next state. In the RWTH Aachen systems, the transition probabilities are replaced by the so-called time distortion penalties (TDPs). The TDPs depend only on the transition type, but not on the state itself. A special case is the HMM for the silence model, which consists only of a single state and has separate TDPs. The HMM for a sequence of words is assembled by concatenating the HMMs of the corresponding triphone sequence. The LVCSR systems at RWTH Aachen use only a single, globally pooled diagonal covariance matrix Σ. This choice is made to avoid data sparseness problems in the acoustic model training. Using a diagonal covariance matrix requires the components of the acoustic features to be decorrelated, which is ensured in the feature extraction step by applying a discrete cosine transform. The free parameters of the acoustic model μ_{sl}, c_{sl}, and Σ are estimated by applying maximum likelihood (ML) estimation in combination with the expectation maximization (EM) algorithm [Dempster & Laird$^+$ 1977]. In the state-of-the-art LVCSR systems, the ML/EM training is followed by a discriminative refinement of the acoustic model parameters [Bahl & Padmanabhan$^+$ 1996; Schlüter 2000; Woodland & Povey 2002]. In the discriminative training step, the objective is to maximize the a-posteriori probability of the correct sentence [Bahl & Brown$^+$ 1986; Normandin & Lacouture$^+$ 1994] or to minimize the word or phoneme error rate on the training data [Juang & Katagiri 1992; Kaiser & Horvat$^+$ 2000; McDermott & Katagiri 2005; Povey & Woodland 2002a].

1.5 Language Model

The language model provides the a-priori probability $p(w_1^N)$ for a word sequence w_1^N. Ideally, it covers the syntax, the semantics, and the pragmatics of the language and the underlying situation. In practice, the m-gram model is the standard for the LVCSR systems. This model makes the assumption that the probability of the current word w_n depends only on the previous $m-1$ words w_{n-m+1}^{n-1} [Bahl & Jelinek$^+$ 1983]. Equation (1.8) motivates the factorization of the a-priori probability under the assumption of the $(m-1)$th order Markov process. Thus, the probability of the word sequence w_1^N is expressed as a product of the conditional probabilities of individual words given their $m-1$ histories. Typically, histories are approximated by a limited number of preceding words such as 1,2 or 3 words, resulting into bigram, trigram or 4-gram language models respectively.

$$p(w_1^N) = \prod_{n=1}^{N} p(w_n|w_1^{n-1}) \tag{1.8}$$

$$= \prod_{n=1}^{N} p(w_n|w_{n-m+1}^{n-1}) \tag{1.9}$$

The consecutive sequence of m words is called an m-gram. In the general case, the history of a word w_n is a function of w_{n-m+1}^{n-1}. For the standard m-gram model, this function is the identity function. Examples for alternative history functions are found in the class-based language models [Brown & deSouza$^+$ 1992] or the trigger models [Martin 2000].

Normally, the language model attempts to reflect how frequently a string of words occurs as a sentence. Therefore, the estimates of $p(w_n|w_{n-m+1}^{n-1})$ are usually based on relative frequencies computed on a large training set of text (e.g. transcripts of speech) that contains typically many millions of running words.

In fact, the number of possible m-grams grows exponentially w.r.t. m; and for typical LVCSR tasks, many m-grams are not seen in the training data or have very few observations. Thus, any word sequence

of the test data containing a single unseen m-gram will have a probability of zero. This means that the recognition system will never be able to recognize such sentences which is unaccepted behavior. Therefore, the relative frequencies have to be smoothed such that all possible word sequences are assigned nonzero probabilities. Common smoothing techniques are based on discounting followed by backing-off or interpolation [Generet & Ney+ 1995; Katz 1987; Ney & Essen+ 1994; Ney & Martin+ 1997]. In the discounting step, the probability mass is removed from the relative frequencies. The backing-off or interpolation step distributes the discounted probability mass over all unseen m-grams (backing-off) or over all m-grams (interpolation). The name smoothing comes from the fact that this technique tends to make distributions more uniform, by adjusting low probabilities such as zero probabilities upward, and high probabilities downward. Not only do smoothing methods generally prevent zero probabilities, but they also attempt to improve the accuracy of the model as a whole. Whenever a probability is estimated from few counts, smoothing has the potential to significantly improve the estimation. A popular method to estimate the parameters of a smoothed language model is the leaving-one-out [Ney & Essen+ 1994]. An extensive empirical comparison of the most widely-used smoothing techniques, including those described by [Bell & Cleary+ 1990; Jelinek & Mercer 1980; Katz 1987; Kneser & Ney 1995; Ney & Essen+ 1994], can be found in [Chen & Goodman 1996].

The most common metric for evaluating a language model is the probability that the model assigns to some test data, or the derivative measures called the cross-entropy and the perplexity. For a smoothed language model probability distribution p and a test set T composed of a number of l_T sentences $(t_1, ..., t_{l_T})$, we can calculate the probability of the test set $p(T)$ as the product of the probabilities of all sentences in the set:

$$p(T) = \prod_{i=1}^{l_T} p(t_i) \tag{1.10}$$

The measure of cross-entropy can be motivated using the well-known relation between prediction and compression [Bell & Cleary+ 1990; Cover & Thomas 1991]. In particular, given a language model that assigns probability $p(T)$ to a text T, we can derive a compression algorithm that encodes the text T using $-\log_2 p(T)$ bits. The cross-entropy $H_p(T)$ of the language model on data T is defined as:

$$H_p(T) = -\frac{1}{W_T} \log_2 p(T) \tag{1.11}$$

where W_T is the number of words in the text T. This value can be interpreted as the average number of bits needed to encode each of the W_T words in the test data using the compression algorithm associated with model p. In a different perspective, the cross-entropy can be interpreted as the average number of different words that could follow any given history (average number of words per position). Sometimes the cross-entropy is referred to as just entropy.

The perplexity $PP_p(T)$ of a model p is the most widely-used evaluation measure for an m-gram language model. It is defined as the reciprocal of the (geometric) average probability assigned by the model to each word in the test set T as in Equation (1.12):

$$\begin{aligned} PP_p(T) &= 2^{H_p(T)} \\ &= [p(T)]^{-1/W_T} \\ &= \left[\prod_{i=1}^{l_T} p(t_i)\right]^{-1/W_T} \\ &= \left[\prod_{i=1}^{l_T} \prod_{n=1}^{N_{t_i}} p(w_n | w_{n-m+1}^{n-1})\right]^{-1/W_T} \end{aligned} \tag{1.12}$$

As clear from the above definitions, it is always desirable that the cross-entropies and the perplexities have lower values. Usually, in order to train a well-performing language model, the perplexity of that model is minimized over some held-out text which should be close to the domain of the test data. As an

example, typical perplexities yielded by m-gram models on English text range from about 50 to almost 1000 (corresponding to cross-entropies from about 6 to 10 bits/word), depending on the type of text [Chen & Goodman 1996].

1.6 Search

The search problem in ASR consists of finding an efficient algorithm and appropriate approximations for finding the word sequence w_1^N which maximizes the a-posteriori probability $p(w_1^N|x_1^T)$ for a given feature sequence x_1^T, i.e. solving Equation (1.1). As shown in Figure 1.1, the search combines the different knowledge sources: the acoustic model (including the pronunciation model) and the language model. If the acoustic model is an HMM as described in Equation (1.4) and the language model is an m-gram model following Equation (1.8), then Equation (1.13) describes the resulting optimization problem.

$$\begin{aligned}\left[w_1^N\right]_{opt} &= \operatorname*{argmax}_{w_1^N}\left\{\left[\prod_{n=1}^{N} p(w_n|w_{n-m+1}^{n-1})\right]\left[\sum_{s_1^T}\prod_{t=1}^{T} p(x_t|s_t;w_1^N)p(s_t|s_{t-1};w_1^N)\right]\right\} \\ &\overset{\text{Viterbi}}{=} \operatorname*{argmax}_{w_1^N}\left\{\left[\prod_{n=1}^{N} p(w_n|w_{n-m+1}^{n-1})\right]\left[\max_{s_1^T}\prod_{t=1}^{T} p(x_t|s_t;w_1^N)p(s_t|s_{t-1};w_1^N)\right]\right\} \end{aligned} \quad (1.13)$$

The optimization problem can be efficiently solved by using dynamic programming [Bellman 1957]. The Markov assumptions and the Viterbi approximation yield a mathematical structure which divides the global optimization problem in Equation (1.13) into sub-problems with local dependencies and allows the application of dynamic programming. The search can be organized in two ways: depth-first search or breadth-first search. Instances of the depth-first search (stack decoding algorithms) are the Dijkstra [Dijkstra 1959] and the A^* algorithm [Jelinek 1969; Paul 1991]. The hypotheses space is explored in a time-asynchronous manner according to the stack organization. In the A^* algorithm, the stack is sorted by a heuristic estimate of the cost to complete a hypothesis. In the breadth-first search, all hypotheses are expanded in a time-synchronous manner [Baker 1975; Ney 1984; Sakoe 1979; Vintsyuk 1971].

In typical LVCSR tasks, the search space is huge and thus the full exploration is impractical. Therefore, modern recognizers use pruning techniques to keep only the promising parts of the search space thereby avoiding search errors. In an A^* decoder, pruning is applied by removing the least promising partial paths from the stack. The quality of the pruning depends on the quality of the heuristic cost estimate. In contrast, the standard pruning for breadth-first search decoders does not require an explicit heuristic. In a breadth-first search implementation, the likelihoods for all hypotheses are computed at each time frame. The so-called beam pruning compares the likelihoods at each time frame and keeps only those hypotheses which have likelihoods that are sufficiently close to the likelihood of the current best hypothesis [Lowerre 1976; Ney & Mergel$^+$ 1987; Ortmanns & Ney 1995]. A careful tuning of the pruning parameters yields a considerable reduction in the search effort without having a significant number of search errors.

The beam search approaches for LVCSR are particularly effective in combination with lexical prefix trees [Ney & Häb-Umbach$^+$ 1992; Ortmanns & Eiden$^+$ 1998]. The pronunciations that have common prefixes are laid together in the lexical prefix tree. Pruning in the early stages of the tree removes whole sub-trees and eventually discards large parts of the search space. Language model look-ahead techniques aim at considering the language model probabilities in the early stages of the lexical prefix tree; which enables the decoder to perform better pruning based on more accurate scores [Alleva & Huang$^+$ 1996; Ortmanns & Ney$^+$ 1996; Steinbiss & Ney$^+$ 1993].

Alternatively, weighted finite state transducers (WFSTs) provide a generic framework to optimize the search space [Allauzen & Mohri$^+$ 2004; Mohri & Riley 1997]. The acoustic model (HMM) and the language model (m-gram model) have WFST representations that can be combined and minimized by using generic algorithms. In particular, the lexical prefix tree and the language model look-ahead technique are implicitly applied by a WFST decoder using a minimized static search space transducer [Kanthak & Ney$^+$ 2002]. Other methods to reduce the computational complexity of the search include the fast likelihood computation [Kanthak & Schütz$^+$ 2000; Parihar & Schlüter$^+$ 2009; Ramasubramansian & Paliwal 1992], several look-ahead techniques [Alleva & Huang$^+$ 1996; Häb-Umbach & Ney 1994; Ortmanns & Ney$^+$

1996], and multi-pass approaches, where a fast first pass reduces the search space for the ultimate Viterbi search [Ljolje & Pereira+ 1999; Murveit & Butzberger+ 1993; Ney & Aubert 1994; Ortmanns & Ney+ 1997; Schwartz & Chow 1990].

1.7 Multi-Pass Search

The state-of-the-art LVCSR recognizers perform multiple recognition and/or re-scoring passes [Evermann & Chan+ 2003; Hoffmeister & Fritz+ 2007; Prasad & Matsoukas+ 2005]. Supervised adaptation techniques like standard VTLN, MLLR, CMLLR, and domain specific language model adaptation require a reference transcription. In a multi-pass decoder the output of the first unadapted recognition run provides the required transcription to the adaptation step, which is followed by a second recognition run with the adapted models and/or adapted features.

Some models and techniques cannot be applied during the Viterbi search because of their complexity, like the language models in [Arisoy & Sainath+ 2012; Bengio & Ducharme+ 2003; Bilmes & Kirchhoff 2003; Brown & deSouza+ 1992; Emami & Papineni+ 2007; Kirchhoff & Vergyri+ 2006; Kombrink & Mikolov+ 2011], the phoneme duration model in [Jennequin & Gauvain 2007], the Bayes risk decoding in [Hoffmeister 2011], or the system combination approaches in [Hoffmeister & Schlüter+ 2008]. Therefore, these models and techniques are applied to a restricted search space which results from a normal Viterbi search. Thus, during the Viterbi search; instead of finding a single hypothesis; the search algorithm narrows the search space by creating N-best lists or lattices containing the sets of hypotheses achieving the highest scores. These N-best lists or lattices are then re-scored via the sophisticated models or techniques.

1.7.1 N-Best Lists and Lattices

The most common way to store a large number of speech recognition hypotheses is to use N-best lists or lattices. The N-best list is a sorted list of the N sentences that are assigned the highest scores after performing the ASR search. Whereas, the lattice is a more compact form which stores a large number of hypotheses that can be much larger than the size of any feasible N-best list.

A lattice is a directed, acyclic graph with time stamps on the states and labels on the arcs. In a word lattice, the label is usually the word together with its pronunciation, in a phoneme lattice the label is simply a phoneme. In addition, each arc stores the acoustic and the language model scores from the Viterbi decoding. An example of a word lattice is shown in Figure 1.3.

The properties of a lattice depend on the search algorithm used by the decoder, and on the subsequent filter steps. The default Viterbi search of the RWTH Aachen LVCSR system is a time-synchronous word-conditioned tree search implementation [Beulen & Ortmanns+ 1999]. A word lattice produced by this decoder follows the word-pair approximation in which an assumption is made that the end time of a word depends only on the current and the preceding word hypothesis [Ney & Aubert 1994; Ortmanns & Ney+ 1997]. The word-pair approximation guarantees that at any time t and for any word w and predecessor word v there exists only one lattice arc labeled with w. As a consequence the lattice is deterministic, i.e. each word sequence exists only once, in particular the same word sequence cannot exist with different word boundaries. This word-pair approximation helps creating a more compact lattice. However, the only guaranteed property of a lattice produced by the RWTH Aachen decoder is that it contains the best sentence hypothesis with the correct scores and correct word boundaries. Due to the word-pair approximation, hypotheses competing with the best one may have inaccurate word boundaries and thus overestimated acoustic scores. Furthermore, it is not guaranteed that a lattice of M hypotheses contains the N-best list for $1 < N \leq M$, i.e. the N best scored hypotheses. The constraints hold not only for the RWTH Aachen decoder but for any popular LVCSR decoder.

The quality of a lattice is measured in terms of the graph error rate (GER)[1] and the graph density (GD). The goal of the lattice construction process is to achieve a low GER for a small GD. The GER of a word lattice $\mathbf{L}$ is defined in Equation (1.14), where $\mathrm{Lev}(w_1^N, \tilde{w}_1^{\tilde{N}})$ is the Levenshtein distance between

[1] Sometimes referred to the lattice error rate (LER) or as the oracle error rate (OER) of the lattice.

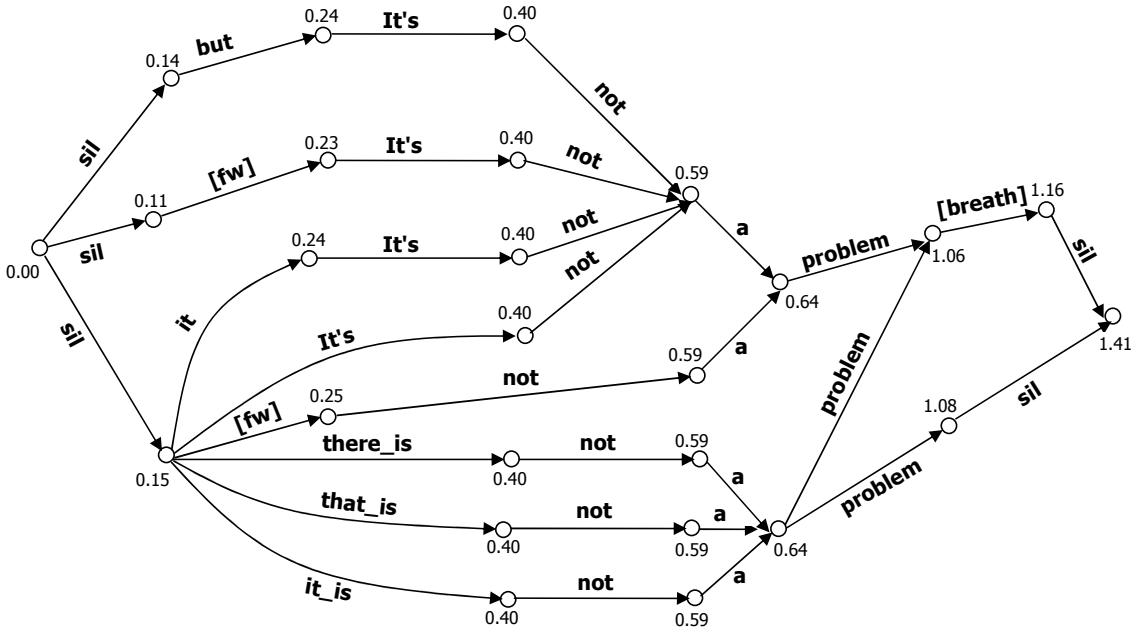


Figure 1.3. An example of a word lattice (taken from [Schwenk 2007]). The lattice is produced using a trigram LM, where each word has a unique bigram context. For simplicity, acoustic and language model scores are not shown on arcs ([**fw**]: filler word; [**breath**]: breath noise).

the word sequence w_1^N and the reference $\tilde{w}_1^{\tilde{N}}$, and $\tilde{N}$ is the number of reference words.

$$\mathrm{GER}(\mathbf{L}) := \min_{w_1^N \in \mathbf{L}} \frac{\mathrm{Lev}(w_1^N, \tilde{w}_1^{\tilde{N}})}{\tilde{N}} \tag{1.14}$$

The GD is defined as the ratio between the number of arcs in the lattice $|E(\mathbf{L})|$ and the number of words in the reference $\tilde{N}$, where $E(\mathbf{L})$ is the set of arcs in the lattice $\mathbf{L}$. If the reference is unknown, then the GD can be approximated by using the number of words in the Viterbi decoding (1-best) result $\hat{N}$.

$$\mathrm{GD}(\mathbf{L}) := \frac{|E(\mathbf{L})|}{\tilde{N}} \approx \frac{|E(\mathbf{L})|}{\hat{N}} \tag{1.15}$$

The lattices produced by the RWTH Aachen LVCSR decoder use the word-conditioned tree search decoder and the word-pair approximation. The resulting lattices are referred to as word-conditioned lattices.

In a similar fashion, the N-best error rate (NER) is the minimum WER achieved by any word sequence in the N-best list. Let $\mathbf{B}$ be a set of N-best sentences, then similar to Equation 1.14, the NER can be defined as:

$$\mathrm{NER}(\mathbf{B}) := \min_{w_1^N \in \mathbf{B}} \frac{\mathrm{Lev}(w_1^N, \tilde{w}_1^{\tilde{N}})}{\tilde{N}} \tag{1.16}$$

1.7.2 Confusion Networks

The lattice has usually a complex topology. Therefore, a common approximation of the lattice, called a confusion network (CN), is sometimes used instead. A confusion network (CN) is a weighted directed graph with the peculiarity that each path from the start node to the end node goes through all the other nodes. Each edge is labeled with a word and a (posterior) probability. Additional scores can also be provided. The total probability of all edges between two consecutive nodes sum up to 1. A path from

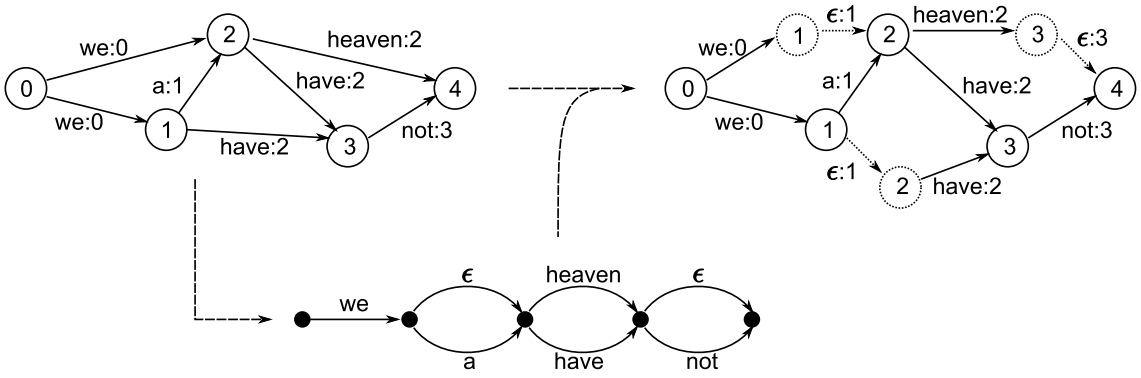


Figure 1.4. An example of a confusion network (CN) derived from a lattice. The figure shows: the original lattice, a derived CN, and an intermediate lattice in which all paths have the same length. The positions for the insertions of the ϵ-arcs are derived from the CN according to the algorithm described in [Hoffmeister 2011]. The number that appears on each arc corresponds to the CN slot to which the arc is assigned.

the start node to the end node is scored by multiplying the scores of its edges. If the previous constrain is satisfied, the product represents the likelihood of the path, and the sum of the likelihoods of all paths equals to 1. Between any two consecutive nodes, at most one special empty word (ϵ) can be inserted. These empty words allow paths to have different lengths. Any path within a CN represents a realization of the CN. Realizations of a CN can differ in terms of either the sequence of words or the total score. It is possible that two or more realizations have the same sequence of words, but different scores. The number of words can also differ due to presence of ϵ-arcs. A CN contains *at least* all the paths of the lattice from which it is originated.

In the recent years, several methods have been proposed to build CNs directly from lattices [Hakkani & Riccardi 2003; Hoffmeister & Schlüter$^+$ 2009; Mangu & Brill$^+$ 2000; Xue & Zhao 2005]. Figure 1.4 shows an example of a CN derived from a lattice using the algorithm described in [Hoffmeister 2011]

1.7.3 Speaker Adaptation

Speaker adaptation requires a speaker label S for each speech utterance. All the utterances spoken by the same speaker form a separate speaker cluster labeled by S. A common approach for unsupervised speaker clustering is based on optimizing the Bayesian information criterion (BIC) on the acoustic features of the clustered utterances [Chenand & Gopalakrishnan 1998; Tritschler & Gopinath 1999]. The commonly applied speaker adaptation methods in the RWTH Aachen decoder are: vocal tract length normalization (VTLN), maximum likelihood linear regression (MLLR), and constrained MLLR (CMLLR).

In VTLN approach, the warping factor for a speaker S is chosen by performing a grid search that aims at maximizing the likelihood of the speaker cluster given the output of the previous recognition pass. This approach is computationally expensive. Therefore, the RWTH Aachen decoder uses the fast-VTLN implementation, where the warping factor is selected by a classifier [Lee & Rose 1996; Molau 2003].

In MLLR approach, the parameters of the GMMs are adapted to match the speaker by applying a speaker-dependent linear transformation to the means and variances. Equation (1.17) shows the unconstrained form of MLLR, where s is the index of the state, and l is the index of the mixture density.

$$\hat{\mu}_{sl}^{(S)} = \mathbf{A}_s^{(S)}\mu_{sl} + \mathbf{b}_s^{(S)}, \qquad \hat{\Sigma}_{sl}^{(S)} = \mathbf{H}_s^{(S)}\Sigma_{sl}\mathbf{H}_s^{(S)\mathrm{T}} \tag{1.17}$$

In the RWTH Aachen decoder, only the means are adapted but not the globally pooled diagonal covariance matrix Σ. The state dependent transformation matrices $\mathbf{A}_s^{(S)}$ for a given speaker S and a state s are tied according to a decision tree [Pitz 2005]. In the estimation step, those transformation matrices are chosen so as to maximize the likelihood of the corresponding speaker cluster, where the output of the previous decoding pass serves as a supervisor.

In the CMLLR approach, a constrained form of MLLR is used, where the means and variances are transformed by the same matrices. The RWTH Aachen decoder uses CMLLR for speaker adaptive training (SAT), where only a single transformation per speaker is used. The resulting transformation is shown in Equation (1.18).

$$\hat{\mu}_{sl}^{(S)} = \mathbf{A}^{(S)}\mu_{sl} + \mathbf{b}^{(S)}, \qquad \hat{\Sigma}^{(S)} = \mathbf{A}^{(S)}\Sigma\mathbf{A}^{(S)\mathrm{T}} \tag{1.18}$$

The advantage of CMLLR is that it can be implemented as a feature transformation. This simplifies the integration of CMLLR in LVCSR systems [Leggetter & Woodland 1995].

1.8 Model and System Combination

A successful approach to achieve less WER in ASR is to combine several models or systems together. In the model combination approach, all the knowledge sources are combined into a single log-linear model from which the posterior probability $p(w_1^N|x_1^T)$ is computed. The knowledge sources combined in the log-linear model usually consist of several language models and acoustic models. In the cross-adaptation approach, two or more independently trained systems are combined, where the interaction between the systems takes place in the speaker adaptation step. The third and most common approach is to introduce the system as a hidden variable and to compute the marginal over the resulting weighted, system-dependent posteriors. Equation 1.19 expresses the combination of J LVCSR systems [Hoffmeister 2011].

$$p(w_1^N|x_1^T) = \sum_{j=1}^{J} p(w_1^N, j|x_1^T) = \sum_{j=1}^{J} p(j|x_1^T)p(w_1^N|j, x_1^T) \tag{1.19}$$

1.8.1 Log-linear Model Combination

The standard approach in ASR is to use a model with only two knowledge sources: the acoustic model and the language model. To achieve the optimal performance, LVCSR systems introduce a language model scale which eventually turns Equation (1.1) into a log-linear model. The log-linear model can be used explicitly for model combination by adding more knowledge sources to the model, usually additional language models or acoustic models [Metze & Waibel 2002a,b; Zolnay 2006].

In the discriminative model combination (DMC), each of the knowledge sources combined in the log-linear model gets its own scaling factor which is optimized for minimal WER [Beyerlein 1997, 1998; Vergyri 2000; Zolnay & Schlüter$^+$ 2005]. In practice, a decoding process with many acoustic models is expensive in terms of both time and memory. Therefore, a common approach is to generate lattices with a base decoder and rescore these lattices with the additional knowledge sources. In the standard LVCSR training procedures, the interaction between the several knowledge sources during search is not fully considered. An approach to compensate for the short-coming of the model training is to capture the interactions in the log-linear model combination using context-dependent scaling factors [Hoffmeister & Liang$^+$ 2009; Huang & Belin$^+$ 1993; Vergyri & Tsakalidis$^+$ 2000].

1.8.2 System Combination

An alternative to the log-linear model combination is the N-best list or lattice-based system combination, where the output of several decoders is combined. In the log-linear model combination, all models are combined into a single system, whereas in the system combination approach each model is used in a separate system to produce a separate output. In the simplest case, only a single best hypothesis from each system are combined like in the ROVER approach [Fiscus 1997]. The quality of the ROVER result can be significantly increased by using confidence scores [Mangu & Brill$^+$ 2000; Wessel & Schlüter$^+$ 2001] or by replacing ROVER's decision rule by a classifier [Hillard & Hoffmeister$^+$ 2007; Zhang & Rudnicky 2006]. Instead of a single hypothesis, N-best lists or confusion networks (CNs) can be combined [Evermann & Woodland 2000; Mangu 2000; Ostendorf & Kannan$^+$ 1991; Stolcke & Bratt$^+$ 2000].

1.8.3 Cross-Adaptation

An alternative method to perform system combination which became popular in the recent years is called cross-adaptation [Soltau & Kingsbury+ 2005a; Stüker & Fügen+ 2006]. Instead of applying the system combination as a post-processing step to the decoding process, the interaction between the systems is put into the speaker adaptation step of a multi-pass decoder. In the cross-adaptation approach, the supervisor for MLLR adaptation is the output of an alternative system. In [Guiliani & Brugnara 2006], the approach is extended to multiple supervisors. The multiple supervisors are either reduced to a single supervisor in a preprocessing step by applying system combination methods, or the ultimate adaptation statistics are derived from the weighted average of the supervisor-dependent statistics [Guiliani & Brugnara 2007; Hoffmeister & Fritz+ 2007; Rybach & Hahn+ 2007; Vergyri & Mandal+ 2008].

1.9 Morphologically Rich Languages

All the experiments in this thesis are performed on Arabic, German and Polish corpora since these languages are good examples of morphologically rich languages. Therefore, in this section, we give a quick overview of the main linguistic properties of these languages. We essentially highlight the morphological properties that are closely related to our work.

1.9.1 Arabic

Arabic belongs to the family of Semitic languages containing Amharic, Aramaic and Hebrew. The standard form of Arabic that is used in writing and in most formal speech is called modern standard Arabic (MSA). Arabic is a highly inflected language. The Arabic words are derived from roots, which in most cases consist of three consonants, by applying patterns to get stems and then attaching different affixes to obtain a large number of word forms. Usually, affixes are used to indicate grammatical categories like person, number and gender. We can think of a stem as being decomposed into a root and a pattern. The root holds the basic meaning of the word; while the pattern adds secondary features like: voice, tense, or causality. The root and the pattern are interspersed to give a typical word form. An example of having several words derived from the same root is shown in Table 1.1.

Moreover, Arabic is a strongly consonantal language having only three vowels, each of which has a short and long form. Some important pronunciation phenomena are indicated by special marks rather than normal letters. These marks are called *diacritics* which are short strokes placed above or below the consonant. These diacritical marks are used to indicate: short vowels, gemination (consonant doubling), nunation (word-final adverbial mark) or sukun (vowel absence). A word with full diacritics is called a *diacritized word*. Usually, Arabic words are written without diacritization, also referred to as vowelization, which the reader should infer according to the context. In fact, very few text sources use explicit diacritics. This is usually found in religious texts but not in the texts of the MSA. For more details about the Arabic language, see e.g. [Bateson 2003].

Arabic is also characterized by a large number of colloquial spoken (usually not written) varieties called *dialects*, such as Egyptian, Levantine, Moroccan, Iraqi, and Gulf Arabic. All the dialects can be categorized into five regional groups: Egyptian, Syro-Lebanese, Maghrebi, Mesopotamian and Arabian Peninsula. These dialects deviate from the MSA in different aspects, like the vocabulary, word ordering, the set of affixes, word fusion, in addition to the way of pronunciation. In this work, we perform experiments on the Egyptian colloquial Arabic (ECA) which is considered as the most widely understood colloquial version of Arabic. ECA inherits all the basic characteristics of the MSA, like the complex morphology, the high degree of inflection, and the rich derivation. However, like most of the other dialects, it shares a large number of vocabulary words with the MSA. It is considered as a low-resource dialectal language for which there are no widely available language resources such as written text, pronunciation dictionaries, morphological analyzers, and so forth.

Table 1.1. Different Arabic words derived from the same root **"ktb"**.

Arabic Word	Buckwalter Transliteration	English Meaning
كَتَبَ	**k**a**t**a**b**a	he wrote
كُتِبَ	**k**u**t**i**b**a	it was written
مَكتوبٌ	ma**k**t**b**uN	is written
كَاتِبٌ	**k**A**t**i**b**uN	a writer
كِتَابٌ	**k**i**t**A**b**uN	a book
كِتَابَةٌ	**k**i**t**A**b**apuN	writing
كُتُبٌ	**k**u**t**u**b**uN	books
أَكتُبُ	>**k**t**b**u	I write
مَكتَبٌ	ma**k**t**b**uN	desk or office

1.9.2 German

German belongs to the family of Germanic languages like: Dutch, Norwegian, Danish and Swedish. It is usually cited as an outstanding example of highly inflected languages, as a large number of words can be derived from the same root. Like most Germanic languages, German uses noun compounds where the first noun modifies the category given by the second, like in the word "Hundehütte : *dog hut*"[2]. Long combinations of nouns are often written in a closed form without spaces, for example "Baumhaus : *tree house*". Also, the meaning of German verbs can be expanded, and sometimes radically changed, through the use of prefixes. An example is the prefix *zer-* refers to the destruction of things, as in "zerbrechen : *to break apart*", or "zerschneiden : *to cut apart*". Many German verbs have a separable prefix which can be split off and moved to the end of the clause. For example, "mitgehen : *to go along*" which can be split like in "Gehen Sie mit? : *Are you going with?*". For more extensive overview of German morphology, see e.g. [Fox 2005].

1.9.3 Polish

Polish is also considered as one of the morphologically rich languages. It belongs to the family of Slavic languages like Russian, Czech, and Bulgarian. It is characterized by a high degree of inflection, having seven cases and two number classes. Polish does not use definite or indefinite articles. It has a complex gender system like almost all the other Slavic languages. This is due to the combination of three categories: gender (masculine, feminine, neutral), personhood (personal versus non-personal) and animacy (animate versus inanimate). Declensional endings depend on case, number and gender. In addition, declension changes if the word is noun or adjective. Moreover, word stems are frequently modified by the addition or absence of endings. Also, verbs are inflected according to gender as well as person and number. For more detailed overview of Polish language, see e.g.[Swan 2002].

1.10 Language Modeling: the State of the Art

Over the last few decades, the backoff m-gram LM estimated over sequences of full-words has been considered as the state of the art in the language modeling field. This is because of the strong recognition performance of this model relative to its simplicity and low computational complexity. The major focus of this thesis is to develop improved language modeling approaches suitable for dealing with morphologically

[2]For non-English words, we give the word and the English translation separated by ":".

rich languages. Recently, several advanced language modeling ideas have been proposed. Some of these ideas have focused on the choice of the proper recognition unit, such as different types of sub-words. Broadly speaking, these ideas produce a set of models called *sub-word based LMs*. Other ideas have focused on incorporating word classes into the LM estimation process, such as morphology-based classes. Other approaches have focused on the fundamental modeling and parameter estimation techniques such as Bayesian LMs, and continuous space LMs.

The following sections give an overview of the most important research in these directions.

1.10.1 Sub-Word Based LMs

One approach to deal with morphologically rich languages is the sub-lexical language modeling. The words of the underlying language are decomposed into some type of sub-words called sub-lexical units, then m-gram LMs are estimated over sequences of these sub-words. Normally, the number of possible sub-words in a given text corpus is smaller than the number of the full-words which leads to higher lexical coverage. Also, the sub-words can be properly combined to produce a wide range of new words achieving lower OOV rates. In addition, the average frequency of sub-words is larger than the average frequency of full-words which helps to reduce the effect of data sparsity leading to more reliable probability estimates. Possible types of sub-word units are: morphemes, syllables, phonemes, and graphones.

Morpheme-based LMs. An example of a sub-word based LM is the morpheme-based LM in which the probability estimation is performed over sequences of morphemes rather than sequences of full-words. Normally, morphemes are generated from the full-words by applying *morphological decomposition* based on supervised or unsupervised approaches. The supervised approaches make use of linguistic knowledge, like in [Lamel & Messaoudi$^+$ 2008; Xiang & Nguyen$^+$ 2006], where decomposition is performed for Arabic words based on Buckwalter Arabic morphological analyzer (BAMA) [Buckwalter 2004] along with some added constraints. In [Afify & Sarikaya$^+$ 2006], a decomposition method starts with a fixed set of affixes and decomposes Arabic words into stems and affixes based on pattern matching approaches. In [Choueiter & Povey$^+$ 2006; Lee & Papineni$^+$ 2003], a LM based morpheme generator is used to perform decomposition for Arabic words, along with a morpheme lattice constrainer to reject illegal sequences of morphemes. In [Byrne & Hajič$^+$ 2000], a carefully built morphological analyzers based on lexical and syntactic knowledge is used for Czech language. However, in [Berton & Fetter$^+$ 1996], a manually decomposed lexicon is used for German speech recognition. Whereas, in [Adda-Decker & Adda 2000], a set of manual rules is developed for German word decomposition. Although the supervised decomposition is normally optimized for high performance, it requires labor-intensive work and still suffers from the so-called *unknown word problem*, that is, words that are not explicitly coded into the system. On the other side, the unsupervised approaches are statistical based data-driven approaches [Adda-Decker 2003; Ordelman & Hassen$^+$ 2003; Rotovnik & Maučec$^+$ 2007]. In [Larson & Willett$^+$ 2000], an algorithm is proposed that decomposes words according to the statistical relevance of the resulting constituents. In [Ordelman & Hassen$^+$ 2003], a compound splitting algorithm is developed for Dutch language that uses sorting, word length, and word frequency information. Other unsupervised methods are based on the minimum description length (MDL) principle like in [Creutz 2006; Creutz & Hirsimäki$^+$ 2007]. On the contrary, unsupervised approaches are language independent as they do not require any language specific knowledge and can be applied to any language.

Syllable-based LMs. Another type of a sub-word based LM is the syllable-based LM in which the main recognition unit is the syllable which consists of one or more written letters representing a unit of speech. A syllable can also be recognized as a phonological building block of words. Syllables are generated from the full-words by performing a process called *syllabification*. In most languages, syllabification can be achieved by applying linguistic and phonetic rules. Syllable-based LMs have been used for languages like Polish [Piotr 2008], and English [Schrumpf & Larson$^+$ 2005]. In [Çarki & Geutner$^+$ 2000], a sub-lexical approach is proposed for Turkish language which starts by initially breaking up words into syllables using syllabification rules and then merging syllables into larger units by defining syllable classes.

Phoneme- and graphone-based LMs. A different type of unit makes use of the word pronunciation and aims at combining the pronunciation model with the language model in one joint model. In [Bazzi & Glass 2000; Creutz & Hirsimäki+ 2007], a phoneme-based model is augmented to a word model so that any OOV word can be recognized as an arbitrary sequence of phonemes. In [Klakow & Rose+ 1999], multi-phoneme fragments are automatically constructed and integrated into the lexicon and the LM, but no attempt is made to convert phoneme sequences into proper words. Alternatively, in [Galescu 2003], a set of automatically derived units based on joint grapheme-phoneme components, called graphones, are augmented to a normal word model leading to small improvements in the WER for an English LVCSR task. In [Bisani & Ney 2005], a set of fragment-based graphones are used for OOV words in order to perform an English dictation task, where the graphones are constructed based on arbitrary fragments with some length constraints but without any linguistic considerations. The choice of the set of graphones is based on a grapheme-to-phoneme (G2P) conversion model as described in [Bisani & Ney 2008].

Criticism. Most of the previously cited work is either based on small vocabulary sizes or is lacking proper optimization, like selecting the most suitable type of units, optimizing the overall vocabulary size, number of different units, and OOV rates. In addition, the existing work on graphone units does not take into consideration the optimization of the set of graphones as recognition units to achieve the best recognition performance in a LVCSR system; rather, they are only optimized to provide the minimum phoneme error rates (PERs) in G2P conversion tasks. In addition, the performance of graphone units has not been tested on very large vocabulary systems.

1.10.2 Word Classes in LMs

Another approach to overcome the data sparseness and to reduce the dependence of the traditional word based LMs on the discourse domain is to incorporate other word classes taken from different knowledge sources into the LMs rather than only normal words. Usually, this approach yields better smoothing and better generalization with regard to unseen word sequences. Word classes can be generated based on linguistic methods as in [Kirchhoff & Vergyri+ 2006; Maltese & Bravetti+ 2001], or via data-driven approaches as in [Brown & deSouza+ 1992; Kneser & Ney 1993a, 1991]. Examples of such classes are the morphology-based classes. Possible types of LMs are: stream-based, class-based LMs, and factored LMs.

Stream- and class-based LMs. Possible approaches for incorporating word classes into LMs are the stream-based LMs [Kirchhoff & Vergyri+ 2006], and the class-based LMs [Brown & deSouza+ 1992; Kneser & Ney 1991]. Each of these two approaches treats every class stream separately without considering any interaction among classes during the backoff process. The stream-based LM is a normal backoff m-gram model built over of a stream of class assignments. Whereas, the class-based LM is a model that combines the m-gram model over a stream of class assignments with the probability distribution of words within classes in order to better estimate smoothed probabilities of word sequences. In [Kirchhoff & Vergyri+ 2006], both stream- and class-based LMs are successfully used to obtain significant reductions in WERs on a conversational Egyptian Arabic speech recognition task. In [Maltese & Bravetti+ 2001], a combination of word-based LMs and different class-based LMs are used to perform LVCSR for French, English, German, Italian, and Spanish leading to significant reductions in both perplexities (PPLs) and WERs. In principle, the generation of word classes can be performed using linguistic or data-driven approaches. The most important work on data-driven word clustering have been introduced by [Brown & deSouza+ 1992; Kneser & Ney 1991, 1993b; Martin & Liermann+ 1998], where clustering is performed using the criterion of perplexity improvement. In [Maltese & Bravetti+ 2001], a comparison is made between different linguistic and data-driven classes based on manual and automatic word-clustering techniques. In [Gao & Goodman+ 2001], a generalized version of the class-based LM called the *cluster-based LM* is utilized for Japanese and Chinese languages. Therein, a major focus is given to investigate the best methods to use the classes. Using these models, significant reductions in PPLs are obtained.

Factored LMs. A different model that is mainly used to incorporate morphology-based word classes into LMs is the factored LM (FLM) [Bilmes & Kirchhoff 2003]. This model uses a complex backoff

mechanism in a form of a predefined backoff graph in order handle different class streams jointly during the backoff. The concept of the generalized parallel backoff (GPB) is introduced by [Bilmes & Kirchhoff 2003; Kirchhoff & Bilmes$^+$ 2008] in which the model backs-off to multiple combinations of classes in parallel during the training time, statistics are then collected and combined from every backoff path to estimate the required probability. The FLMs have been successfully applied to Arabic and Amharic ASR tasks, like in [Kirchhoff & Bilmes$^+$ 2002; Tachbelie 2010; Tachbelie & Abate$^+$ 2011]. However, no attempt has been made to use such models in other morphologically-rich languages.

Criticism. Most of the previously cited work on class-based LMs is focused on the use of data-driven classes. However, for morphologically rich languages, highly reliable morphology-based classes can be efficiently derived based on existing morphological analyzers. In addition, factored LMs have not been properly compared to interpolated class-based LMs. In fact, the use of factored LMs has not been tried for languages beyond Arabic and Amharic, and most of the existing work does not consider the use of very large vocabularies which is essential in typical tasks. Moreover, no trial has been made to use classes for sub-word based LMs.

1.10.3 Novel LM Estimation

There has been a considerable amount of research aimed at improving the conventional backoff m-gram LMs. Examples of such trials are: the hierarchical Pitman-Yor LM (HPYLM) and the continuous space LMs based on neural networks.

Hierarchical Pitman-Yor LMs. Recently, the hierarchical Bayesian LMs [Blei & Ng$^+$ 2003; Gelman & Carlin$^+$ 2003] have succeeded to achieve a comparable performance to the state-of-the art m-gram LMs smoothed with the modified Kneser-Ney (MKN) smoothing [Chen & Goodman 1996]. A hierarchical Pitman-Yor LM (HPYLM), initially introduced in [Teh 2006a], is a type of Bayesian LM based on the Pitman-Yor (PY) process that has been shown to improve both the PPL and the WER over the modified Kneser-Ney smoothed m-gram LMs [Huang & Renals 2007]. The Pitman-Yor (PY) process is a generalization of the widely used Dirichlet distribution [Ishwaran & James 2001; Pitman 2002; Pitman & Yor 1997]. This PY process produces *power-law* distributions over word frequencies [Goldwater & Griffiths$^+$ 2006, 2011]. This means that few number of words occur with high frequencies, while most words occur with low frequencies. This distribution has been found to be one of the most striking statistical properties of word frequencies in natural languages.

Continuous space LMs. Recently, continuous space LMs have shown significant performance improvements in LVCSR tasks. There is currently a growing research interest in such models because they allow for higher levels of generalization as a result of the inherent smoothing capabilities in continuous space. In fact, there is a large group of techniques that aim at estimating LMs in continuous space. This includes: the shallow neural network LM (SNNLM), investigated in [Bengio & Ducharme 2001; Bengio & Sénécal 2003; Bengio & Ducharme$^+$ 2003], which uses a single hidden layer feed-forward neural network to estimate the LM. In [Arisoy & Sainath$^+$ 2012], a deep neural network LM (DNNLM) is investigated, where a feed-forward neural network employing multiple hidden layers is used as the probability estimator in continuous space to perform speech recognition on a Wall Street Journal (WSJ) task. On the other hand, in [Kombrink & Mikolov$^+$ 2011; Mikolov 2012; Mikolov & Karafiát$^+$ 2010], a different type of continuous space LM is proposed that uses a recurrent neural network (RNN), this LM is called RNNLM. An improved type of RNN that is recently used to estimate continuous space LMs is called the long short-term memory (LSTM) RNN [Gers 2001; Graves & Schmidhuber 2005; Hochreiter & Schmidhuber 1997]. This LM is called LSTMLM [Sundermeyer & Schlüter$^+$ 2012; Sundermeyer & Oparin$^+$ 2013]. Using these continuous space LM approaches, significant reductions in both PPLs and WERs are reported.

Criticism. The use of hierarchical Pitman-Yor LMs has not been sufficiently investigated on very large vocabulary speech recognition tasks. In addition, no trial has been made to use hierarchical Pitman-Yor models to estimate class-based LMs on sub-word level. On the other hand, although there exist many

publications on recurrent neural network LMs, to the best of our knowledge, only one trial has been made to investigate the use of feed forward deep neural network LMs [Arisoy & Sainath$^+$ 2012]. This previous work is performed on a small WSJ task with small vocabularies. Furthermore, no trial has been made to use word classes with the neural network LMs.

Chapter 2

Scientific Goals

The major interest of this thesis is to develop improved language modeling approaches to deal with the problems related to the LVCSR of morphologically rich languages, like Arabic, German, and Polish. Examples of such problems are: data sparsity, high OOV rates, poor LM probability estimates, and lack of generalization to unseen word sequences. The first objective of this thesis is to investigate and extend the use of sub-word based LMs using different types of sub-word units like, morphemes and syllables, as well as combining the pronunciation model with the language model using graphonemic units. This also includes the use of hybrid LMs containing a mixture of various sub-word units along with full-words in one flat model. A major attention is paid to perform a careful optimization of the sub-word based models as well as the competing full-word models in order to show the actual potential of this approach. The second objective is to develop a novel methodology in which morphology-based classes are used for LM estimation, where the classes are derived on sub-word level rather than full-word level. Thereby, the benefits of sub-word based language modeling are retained simultaneously with the advantages of using classes. The third objective is to explore the combination of the aforementioned approaches with the recent state-of-the-art language molding techniques, like hierarchical Pitman-Yor LMs (HPYLMs), and continuous space LMs based on feed forward deep neural networks (DNNLMs). Starting from these objectives, a set of scientific goals are derived, which include:

Development of optimized sub-word based language models. A language modeling approach which is more appropriate for morphologically rich languages is the sub-word based LMs, also known as sub-lexical LMs. The words of the underlying language are decomposed into some type of sub-word units, called sub-lexical units, then m-gram LMs are estimated over sequences of these sub-words. Normally, the number of possible sub-words in a given text corpus is smaller than the number of full-words in the same corpus which leads to higher lexical coverage. Also, sub-word units can be properly combined to produce a wide range of previously unseen words achieving lower OOV rates. In addition, the average frequency of such units is larger than the average frequency of full-words, which helps to reduce the effect of data sparsity and leads to more reliable probability estimates. This approach has been successfully used for various languages. However, most of the previous work is either based on small vocabulary sizes or is lacking proper optimization, like selecting the most suitable type of units, optimizing the overall vocabulary size, number of different units, and OOV rates. In this thesis, an extensive study is performed on the use of carefully optimized sub-word based LMs for LVCSR of morphologically rich languages like Arabic [El-Desoky & Gollan$^+$ 2009], German [El-Desoky & Shaik$^+$ 2010] and Polish [Shaik & El-Desoky$^+$ 2011b]. Investigations are made on different types of sub-word units based on supervised and unsupervised word decomposition. The proposed LMs are optimized over different types of units, like words, morphemes and syllables using very large vocabulary sizes that go up to one million.

Joint language and pronunciation models. Joint language and pronunciation models are investigated in oder to help improving the performance of LVCSR systems. This can be accomplished by incorporating pronunciations into LMs using sub-word units combined with their context dependent pronunciations. In a previous work, fragments of words are combined with their pronunciations to form a so called fragment-based graphone units that are mainly used to model OOV words [Bisani & Ney 2005]. Therein, sub-words are defined by arbitrary word fragments with some length constraints. These graphone units are derived from a grapheme-to-phoneme (G2P) conversion model [Bisani & Ney 2008]. However, they are not well optimized as recognition units to achieve the best recognition performance in a LVCSR system; rather, they are only optimized to provide the optimum G2P conversion performance. In addition, the

performance of such units has not been tested on very large vocabulary systems. In this thesis, novel types of graphones are introduced, in which optimized morphemic or syllabic sub-words are combined with their context dependent pronunciations. This serves as a combination of language and pronunciation models in one joint probability distribution. To create such models, a modification procedure is proposed on top of the conventional fragment-based graphones using letter-phoneme sequence alignment via dynamic programing (DP) and expectation maximization (EM) [Shaik & El-Desoky$^+$ 2011b]. These novel models are called morpheme- or syllable-based graphone models and are tested on very large vocabulary systems.

Development of extended hybrid language models. Another novel approach is introduced, in which the use of lexicons and LMs comprising mixed types of lexical and sub-lexical units is investigated like: full-words, morphemes, syllables, morpheme-based graphones, and syllable-based graphones [Shaik & El-Desoky$^+$ 2011a]. For example: full-words, morphemes, and morpheme-based graphones; or full-words, syllables, and syllable-based graphones are used in one hybrid lexicon and LM. This novel mixture of units is forming a so called extended hybrid LMs suitable for open vocabulary LVCSR tasks, where systems operate over open, constantly changing vocabularies. The numbers of the vocabulary items of each type of unit are optimized so as to obtain the minimum WER.

Investigations on using morphology-based classes in language models. An approach to overcome the data sparseness and to reduce the dependence of the traditional word-based LMs on the discourse domain is to incorporate word classes into the LM estimation process rather than using only words or sub-words. Usually, this approach yields better smoothing and better generalization with regard to unseen word sequences. In principle, word classes can be generated based on linguistic or data-driven methods. Most of the previous work has been focused on the use of data-driven classes in LMs. Whereas, less work has been performed using linguistic classes. This thesis pays a major attention to the use of morphology-based classes for LMs of morphologically rich languages. It shows how efficient morphology-based classes can be generated based on existing morphological analyzers for Arabic and German languages.

Examples of LMs that operate on word classes are the stream-based LMs [Kirchhoff & Vergyri$^+$ 2006] and the class-based LMs [Brown & deSouza$^+$ 1992; Kneser & Ney 1993a, 1991]. In these models, every class stream is treated separately without considering any interaction among different classes during the backoff process. Thus, a separate stream- or class-based LM is built over every individual class. In the work of this thesis, multiple stream- and class based LMs are estimated over morphology-based classes [El-Desoky & Schlüter$^+$ 2012]. A combination of these models is accomplished via model interpolation or N-best score combination.

Another approach that makes use of word classes is the factored LM [Bilmes & Kirchhoff 2003; Kirchhoff & Bilmes$^+$ 2002]. In this model, both words and their classes are viewed as generic factors. Every word is considered as a vector consisting of a set of parallel factors over which the probability estimation is to be performed. The factored LM uses a complex backoff mechanism in a form of a predefined backoff graph in order handle different class streams jointly during the backoff. The main idea of the model is to backoff to different combinations of classes when some word m-gram is not sufficiently observed in the training data. Thereby, the probability estimates are improved by taking into account the joint interaction among classes in training time. This thesis shows how optimized FLMs can be estimated over morphology-based classes for different languages, namely Arabic and German [El-Desoky & Schlüter$^+$ 2010; El-Desoky & Shaik$^+$ 2011]. A detailed comparison with interpolated word- and class-based LMs is performed.

Combining the benefits of sub-word based LMs and morphology-based classes. A novel approach is introduced that retains the benefits of the sub-word based LMs along with the advantages of using morphology-based classes. This is accomplished via generating classes on the sub-word level, namely the morpheme level, rather than the level of full-words. Hence; stream-based, class-based, and factored LMs are estimated over sequences of morphemes and their classes [El-Desoky & Schlüter$^+$ 2010; El-Desoky & Shaik$^+$ 2011; El-Desoky & Schlüter$^+$ 2012].

In case that language model interpolation is not directly possible, score combination is performed over N-best sentences in order to benefit from different LMs during rescoring. Usually, for N-best rescoring, scores used for re-ranking the N-best hypotheses are normally a weighted combination of several component

scores representing the acoustic score, the LM score and the number of words. However, scores from various LMs can be also added. The final score for each hypothesis is computed as a log-linear combination of the invoked scores. The weights of this combination can be optimized to achieve the minimum WER on the development set.

Investigations on hierarchical Pitman-Yor language models. A well known fact about the backoff m-gram LM is that it is not based on an internally coherent Bayesian probabilistic model. This makes it difficult to describe the advantages of the backoff m-gram model in terms of how closely it copes with the inherent properties of natural languages. Among the approaches that aim at improving the m-gram LM, hierarchical Bayesian LMs [Blei & Ng$^+$ 2003; Gelman & Carlin$^+$ 2003] have succeeded to achieve a comparable performance to the state-of-the art m-gram LMs smoothed with the modified Kneser-Ney smoothing [Chen & Goodman 1996]. A hierarchical Pitman-Yor LM (HPYLM) is a type of hierarchical Bayesian LM that is based on a coherent Bayesian probabilistic model which explicitly declares prior assumptions over the LM parameters [Huang & Renals 2007; Teh 2006a]. HPYLM is based on the Pitman-Yor process which is a generalization of the widely used Dirichlet distribution [Pitman & Yor 1997]. The resulting model is considered as a direct generalization of the hierarchical Dirichlet LM proposed by [MacKay & Peto 1994]. The PY process produces *power-law* distributions over word frequencies. This means that few number of words occur with very high probabilities, whereas most words occur with low probabilities. This distribution has been found to be one of the most striking statistical properties of word frequencies in natural languages.

In this thesis, hierarchical Pitman-Yor models are utilized to estimate class-based LMs on morpheme level. In other words, the traditional modified Kneser-Ney smoothed models are replaced with the hierarchical Pitman-Yor models. This is a novel approach that aims at combining the benefits of sub-word based LMs and morphology-based classes with the advantages of the HPYLMs.

Investigations on continuous space language models using feed-forward deep neural networks. One of the major disadvantages of the backoff m-gram LM is its poor modeling performance in cases of data sparseness. In fact, data sparseness is an essential problem in morphologically rich languages. Even when large training corpora are used, still extremely small probabilities can be assigned to many valid word sequences. This is an inherent disadvantage of all LMs estimated in a discrete space. The discrete nature of such models makes it difficult to reach high levels of generalization even after applying the most efficient smoothing techniques, like the modified Kneser-Ney (MKN) smoothing of the backoff m-gram models [Chen & Goodman 1996]. Actually, the main problem comes from the lack of a notion of word similarity. Indeed, the use of morphology-based classes with discrete space LMs, like the stream-based; class-based; and factored LMs, introduce a partial solution to this problem by supporting the probability estimation process with word classes in cases of data sparseness.

In contrast, the neural network LM (NNLM) performs probability estimation in a continuous space using a single hidden layer (shallow) feed-forward neural network [Bengio & Ducharme$^+$ 2003; Schwenk 2007; Schwenk & Gauvain 2005]. The projection of words into continuous space is done jointly with the neural network training in a single process. This ensures the learning of the most suitable projection matrix that best fits the probability estimation task. Thereby, words that are semantically or grammatically related are mapped to similar locations in the continuous space. This is considered as a built-in smoothing capability that enables the model to achieve better generalization to unseen m-grams.

On the other hand, deep neural networks (DNNs) with multiple hidden layers are able to capture high-level and abstract information about the input data. Recently, DNNs have shown improved performance compared to shallow networks with a single hidden layer across different tasks [Arisoy & Sainath$^+$ 2012; Bengio 2009; Dahl & Ranzato$^+$ 2010; Mohamed & Dahl$^+$ 2009; Seide & Li$^+$ 2011].

In this thesis, the use of DNNs is explored for language modeling (DNNLMs). In addition, DNNs are used to estimate sub-word based LMs rather than full-word based LMs. Moreover, the input of the DNNs is augmented with morphology-based word and sub-word classes in order to estimate robust LM probabilities for morphologically rich languages. To achieve the best performance, interpolation is performed between the DNNLMs and the backoff LMs. This is a novel approach that combines the benefits of sub-word based LMs and morphology-based classes with the improved modeling of the DNNLMs.

The remainder of the thesis is organized as follows: Chapter 3 introduces the use of sub-word based LMs as well as the combination of language and pronunciation models. It also introduces the extended hybrid LMs. Chapter 4 introduces the use of stream-based, class-based, and factored LMs that make use of various morphology-based classes. In addition, it presents the use of morphology-based classes with sub-word based LMs. Moreover, this chapter also introduces the utilization of the hierarchical Pitman-Yor approach to estimate class-based LMs. Chapter 5, introduces the use of continuous space LMs based on feed-forward deep neural networks to estimate sub-word based LMs with morphology-based classes. Chapter 6 concludes the thesis by a summary of the scientific contributions. An outlook is given in Chapter 7. The appendix contains a detailed description of systems and corpora used throughout the experiments conducted in this thesis.

Chapter 3

Sub-Word Based Language Models

A sub-word based LM refers to this type of LM that is estimated over some fractions of the graphemic word, called *sub-word units* or equivalently *sub-lexical units*, rather than full-words. However, it is also possible that the graphemic sub-word unit is combined with some phonetic sequence representing its pronunciation. The resulting sub-lexical unit in this case is called a *graphone*. A model estimated over such units represents an implicit combination of language model and pronunciation model in one joint probability distribution. Normally, the number of possible sub-words in a given text corpus is smaller than the number of full-words which leads to higher lexical coverage. Besides that, these sub-words can be properly combined to produce a wide range of unseen words achieving lower OOV rates. In addition, the average frequency of sub-words is larger than the average frequency of full-words which helps to reduce the effect of data sparsity leading to more reliable probability estimates. Previously used types of sub-lexical units are: morphemes [Choueiter & Povey$^+$ 2006; Creutz 2006; Creutz & Hirsimäki$^+$ 2007; Lamel & Messaoudi$^+$ 2008; Lee & Papineni$^+$ 2003; Xiang & Nguyen$^+$ 2006], syllables [Piotr 2008; Schrumpf & Larson$^+$ 2005; Xu & Ma$^+$ 1996], phonemes [Bazzi & Glass 2000; Creutz & Hirsimäki$^+$ 2007; Klakow & Rose$^+$ 1999] and graphones based on arbitrary word fragments [Bisani & Ney 2005, 2008; Galescu 2003].

In this chapter, we perform an extensive study of the sub-word based language modeling approach used to perform LVCSR for morphologically rich languages. We use different types of units; such as full-words, morphemes, syllables, and graphones. We show how to carefully optimize both the sub-word based models and the competing full-word based models. In addition to the previously known fragment-based graphones, we propose novel types of graphones where morpheme or syllable sub-words are combined with their context dependent pronunciation in order to effectively combine the language and pronunciation model in one joint distribution. Another novel contribution comes in the use of extended hybrid lexicons and LMs comprising multiple types of units in one flat model. The numbers of the used vocabulary items of each type of unit are carefully optimized so as to achieve the minimum WER. The recognition experiments are performed on Arabic, German, and Polish tasks. The approaches described in this chapter have been introduced in [El-Desoky & Gollan$^+$ 2009; El-Desoky & Shaik$^+$ 2010], and further investigated in [Shaik & El-Desoky$^+$ 2011a,b].

Section 3.1 describes the sub-word based m-gram models and their perplexities. Section 3.2 introduces different the types of sub-word units used in our work. Section 3.3 discusses the word decomposition approaches used to generate different types of sub-words. Section 3.4 shows how these sub-words are combined with pronunciations and incorporated into LMs. Section 3.5 presents experimental results performed on different development and evaluation corpora. Whereas, Section 3.6 presents a summary of recognition results recorded for external evaluation campaigns. Section 3.7 summarizes the chapter.

3.1 Sub-Word Based m-gram Models

As previously discussed in Section 1.5, a standard word-based statistical m-gram LM computes the probability of some text corpus T composed of a word sequence w_1^N as a product of the conditional probabilities of individual words given their histories. Histories are approximated by a limited number of preceding

$m-1$ words such as 1,2 or 3 words, resulting into bigram, trigram or 4-gram LMs respectively. Thus:

$$\begin{aligned} p(T) = p(w_1^N) &= \prod_{n=1}^{N} p(w_n|w_1^{n-1}) \\ &= \prod_{n=1}^{N} p(w_n|w_{n-m+1}^{n-1}) \end{aligned} \tag{3.1}$$

The conditioning part w_{n-m+1}^{n-1} is called the *history* which is composed of $m-1$ previous words. The conditional probabilities $p(w_n|w_{n-m+1}^{n-1})$ make up the LM.

A similar model as in Equation 3.1 can be built over a decomposed text, where words are replaced by sub-word units. In this case, the resulting m-gram model is called a *sub-word based m-gram LM*. Such a LM is used to model the regularities governing sequences of sub-word units such as sequences of morphemes or syllables rather than sequences of words.

Usually, the quality of the LM is measured by its perplexity over some text corpus $T = w_1^N$ defined as:

$$PP(T) = PP(w_1^N) = \left[\prod_{n=1}^{N} p(w_n|w_{n-m+1}^{n-1})\right]^{-1/N} \tag{3.2}$$

It is also widely common to compute the log perplexity, which takes the form:

$$\log PP(T) = \log PP(w_1^N) = \frac{-1}{N}\sum_{n=1}^{N} \log p(w_n|w_{n-m+1}^{n-1}) \tag{3.3}$$

Character-level perplexity. In order to be able to compare perplexities across different types of recognition units; such as full-words, morphemes, syllables ... etc.; we define a so-called character-level perplexity in which the conventional word or sub-word level perplexity is normalized on character level. Consider a text corpus T having N words[1], the word-level perplexity $PP(T)$ is computed as given in Equation 3.2. Assume that T consists of M characters[2], then, to go from the word-level perplexity $PP(T)$ to the character-level perplexity $PP_c(T)$, we use the following equation:

$$\begin{aligned} PP_c(T) &= [p(T)]^{\frac{-1}{M}} \\ &= \left[PP(T)^{-N}\right]^{\frac{-1}{M}} \\ &= [PP(T)]^{\frac{N}{M}} \end{aligned} \tag{3.4}$$

Similar to Equation 3.3, the log character-level perplexity takes the form:

$$\log PP_c(T) = \log PP_c(w_1^N) = \frac{N}{M}\log PP(T) \tag{3.5}$$

Perplexity for in- and out-of-vocabulary text. It is usually a common practice during the computation of the perplexity to replace all the unknown words that are not members of the LM vocabulary with a fixed *unk* symbol that is explicitly assigned a probability during the model training. This normally leads to some untruthful reduction in the overall perplexity when computed for some text corpus. For this reason, it would be interesting to compute the perplexity after excluding this *unk* symbol. This is in fact the perplexity of only the in-vocabulary region of the text corpus. To illustrate the relationship between the perplexities in the in-vocabulary and the out-of-vocabulary regions, consider a text corpus T having N words divided into two regions T_{inv} and T_{oov} representing the in-vocabulary and the out-of-vocabulary regions of the corpus respectively. Let N_{inv} and N_{oov} be the number of words in every region respectively, where $N = N_{inv} + N_{oov}$, and let $PP(T)$ be the perplexity of the text T with probability $p(T)$[3], then the

[1] The count of words includes the count of sentence end symbols ($< /s >$).
[2] The count of characters includes the count of word and sentence boundaries.
[3] Contexts of words are preserved while computing $p(T_{inv})$ and $p(T_{oov})$ whether they fall into *inv* or *oov* regions.

following relationship can be drawn:

$$\begin{aligned} PP(T) &= [p(T)]^{\frac{-1}{N}} \\ &= [p(T_{inv}) \cdot p(T_{oov})]^{\frac{-1}{N}} \\ &= \left[PP(T_{inv})^{-N_{inv}} \cdot PP(T_{oov})^{-N_{oov}}\right]^{\frac{-1}{N}} \\ &= PP(T_{inv})^{\frac{N_{inv}}{N}} \cdot PP(T_{oov})^{\frac{N_{oov}}{N}} \end{aligned} \tag{3.6}$$

In the log domain, the relationship takes the form:

$$\log PP(T) = \frac{N_{inv}}{N} \log PP(T_{inv}) + \frac{N_{oov}}{N} \log PP(T_{oov}) \tag{3.7}$$

If T has M characters, T_{inv} has M_{inv} characters, and T_{oov} has M_{oov} characters with $M = M_{inv} + M_{oov}$; then a similar relationship can be deduced for the character-level perplexity, thus:

$$\log PP_c(T) = \frac{M_{inv}}{M} \log PP_c(T_{inv}) + \frac{M_{oov}}{M} \log PP_c(T_{oov}) \tag{3.8}$$

3.2 Sub-Word Units

In this section, we define the types of sub-word units that we use in our LMs. A detailed description of how these units are generated is given in the next section.

3.2.1 Morphemes

One possible type of unit is the *morpheme* defined as the smallest linguistic component of the word that holds a semantic meaning. Normally, morphemes are generated from the full-words by applying morphological decomposition based on supervised or unsupervised approaches. The supervised approaches make use of linguistic knowledge, like the knowledge provided by carefully designed morphological analyzers. Whereas, the unsupervised approaches are statistical data-driven approaches that are essentially language independent and can be applied to any language.

In our experiments on Arabic LVCSR, we use morphemes generated via supervised approaches based on an available morphological analyzer called MADA [Habash & Rambow 2005, 2007]. However, for both German and Polish LVCSR experiments, we use morphemes generated via unsupervised approaches implemented in a tool called Morfessor [Creutz & Lagus 2005]. More details about generating morphemes are given in Sections 3.3.1, and 3.3.2.

3.2.2 Syllables

Another type of sub-word unit is the *syllable* which consists of one or more written letters representing a unit of speech. It can be also recognized as a phonological building block of words. Although the syllables of a given word are more related to its pronunciation, they are still representing a set of written sub-words that can be used for sub-word based language modeling. Usually, a syllable consists at least of a nucleus that can either be a vowel or a diphthong. Consonant clusters can enclose the nucleus and must fulfill the phonotactic[4] restrictions to form a valid syllable [Kemp & Jusek 1996; Möbius 1998; Rubach & Booij 1990]. In most languages, syllabification can be achieved by applying linguistic and phonetic rules.

In this work, we use syllables for Polish LVCSR experiments. Syllables are generated using a rule based tool. More details about syllabification are discussed in Section 3.3.3.

[4] phonotactics: the set of allowed arrangements or sequences of speech sounds in a given language.

3.2.3 Graphones

A different type of unit is the *graphone* which is a combination of the graphemic sub-word with its context dependent pronunciation forming one joint unit. Thus, a graphone consists of two components: a graphemic component and a phonemic component. Using such units in LMs allows different context dependent pronunciations of sub-words to be captured on the level of the LM rather than the lexicon level. This is an implicit combination of pronunciation model and language model in one joint distribution. This approach is mainly used to cope with high OOV rates. The traditional word model is augmented with a specialized graphone-based model dedicated for modeling OOV words. The goal of this OOV modeling is to be able to spell out new words as sequences of graphones. Usually, the presence of the OOV words can also affect the adjacent words leading to mis-recognition of in-vocabulary words. According to [Bisani & Ney 2005], each OOV word causes 1.5 to 2 errors on average. Therefore, the successful recognition of OOV words can positively affect the recognition of the adjacent words. Usually, the type of the graphone is determined according to the type of its graphemic component. In literature, like in [Bisani & Ney 2005; Galescu 2003], only fragment-based graphones are used, where the graphemic components are just arbitrary fragments with some length constraints but without any linguistic considerations.

In this thesis, we investigate the use of *fragment-based* graphones in addition to a novel type of unit in which we redefine the graphones so that the graphemic components represent morphemes or syllables with the phonemic components kept correspondingly [El-Desoky & Shaik$^+$ 2010; Shaik & El-Desoky$^+$ 2011b]. We call the resulting units *morpheme-based* and *syllable-based* graphones. Details are discussed in Section 3.4.

3.2.4 Arabic Diacritized Sub-Words

Due to the unique properties of Arabic language, a special treatment is usually applied for incorporating pronunciations into Arabic LMs. As previously discussed in Section 1.9.1, Arabic script is usually written without short vowels. If the short vowels are explicitly given, then they are indicated by diacritical marks rather than normal letters. A word with full diacritics is called a *diacritized word*. For a non-diacritized word, there are several possible pronunciations that may occur during speaking. The speaker infers the suitable pronunciation based on the context. However, a completely diacritized word determines only one possible pronunciation. This means that a diacritized word is in fact a combination of the word with its pronunciation in one encapsulated unit. Hence, the diacritized sub-words can be used in place of graphones to incorporate pronunciations into Arabic LMs. In traditional Arabic systems, the vowel information is not captured in the LMs. Instead, a relatively high number of pronunciation variants are used during the ASR search in order to fill the gap between the spoken and the written language. Recently, it has been shown that modeling short vowels into Arabic LMs can improve performance [Afify & Nguyen$^+$ 2005; Messaoudi & Gauvain$^+$ 2006].

In this work, we automatically generate diacritized Arabic words based on an MADA morphological analyzer that performs both decomposition and diacritization in one process through full morphological tagging [Habash & Rambow 2005, 2007]. More details are given in Section 3.3.1.

3.3 Word Decomposition

As mentioned in Section 3.2.1, morphemes can be generated from words by applying supervised or unsupervised morphological decomposition approaches. For Arabic language, we used supervised decomposition based on linguistic knowledge. Fortunately, powerful morphological analyzers are freely available for Arabic. For German and Polish experiments, we used unsupervised morphological decomposition based on statistical approaches that follow the minimum description length (MDL) principle. For this purpose, another powerful tool is also freely available. In addition to morphemes, we also used syllables for German and Polish LVCSR systems. To generate syllables, we used a syllabification tool that applies a set of linguistic and phonetic syllabification rules. The details are given in the following sections.

3.3.1 Supervised Morphological Decomposition

In our Arabic LVCSR experiments, we generate morphemes from full-words by applying supervised morphological decomposition based on linguistic knowledge provided by the morphological analyzer and disambiguator tool for Arabic (MADA)[5] [Habash & Rambow 2005, 2007]. MADA is a tool that is built on top of the Buckwalter Arabic morphological analyzer (BAMA) [Buckwalter 2004] in order to perform morphological tagging, disambiguation [Habash & Rambow 2005], diacritization [Habash & Rambow 2007], along with tokenization [Habash & Sadat 2006] of modern standard Arabic (MSA). First, BAMA is used to generate all possible analyses for a word in a given sentence. Then, MADA applies a set of classifiers to the word morphological features. Given the output of these classifiers, a combiner is used to rank the potential word analyses returned by BAMA, and the highest ranked analysis is chosen for this word. At the end, MADA is able to associate a complete set of morphological tags with each word with a reliable accuracy score of around 95%. These tags are used to produce robust word diacritization and tokenization along with stem orthographic normalization.

In our experiments, MADA tool is slightly modified in order to apply the following transformation to all the words of the LM training corpora:

$$w \rightarrow \bar{w}/p_1+ \quad p_2+ \ldots p_n+ \quad s \quad +f$$

where w is the original non-diacritized word, $\bar{w}$ is the diacritized word, $p_1+ \quad p_2+ \ldots p_n+$ is an optional sequence of prefixes, where $0 \leq n \leq 2$ is the number of prefixes, s is a mandatory stem of the word, $+f$ is an optional suffix, and "/" is a separator. The stem and affixes are also diacritized. A typical example is the transformation of the word "وبداخلها : *and inside it*" using Buckwalter transliteration as:

$$wbdAxlhA \rightarrow wabidAxilihA/wa+ \quad bi+ \quad dAxili \quad +hA$$

In case that MADA fails to obtain a proper analysis for a given word, the original word is written between double at-marks *"@@"* without decomposition or diacritization. An example is the word "الجاي : *the coming*" which is a dialectal word transformed to *@@AljAy@@*.

By applying this transformation, we obtain a flexible baseline text corpus which can be easily customized. For example, the diacritical characters can be removed to obtain non-diacritized text, or prefixes can be properly concatenated to obtain one prefix per word, or even we can backoff to the original full-word under certain conditions. In our experiments, Two different sets of affixes are examined:

1. **simple affixes:**
   - ***prefixes***: {Al, b, f, k, l, ll, w}.
   - ***suffixes***: {h, hA, hm, hmA, hn, k, km, kmA, kn, nA}.
2. **compound affixes:**
   - ***prefixes***: {Al, b, bAl, f, fAl, fb, fbAl, fk, fl, fll, k, kAl, l, ll, w, wAl, wb, wbAl, wk, wkAl, wl, wll}.
   - ***suffixes***: {h, hA, hm, hmA, hn, k, km, kmA, kn, nA}.

Both sets have the same suffixes but different prefixes. Simple prefixes can not be further decomposed. A word can have multiple simple prefixes. However, compound prefixes result from valid compounding of simple prefixes. A word can not have more than a single compound prefix. We attach a "+" marker to the end of each prefix and the start of each suffix, such as "b+" and "+h", in order to allow for easy and deterministic full-word recovery via attaching affixes to the corresponding stems. In order to get valid full-words after recovery, the MADA stem orthographic normalization is resolved by applying a small set of linguistic rules to recover the original form of the stem. Examples of applied transformations using

[5] In this work, MADA version *2.0* is used.

simple affixes, and compound affixes for the word "فبمتابعتهم : *and by following them*" are:

$$fbmtAbEthm \xrightarrow{\text{simple affixes}} f+ \quad b+ \quad mtAbEt \quad +hm$$

$$fbmtAbEthm \xrightarrow{\text{compound affixes}} fb+ \quad mtAbEt \quad +hm$$

A well known phenomenon in Arabic pronunciation occurs when a prefix that ends with *"Al"* or *"ll"* is followed by a type of consonant called *solar consonants* (Table 3.1), then the final *"l"* of the prefix is not pronounced and the solar consonant that comes after is geminated. Therefore, an *"@"* marker is attached to the end of these prefixes to distinguish them from those followed by other consonants called *lunar consonants*. Hence, for every prefix ending with *"Al"* or *"ll"*, we have indeed *two* versions, one with *"@"* marker and one without, like *"Al+"* and *"Al@+"*. In Arabic phonology, the final *"l"* of the prefix is called either a *solar "l"* or a *lunar "l"* according to the type of the following consonant. An example of a solar *"l"* occurs in the word "الشمس : *the sun*" transliterated as *Al$ms*, whereas an example of a lunar *"l"* occurs in the word "القمر : *the moon*" transliterated as *Alqmr*.

In fact, the use of morphological decomposition and diacritization in Arabic ASR could lead to some negative side effects. For example, the decomposition process could result into some very short and rare morphemes that are difficult to recognize. Also, dividing the words into morphemes breaks down the LM contexts, thus it could be useful to use larger LM contexts to cover the same word m-grams as in the case of full-words. In addition, diacritizing words leads to a division in the LM probability mass among different diacritized forms of the same word which could lead to performance degradation. For these reasons, we need to rationalize the decomposition and the diacritization processes in order to counteract such negative side effects. Therefore, we apply the following constraints while processing the transformed MADA output:

1. Do not decompose words with stems of length ≤ 2 letters.
2. Do not decompose the top N most frequent decomposable words.
3. Do not diacritize the top M most frequent words.

The first constraint is used to avoid very short stems which are usually difficult to recognize specially when surrounded by the high frequent affixes. The second constraint is found very useful, because recognizing the top most frequent words is very important and highly effective on the final word error rate, therefore it is not desired to divide the probability mass of those most frequent words given their contexts over sequences of multiple morphemes. The third constraint is also used to avoid the division of the probability mass of those most frequent words among multiple diacritized forms. Indeed, we focus the utilization of the diacritized word forms in the region of less frequent words. The values of N and M in the last two constraints are optimized over the development corpora.

3.3.2 Unsupervised Morphological Decomposition

As described in the previous section, Arabic has a limited and well known set of prefixes and suffixes. Whereas, in German and Polish the number of morphemes per word is varying so much and not known in advance. This is even more prominent in German language which allows arbitrarily long compounds (review Section 1.9.2). This makes it difficult to rely on linguistic knowledge to perform word decomposition for such languages. On the other hand, as per our knowledge, there are no reliable and readily available linguistic based tools to perform word decomposition for German or Polish. Therefore, the use of the unsupervised decomposition is considered a suitable choice.

For German and Polish, word decomposition is performed using unsupervised techniques implemented by the *Morfessor* tool [Creutz & Lagus 2005]. It is a language independent tool that works in an unsupervised manner, and autonomously discovers segmentations for the words observed in an unannotated text corpus. It is considered as a general model for unsupervised induction of morphology from raw text. It is mainly designed to cope with languages having a concatenative morphology and unrestricted number of morphemes per word. The obtained morphemes often resemble linguistic morphemes [Creutz 2006].

The Morfessor tool follows the principle of the minimum description length (MDL), where the main

Table 3.1. Arabic solar and lunar consonants (**bw**: using Buckwalter transliteration; **ar**: using Arabic script).

solar		**lunar**	
Buckwalter	Arabic	Buckwalter	Arabic
t	ت	>	أ
v	ث	<	إ
d	د	b	ب
*	ذ	j	ج
r	ر	H	ح
z	ز	x	خ
s	س	E	ع
$	ش	g	غ
S	ص	f	ف
D	ض	q	ق
T	ط	k	ك
Z	ظ	m	م
l	ل	h	ه
n	ن	w	و
		y	ي

goal is to discover as compact a description of the input text data as possible. As described in [Creutz & Hirsimäki$^+$ 2007], substrings occurring frequently enough in several different words are proposed as *morphs*. The words are then represented as a concatenation of morphs. An optimal balance is searched between the compactness of the *morph lexicon* and the compactness of the *corpus* representation, where the morph lexicon is a list of all distinct morphs, and the corpus is represented as a sequence of pointers to entries in the morph lexicon. The morfessor model used in our work is the *Morfessor Baseline* model originally introduced in [Creutz 2003; Creutz & Lagus 2002].

It is stated in [Creutz & Lagus 2005] that ignoring word counts and using only the corpus vocabulary to train the Morfessor model, rather than the corpus itself, produces segmentations that are closer to linguistic morphemes. Therefore, we train our Morfessor model using a vocabulary of distinct words that occur more than *five* times in the training corpus. We do not include less frequent words in order to avoid irregularities that are harmful to the training process. In addition, this implementation of Morfessor allows to use the trained model to decompose unseen words. The resulting segmentations are modified so as to remove very short and noisy morphemes, this is found helpful to improve the final WER. Moreover, no decomposition is done for the top N most frequent decomposable full-words, where the value of N is optimized over the development corpora [El-Desoky & Shaik$^+$ 2010; Shaik & El-Desoky$^+$ 2011b].

To allow for a deterministic recovery to full-words in the recognition output, we attach a "+" marker to the end of every non-final morpheme. For example, the German word *"aufmachen : to open"*, and the Polish word "*niejednokrotnie : repeatedly*" are decomposed as:

$$aufmachen \rightarrow auf+ \quad machen$$

$$niejednokrotnie \rightarrow nie+ \quad jedno+ \quad krotnie$$

3.3.3 Syllabification

A Syllable is a phonologically motivated type of sub-word that aims at breaking down the word into a sequence of speech sounds by breaking down its written form. Therefore, it is expected that syllables perform well when used as units for speech recognition especially when combined with pronunciations in graphone-based units (review Section 3.2.3). Since a syllable sub-word corresponds to one unit of sound, it is expected that the mapping from word pronunciation to syllable pronunciation will be more natural than for other types of sub-words. This is because the pronunciation symbols are supposed to align more

naturally to the syllable boundaries.

Syllabification is the process of dividing words into syllables whether spoken or written. Usually, the spoken syllables form the basis of the written syllables. This can be achieved by applying linguistic and phonetic rules. In our German and Polish experiments, we perform syllabification using a rule based tool called *KombiKor v.8.0*[6]. As performed with other types of sub-words, the syllables are modified so as to avoid very short syllables. For example, the syllabification of the German word "*naturwissenschaft : natural science*", and the Polish word "*ładunkowych : load*" are:

$$naturwissenschaft \rightarrow nat+ \quad ur+ \quad wiss+ \quad en+ \quad schaft$$

$$Ładunkowych \rightarrow La+ \quad dun+ \quad ko+ \quad wych$$

Moreover, no syllabification is performed for the top N most frequent words. The value of N is optimized over the development corpora [Shaik & El-Desoky$^+$ 2011b].

3.4 Sub-Word Units Combined with Pronunciations

In this section, we show how the pronunciations of words and sub-words are generated, and how different types of sub-words are joint with pronunciations to form a new set of sub-lexical units that are used to effectively combine pronunciation models with LMs.

3.4.1 Grapheme-to-Phoneme Conversion

Graphemes are defined as the fundamental units in a written language usually indicated by the alphabetic letters. However, *phonemes* are the smallest segmental units of sound employed to form meaningful contrasts between utterances. The process of generating a pronunciation for a word is the process of converting a given sequence of graphemes to the corresponding sequence of phonemes. This is known as *grapheme-to-phoneme (G2P) conversion*. Thus, for words or sub-words whose pronunciations are unknown, a statistical G2P approach is used to generate the missing pronunciations. Our approach is based on the joint-sequence models described in [Bisani & Ney 2008], where we search the most likely pronunciation $\varphi \in \Phi^*$ for a given orthographic form $g \in G^*$, where Φ and G are the sets of phonemes and letters respectively:

$$\varphi(g) = \underset{\acute{\varphi} \in \Phi^*}{\operatorname{argmax}}\, p(\acute{\varphi}, g) \tag{3.9}$$

The joint probability distribution $p(\varphi, g)$ is referred to as a graphonemic joint sequence model. It is assumed that for each word, its orthographic form and its pronunciation are generated by a common sequence of graphonemic units called *graphones* (review Section 3.2.3). Each graphone is a pair q taken from the graphone inventory Q, where:

$$q = (g, \varphi) \in Q \subseteq G^* \times \Phi^* \tag{3.10}$$

Thus, q is a pair of a letter sequence and a phoneme sequence of possibly different lengths. For example, the sequence of graphones looks like:

"mixing" [mɪksɪŋ]	=	m [m]	i [ɪ]	x [ks]	in [ɪŋ]	g —

The joint probability distribution $p(\varphi, g)$ is reduced to a probability distribution over graphone sequences $p(q)$ which is modeled by a standard m-gram model:

$$p(q_1^N) = \prod_{n=1}^{N} p(q_n | q_{n-m+1}, ..., q_{n-1}) \tag{3.11}$$

[6] http://www.3n.com.pl/kombi.php

This model has two parameters: the order m of the m-gram model, and the *graphone size limit* L which is the maximum number of letters or phonemes per graphone. In other words, L is the permissible size of the graphones, where the number of letters and phonemes of any graphone are allowed to vary between zero and an upper limit L. As presented in [Bisani & Ney 2008], such a model can be trained using Maximum Likelihood (ML) training via Expectation Maximization (EM) algorithm. To produce a pronunciation for a given word, we maximize over the set $S(g, \varphi)$ of all joint segmentations of g and φ:

$$p(\varphi, g) \approx \max_{q_1^N \epsilon S(g,\varphi)} p(q_1, ..., q_N) \tag{3.12}$$

3.4.2 Graphones as Recognition Units

During G2P training, the graphone inventory Q is automatically inferred. The size of this inventory depends on the graphone size parameter L. Normally, we optimize the value of L so as to achieve the minimum phoneme error rate (PER) over some test dictionary. This guarantees the best possible pronunciation for a given sequence of letters. The set of inferred graphones constitutes a graphone model that can be integrated with the normal word or sub-word model to form a unified set of recognition units. Yet, the graphemic components of the graphones are not representing any linguistic units; rather, they are just fragments of a limited length [Bisani & Ney 2005]. These are called *fragment-based graphones*.

However, optimizing the set of graphones for the best PER does not guarantee to obtain the set of graphones which achieves the best WER during ASR. Therefore, a novel approach is experimented by which a modified set of graphones is generated by performing the following steps:

1. Estimate a normal set of graphones using the optimum value of L achieving the minimum PER.
2. Modify the letter sequences of graphones such that they represent morphemes or syllables of the underlying words.
3. Realign phoneme sequences to letter sequences to obtain a novel set of graphones.

The new set of graphones are called *morpheme- or syllable-based graphones* according to the chosen type of sub-word. To perform step (3), we need to perform letter-phoneme sequence alignment. This will be discussed in the next section. The following are examples of graphone sequences for the German word "*naturwissenschaft : natural science*":

- Fragment-based graphones (L = 4):
  → [**na**:na] [**tur**:tʊr] [**wiss**:vɪs] [**en**:=ŋ] [**sch**:ʃ] [**aft**:aft]
- Morpheme-based graphones:
  → [**natur**:natʊr] [**wissenschaft**:vɪs=ŋʃaft]
- Syllable-based graphones:
  → [**nat**:nat] [**ur**:ʊr] [**wiss**:vɪs] [**en**:=ŋ] [**schaft**:ʃaft]

3.4.3 Letter-Phoneme Sequence Alignment

As discussed in the previous section, the main step in building morpheme- or syllable-based graphone model is to perform letter-phoneme sequence alignment. For this, we follow an approach based on dynamic programming (DP), and expectation maximization (EM) as described in [Damper & Marchand$^+$ 2004]. The alignment process can be seen as a path-finding problem which can be solved by following a sequence of locally optimal steps. Thus, we create a matrix **A** that is indexed by all letters and all phonemes that occur in the alignment task. The matrix **A** holds the degrees of association between each letter and each phoneme in the task. Then, we use two other matrices **B** and **C** that are both indexed by the letters of the word and the phonemes of its pronunciation to be aligned to. The matrix **B** holds the accumulated associations up to some point in the matrix (the alignment grid). The matrix **C** holds the trace-back pointers indicating the cell from which the DP moves. The matrix **B** is filled out in a left-to-right, top-to-bottom order according to the following recursive maximization equation [Needleman & Wunsch 1970]:

$$B_{i,j} = \max \left\{ \begin{array}{l} B_{i-1,j-1} + A_{l(i),p(j)}, \\ B_{i-1,j}, \\ B_{i,j-1} \end{array} \right\} \begin{array}{l} 1 \leq i \geq L \\ 1 \leq j \geq P \end{array} \tag{3.13}$$

where L is the number of letters of the given word, and P is the number of phonemes in the given pronunciation. The functions $l(\cdot)$ and $p(\cdot)$ provide the letter and phoneme identity at the given index respectively. The entries of $\mathbf{C}$ are filled accordingly based on the chosen maximum. At the end, the matrix $\mathbf{B}$ holds the maximum accumulated association for the complete word in its bottom-right cell, and the best alignment is found by tracing pointers back from the bottom-right cell of the matrix $\mathbf{C}$.

Estimating the association matrix. In order to estimate the association matrix $\mathbf{A}$, we apply an expectation maximization (EM) algorithm using the word-pronunciation pairs of an available pronunciation dictionary as training examples. The EM algorithm works in the following steps:

1. Let $k = 0$; then initialize $\mathbf{A}^k$ such that, for every word-pronunciation pair in the dictionary, the entry a^k_{lp} is incremented if the letter l and phoneme p appear in the same pair.
2. Use $\mathbf{A}^k$ in DP to align all the word-pronunciation pairs of the dictionary, then increment k to $k = k + 1$.
3. Compute $\mathbf{A}^k$ such that, for every word-pronunciation pair in the dictionary, the entry a^k_{lp} is incremented if the letter l and phoneme p appear in the same aligned position.
4. Go to step 2 until convergence indicated by having insignificant difference between $\mathbf{A}^k$ and $\mathbf{A}^{k-1}$.

To illustrate the alignment process, the alignment of the word-pronunciation pair *(phase,feɪz)* is shown in Table 3.2. Table 3.2(a) shows a part of matrix $\mathbf{A}$ that stores the letter-phoneme associations only for the letters and phonemes occurring in the pair *(phase,feɪz)*, where # and $ are treated as delimiters that always align together. Table 3.2(b) shows a superposition of matrix $\mathbf{B}$ and matrix $\mathbf{C}$ that shows the accumulative associations, filled out according to the recursive Equation 3.13, along with the trace-back pointers. The trace-back pointers are indicated by cursors to determine the movement direction during the trace-back. As described above, the process of tracing back starts from the bottom-right cell and moves up to the top-left cell. Reading a diagonal cursor ↘ means to align the current letter and phoneme together and then move to the previous diagonal cell. Reading a down cursor ↓ means to align the current letter to an empty phoneme and move up. Whereas, reading a right cursor → means to align an empty letter to the current phoneme and move left. The reading of the symbol ϵ means to stop the alignment process always at the top-left cell. This is in fact a flexible alignment scheme that enables the alignment of a letter or phoneme to nothing (empty symbol). The final alignment is given by:

"phase"	=	p	h	a	s	e
[feɪz]		—	f	eɪ	z	—

3.5 Experimental Results

In this section, experimental results are presented on sub-word based LMs based on different types of sub-word units that are discussed earlier in Section 3.2. The Experiments are performed using Arabic, German, and Polish testing systems. A detailed description of these systems along with a description of the development and evaluation corpora is given in Appendix A.

3.5.1 Experiments on Arabic

Table 3.3 summarizes the results of the recognition experiments performed on Arabic corpora using sub-word based LMs that use supervised morphemes generated using MADA toolkit (see Section 3.3.1) along with the baseline experiment. We follow the 3 passes recognition setup of the modern standard Arabic (MSA) testing system described in Appendix A, where a bigram LM is used to produce lattices which are then rescored using 4 to 7-gram LMs. The baseline experiment uses a traditional LM based on full-words

Table 3.2. An example of the alignment process for the word-pronunciation pair *(phase,feɪz)*.

(a) A part of matrix **A** showing letter-phoneme associations for the letters and phonemes occurring in the pair (phase,feɪz), where # and $ are delimiters.

	$	f	eɪ	z	$
#	0	0	0	0	0
p	0	9	0	0	0
h	0	2580	27	35	0
a	0	42	23098	937	0
s	0	79	3	45788	0
e	0	947	1732	2641	0
#	0	0	0	0	0

(b) A superposition of matrices **B** and **C** showing the accumulative associations and the trace-back pointers indicated by the cursor movement during the Dynamic Programming.

	$	f	eɪ	z	$
#	**0,**ϵ	0,→	0,→	0,→	0,→
p	**0,**↓	9,↘	9,→	9,→	9,→
h	0,↓	**2580,**↘	2580,→	2580,→	2580,→
a	0,↓	2580,↓	**25678,**↘	25678,→	25678,→
s	0,↓	2580,↓	25678,↓	**71446,**↘	71446,→
e	0,↓	2580,↓	25678,↓	**71446,**↓	71446,↘
#	0,↓	2580,↓	25678,↓	71446,↓	**71446,**↘

without any morphemes. For this initial set of experiments, the total vocabulary size is fixed to 70k. In addition, the use of both simple and compound affixes is examined. The number of the most frequent decomposable full-words retained without decomposition is optimized over the development corpus by performing a series of recognition experiments using a gradually increased number of full-words (see decomposition constraint (2), Section 3.3.1)). The objective of these experiments is to discover the best number of full-words, the most suitable type of affixes, and the optimum order of the sub-word based LM.

Table 3.3. Recognition experiments on Arabic corpora using morpheme-based LMs with 70k vocabularies.

				ar-dev07		**ar-eval07**	
experiment	**LM**	**full-words**	**morphemes**	**OOV** [%]	**WER** [%]	**OOV** [%]	**WER** [%]
full-words	4-gram	70k	-	**3.7**	**16.2**	**4.8**	**18.5**
simple affixes	4-gram	-	70k	1.1	17.0	1.7	18.9
compound affixes	4-gram	-	70k	1.1	17.0	1.7	18.7
		5k	65k	1.2	15.0	1.8	-
		10k	60k	1.3	14.6	1.9	-
		20k	50k	**1.4**	**14.5**	**2.0**	**16.5**
		30k	40k	1.7	14.7	2.5	-
		40k	30k	2.0	14.9	2.8	-
	6-gram	20k	50k	1.4	14.5	2.0	16.5
	7-gram	20k	50k	1.4	14.5	2.0	16.5

It can be seen that the use of compound affixes is a little more beneficial than the use of simple affixes. This is because the negative side effect of reducing the LM span as a result of using smaller units

(morphemes) is less in case of compound affixes compared to the case of simple affixes. Also, during LM training, the existence of multiple prefixes in LM training data rises the number of sequences of prefixes giving them high probabilities which leads to observing high insertion rates in the recognition output. Therefore, we continue the experiments using compound affixes.

Optimizing the number of full-words as a part of the sub-word based vocabulary has shown that the best number of full-words is 20k given the experimental conditions. Thus, the minimum observed WERs are achieved using *20k full-words + 50k morphemes*. The WERs are reduced by [ar-dev07: 1.7% absolute (10.5% relative); ar-eval07: 2.0% absolute (10.8% relative)] compared to the full-word baseline. Moreover, significant reductions can be observed in the sub-word OOV rate compared to the full-word OOV rate. In fact, the OOV rates recorded in Table 3.3 are the *effective OOV rates*[7], where a word is considered an OOV if and only if it is not found in the vocabulary and it is not possible to compose it using in-vocabulary sub-words.

On the other hand, it is noted that using higher order LMs does not have a noticeable effect on the WER. Although the use of sub-word based models counteracts the negative effect of data sparsity, still much more training data is needed to feed the higher order models. In other words, increasing the LM order brings the data sparsity problem again into picture leading to unimproved probability estimates. For this reason, 4-gram LMs are used for the rest of experiments.

In order to examine the full potential of the sub-word based LMs compared to the full-word based LMs, it is important to optimize the total vocabulary size of the full-words. This aims at finding the best operating point to which the sub-word based LM should be particularly compared. Table 3.4 introduces a set of experiments in which both full-word and sub-word vocabulary sizes are increased gradually up to one million. In addition, an extended hybrid version of LM is experimented which includes mixed types of units; namely, full-words, non-diacritized morphemes, as well as diacritized morphemes. In this extended hybrid approach, full-words are the highest frequent units in the vocabulary; and non-diacritized morphemes are less frequent units; whereas diacritized morphemes represent the least frequent part of the vocabulary.

In the recognition lexicon, multiple pronunciation variants are provided for every non-diacritized unit, whereas a single pronunciation is provided for every diacritized unit; this is the pronunciation that corresponds exactly to the diacritized form obtained by the MADA toolkit (review Sections 3.2.4, and sec:chp3:word-decomp-supervised). Since they are very frequent, all the compound affixes are kept non-diacritized with all their possible pronunciations included in the lexicon. The probabilities assigned by the LM to the diacritized units can be recognized as a combination of pronunciation probability and LM probability in one join distribution.

Table 3.4. Recognition experiments on Arabic corpora using full-words, morphemes, and diacritized morphemes for LMs with very large vocabularies.

				ar-dev07			ar-eval07		
voc. size	**full-words**	**morph-emes**	**diacri-tized morph-emes**	**OOV** [%]	**WER** (ins/del) [%]	**CER** (ins/del) [%]	**OOV** [%]	**WER** (ins/del) [%]	**CER** (ins/del) [%]
70k	70k	-	-	3.7	16.2 (3.0/1.7)	7.3 (2.6/3.2)	4.8	18.5 (2.6/1.7)	9.5 (2.2/4.9)
140k	140k	-	-	2.0	15.2 (2.8/1.6)	6.9 (2.5/3.0)	2.7	17.0 (2.5/1.4)	8.9 (2.2/4.5)
256k	256k	-	-	1.4	14.9 (2.7/1.6)	6.8 (2.5/2.9)	1.9	16.7 (2.3/1.4)	8.7 (2.2/4.4)
500k	500k	-	-	0.6	14.7 (2.7/1.5)	6.8 (2.5/2.9)	0.8	16.3 (2.2/1.4)	8.7 (2.1/4.4)
750k	**750k**	**-**	**-**	**0.5**	**14.6 (2.7/1.5)**	**6.6 (2.5/2.7)**	**0.7**	**16.3 (2.3/1.4)**	**8.7 (2.2/4.3)**
1M	1M	-	-	0.4	14.6 (2.7/1.4)	6.6 (2.6/2.7)	0.6	16.3 (2.3/1.4)	8.7 (2.2/4.3)
1.3M	1.3M	-	-	0.3			0.5		
70k	20k	50k	-	1.4	14.5 (2.5/1.5)	6.7 (2.5/2.8)	2.0	16.5 (2.2/1.6)	8.8 (2.1/4.5)
256k	**20k**	**236k**	**-**	**0.5**	**14.1 (2.4/1.6)**	**6.5 (2.5/2.7)**	**0.7**	**16.1 (2.1/1.5)**	**8.6 (2.1/4.4)**
500k	20k	480k	-	0.3	14.3 (2.4/1.6)	6.6 (2.5/2.8)	0.5	16.1 (2.0/1.5)	8.7 (2.0/4.4)
750k	20k	730k	-	0.3	14.3 (2.4/1.6)	6.6 (2.5/2.8)	0.4	16.1 (2.0/1.6)	8.7 (2.0/4.4)
850k	20k	830k	-	0.2			0.36		
400k	20k	-	380k	0.5	15.3 (2.8/1.6)	7.00 (2.5/3.0)	0.7	17.0 (2.5/1.4)	8.9 (2.2/4.5)
350k	**20k**	**120k**	**210k**	**0.5**	**14.5 (2.5/1.6)**	**6.7 (2.5/2.7)**	**0.7**	**16.4 (2.1/1.5)**	**8.8 (2.1/4.4)**

[7] This is how the OOV rate is computed throughout the experiments presented in this thesis.

Table 3.4 shows that the best operating point for the full-word LM occurs at a vocabulary size of 750k full-words. Whereas, the best operating point for the morpheme-based LM occurs at a vocabulary size of 256k *(20k full-words + 236k morphemes)*. Even if the OOV rates are similar for both full-word and sub-word based vocabularies, improvements in WER can still be observed. Thus, using a morpheme-based LM, WER reductions of [ar-dev07: 0.5% absolute (3.4% relative); ar-eval07: 0.2% absolute (1.2% relative)] are achieved compared to the best full-word LM.

However, an extended hybrid LM containing *20k full-words + 120k morphemes + 210k diacritized morphemes* could only improve the WER over the 256k full-word LM for both ar-dev07 and ar-eval07 corpora [ar-dev07: 0.4% absolute (2.7% relative); ar-eval07: 0.3% absolute (1.8% relative)].

Table 3.5 shows the word- and character-level perplexities (PPLs) for the most important experiments listed in Table 3.4. The perplexity is reported for both in-vocabulary text; where the probabilities of the *unk* symbol are excluded from the perplexity computation; and for the whole text, where the probabilities of the *unk* symbol are included. The character-level perplexity and the relationship between different perplexities are previously discussed in Section 3.1.

Table 3.5. word- and character-level perplexities for full-word and sub-word based LMs on Arabic corpora (**inv**: perplexity for in-vocabulary text excluding the unk symbol; **all**: perplexity for the whole text including the unk symbol).

corpus	voc. size	full-words	morph-emes	diacritized morphemes	word-level PPL		char-level PPL	
					inv (#units)	all (#units)	inv (#chars)	all (#chars)
ar-dev07	750k	750k	-	-	502.6 (18920)	500.8 (19002)	3.084 (104489)	3.075 (105142)
	256k	20k	236k	-	392.8 (19600)	391.7 (19724)	**3.051 (104967)**	**3.041 (105869)**
	350k	20k	120k	210k	397.7 (19655)	395.0 (19776)	3.061 (105173)	3.050 (106032)
ar-eval07	750k	750k	-	-	679.0 (29249)	673.9 (29430)	3.297 (159882)	3.280 (161384)
	256k	20k	236k	-	513.1 (30538)	509.7 (30752)	**3.266 (161028)**	**3.248 (162719)**
	350k	20k	120k	210k	519.9 (30543)	514.6 (30790)	3.273 (161073)	3.253 (162976)

3.5.2 Experiments on German

Table 3.6 summarizes the results of the recognition experiments performed on German corpora using sub-word based LMs that use unsupervised morphemes generated using the Morfessor tool (see Section 3.3.2) along with the baseline experiment. We follow the 2 passes recognition setup of the German testing system described in Appendix A, where a 4 or 6-gram LM is used to construct the search space without a subsequent lattice or N-best rescoring. In the baseline experiment, a traditional LM is used based on full-words without any morphemes. For this initial set of experiments, the total vocabulary size is fixed to 100k. The number of the most frequent decomposable full-words retained without decomposition is optimized over the development corpus. Therefore, the number of full-words is increased gradually starting from zero. The objective of these initial experiments is to discover the best number of full-words, and the optimum order of the sub-word based LM.

Table 3.6. Recognition experiments on German corpora using morpheme-based LMs with 100k vocabularies.

LM	full-words	morphemes	gr-dev09		gr-eval09	
			OOV [%]	WER [%]	OOV [%]	WER [%]
4-gram	100k	-	**5.0**	**33.9**	**4.8**	**29.7**
	-	100k	1.0	32.2	-	-
	2k	98k	1.2	31.8	-	-
	5k	95k	**1.5**	**31.7**	**1.4**	**28.5**
	7k	93k	1.6	31.7	-	-
	10k	90k	1.8	31.8	-	-
	20k	80k	1.9	31.8	-	-
	30k	70k	2.1	31.9	-	-
6-gram	5k	95k	1.5	31.6	1.4	28.5

Table 3.6 shows that the best number of full-words to retain in the sub-word based vocabulary is 5k.

The minimum observed WERs are achieved using *5k full-words + 95k morphemes*. Thereby, the WERs are reduced by [gr-dev09: 2.2% absolute (6.5% relative); gr-eval09: 1.2% absolute (4.0% relative)] compared to the full-word baseline. In addition, significant reductions can be observed in the sub-word OOV rates compared to the full-word OOV rates. On the other hand, it is noted that using a 6-gram rather than 4-gram LM does not help as almost the same WERs are observed for both corpora.

Table 3.7 compares the previously known fragment-based graphones to the newly proposed morpheme-based graphones. The generation of graphones is based on G2P models as discussed previously in Section 3.4.2. To train these G2P models, we use a base-lexicon containing pronunciations for about 118k words divided into a training set of 112k words, and a test set of 6k words. Multiple G2P models are trained using different model parameters. For each model, the phoneme error rate (PER) is measured on the test set. The morpheme-based graphones are obtained by modifying a set of graphones based on a G2P model trained with a graphone size parameter $L = 4$ since it gives the least PER (review Section 3.4.2). The size of the baseline full-word vocabulary is set to 100k words on top of which different types of graphones are added. It is worth noting that the used number of fragment-based graphones represents all the graphones found in the training data other than the original 100k full-words. This interprets the very low OOV rates observed in these cases. Nevertheless, we could not set the graphone size parameter L to a value more than 4 as this increases the graphone inventory leading to impractically very large resource requirements during the G2P model training.

Table 3.7. Recognition experiments on German corpora using 100k full-words as a baseline vocabulary and adding different fragment-based and morpheme-based graphones.

			gr-dev09		gr-eval09	
experiment	**voc. size**	**graphones**	**OOV** [%]	**WER** [%]	**OOV** [%]	**WER** [%]
full-words	100k	-	**5.0**	**33.9**	**4.8**	**29.7**
fragment-based graphones						
graphone size (L) = 2	102k	2k	0.1	34.2	-	-
3	110k	10k	0.1	32.8	-	-
4	124k	24k	**0.1**	**32.4**	**0.1**	**29.4**
morpheme-based graphones	177k	77k	2.8	32.5	2.6	29.5
	300k	200k	**1.0**	**32.1**	**1.1**	**29.3**

It can be seen from Table 3.7 that the morpheme-based graphones outperform the fragment-based graphones. Therefore, fragment-based graphones are going to be utilized in further experiments.

Similar to the experiments on Arabic, to find the best operating point for both the full-word and sub-word based LMs, Table 3.8 introduces a set of experiments in which both full-word and sub-word based vocabulary sizes are increased gradually up to one million. In addition, an extended hybrid LM is experimented which includes full-words, morphemes, as well as morpheme-based graphones. Therein, full-words are the highest frequent units in the vocabulary; and morphemes are less frequent units; whereas morpheme-based graphones are the least frequent part of the vocabulary. In the recognition lexicon, multiple pronunciation variants are provided for every unit except for graphones, where a single pronunciation is provided that corresponds to the phonemic part of every graphone. The probability distribution over graphones can be seen as a combination of pronunciation probability and LM probability in one join distribution.

Table 3.8 shows that the best operating point for the full-word LM occurs at a vocabulary size of 750k full-words. Whereas, the best operating point for the morpheme-based LM occurs at a vocabulary size of 500k *(5k full-words + 495k morphemes)*. Using this morpheme-based LM, WER reductions of [gr-dev09: 0.3% absolute (1.0% relative); gr-eval09: 0.2% absolute (0.7% relative)] are achieved compared to the best full-word LM. At the same time, significant reductions in OOV rates are observed for the best morpheme-based LM compared to the best full-word LM.

Using an extended hybrid LM containing *5k full-words + 295k morphemes + 200k morpheme-based graphones*, WER reductions of [gr-dev09: 0.3% absolute (1.0% relative); gr-eval09: 0.4% absolute (1.5% relative)] are achieved compared to the best full-word LM. Table 3.9 shows the word- and character-level perplexities (PPLs) for the most important experiments listed in Table 3.8.

Table 3.8. Recognition experiments on German corpora using full-words, morphemes, and morphemic graphones for LMs with very large vocabularies.

				gr-dev09			gr-eval09		
voc. size	**full-words**	**morph-emes**	**morph-emic graph-ones**	**OOV** [%]	**WER** (ins/del) [%]	**CER** (ins/del) [%]	**OOV** [%]	**WER** (ins/del) [%]	**CER** (ins/del) [%]
100k	100k	-	-	5.0	33.9 (5.3/6.8)	15.1 (2.7/6.9)	4.8	29.7 (3.4/7.1)	13.8 (2.5/6.9)
200k	200k	-	-	3.8	32.7 (4.7/7.0)	14.7 (2.7/6.8)	3.5	28.8 (3.0/7.3)	13.5 (2.4/6.8)
300k	300k	-	-	3.3	32.2 (4.4/7.0)	14.6 (3.0/6.9)	3.0	28.4 (2.9/7.3)	13.2 (2.3/6.7)
500k	500k	-	-	2.7	32.0 (4.0/7.3)	14.7 (2.9/7.1)	2.4	28.6 (2.7/7.8)	13.4 (2.2/7.0)
750k	**750k**	**-**	**-**	**2.3**	**31.3 (4.6/6.0)**	**14.3 (3.5/5.9)**	**2.1**	**27.4 (3.2/6.5)**	**12.8 (2.7/5.9)**
1M	1M	-	-	2.2	31.4 (4.6/6.0)	14.3 (3.5/6.0)	1.9	27.5 (3.1/6.5)	12.7 (2.7/5.8)
2.5M	2.5M	-	-	1.7			1.4		
100k	5k	95k	-	1.5	31.7 (3.8/7.3)	14.6 (2.6/6.7)	1.4	28.5 (2.8/7.5)	13.3 (2.3/6.8)
500k	**5k**	**495k**	**-**	**0.9**	**31.0 (4.4/5.8)**	**14.2 (3.5/5.8)**	**0.7**	**27.2 (3.1/6.1)**	**12.5 (2.7/5.6)**
750k	5k	745k	-	0.8	31.0 (4.3/5.9)	14.2 (3.5/5.9)	0.7	27.2 (3.1/6.2)	12.5 (2.7/5.6)
1M	5k	995k	-	0.8	31.2 (4.3/6.1)	14.3 (3.5/6.0)	0.7	27.2 (3.1/6.1)	12.5 (2.7/5.6)
2.1M	5k	2095k	-	0.7			0.5		
300k	100k	-	200k	1.0	32.1 (4.4/7.0)	14.5 (3.0/6.8)	1.1	29.3 (3.2/7.1)	13.5 (2.4/6.8)
500k	**5k**	**295k**	**200k**	**0.9**	**31.0 (4.7/5.8)**	**14.1 (3.5/5.8)**	**0.8**	**27.0 (3.3/5.9)**	**12.3 (2.7/5.6)**

Table 3.9. word- and character-level perplexities for full-word and sub-word based LMs on German corpora (**inv**: perplexity for in-vocabulary text excluding the unk symbol; **all**: perplexity for the whole text including the unk symbol).

	voc.	**full-**		**morphemic**	**word-level PPL**		**char-level PPL**	
corpus	**size**	**words**	**morphemes**	**graphones**	inv (#units)	all (#units)	inv (#chars)	all (#chars)
gr-dev09	750k	750k	-	-	509.0 (69548)	490.4 (71133)	2.818 (418447)	2.725 (439560)
	500k	5k	495k	-	403.9 (72391)	393.2 (73906)	**2.799 (422086)**	**2.713 (442333)**
	500k	5k	295k	200k	398.5 (74650)	397.1 (76633)	2.889 (421293)	2.802 (445060)
gr-eval09	750k	750k	-	-	520.0 (35591)	503.3 (36319)	2.793 (216684)	2.713 (226395)
	500k	5k	495k	-	403.0 (37151)	393.8 (37845)	**2.772 (218582)**	**2.697 (227921)**
	500k	5k	295k	200k	382.7 (38433)	382.5 (39288)	2.838 (219126)	2.769 (229364)

3.5.3 Experiments on Polish

Table 3.10 summarizes the results of the recognition experiments performed on Polish corpora using morpheme- and syllable-based LMs along with the baseline experiment. We follow the 3 passes recognition setup of the Polish testing system described in Appendix A, where a 5-gram LM is used to construct the search space without lattice or N-best rescoring. In the baseline experiment, a traditional LM is used based on full-words without any morphemes or syllables. The total vocabulary size is fixed to 300k. The objective of this initial set of experiments is to find out the best performing type of sub-word together with the best number of full-words to retain in the sub-word based vocabulary for each type of sub-word.

Table 3.10. Recognition experiments on Polish corpora using morpheme- and syllable-based LMs with 300k vocabularies.

			pl-dev10		**pl-eval10**	
experiment	**full-words**	**sub-words**	**OOV** [%]	**WER** [%]	**OOV** [%]	**WER** [%]
full-words	300k	-	**1.7**	**22.7**	**1.9**	**26.8**
morphemes	30k	270k	1.5	23.0	-	-
	50k	250k	1.5	22.8	-	-
	70k	230k	**1.6**	**22.7**	**1.8**	**26.2**
	90k	210k	1.7	22.8	-	-
	100k	200k	1.7	22.8	-	-
syllables	50k	250k	0.5	23.0	-	-
	70k	230k	0.5	22.6	-	-
	90k	210k	0.6	22.7	-	-
	110k	190k	0.6	22.5	-	-
	130k	170k	**0.6**	**22.3**	**0.5**	**26.1**
	150k	150k	0.7	22.4	-	-

Table 3.10 shows that WER improvements can be achieved by using either morphemes- or syllable-based LMs. Nevertheless, the use of syllable-based LMs outperforms the use of morpheme-based LMs. The best number of full-words to retain in the syllable-based vocabulary is 130k. The minimum observed WERs are achieved using *130k full-words + 170k syllables*. Thereby, the WERs are reduced by [pl-dev10: 0.4% absolute (1.8% relative); pl-eval10: 0.7% absolute (2.6% relative)] compared to the full-word baseline. In addition, significant reductions can be observed in the OOV rates. The reductions in OOV rates using syllables are generally more than the reductions in OOV rates using morphemes. This is because syllables are essentially smaller and more frequent units. The average morpheme length is around 6 letters, whereas the average syllable length is around 4 letters.

Similar to previous experiments on Arabic and German, to find the best operating point for both the full-word and sub-word based LMs, Table 3.11 introduces a set of experiments in which both full-word and syllable-based vocabulary sizes are increased gradually up to one million. In addition, an extended hybrid LM is experimented that includes full-words, syllables, as well as syllable-based graphones.

Table 3.11 shows that the best operating point for the full-word LM occurs at a vocabulary size of 750k full-words. Whereas, the best operating point for the syllable-based LM occurs at a vocabulary size of 300k *(130k full-words + 170k syllables)*. Using the syllable-based LM, WER reductions of [pl-dev10: 0.4% absolute (1.8% relative); pl-eval10: 0.7% absolute (2.6% relative)] are achieved compared to the 300k full-word system. However, no improvements could be achieved compared to the best full-word system.

Using an extended hybrid LM containing *130k full-words + 70k syllables + 300k syllable-based graphones*, limited improvements in WERs are achieved [pl-dev10: 0.1% absolute (0.5% relative); pl-eval10: 0.1% absolute (0.4% relative)] compared to the best full-word system. However, comparing to the 300k full-word system, WER reductions of [pl-dev10: 0.8% absolute (3.5% relative); pl-eval10: 1.3% absolute (4.9% relative)] are obtained. Table 3.12 shows the word- and character-level perplexities (PPLs) for the most important experiments listed in Table 3.11.

Table 3.11. Recognition experiments on Polish corpora using full-words, syllables, and syllabic graphones for LMs with very large vocabularies.

				pl-dev10			pl-eval10		
voc. size	**full-words**	**syll-ables**	**syll-abic graph-ones**	**OOV** [%]	**WER** (ins/del) [%]	**CER** (ins/del) [%]	**OOV** [%]	**WER** (ins/del) [%]	**CER** (ins/del) [%]
100k	100k	-	-	4.5			5.2		
300k	300k	-	-	1.7	22.7 (1.9/6.7)	10.3 (3.0/4.4)	1.9	26.8 (1.8/8.3)	20.0 (9.2/6.0)
500k	500k	-	-	1.1	22.1 (1.6/7.4)	10.4 (1.7/5.6)	1.2	25.6 (2.1/7.3)	19.0 (9.5/5.3)
750k	**750k**	**-**	**-**	**0.8**	**22.0 (1.4/7.4)**	**10.2 (1.7/5.5)**	**0.9**	**25.6 (2.1/7.3)**	**18.6 (9.4/5.0)**
1M	1M	-	-	0.7	22.0 (1.4/7.4)	10.2 (1.6/5.5)	0.8	25.6 (2.0/7.4)	18.6 (9.5/5.0)
2M	2M	-	-	0.5			0.6		
3M	3M	-	-	0.5			0.6		
300k	**130k**	**170k**	**-**	**0.6**	**22.3 (1.5/7.1)**	**10.0 (2.7/4.5)**	**0.5**	**26.1 (2.0/7.6)**	**19.0 (9.5/5.1)**
500k	130k	370k	-	0.5	22.4 (1.4/7.4)	10.2 (1.6/5.4)	0.3	26.2 (2.0/7.8)	19.2 (9.5/5.3)
750k	130k	620k	-	0.4	22.4 (1.4/7.4)	10.2 (1.6/5.5)	0.2	26.2 (2.0/7.9)	19.2 (9.5/5.4)
1M	130k	870k	-	0.4			0.2		
300k	130k	70k	100k	0.7	22.2 (1.6/6.8)	10.2 (2.8/4.5)	0.6	25.8 (2.1/7.0)	18.9 (9.6/4.9)
500k	**130k**	**70k**	**300k**	**0.4**	**21.9 (1.6/6.9)**	**10.1 (1.6/5.4)**	**0.3**	**25.5 (2.2/7.1)**	**19.0 (9.4/5.3)**

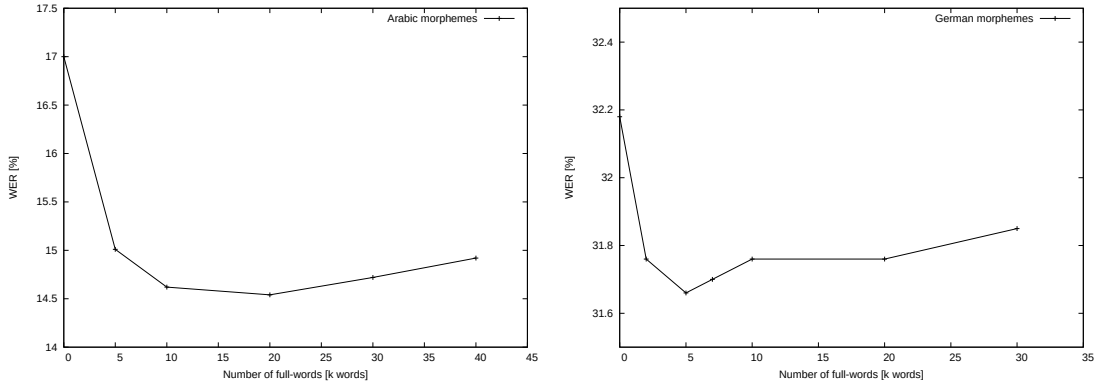
Table 3.12. word- and character-level perplexities for full-word and sub-word based LMs on Polish corpora (**inv**: perplexity for in-vocabulary text excluding the unk symbol; **all**: perplexity for the whole text including the unk symbol).

	voc.	**full-**		**syllabic**	**word-level PPL**		**char-level PPL**	
corpus	**size**	**words**	**syllables**	**graphones**	inv (#units)	all (#units)	inv (#chars)	all (#chars)
pl-dev10	750k	750k	-	-	716.0 (30732)	697.6 (31029)	2.881 (190932)	2.849 (194063)
	300k	130k	170k	-	591.3 (32797)	580.8 (33069)	2.949 (193552)	2.922 (196255)
	500k	130k	70k	300k	693.5 (31150)	683.2 (31407)	2.892 (191904)	2.870 (194441)
pl-eval10	750k	750k	-	-	726.4 (31507)	710.4 (31771)	2.984 (189865)	2.952 (192722)
	300k	130k	170k	-	632.1 (32189)	617.3 (32542)	2.981 (190083)	2.947 (193493)
	500k	130k	70k	300k	702.3 (31899)	691.1 (32151)	2.994 (190653)	2.970 (193102)

3.5.4 Overview of Experimental Results

It is noted from the experimental results discussed in the previous sections that the most influential factor to gain improvements in WER via sub-word based LMs is to optimize the number of different units used in the recognition system. Since full-words fall into the highest frequency region of the recognition vocabulary, optimizing the number of full-words is considered the most important task.

Figure 3.1 shows the change in WER on development corpora during the optimization of the number of full-words as a part of the sub-word based vocabulary. It is noted that, by increasing the number of full-words, the WER decreases rapidly until reaching an optimum value and then starts to increase again. This U-shaped curve can be considered as a general behavior as it is true for all corpora and all types of sub-words across different languages. The average reduction in the WER after performing this optimization is around [1.0% absolute (6.0% relative)] compared to the use of *zero* full-words.



(a) Arabic morphemes (ar-dev07)

(b) German morphemes (gr-dev09)

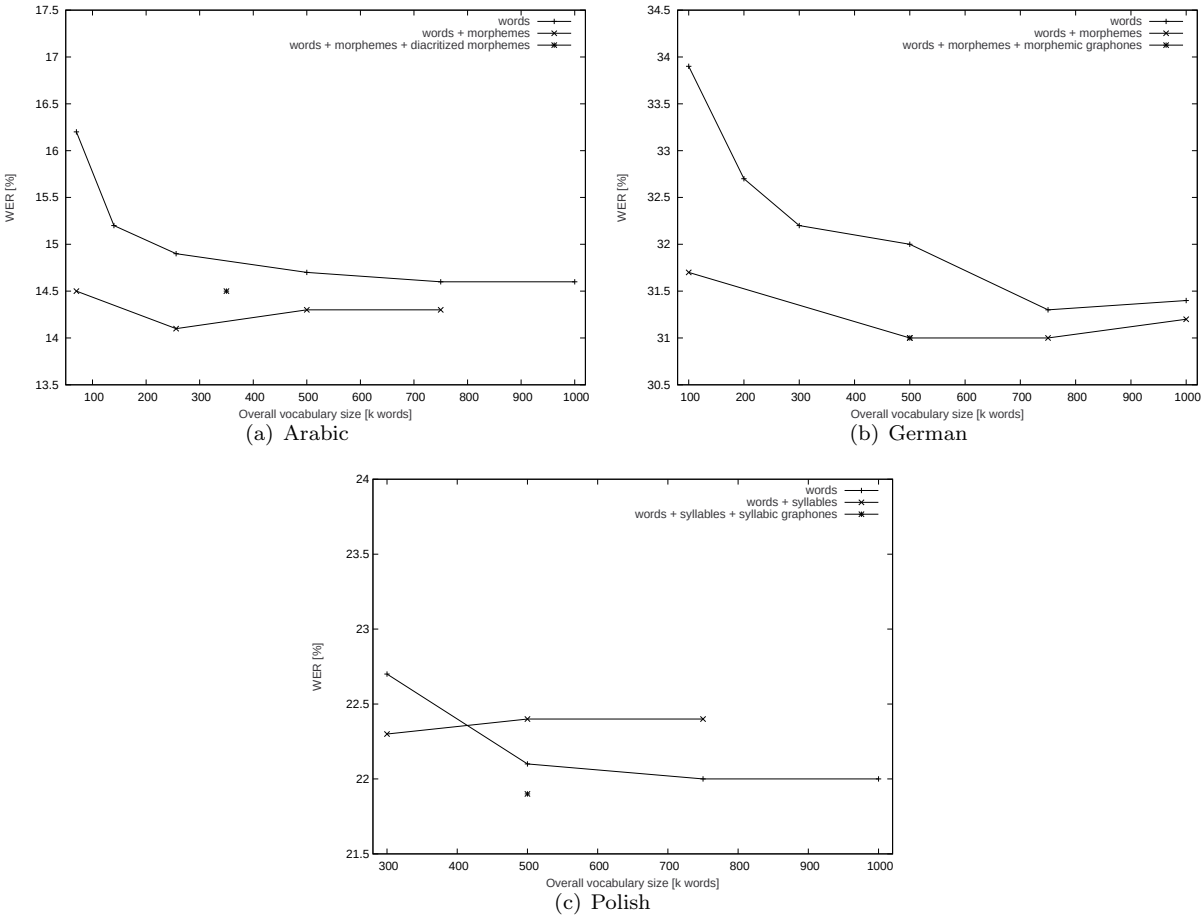
(c) Polish morphemes (pl-dev10)

(d) Polish syllables (pl-dev10)

Figure 3.1. Optimization of the number of full-words retained in the sub-word based vocabularies.

Another important factor that heavily influences the WERs of both full-word and sub-word based systems is the overall vocabulary size. Optimizing the overall vocabulary sizes helps to discover the exact potential of the sub-word based systems compared to the full-word based counterparts. Figure 3.2 shows this optimization performed on development corpora. It can be seen that, in most cases, the whole curve of the sub-word based systems is located under the curve of the full-word based systems. An exception can be observed in Polish experiments.

Figure 3.3 shows a summary of the best achieved WERs, and the corresponding OOV rates on different corpora for the best sub-word based systems compared to the best full-word based systems. Using a bootstrap method of significance analysis described in [Bisani & Ney 2004], we can see that these WER reductions are statistically significant. The probability of improvement (POI_{boot}) ranges between 94% and 97%, which indicates that we can be quite confident that this reflects a real superiority of sub-word



(a) Arabic (b) German

(c) Polish

Figure 3.2. Optimization of the overall vocabulary sizes for full-word and sub-word based experiments.

based systems compared to full-word based systems.

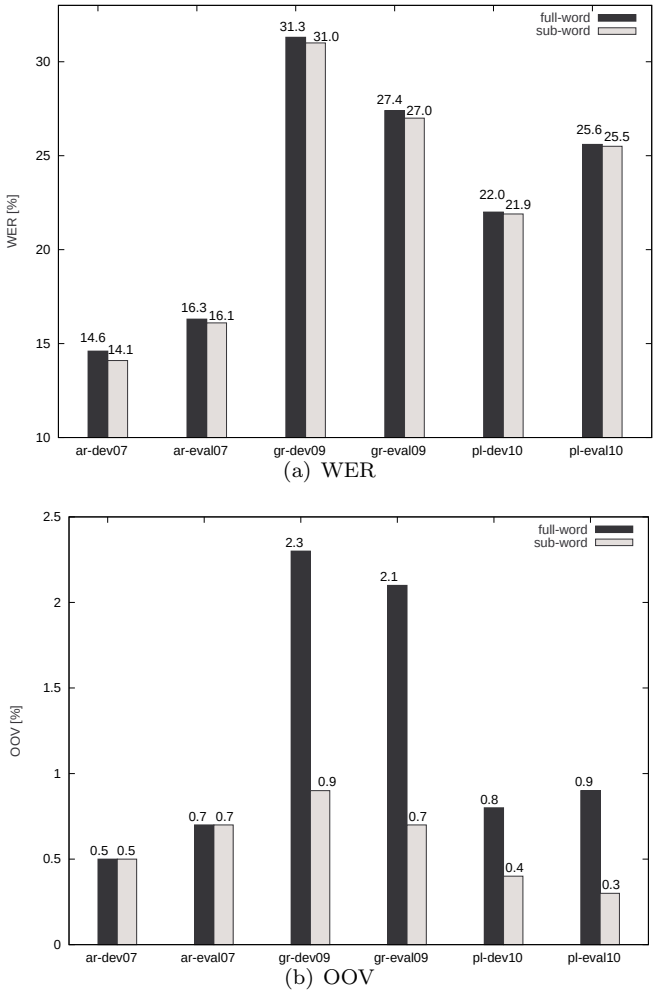


Figure 3.3. The best sub-word based experiments compared to the best full-word based experiments on Arabic, German, and Polish corpora.

The WER of a given speech recognition system is measured as the sum of *insertion, deletion,* and *substitution* rates after aligning the hypothesized sentences with the reference sentences. Any improvement in the WER comes from reductions in any of these three factors. Normally, the inserted words in a given hypothesized sentence are not related to any words from the reference sentence, i.e. they are aligned to empty reference words. However, the deleted and the substituted words are aligned to usual words from the reference sentence.

In order to analyze the WER improvement of a particular sub-word based system compared to a given full-word baseline, we assume that, for the sub-word based system, every word in the reference is marked as an out-of-vocabulary (OOV) or in-vocabulary (INV) word with respect to the full-word baseline vocabulary. The sum of deletion and substitution rates is then redistributed over two different types of errors, namely: (1) error rate by deletion or substitution of out-of-vocabulary words, (2) error rate by deletion or substitution of in-vocabulary words.

Table 3.13 records the amount of reduction in OOV rate and WER when going from the best full-word based system to the best sub-word based system for Arabic, German, and Polish experiments. The overall amount of reduction in WER is divided into three types of error rate reductions:

1. Reduction in insertion rate
2. Reduction in deletion/substitution rate of OOV words

3. Reduction in deletion/substitution rate of INV words

Here, it is worth noting that any negative value of reduction means an increase in the underlying rate. Table 3.14 gives some examples of words for which recognition is improved by reducing the number of deletions or substitutions using the best sub-word based systems.

Table 3.13. Analysis of improvements in the best sub-word based system compared to the best full-word based system for Arabic, German, and Polish corpora. Amount of reduction in WER is divided into (**ins**: reduction in insertion rate; **OOV del/sub**: reduction in deletion/substitution rate of OOV words; **INV del/sub**: reduction in deletion/substitution rate of INV words). Note: a negative reduction means an increase.

		absolute reduction in				
language	**corpus**	**OOV** [%]	**WER** [%]	**ins** [%]	**OOV del/sub** [%]	**INV del/sub** [%]
Arabic	**ar-dev07**	0.0	0.5	0.3	0.0	0.2
	ar-eval07	0.0	0.2	0.2	0.0	0.0
German	**gr-dev09**	1.4	0.3	0.2	0.1	0.0
	gr-eval09	1.4	0.4	-0.1	0.3	0.2
Polish	**pl-dev10**	0.4	0.1	-0.2	0.2	0.1
	pl-eval10	0.6	0.1	-0.1	0.1	0.1

Table 3.14. Examples of words for which recognition is improved using the best sub-word based systems.

Arabic	**German**	**Polish**
Aljmyl	*eigentlich*	*dwudziestotrzylatkiem*
Almst$Ar	*heimtrainer*	*prezydentura*
ystlhmh	*dreiecksungleichung*	*wypracowana*
yktnfh	*rückführungsrichtlinie*	*terminowe*
yHddhA	*justizkomitee*	*zapunktować*

3.6 External Evaluations

The approaches discussed in this chapter have been employed in the RWTH evaluation systems used in many evaluation campaigns during the years from 2010 up to 2013. In those evaluations, RWTH has achieved advanced positions among the participants, namely the first or the second position. This section presents a summary of the recognition results achieved in those evaluations. Initially, Table 3.15 presents a list of the participant sites.

3.6.1 Quaero German ASR Evaluation 2010

Table 3.16 shows the results of the Quaero[8] evaluation on German ASR held in 2010. Two types of evaluation data have been used, broadcast news (BN) and broadcast conversations, in a rough 50-50% ratio. The RWTH system uses morpheme-based LMs. The first position has been achieved out of three participants. A detailed description of the system is given in [Sundermeyer & Nußbaum-Thom$^+$ 2011].

3.6.2 Quaero German ASR Evaluation 2011

Table 3.17 shows the results of the Quaero evaluation on German ASR held in 2011. Similar to the year 2010, broadcast news (BN) and broadcast conversations data have been used in a rough 50-50%

[8] http://www.quaero.org

Table 3.15. List of participants in different evaluation campaigns.

RWTH	RWTH Rheinisch-Westfälische Technische Hochschule Aachen, Germany
KIT	Karlsruhe Institute of Technology, Germany
CITLAB	Computational Intelligence Technology Laboratory, University of Rostock, Germany
LIMSI	Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur, France
VR	Vocabia Resarch, France
A2IA	Artificial Intelligence and Image Analysis, France
LITIS	Laboratory for Computer Science, Information Technology and Systems, University of Rouen, France
UOB-TELE	Telecom ParisTech research lab, France
UPV	Universitat Politècnica de València, Spain
FBK	Fondazione Bruno Kessler, Italy
UEDIN	University of Edinburgh, UK

Table 3.16. Quaero German ASR evaluation 2010.

participant	**WER** [%]
RWTH	**16.94**
KIT	24.14
LIMSI + VR	21.05

ratio. The RWTH system uses morpheme-based LMs. The second position has been achieved out of four participants. The WER of the RWTH system is only 0.09% (absolute) higher than the best WER achieved by KIT in this year.

Table 3.17. Quaero German ASR evaluation 2011.

participant	**WER** [%]
RWTH	**17.49**
KIT	17.40
LIMSI + VR	18.04
VR	20.17

3.6.3 Quaero German and Polish ASR Evaluation 2012

Tables 3.18 and 3.19 show the results of the Quaero ASR evaluation on German and Polish ASR held in 2012. The evaluation data was a mix of broadcast news and broadcast conversations/podcasts, with an emphasis of the conversations. The RWTH German system uses morpheme-based LMs, whereas, the Polish system uses syllable-based LMs. The first and second positions have been achieved out of three participants for German and Polish respectively.

Table 3.18. Quaero German ASR evaluation 2012.

participant	**WER** [%]
RWTH	**18.71**
KIT	19.38
LIMSI + VR	19.63

Table 3.19. Quaero Polish ASR evaluation 2012.

participant	WER [%]
RWTH	**13.57**
LIMSI + VR	12.67
VR	14.79

3.6.4 Quaero German ASR Evaluation 2013

Table 3.20 shows the results of the Quaero evaluation on German ASR held in 2013. In fact, two different evaluation tracks have been considered. The first is based on lecture speech data, whereas the second is based on a mix of broadcast news and broadcast conversations/podcasts data. The RWTH systems use morpheme-based LMs. The first position has been achieved out of two participants in each domain.

Table 3.20. Quaero German ASR evaluation 2013.

participant	lecture data WER [%]	BN + BC data WER [%]
RWTH	**25.23**	**14.38**
VR	31.86	-
KIT	-	15.95

3.6.5 IWSLT German ASR Evaluation 2013

Table 3.21 shows the results of the IWSLT evaluation on German ASR held in 2013[9]. This is the tenth evaluation campaign organized by the IWSLT workshop. The 2013 evaluation has offered a track on lecture transcription based on the TED Talks[10] corpus. The RWTH system uses morpheme-based LMs. The first position has been achieved out of four participants. A detailed description of the recognition system is given in [Shaik & Tüske+ 2013].

Table 3.21. IWSLT German ASR evaluation 2013.

participant	WER [%]
RWTH	**16.94**
KIT	24.14
LIMSI + VR	21.05

3.6.6 OpenHaRT Arabic Handwriting Recognition Evaluation 2013

Table 3.22 shows the results of the 2013 NIST open handwriting recognition and translation evaluation (OpenHaRT 2013[11]) performed on Arabic handwriting text. The RWTH system uses morpheme-based Arabic LMs. The Arabic word decomposition is performed using MADA toolkit [Habash & Rambow 2005, 2007]. Two different evaluation tasks have been offered. The first is a constrained task, where only the official OpenHaRT data have been used. The second is an unconstrained task, where all the available data have been used. The second position has been achieved out of six participants in the constrained

[9] http://www.iwslt2013.org/59.php
[10] A collection of public speeches covering many different topics (http://www.ted.com).
[11] http://www.nist.gov/itl/iad/mig/hart2013.cfm

task. Whereas, the first position has been achieved out of two participants in the unconstrained task. A detailed description of the recognition system is given in [Hamdani & Doetsch+ 2014].

Table 3.22. OpenHaRT Arabic handwriting recognition evaluation 2013.

participant	constrained task WER [%]	unconstrained task WER [%]
RWTH	**23.91**	**16.15**
A2IA	20.32	18.50
CITLAB	26.81	-
LITIS	78.40	-
UOB-TELE	48.66	-
UPV	30.01	-

3.7 Summary

In this chapter, a set of sub-word based language modeling approaches have been investigated in order to deal with the challenges related to LVCSR of morphologically rich languages. Experiments have been conducted on Arabic, German, and Polish corpora. Different types of units have been investigated like full-words, morphemes, syllables and graphones. Novel graphones have been experimented based on morphemes or syllables. Supervised and unsupervised word decomposition techniques have been used to generate sub-word units. Trials have been made to use a mixture of multiple types of units in the same lexicon and LM. A careful optimization of the proposed approaches has been performed.

It has been shown that a carefully optimized system that uses a mixture of recognition units could achieve significant improvement in WER over the best traditional full-word based system. The most important optimizations are: the number of each type of unit, and the overall vocabulary size. Nevertheless, the performance of the sub-word based approach depends on the language, the corpus, and the used types of units. Usually, the best operating point for a sub-word based system occurs at a lower vocabulary size compared to the best operating point for a full-word based system. The influence of the sub-word based approach on the recognition performance is naturally higher when dealing with corpora having inherently high OOV rates. In this case, reductions in OOV rates can not easily be achieved by simply increasing the full-word vocabulary size.

The reason behind the success of sub-word based systems lies mainly in two important properties. The first is their capability to model unseen words and thus recognize OOV words. The second is the more reliable probability estimates obtained over more frequent units which leads to less mis-recognition of in-vocabulary words. On the other side, the main drawback of the sub-word based approach is that the degree of acoustic confusion among different recognition units becomes higher due to the shorter length of the units.

Generally, it has been observed that morphemes perform better for both Arabic and German experiments, whereas syllables perform better for Polish experiments. In most cases, the larger improvement gain comes from the use of full-words combined with normal morphemes or syllables, while a smaller or no gain is achieved by using graphones.

Chapter 4

Language Modeling with Morphology-Based Classes

An approach that attempts to improve the language modeling of morphologically rich languages is to incorporate morphology-based classes into the LM estimation process rather than using only words or sub-words. Normally, word classes are used to define some sort of similarity between words that helps to produce more knowledgeable estimates of probabilities of word sequences. In this chapter, we aim to exploit the morphological richness of the underlying languages to define morphology-based classes that can be used to estimate efficient LMs for such languages. This has been demonstrated as an effective way to handle the problem of data sparseness and to reduce the dependence of the traditional word-based LM on the discourse domain. In addition, this approach yields better smoothing and better generalization with regard to unseen word sequences.

In general, word classes can be generated based on linguistic methods as in [Bilmes & Kirchhoff 2003; Kirchhoff & Vergyri$^+$ 2006; Maltese & Bravetti$^+$ 2001; Tachbelie 2010; Tachbelie & Abate$^+$ 2011], or via data-driven approaches as in [Brown & deSouza$^+$ 1992; Kneser & Ney 1991, 1993b; Martin & Liermann$^+$ 1998; Matsuzaki & Miyao$^+$ 2003]. In this chapter, morphology-based classes are generated based on carefully designed and freely available morphological analyzers. Various models that utilize word classes are investigated in this chapter, including stream-based LMs [Kirchhoff & Vergyri$^+$ 2006], class-basedLMs [Brown & deSouza$^+$ 1992], and factored LMs [Bilmes & Kirchhoff 2003; Kirchhoff & Bilmes$^+$ 2002]. In stream- and class-based LMs, every class stream is treated separately without considering any interaction among different classes during the backoff. Whereas, in factored LMs, classes are viewed as generic factors and a so called backoff graph is used to define the backoff mechanism which handles different class streams jointly during the backoff.

In addition, a novel approach is introduced that attempts to retain the benefits of sub-word based LMs presented in Chapter 3 along with the advantages of using morphology-based classes. For this purpose, classes are generated on sub-word level, namely the morpheme level, rather than the level of full-words. Thus, stream-based, class-based, and factored LMs are estimated over morphemes and their classes, and then utilized to perform rescoring of N-best lists generated by performing a recognition pass via a traditional morpheme-based LM. Moreover, linear interpolation as well as log-linear score combination of different LMs are experimented.

In order to improve the smoothness of the LMs, a recent language modeling approach is utilized based on the hierarchical Pitman-Yor model (HPYLM) [Huang & Renals 2007; Teh 2006a]. It is a type of hierarchical Bayesian LM based on a coherent Bayesian probabilistic model that explicitly declares prior assumptions over the LM parameters. The HPYLM is used to estimate class-based LMs on morpheme level as well normal morpheme-based LMs. Thus, the traditional modified Kneser-Ney (MKN) smoothing is replaced with the hierarchical Pitman-Yor based estimation. This is a novel approach that aims at combining the benefits of sub-word based LMs and morphology-based classes with the advantages of the HPYLMs. The recognition experiments are performed on Arabic and German tasks. The approaches described in this chapter have been introduced in [El-Desoky & Schlüter$^+$ 2010; El-Desoky & Shaik$^+$ 2011; El-Desoky & Schlüter$^+$ 2012; El-Desoky & Shaik$^+$ 2012, 2013].

Section 4.1 shows how morphology-based classes are generated for Arabic and German words and morphemes. In addition, it presents a data-driven clustering algorithm used to generate an additional data-driven class for German words or morphemes. Sections 4.2, 4.3, and 4.4 introduce stream-based, class-based, and factored LMs respectively. Section 4.5 illustrates the foundations of the hierarchical Pitman-Yor LMs. Section 4.6 describes the methods of combining multiple LMs together. Experimental results are presented in Section 4.7, followed by a summary in Section 4.8.

4.1 Generating Classes

In this section, we show how to generate a set of reliable classes on both word and sub-word levels in order to use them for both word and sub-word based LMs. Since classes are essentially language dependent, we discuss the generation of Arabic and German classes separately.

4.1.1 Morphology-Based Classes for Arabic

For Arabic, we use the MADA morphological tags previously referred to in Section 3.3.1. Based on these tags, we generate two different classes, namely *lexeme* and *morph*. Lexeme is defined as an abstraction over the inflected word forms which represents all those forms that differ only in one of the morphological categories such as number, gender, aspect, or voice [Habash & Rambow 2007]. In other words, lexeme can be viewed as the canonical form of a word. Whereas, morph represents the morphological description of the word; it includes the word part-of-speech (POS) tag and indicates whether a conjunction, particle, article, or a clitic is agglutinated to the word. In addition to lexeme and morph, a third class called *pattern* is derived by subtracting *root* letters from the word. The root is generated via the *Sebawai* tool [Darwish 2002]. For more information about the root of the Arabic word, review Section 1.9.1.

The LM training corpus is preprocessed such that every word is replaced by a vector of classes including the word itself. The components of the vector are separated by a colon character ":". A single class is written in the form of a "$< tag > - < value >$" pair, thus:

$$word \rightarrow \mathbf{W}-word : \mathbf{M}-morph : \mathbf{L}-lexeme : \mathbf{P}-pattern.$$

A sequence of individual vector components defines a class stream. This format of writing classes has been adopted by [Bilmes & Kirchhoff 2003; Kirchhoff & Bilmes$^+$ 2008] n the development of factored LMs. Therefore, this format is called the *factored word representation*. For example, consider the Arabic word "واستخدامها : *and using it*" transliterated as *wAstxdAmhA*. The factored representation of this word is given as:

$$\mathbf{W}-wAstxdAmhA : \mathbf{M}-conj+N-3+clitic-FEM-SG-3 : \mathbf{L}-AstxdAm : \mathbf{P}-wAstCCAChA.$$

The morph class of the word that follows the tag "**M**" indicates that the word consists of a conjunction prefix, followed by a third person noun, followed by a pronominal clitic which is feminine and third person singular. The lexeme class following the tag "**L**" indicates that the canonical word form is *AstxdAm*. Whereas, the pattern class of the word following the tag "**P**" is given the value *wAstCCAChA*, where:

$$Root(wAstxdAmhA) = xdm.$$

The three letters of the root *xdm* are replaced by the placeholder letter "C". This means that other words can be generated following the same pattern by inserting new root letters in the places of the "C" letters. Now, given the morphemic decomposition of this word as:

$$wAstxdAmhA \rightarrow w+ \quad AstxdAm \quad +hA,$$

the factored representation can be readjusted by assigning classes to morphemes rather than to full-word.

Thus, the following *decomposed factored representation* can be generated:

$$\mathbf{W}-w+ : \mathbf{M}-conj : \mathbf{L}-w+ : \mathbf{P}-NUL$$
$$\mathbf{W}-AstxdAm : \mathbf{M}-N-3 : \mathbf{L}-AstxdAm : \mathbf{P}-AstCCAC$$
$$\mathbf{W}-+hA : \mathbf{M}-clitic-FEM-SG-3 : \mathbf{L}-+hA : \mathbf{P}-NUL$$

From this example, it can be seen that a careful handling of word morphological classes could help to produce valid class assignments to morphemes. This is what we call *morpheme-level classes*.

4.1.2 Morphology-Based Classes for German

For German, classes are generated using the *TreeTagger* [Schmid 1994]. It is a probabilistic tool that uses decision trees for annotating text with *part-of-speech* and *lemma* information. Lemma is defined as a particular form of the lexeme that serves as the canonical or the citation word form. Since the concept of lemma is closely related to the concept of lexeme, in this work, we refer to both terms as *lexeme*. In fact, the TreeTagger has been successfully used to tag words of many languages including German [Schmid 1995]. Indeed, it is adaptable to other languages if a lexicon and a manually tagged training corpus are available.

The LM training data is rewritten in a factored representation analogous to the one given in the Section 4.1.1, thus:

$$word \rightarrow \mathbf{W}-word : \mathbf{P}-POS-tag : \mathbf{L}-lexeme : \mathbf{I}-cluster-index$$

The class that follows the "**I**" tag is called a *cluster index*. It is a numeric class which is generated via running a data-driven clustering algorithm over the German text corpus. In fact, this is the only data-driven class used in our work. This class will be discussed in details in the following section. To illustrate by an example, consider the word *"eingeschlafen : [have] fallen asleep"*. Its factored representation is given as:

$$\mathbf{W}-eingeschlafen : \mathbf{P}-VVPP : \mathbf{L}-einschlafen : \mathbf{I}-224,$$

where VVPP means *past participle verb*. One of the important properties of the TreeTagger is that it operates successfully over morphemes as well as full-words provided that the input morphemes are linguistically meaningful. For the experiments performed in these thesis, morphemes are generated for German words using Morfessor [Creutz & Lagus 2005]. As mentioned in Chapter 3, the morphemes obtained by Morfessor are almost linguistically meaningful [Creutz 2006] (review Section 3.3.2). Therefore, the TreeTagger can be successfully used to generate valid part-of-speech tags and lexemes for German morphemes. It is also worth noting that, in most of the cases, the decomposition process of German words produces smaller valid words due to the abundance of compound words in German language. For example, given the following morphemic decomposition:

$$eingeschlafen \rightarrow ein+ \quad geschlafen,$$

the factored representation of this word can be readjusted such that classes are assigned to the morphemes. Thus, the following decomposed factored representation can be generated, where ART ≡ article:

$$\mathbf{W}-ein+ : \mathbf{P}-ART : \mathbf{L}-ein : \mathbf{I}-15$$
$$\mathbf{W}-geschlafen : \mathbf{P}-VVPP : \mathbf{L}-schlafen : \mathbf{I}-192$$

4.1.3 Data-Driven Word Clustering

One additional class called *cluster index* is generated for German words. It is a numerical index assigned to each word or morpheme after running a data-driven clustering procedure over the German text corpus. It indicates the cluster to which the word or the morpheme belongs. In fact, the class assigned to some word can be considered as a data-driven approximation to the real semantic class of the word. For generality, the term *word* is utilized in this section to refer to a word or a morpheme.

To perform word clustering, all discrete vocabulary words are first mapped into a continuous space in the form of vectors of real numbers using a method proposed in [Sarikaya & Afify$^+$ 2009]. Then, this

continuous space of vectors is clustered into some selected number of clusters via the standard *K-means* clustering algorithm. Then, every word is assigned a cluster index which acts as a data-driven class of the underlying word.

Discrete to continuous mapping. In order to map discrete words into a continuous space, an approach inspired from latent semantic analysis (LSA) is used [Deerwester & Dumais$^+$ 1990; Sarikaya & Afify$^+$ 2009]. Starting from the text corpus, a word-pair co-occurrence matrix is created based on the bigram counts extracted from the text. All the word bigrams are accumulated from the entire text corpus to fill in the entries of a co-occurrence matrix $\mathbf{C}$, where $\mathbf{C}(w_i, w_j)$ denotes the count of the bigram $w_j w_i$ in the text corpus. This forms a large but very sparse matrix, since typically a small number of words follow a given word. The matrix dimension is $M \times M$, where M is the vocabulary size. Because of its large size and sparsity, singular value decomposition (SVD) is a good choice to produce a reduced-rank approximation of the matrix $\mathbf{C}$. This co-occurrence matrix typically contains few high frequency events and many low frequency events. Since SVD derives a compact approximation of the co-occurrence matrix that is optimum in the least-square sense, it is normally over-fitted to the high frequency events which may not be the most informative. Therefore, the entries of the co-occurrence matrix are log-smoothed using to the following operation:

$$\hat{\mathbf{C}}(w_i, w_j) = log\big[\mathbf{C}(w_i, w_j) + 1\big]. \tag{4.1}$$

Then, using a similar approach as described in [Bellegarda 2000; Sarikaya & Afify$^+$ 2009], SVD is performed over the log-smoothed matrix $\hat{C}$, such that :

$$\hat{\mathbf{C}} \approx \mathbf{USV}^T. \tag{4.2}$$

Assuming that an order of decomposition $R : R \ll M$ is used, then $\mathbf{U}_{[M\times R]}$[1] is a left singular matrix, $\mathbf{S}_{[R\times R]}$ is a diagonal matrix of singular values, and $\mathbf{V}_{[M\times R]}$ is a right singular matrix. The continuous space of words is defined as the space spanned by the column vectors of $\mathbf{A}_{[M\times R]} = \mathbf{US}$.

Now, assuming that a word w_i is represented by an indication column vector $\vec{w}_{i[M\times 1]}$, where the i^{th} entry of $\vec{w}_i$ is equal to *one* and all the remaining $M-1$ entries are equal to *zero*. Then, this indication vector $\vec{w}_{i[M\times 1]}$ is mapped to a lower dimensional vector $\hat{w}_{i[R\times 1]}$ after applying a dimensionality reduction operation given by:

$$\hat{w}_i = \mathbf{A}^T \vec{w}_i. \tag{4.3}$$

In simpler words, a word w_i is represented by the i^{th} row vector of matrix $\mathbf{A}$. These row vectors are called *latent word vectors* which define the continuous space of the original discrete words.

4.2 Stream-Based m-gram Models

The first type of language model that utilizes word classes is called *stream-based LM*. This is a simplified model in which a traditional m-gram model is built over sequences of classes assigned to words rather than sequences of words themselves. The sequence of classes is called a class stream, like sequences of word roots, stems, part-of-speech (POS) tags, ... etc. It completely ignores the use of words in the probability estimation, rather it directly models the regularities governing sequences of class assignments. Thus, the probability $p(w_n|w_{n-m+1}^{n-1})$ of a traditional word m-gram model is replaced with $p(c_n|c_{n-m+1}^{n-1})$, where c_n is the class assigned to w_n at time n. Analogous to the traditional word m-gram model, given a sequence of class assignments c_1^N, the corresponding m-gram stream-based LM is given as:

$$p(c_1^N) \quad = \quad \prod_{n=1}^{N} p(c_n|c_{n-m+1}^{n-1}) \tag{4.4}$$

In order to train this model, the LM training corpus is preprocessed such that, for all sentences, word sequences are replaced with class sequences. This type of model built over classes can be used for rescor-

[1] The subscript defines the dimensions of the matrix.

ing N-best lists. To perform rescoring, the hypothesized N-best sentences are initially mapped to the corresponding sequences of classes suitable for the applied stream-based LM. Usually, the preprocessing of the N-best sentences follows the same steps of preprocessing the LM raining data.

4.3 Class-Based m-gram Models

The second type of language model that utilizes word classes is called *class-based LM*. The basic idea of the class-based LMs are initially mentioned in [Derouault & Merialdo 1986] and later described in [Brown & deSouza+ 1992; Kneser & Ney 1991]. This type of model combines the probability distribution over sequences of classes with the probability distribution of words given classes in order to produce more robust estimates of the probabilities of word sequences. In principle, to derive such a model, classes are utilized for both the predicted and the conditional words.

Assigning a word w_n to a class c_n can occur in two different forms. In the first form, there is many-to-one mappings from words and classes, i.e. a given word can only be uniquely mapped to one specific class. In this case, the class is called a *hard class* and the relationship is called *unambiguous class membership*. In the second more complex case, there may be many-to-many mappings from words to classes, i.e. a given word may belong to more than one class, and a given class will typically contain more than one word. In this case, the class is called a *soft class* and the relationship is called *ambiguous class membership*. In this section, we will consider the derivation of the class-based LMs in cases of using both hard and soft classes. In principle, hard classes can always be considered as a special case of soft classes.

Soft classes. Consider a trigram probability $p(w_n|w_{n-1}w_{n-2})$, where w_n denotes the word to be predicted, w_{n-1} and w_{n-2} are the context words. Let c_n denotes the class assigned to w_n at time (position) n, and assume that soft classes are used for both the predicted and the context words, then the following derivation of a trigram class-based LM takes place:

$$\begin{aligned} p(w_n|w_{n-1}w_{n-2}) &= \sum_{c_n,c_{n-1},c_{n-2}} p(w_n c_n c_{n-1} c_{n-2}|w_{n-1}w_{n-2}) \\ &= \sum_{c_n,c_{n-1},c_{n-2}} \Big\{ p(w_n|c_n c_{n-1} c_{n-2} w_{n-1} w_{n-2})\ p(c_n|c_{n-1} c_{n-2} w_{n-1} w_{n-2}) \\ &\qquad p(c_{n-1}|c_{n-2} w_{n-1} w_{n-2})\ p(c_{n-2}|w_{n-1} w_{n-2}) \Big\} \end{aligned} \tag{4.5}$$

Some approximations can be used to simplify Equation 4.5. The first is to consider only the dependence of w_n on c_n and ignore all the other complex dependencies. The second is to assume that the probability of a class c_n depends only on the classes of the previous words $c_{n-1}c_{n-2}$ ignoring the dependence on the previous words $w_{n-1}w_{n-2}$ themselves. A third approximation is to assume that the probability of a class at position n is independent on any word or class at different positions given the word at the same position n. These approximations yield the following simplified form of a trigram class-based LM:

$$p(w_n|w_{n-1}w_{n-2}) = \sum_{c_n,c_{n-1},c_{n-2}} p(w_n|c_n)\ p(c_n|c_{n-1}c_{n-2})\ p(c_{n-1}|w_{n-1})\ p(c_{n-2}|w_{n-2}) \tag{4.6}$$

Hard classes. In case that hard classes are used, then the summation of Equation 4.6 is removed considering only one exact class for each of w_n, w_{n-1}, w_{n-2}. This can be described by a deterministic mapping function $\hat{c}(\cdot)$ that maps words to the corresponding hard classes. Thus, given the Kronecker delta function $\delta(\cdot)$[2], the estimation takes on the form:

$$\begin{aligned} p(w_n|w_{n-1}w_{n-2}) &= \sum_{c_n,c_{n-1},c_{n-2}} p(w_n|c_n)\ p(c_n|c_{n-1}c_{n-2})\ \delta_{c_{n-1}=\hat{c}(w_{n-1})}\ \delta_{c_{n-2}=\hat{c}(w_{n-2})} \\ &= p(w_n|\hat{c}(w_n))\ p(\hat{c}(w_n)|\hat{c}(w_{n-1})\hat{c}(w_{n-2})) \end{aligned} \tag{4.7}$$

[2]The Kronecker delta function used with a boolean subscript notation: $\delta_{i=j} = 1$ iff $i = j$; otherwise $\delta_{i=j} = 0$.

Generalization. A generalization to Equation 4.5 for the m-gram case can be inductively derived as:

$$p(w_n|w_{n-m+1}^{n-1}) = \sum_{c_{n-m+1}^{n}} p(w_n|c_n)\, p(c_n|c_{n-m+1}^{n-1}) \prod_{k=n-m+1}^{n-1} p(c_k|w_k) \tag{4.8}$$

The probability $p(w_n|c_n)$ can be estimated using count statistics on the training data as follows:

$$p(w_n|c_n) = \frac{N(w_n, c_n)}{N(c_n)}, \tag{4.9}$$

where $N(w_n, c_n)$ is the count of occurrences of the word w_n in the class c_n, and $N(c_n)$ is the count of occurrences of the class c_n. The probability $p(c_n|w_n)$ can be written as follows:

$$\begin{aligned} p(c_n|w_n) &= \frac{p(w_n, c_n)}{p(w_n)} \\ &= \frac{p(w_n, c_n)}{\sum_{c_i} p(w_n, c_i)} \\ &= \frac{p(w_n|c_n)\, p(c_n)}{\sum_{c_i} p(w_n|c_i)\, p(c_i)} \end{aligned} \tag{4.10}$$

Substituting Equation 4.10 into Equation 4.8, we get the form:

$$p(w_n|w_{n-m+1}^{n-1}) = \sum_{c_{n-m+1}^{n}} p(w_n|c_n)\, p(c_n|c_{n-m+1}^{n-1}) \prod_{k=n-m+1}^{n-1} \frac{p(w_k|c_k)\, p(c_k)}{\sum_{c_i} p(w_k|c_i)\, p(c_i)} \tag{4.11}$$

In Equation 4.11, we can see that there are only two distributions required to estimate the class-based probability. The first is the probability distribution over sequences of classes, called *class m-gram*, at different orders including the unigram order. The second is the probability distribution of words given classes, called *class membership distribution* or *classification probability*.

4.4 Factored Language Models

The factored LM (FLM) is a general and flexible framework for incorporating classes into the LM estimation process. The FLM was initially introduced by [Bilmes & Kirchhoff 2003; Kirchhoff & Bilmes$^+$ 2002] for incorporating various morphological information into Arabic LMs. Later, in [Tachbelie 2010; Tachbelie & Abate$^+$ 2011], it has been successfully used for Amharic LMs.

In FLMs, words and their classes are considered as generic factors. Every word in a text corpus is replaced by a vector of parallel factors over which the probability estimation is to be performed. The main difference between the FLM and the other models, like the stream- and class-based LMs, is that the FLM uses a complex backoff mechanism to handle different class streams jointly during the backoff. Here, the main idea is to backoff to different combinations of classes when some word m-gram is not sufficiently observed in the training data.

Definition. The FLM views a word as a vector of K parallel factors, such that:

$$w_n \rightarrow f_n^1, f_n^2, ..., f_n^K = f_n^{1:K}. \tag{4.12}$$

A factor could be the word itself or any class assigned to the word such as a morphological or a data-driven class. Hence, a sequence of words w_1^N is viewed as a sequence of K parallel factors, such that:

$$w_1^N \rightarrow f_1^{1:K}, f_2^{1:K}, ..., f_N^{1:K} = f_{1:N}^{1:K}. \tag{4.13}$$

The objective of the FLM is to produce a statistical model over these factors, in the form:

$$p(f_{1:N}^{1:K}) = p(f_1^{1:K}, f_2^{1:K}, ..., f_N^{1:K}). \tag{4.14}$$

Using an m-gram-like formula, the model takes the form:

$$p(f_{1:N}^{1:K}) \approx \prod_{n=1}^{N} p(f_n^{1:K} | f_{n-m+1:n-1}^{1:K}). \tag{4.15}$$

Thus, the goal is to produce accurate models of the form:

$$p(f_n^{1:K} | f_{n-m+1:n-1}^{1:K}).$$

This form opens up the possibility for many modeling option. Applying the chain rule, the conditional probability used in Equation 4.15 can be expressed as:

$$p(f_n^{1:K} | f_{n-m+1:n-1}^{1:K}) = \prod_k p(f_n^k | f_n^{1:k-1}, f_{n-m+1:n-1}^{1:K}) \tag{4.16}$$

This is one possible chain rule ordering of the factors. Given the large number of all the possible chain rule orderings and the number of all possible subsets of conditioning factors, it can be seen that this model represents a large family of statistical LMs [Kirchhoff & Bilmes$^+$ 2008].

Backoff strategy. In a typical FLM, the required distributions take the form:

$$p(f|f_1, f_2, ..., f_M), \tag{4.17}$$

which represents a conditional probability over a set of $M + 1$ random variables. It is not necessary that all these variables are words nor they represent a certain chronological order[3]. In this case, the application of a backoff procedure with a specific dropping order is not straightforward. There are in fact quite a large number of possible orders. Each single dropping order is called a *backoff path*, and all the possible backoff paths can be depicted in a so-called *backoff graph* in which each node corresponds to a particular statistical model (see e.g. Figure 4.1) [Kirchhoff & Bilmes$^+$ 2008]. In [Bilmes & Kirchhoff 2003],

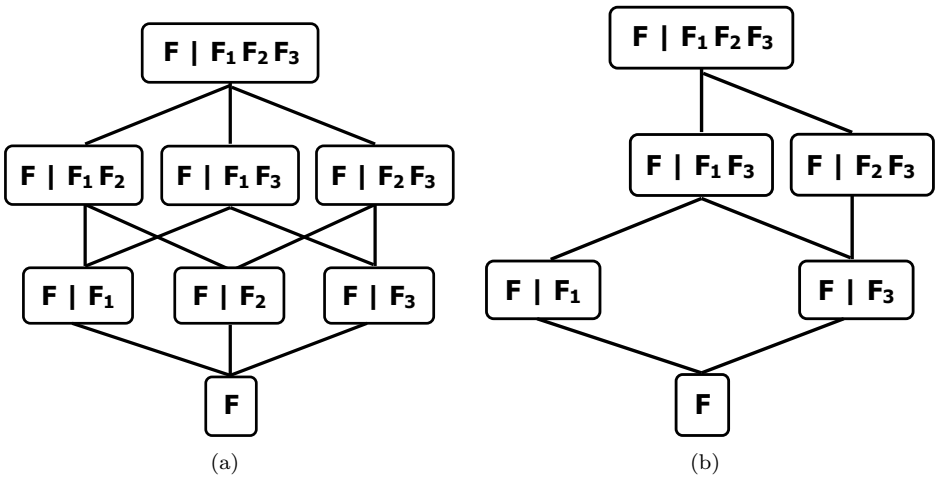


Figure 4.1. (a) An example of a general backoff graph showing all possible backoff paths from top to bottom. (b) An example of a backoff graph where only a subset of the possible backoff paths are allowed.

two alternative approaches are introduced in order to select backoff paths, which are:

[3]Usually, the chronological order applies to sets of variables that are related to words at different times.

1. **Single path:** Only one particular backoff path is chosen at run time to produce the probability, depending on the particular sequence of factors for which the probability is estimated.
2. **Multiple paths:** Multiple backoff paths are used simultaneously at run time to produce the probability, where the set of the used multiple paths might change depending on the particular sequence of factors for which the probability is estimated.

This methodology is called *generalized backoff*. It can be used to improve over the fixed backoff path, since the paths can be designed to best suit the given instances of factors. The generalized backoff procedure is a generalization of the standard Katz's backoff, thus:

$$p_{BO}(f|f_1,f_2,...,f_M) = \begin{cases} d(f,f_1,f_2,...,f_M)p_{ML}(f|f_1,f_2,...,f_M) & \text{if } N(f,f_1,f_2,...,f_M) > \tau_{M+1} \\ \gamma(f_1,f_2,...,f_M)\,g(f,f_1,f_2,...,f_M) & \text{otherwise} \end{cases} \tag{4.18}$$

where $p_{ML}(f|f_1,f_2,...,f_M)$ is the ML distribution , ${\tau_{M+1}}^4$ is a user specified threshold used to determine when a language model *hit* occurs at the current level of the backoff path, or whether to backoff to the next level. The function $g(f,f_1,f_2,...,f_M)$ is the backoff distribution, which is only guaranteed to be a non-negative function. The function $\gamma(f_1,f_2,...,f_M)$ is selected such that the entire distribution is valid, thus to be non-negative and sums to unity. In fact, the choice of the function $g(f,f_1,f_2,...,f_M)$ determines the used backoff strategy. In [Kirchhoff & Bilmes$^+$ 2008], a number of possible forms of $g(f,f_1,f_2,...,f_M)$ are introduced that are implemented as a part of the SRILM toolkit [Stolcke 2002].

Parameter tuning. The following three types of parameters are required to fully define a FLM:

1. **Initial conditioning factors:** factors used in estimating the m-gram probabilities.
2. **Backoff graph:** order of visiting backoff nodes to apply a backoff procedure.
3. **Smoothing options:** smoothing technique applied at every node of the backoff graph.

It can be recognized that the parameter space of the FLM is extremely large. Usually, this space cannot be searched exhaustively, and optimizing models by knowledge-based manual search procedures often leads to suboptimal results. Therefore, it is very important to use automatic search procedures. In [Duh & Kirchhoff 2004], an efficient *genetic algorithm* called `GA-FLM` is introduced to solve this problem.

4.5 Hierarchical Pitman-Yor Language Models

A hierarchical Pitman-Yor LM (HPYLM) is a type of hierarchical Bayesian LM based on a coherent Bayesian probabilistic model that explicitly declares prior assumptions over the LM parameters [Huang & Renals 2007; Teh 2006a]. The concept of statistical priors has been previously investigated using different types of priors like in [Brand 1999; Chen & Rosenfeld 2000; Goodman 2004]. The HPYLM in particular is based on the Pitman-Yor (PY) process which is a nonparametric generalization of the widely used Dirichlet distribution [Ishwaran & James 2001; Pitman 2002; Pitman & Yor 1997]. The resulting HPYLM is considered as a direct generalization of the hierarchical Dirichlet LM proposed by [MacKay & Peto 1994].

The PY process produces *power-law* distributions over word frequencies [Goldwater & Griffiths$^+$ 2006, 2011]. This means that few number of words occur with very high probabilities, while most words occur with low probabilities. This distribution has been found to be one of the most striking statistical properties of word frequencies in natural languages.

Pitman-Yor process. Initially, as adopted by [Teh 2006b], the Pitman-Yor (PY) process [Pitman 1995; Pitman & Yor 1997] can be described in the context of a unigram LM. Thus, Let W be a finite vocabulary of V words. For each word $w \in W$, let $G(w)$ be the probability of w, and let $G = [G(w)]_{w \in W}$ be the vector of word probabilities. We place a PY process prior on G, such that:

$$G \sim PY(d,\theta,G_0), \tag{4.19}$$

[4]The subscript of τ indicates the number of random variables at the current level of the backoff graph. Generally, $M+1$ variables are considered in our equations (1 child variable + M parent variables).

where the PY process is a distribution over distributions over words. The parameters of the process are: a discount parameter $0 \leq d < 1$, a strength parameter $\theta > -d$, and a mean vector $G_0 = [G_0(w)]_{w \in W}$. $G_0(w)$ is the prior probability of word w before observing any data. Usually, a uniform distribution is used for G_0, such that $G_0(w) = 1/V \ \forall \ w \in W$.

We are interested in the distribution over sequences of words induced by the PY process. It can be seen that both G and G_0 are distributions over W, and word $w \in W$ has probabilities $G(w)$ and $G_0(w)$ respectively. Let $[x_l] = x_1, x_2, ...$ be a sequence of words drawn from G. The PY process is described as a generative procedure that iteratively produces $[x_l]$ with G marginalized out. This can be achieved by relating $[x_l]$ to another separate sequence of draws $[y_k] = y_1, y_2, ...$ from the mean distribution G_0.

The first word x_1 is assigned the value of the first draw y_1 from G_0. Let t be the current number of draws from G_0 (currently $t = 1$), c_k be the number of words assigned the value of draw y_k (currently $c_1 = 1$), and $c = \sum_{k=1}^{t} c_k$ be the current number of draws from G. For each subsequent word x_{c+1}, we either assign it the value of a previous draw y_k with probability $\frac{c_k - d}{\theta + c}$ (increment c_k; set $x_{c+1} \leftarrow y_k$), or we assign it the value of a new draw from G_0 with probability $\frac{\theta + dt}{\theta + c}$ (increment t; set $c_t = 1$; draw $y_t \sim G_0$; *set* $x_{c+1} \leftarrow y_t$). This process has been shown to produce a so-called *power-law* distribution, where many unique words are observed, most of them rarely.

Chinese restaurant process. The procedure for generating words from G is often referred to as the Chinese restaurant process (CRP) [Pitman 2002]. Imagine a sequence of customers (corresponding to draws from G) visiting a Chinese restaurant with an infinite number of tables (corresponding to draws from G_0), each of which can accommodate an infinite number of customers. The first customer sits at the first available table, and each subsequent customer either joins an already occupied table (assign a word to a previous draw from G_0), or sits at a new table (assign a word to a new draw from G_0). The exact words drawn from G_0 can be thought of as dishes served by tables, such that the customers sitting at each table eat the dish served by this table [Teh 2006b]. Here, it should be noted that it is possible for the same dish to be served on multiple tables.

Hierarchical Pitman-Yor process. A hierarchical Pitman-Yor LM (HPYLM) can be described based on a hierarchical extension of the PY process described above. An m-gram LM defines probabilities over the current word given contexts up to $m-1$ words. Given a context $\mathbf{u}$, let $G_{\mathbf{u}}(w)$ be the probability of the current word taking on value w. We use a PY process as the prior for $G_{\mathbf{u}} = [G_{\mathbf{u}}(w)]_{w \in W}$, thus:

$$G_{\mathbf{u}} \sim PY(d_{|\mathbf{u}|}, \theta_{|\mathbf{u}|}, G_{\pi(\mathbf{u})}), \tag{4.20}$$

where $\pi(\mathbf{u})$ is the suffix context of $\mathbf{u}$ after dropping the earliest word. The strength and discount parameters are functions of the length $|\mathbf{u}|$ of the context, while the mean vector is $G_{\pi(\mathbf{u})}$, the vector of probabilities of the current word given all but the earliest word in the context. Since $G_{\pi(\mathbf{u})}$ is also unknown, we recursively place a prior over $G_{\pi(\mathbf{u})}$ using 4.20, but with parameters $\theta_{|\pi(\mathbf{u})|}, d_{|\pi(\mathbf{u})|}$, and a mean vector $G_{\pi(\pi(\mathbf{u}))}$. This is repeated until reaching G_ϕ, which is the vector of probabilities over the current word given the empty context ϕ. Then, we place a prior over G_ϕ as:

$$G_\phi \sim PY(d_0, \theta_0, G_0), \tag{4.21}$$

where G_0 is the global uniform mean vector, $G_0(w) = 1/V \ \forall \ w \in W$. This choice of the prior structure expresses the common belief that earlier words in context have the least importance in modeling the probability of the current word.

Starting from a posterior distribution over seating arrangements in a hierarchical Chinese restaurant, the predictive probability of a word w after a context $\mathbf{u}$ can be inferred using Gibbs sampling [Neal 1993]. A detailed inference scheme is described in Teh [2006b].

4.6 Combining Multiple Language Models

Normally, the standard word-based m-gram LMs perform betters than stream- and class-based LMs in capturing the relationships between words for a text belonging to the same discourse domain as the training text. Therefore, an effective way to retain the advantages of these types of LMs is to combine them. The most common combination approaches may rely on: linear interpolation [Samuelsson & Reichl 1999], or N-best score combination [Kirchhoff & Vergyri$^+$ 2006]. The following two sections discuss these methods.

4.6.1 Linear Interpolation

Since word-based and class-based LMs are used to estimate the same target m-gram probability distribution $p(w_n|w_{n-m+1}^{n-1})$, they can be easily interpolated. The linear interpolation of multiple word-based and class-based LMs is expressed as:

$$p(w|h) = \sum_{i=1}^{I} \lambda_i \, p_w^{(i)}(w|h) + \sum_{j=1}^{J} \lambda_j \, p_c^{(j)}(w|h) \tag{4.22}$$

where w is the word, h is the history, $p_w^{(i)}(w|h)$ is i^{th} word-based conditional probability, $p_c^{(j)}(w|h)$ is the j^{th} class-based conditional probability, λ_i, λ_j are the interpolation weights optimized on some development data, such that $\sum_{i=1}^{I} \lambda_i + \sum_{j=1}^{J} \lambda_j = 1$, and $(I + J)$ is the total number of the interpolated models. This is considered a general interpolation formula which expresses the different linear interpolations performed in this thesis. For instance, different class-based LMs could be estimated over different class streams (e.g. lexemes, part-of-speech tags, ... etc.). Furthermore, different word-based LMs could be estimated using different estimation models (e.g. backoff m-gram, hierarchical Pitman-Yor, neural network, ... etc.). The neural network LMs are to be discussed in Chapter 5.

4.6.2 Score Combination

If the target probabilities are different among multiple LMs; such as in the case of word-based and stream-based LMs; then the interpolation becomes not straight forward to perform. In this case, the N-best score combination can be used. Thus, every LM is separately used to generate different scores for the N-best sentences, then these scores are combined together to provide a single score used to re-rank the N-best sentences and thus reselect the best hypothesis with the highest score.

During the LM rescoring of N-best lists, the scores used for re-ranking the N-best hypotheses are normally a weighted combination of several component scores representing the acoustic score, the LM score and the number of words. However, as mentioned before, scores from various LMs can be added. The final score for each hypothesis can be computed as a log-linear combination of the invoked scores. The weights of this combination can be optimized such that the word error rate (WER) is minimized on some development data [Kirchhoff & Vergyri$^+$ 2006]. This is similar to the framework of the discriminative model combination described in [Beyerlein 1998]. Thus, assume that there are K different models $p_k(w_1^N)$, where $k = 1, ..., K$. These models can be log-linearly combined into a distribution of the following form:

$$p_{\{\Lambda\}}(w_1^N) = exp\left\{ log\, C(\Lambda) + \sum_{k=1}^{K} \lambda_k \, log\, p_k(w_1^N) \right\} \tag{4.23}$$

The coefficients $\Lambda = (\lambda_1, ..., \lambda_K)$ can be considered as weights of the models p_k within the model combination. The value $C(\Lambda)$ is a normalization factor. The weights are optimized so as to achieve the minimum WER over a development set. In this work, the weight optimization is performed using *"Amoeba simplex"* search [Press & Flannery$^+$ 1988] which is implemented as a part of the publicly available SRILM toolkit [Stolcke 2002].

4.7 Experimental Results

In this section, experimental results are presented on stream-based, class-based, and factored LMs estimated using word and sub-word level classes presented in Section 4.1. In addition, results are reported on using hierarchical Pitman-Yor LMs in the estimation of traditional word and sub-word models as well as class-based models. The Experiments are performed using Arabic and German testing systems. A detailed description of these systems along with a description of the development and evaluation corpora is found in Appendix A. Before reporting the experimental results, we discuss the optimization of the factored LMs over the given classes for both Arabic and German.

4.7.1 Optimization of Factored Language Models

Optimizing Arabic FLMs. Arabic FLMs are estimated over words and morphemes along with lexeme, morph, and pattern classes presented in Section 4.1.1. In order to obtain a good performance via FLMs, the FLM parameters are optimized using the GA-FLM tool [Duh & Kirchhoff 2004]. This tool can not search all the possible FLM structures due to the design limitations of the backoff graph encoding (review [Duh & Kirchhoff 2004]). Therefore, the GA-FLM optimization is followed by a manual optimization in order to fine tune the FLM parameters. Because of memory limitations, factors up to only *two* previous time slots are used (trigram-like models).

After performing the FLM parameter optimization, we come up with a set of competing FLMs having comparably low perplexities. The exact topologies of those FLMs are given in Figure 4.2. In this figure, we use the FLM format specifications adopted by the SRILM-FLM extensions [Kirchhoff & Bilmes$^+$ 2008; Stolcke 2002]. For a detailed description of these specifications, see [Kirchhoff & Bilmes$^+$ 2008]. The letters {*W, M, L, P*} refer to the factors {*word, lexeme, morph, pattern*} respectively. The first model ($AR\text{–}FLM_1$) represents the FLM equivalent of the standard trigram LM used as a reference baseline. Whereas, the two FLMs ($AR\text{–}FLM_{2,3}$) correspond to the model $P(W_n|W_{n-1}, M_{n-1}, L_{n-1}, P_{n-1}, W_{n-2})$ with only slight differences in the smoothing options. The last two FLMs ($AR\text{–}FLM_{4,5}$) correspond to the model $P(W_n|W_{n-1}, M_{n-1}, L_{n-1}, W_{n-2}, M_{n-2}, L_{n-2})$ also with slight differences in the smoothing options. The *gtmin* option refers to the count threshold that is sufficient to have a language model hit at some node of the backoff graph. The backoff graphs of these FLMs are shown in Figure 4.3.

It is well known in the ASR community that there is a strong correlation between the perplexity of a LM and its performance in a speech recognition system [Klakow & Peters 2002]. However, this correlation is not so strict as might be thought [Clarkson & Robinson 1999, 2001]. This means that the LM which achieves the minimum perplexity is not necessarily the one that yields the minimum WER in the real ASR task. Therefore, the obtained Arabic FLMs that have the least perplexities need to be tested on a real speech recognition experiment in order to discover the best performing FLM.

Table 4.1 shows the results of the recognition experiments performed on a small Arabic tuning corpus *ar-tune07* using the first pass of the modern standard Arabic (MSA) system described in Appendix A. A small vocabulary of 70k full-words is used. The system performs one recognition pass that uses a bigram LM to produce N-best lists *(N = 1000)* which are preprocessed such that the words are converted into the factored word representation shown in Section 4.1.1. All the competing FLMs ($AR\text{–}FLM_{2:5}$) are used to rescore these N-best lists. Here, only word-based FLMs are used to rescore word N-best lists in order to test the performance of the corresponding topology. The resulting WERs are presented in Table 4.1.

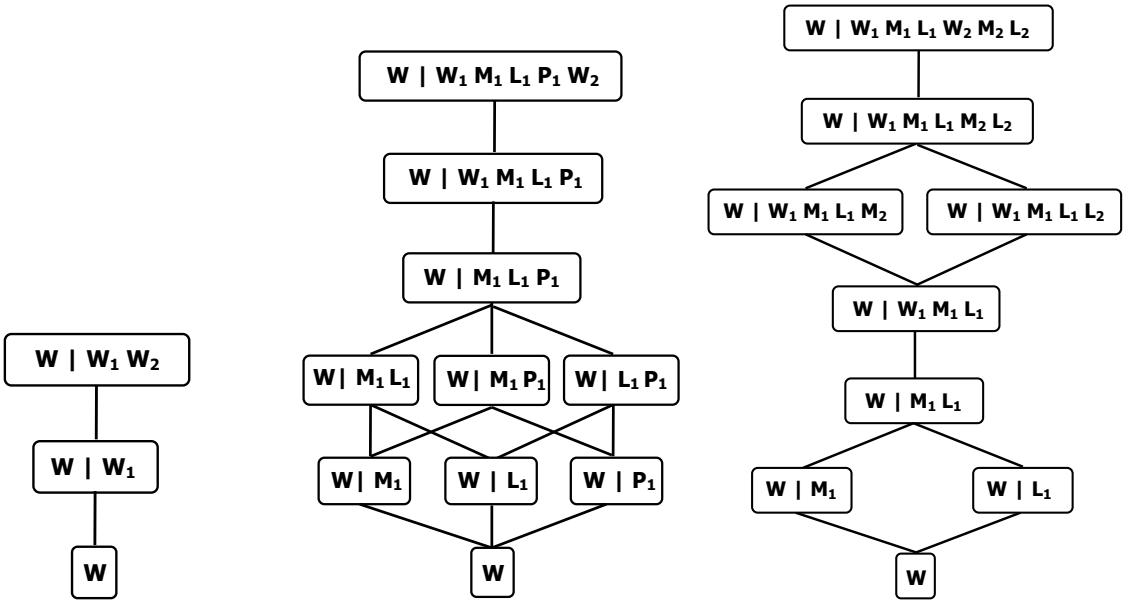
Comparing the baseline perplexity with the perplexities of other FLMs, it can be seen that most of the FLMs are able to improve the perplexity compared to the standard trigram model. The best WER is obtained using $AR\text{–}FLM_4$ although it is not the FLM that achieves the least perplexity. Therefore, $AR\text{–}FLM_4$ is selected for our recognition experiments.

Optimizing German FLMs. German FLMs are estimated over words and morphemes along with lexeme, POS-tag, and cluster-index classes presented in Section 4.1.2. Similar to Arabic FLMs, the FLM parameters are optimized using the GA-FLM tool in addition to manual fine tuning. Also, due to memory limitations, factors up to only *two* previous time slots are considered.

After performing the parameter optimization, a set of competing FLMs with comparably low per-

```
## AR-FLM1 ≡ P(Wn|Wn-1,Wn-2) [standard trigram]
W : 2 W(-1) W(-2)  flm1.count.gz  flm1.lm.gz 3
  W1,W2  W2 kndiscount gtmin 1 interpolate
  W1     W1 kndiscount gtmin 1 interpolate
  0      0 kndiscount gtmin 1

## AR-FLM2 ≡ P(Wn|Wn-1,Mn-1,Ln-1,Pn-1,Wn-2)
W : 5 W(-1) M(-1) L(-1) P(-1) W(-2)  flm2.count.gz  flm2.lm.gz 10
  W1,M1,L1,P1,W2   W2  kndiscount gtmin 3 interpolate
  W1,M1,L1,P1      W1  kndiscount gtmin 2 interpolate
  M1,L1,P1         M1,L1,P1  kndiscount gtmin 1000000 combine max strategy bog_node_prob
  L1,P1            L1,P1 kndiscount gtmin 1000000 combine max strategy bog_node_prob
  M1,P1            M1,P1 kndiscount gtmin 1000000 combine max strategy bog_node_prob
  M1,L1            M1,L1 kndiscount gtmin 1000000 combine max strategy bog_node_prob
  M1               M1  kndiscount gtmin 3 kn-count-parent W1,M1,L1,P1
  P1               P1  kndiscount gtmin 3 kn-count-parent W1,M1,L1,P1
  L1               L1  kndiscount gtmin 1 kn-count-parent W1,M1,L1,P1
  0                0   kndiscount gtmin 1 kn-count-parent W1,M1,L1,P1

## AR-FLM3 ≡ P(Wn|Wn-1,Mn-1,Ln-1,Pn-1,Wn-2)
W : 5 W(-1) M(-1) L(-1) P(-1) W(-2)  flm3.count.gz  flm3.lm.gz 10
  W1,M1,L1,P1,W2   W2  kndiscount gtmin 1 interpolate
  W1,M1,L1,P1      W1  kndiscount gtmin 1 interpolate
  M1,L1,P1         M1,L1,P1 kndiscount gtmin 1000000 combine max strategy bog_node_prob
  L1,P1            L1,P1  kndiscount gtmin 1000000 combine max strategy bog_node_prob
  M1,P1            M1,P1  kndiscount gtmin 1000000 combine max strategy bog_node_prob
  M1,L1            M1,L1  kndiscount gtmin 1000000 combine max strategy bog_node_prob
  M1               M1   kndiscount gtmin 1 kn-count-parent W1,M1,L1,P1
  P1               P1   kndiscount gtmin 1 kn-count-parent W1,M1,L1,P1
  L1               L1   kndiscount gtmin 1 kn-count-parent W1,M1,L1,P1
  0                0    kndiscount gtmin 1 kn-count-parent W1,M1,L1,P1

## AR-FLM4 ≡ P(Wn|Wn-1,Mn-1,Ln-1,Wn-2,Mn-2,Ln-2)
W : 6 W(-1) M(-1) L(-1) W(-2) M(-2) L(-2)   flm4.count.gz  flm4.lm.gz 9
  W1,M1,L1,W2,M2,L2 W2     kndiscount gtmin 3 interpolate
  W1,M1,L1,M2,L2    L2,M2  kndiscount gtmin 1000000 combine mean
  W1,M1,L1,M2       M2  kndiscount gtmin 4 kn-count-parent 0b111111
  W1,M1,L1,L2       L2  kndiscount gtmin 2 kn-count-parent 0b111111
  W1,M1,L1          W1  kndiscount gtmin 2 interpolate
  M1,L1             L1,M1  kndiscount gtmin 1000000 combine max strategy bog_node_prob
  M1                M1  kndiscount gtmin 3 kn-count-parent W1,M1,L1
  L1                L1  kndiscount gtmin 1 kn-count-parent W1,M1,L1
  0                 0   kndiscount gtmin 1 kn-count-parent W1,M1,L1

## AR-FLM5 ≡ P(Wn|Wn-1,Mn-1,Ln-1,Wn-2,Mn-2,Ln-2)
W : 6 W(-1) M(-1) L(-1) W(-2) M(-2) L(-2)  flm5.count.gz  flm5.lm.gz 9
  W1,M1,L1,W2,M2,L2 W2     kndiscount gtmin 1 interpolate
  W1,M1,L1,M2,L2    L2,M2  kndiscount gtmin 1000000 combine mean
  W1,M1,L1,M2       M2  kndiscount gtmin 1 kn-count-parent 0b111111
  W1,M1,L1,L2       L2  kndiscount gtmin 1 kn-count-parent 0b111111
  W1,M1,L1          W1  kndiscount gtmin 1 interpolate
  M1,L1             L1,M1  kndiscount gtmin 1000000 combine max strategy bog_node_prob
  M1                M1  kndiscount gtmin 1 kn-count-parent W1,M1,L1
  L1                L1  kndiscount gtmin 1 kn-count-parent W1,M1,L1
  0                 0   kndiscount gtmin 1 kn-count-parent W1,M1,L1
```

Figure 4.2. Topologies of the Arabic FLMs using the format specifications of the SRILM-FLM extensions (**W**: word; **M**: morph; **L**: lexeme; **P**: pattern).



(a) $AR{-}FLM_1$ (b) $AR{-}FLM_{2,3}$ (c) $AR{-}FLM_{4,5}$

Figure 4.3. Backoff graphs for $AR{-}FLM_{1:5}$, detailed topologies are given in Figure 4.2 (**W**: word; **M**: morph; **L**: lexeme; **P**: pattern).

Table 4.1. Recognition experiments on Arabic ar-tune07 corpus using different factored LMs (**vocabulary**: 70k full-words, OOV rate = 3.6%, N-best size = 1000, **N-best error rate (NER)** = 7.3%; **W**: word; **M**: morph; **L**: lexeme; **P**: pattern).

	ar-tune07	
factored LM	**PPL**	**WER[%]**
$P(W_n \mid W_{n-1}, W_{n-2})$		
$AR{-}FLM_1$	302.6	20.4
$P(W_n \mid W_{n-1}, M_{n-1}, L_{n-1}, P_{n-1}, W_{n-2})$		
$AR{-}FLM_2$	306.2	20.2
$AR{-}FLM_3$	290.9	20.4
$P(W_n \mid W_{n-1}, M_{n-1}, L_{n-1}, W_{n-2}, M_{n-2}, L_{n-2})$		
$AR{-}FLM_4$	300.2	**19.9**
$AR{-}FLM_5$	294.5	20.3

plexities are obtained. The exact topologies of those FLMs are shown in Figure 4.4. The letters {*W, L, P, I*} refer to the factors {*word, lexeme, POS-tag, cluster-index*} respectively. The first model ($GR-FLM_1$) represents the FLM version of the standard trigram LM. Whereas, the three FLMs ($GR-FLM_{2:4}$) correspond to the model $P(W_n|W_{n-1}, L_{n-1}, I_{n-1}, P_{n-1}, W_{n-2}, L_{n-2}, I_{n-2}, P_{n-2})$ with differences in backoff graphs and smoothing options. The two FLMs ($GR-FLM_{5,6}$) correspond to the model $P(W_n|W_{n-1}, L_{n-1}, P_{n-1}, W_{n-2}, L_{n-2}, P_{n-2})$ also with differences in backoff graphs and smoothing options. The last FLM ($GR–FLM_7$) corresponds to the model $P(W_n|W_{n-1}, I_{n-1}, P_{n-1}, W_{n-2}, I_{n-2}, P_{n-2})$. The backoff graphs of these FLMs are shown in Figure 4.5.

Table 4.2 shows the perplexities of all the FLMs measured on the German development corpus *gr-dev09*. The FLMs are estimated on word and morpheme levels. The word-based FLMs are using a vocabulary of 100k full-words. Whereas, the morpheme-based FLMs are using a 100k vocabulary having *5k full-words + 95k morphemes*. This particular choice of the number of full-words and morphemes is motivated by the previous optimization presented in Section 3.5.2. It can be seen that most of the FLMs are able to decrease the perplexity compared to the traditional trigram LM ($GR-FLM_1$). The pattern of the perplexity reduction is consistent for both word- and morpheme-based FLMs.

Table 4.2. Perplexities for the German FLMs $GR-FLM_{1:7}$ measured on the German gr-dev09 corpus. Exact FLM topologies are given in Figures 4.4 and 4.5 (**word-based**: 100k full-words vocab; **morpheme-based**: 100k morpheme-based vocab with 5k full-words + 95k morphemes; **W**: word; **L**: lexeme; **I**: class-index; **P**: POS-tag).

	gr-dev09	
factored LM	**word-based**	**morpheme-based**
$P(W_n\|W_{n-1}, W_{n-2})$		
$GR-FLM_1$	349.7	311.0
$P(W_n\|W_{n-1}, L_{n-1}, I_{n-1}, P_{n-1}, W_{n-2}, L_{n-2}, I_{n-2}, P_{n-2})$		
$GR-FLM_2$	311.7	280.6
$GR-FLM_3$	314.8	283.8
$GR-FLM_4$	330.4	296.4
$P(W_n\|W_{n-1}, L_{n-1}, P_{n-1}, W_{n-2}, L_{n-2}, P_{n-2})$		
$GR-FLM_5$	342.9	306.0
$GR-FLM_6$	384.7	343.0
$P(W_n\|W_{n-1}, I_{n-1}, P_{n-1}, W_{n-2}, I_{n-2}, P_{n-2})$		
$GR-FLM_7$	326.2	294.7

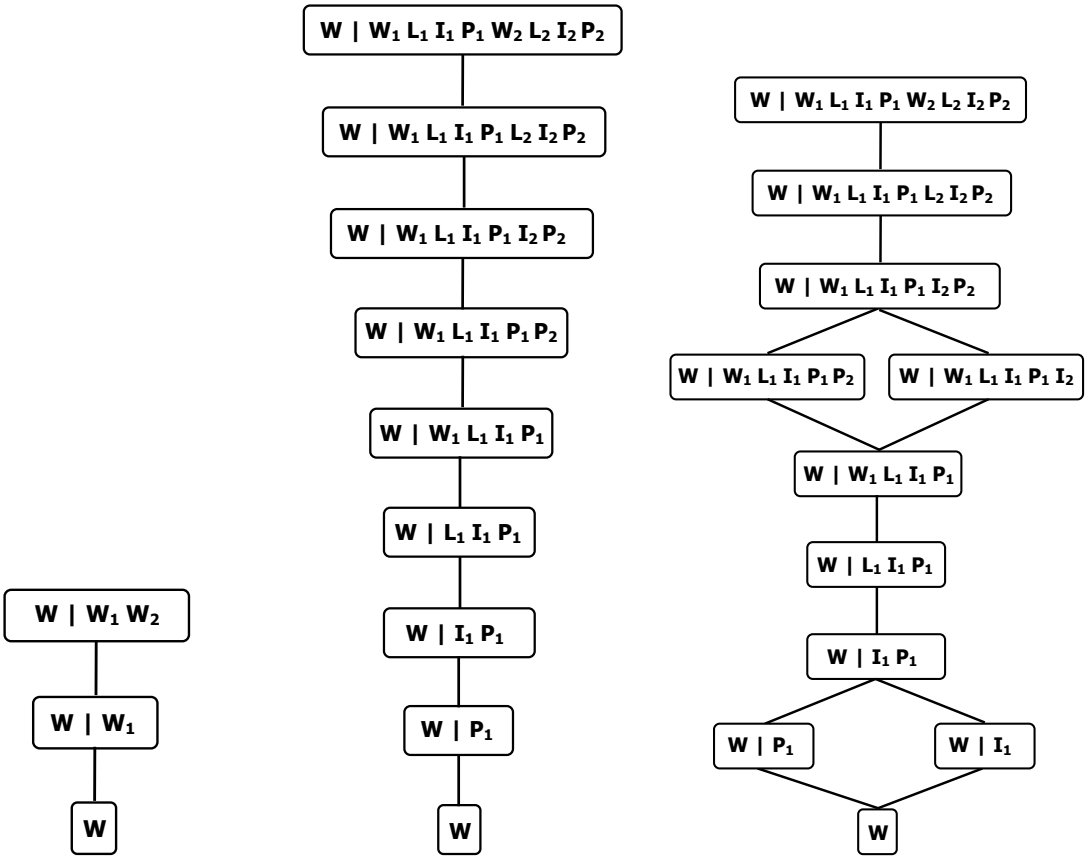
In order to select the best FLM, all the FLMs shown in Table 4.2 are directly employed in recognition experiments. Table 4.3 shows the results of the recognition experiments performed on German corpora using the German testing system described in Appendix A. A vocabulary of size 100k is used. The system performs 2 recognition passes. The second pass uses a 4-gram LM to generate N-best lists *(N = 1000)* which are preprocessed and rescored using the FLMs $GR-FLM_{2:7}$ (see Figures 4.4 and 4.5). The recognition results are shown for both word-based and morpheme-based systems. It can be seen that the best results are obtained using $GR-FLM_5$. Therefore, it is selected for our recognition experiments.

4.7.2 Experiments on Arabic

In order to compare the performance of stream-based LMs versus class-based LMs, Table 4.4 shows the results of recognition experiments performed on the Arabic development corpus *ar-dev07* using the modern standard Arabic (MSA) testing system described in Appendix A. The system performs 3 recognition passes, each of which uses a bigram LM to produce lattices which are then rescored using a 4-gram LM. Additionally, in the third recognition pass, the system produces N-best lists *(N = 1000)* which are rescored using different LMs. The results are shown for both word-based and morpheme-based systems. The word-based system uses 70k full-words, whereas, the morpheme-based system uses 20k full-words + 50k morphemes. The configuration of the morpheme-based system is motivated by the optimization performed in Section 3.5.1. In addition, class-based LMs are linearly interpolated with a traditional word- or morpheme-based LM and used to perform the N-best rescoring. Since a direct interpolation is not possible between stream-based LMs and traditional word- or morpheme-based LMs, log-linear score

```
## GR-FLM1 ≡ P(W_n|W_{n-1},W_{n-2}) [standard trigram]
W : 2 W(-1) W(-2)  flm1.count.gz flm1.lm.gz 3
  W1,W2   W2 kndiscount gtmin 1 interpolate
  W1      W1 kndiscount gtmin 1 interpolate
  0       0  kndiscount gtmin 1

## GR-FLM2 ≡ P(W_n|W_{n-1},L_{n-1},I_{n-1},P_{n-1},W_{n-2},L_{n-2},I_{n-2},P_{n-2})
W : 8 W(-1) L(-1) I(-1) P(-1 ) W(-2) L(-2) I(-2) P(-2 ) flm2.count.gz flm2.lm.gz 9
  W1,L1,I1,P1,W2,L2,I2,P2   W2 kndiscount gtmin 3 interpolate
  W1,L1,I1,P1,L2,I2,P2      L2 kndiscount gtmin 3 interpolate
  W1,L1,I1,P1,I2,P2         I2 kndiscount gtmin 3 interpolate
  W1,L1,I1,P1,P2            P2 kndiscount gtmin 3 interpolate
  W1,L1,I1,P1               W1 kndiscount gtmin 1 interpolate
  L1,I1,P1                  L1 kndiscount gtmin 1 interpolate
  I1,P1                     I1 kndiscount gtmin 1 interpolate
  P1                        P1 kndiscount gtmin 1 interpolate
  0                         0  kndiscount gtmin 1

## GR-FLM3 ≡ P(W_n|W_{n-1},L_{n-1},I_{n-1},P_{n-1},W_{n-2},L_{n-2},I_{n-2},P_{n-2})
W : 8 W(-1) L(-1) I(-1) P(-1) W(-2) L(-2) I(-2) P(-2) flm3.count.gz flm3.lm.gz 11
  W1,L1,I1,P1,W2,L2,I2,P2   W2 kndiscount gtmin 3 interpolate
  W1,L1,I1,P1,L2,I2,P2      L2 kndiscount gtmin 3 interpolate
  W1,L1,I1,P1,I2,P2         I2,P2 kndiscount gtmin 1000000 combine max strategy bog_node_prob
  W1,L1,I1,P1,P2            P2 kndiscount gtmin 3 kn-count-parent W1,L1,I1,P1,W2,L2,I2,P2
  W1,L1,I1,P1,I2            I2 kndiscount gtmin 3 kn-count-parent W1,L1,I1,P1,W2,L2,I2,P2
  W1,L1,I1,P1               W1 kndiscount gtmin 1 interpolate
  L1,I1,P1                  L1 kndiscount gtmin 1 interpolate
  I1,P1                     I1,P1 kndiscount gtmin 1000000 combine max strategy bog_node_prob
  P1                        P1 kndiscount gtmin 1 kn-count-parent W1,L1,I1,P1
  I1                        I1 kndiscount gtmin 1 kn-count-parent W1,L1,I1,P1
  0                         0  kndiscount gtmin 1 kn-count-parent W1,L1,I1,P1

## GR-FLM4 ≡ P(W_n|W_{n-1},L_{n-1},I_{n-1},P_{n-1},W_{n-2},L_{n-2},I_{n-2},P_{n-2})
W : 8 W(-1) L(-1) I(-1) P(-1) W(-2) L(-2) I(-2) P(-2) flm4.count.gz flm4.lm.gz 11
  W1,L1,I1,P1,W2,L2,I2,P2   W2 kndiscount gtmin 3 interpolate
  W1,L1,I1,P1,L2,I2,P2      L2 kndiscount gtmin 3 interpolate
  W1,L1,I1,P1,I2,P2         I2,P2 kndiscount gtmin 1000000 combine mean
  W1,L1,I1,P1,P2            P2 kndiscount gtmin 3 kn-count-parent W1,L1,I1,P1,W2,L2,I2,P2
  W1,L1,I1,P1,I2            I2 kndiscount gtmin 3 kn-count-parent W1,L1,I1,P1,W2,L2,I2,P2
  W1,L1,I1,P1               W1 kndiscount gtmin 1 interpolate
  L1,I1,P1                  L1 kndiscount gtmin 1 interpolate
  I1,P1                     I1,P1 kndiscount gtmin 1000000 combine mean
  P1                        P1 kndiscount gtmin 1 kn-count-parent W1,L1,I1,P1
  I1                        I1 kndiscount gtmin 1 kn-count-parent W1,L1,I1,P1
  0                         0  kndiscount gtmin 1 kn-count-parent W1,L1,I1,P1

## GR-FLM5 ≡ P(W_n|W_{n-1},L_{n-1},P_{n-1},W_{n-2},L_{n-2},P_{n-2})
W : 6 W(-1) L(-1) P(-1) W(-2) L(-2) P(-2) flm5.count.gz flm5.lm.gz 9
  W1,L1,P1,W2,L2,P2         W2 kndiscount gtmin 3 interpolate
  W1,L1,P1,L2,P2            L2,P2 kndiscount gtmin 1000000 combine mean
  W1,L1,P1,P2               P2 kndiscount gtmin 4 kn-count-parent W1,L1,P1,W2,L2,P2
  W1,L1,P1,L2               L2 kndiscount gtmin 2 kn-count-parent W1,L1,P1,W2,L2,P2
  W1,L1,P1                  W1 kndiscount gtmin 2 interpolate
  L1,P1                     L1,P1 kndiscount gtmin 1000000 combine max strategy bog_node_prob
  P1                        P1 kndiscount gtmin 3 kn-count-parent W1,L1,P1
  L1                        L1 kndiscount gtmin 1 kn-count-parent W1,L1,P1
  0                         0  kndiscount gtmin 1 kn-count-parent W1,L1,P1

## GR-FLM6 ≡ P(W_n|W_{n-1},L_{n-1},P_{n-1},W_{n-2},L_{n-2},P_{n-2})
W : 6 W(-1) L(-1) P(-1) W(-2) L(-2) P(-2) flm6.count.gz flm6.lm.gz 9
  W1,L1,P1,W2,L2,P2         W2 kndiscount gtmin 5 interpolate
  W1,L1,P1,L2,P2            L2,P2 kndiscount gtmin 1000000 combine mean
  W1,L1,P1,P2               P2 kndiscount gtmin 6 kn-count-parent W1,L1,P1,W2,L2,P2
  W1,L1,P1,L2               L2 kndiscount gtmin 4 kn-count-parent W1,L1,P1,W2,L2,P2
  W1,L1,P1                  W1 kndiscount gtmin 4 interpolate
  L1,P1                     L1,P1 kndiscount gtmin 1000000 combine max strategy bog_node_prob
  P1                        P1 kndiscount gtmin 5 kn-count-parent W1,L1,P1
  L1                        L1 kndiscount gtmin 1 kn-count-parent W1,L1,P1
  0                         0  kndiscount gtmin 1 kn-count-parent W1,L1,P1

## GR-FLM7 ≡ P(W_n|W_{n-1},I_{n-1},P_{n-1},W_{n-2},I_{n-2},P_{n-2})
W : 6 W(-1) I(-1) P(-1) W(-2) I(-2) P(-2) flm7.count.gz flm7.lm.gz 9
  W1,I1,P1,W2,I2,P2         W2 kndiscount gtmin 3 interpolate
  W1,I1,P1,I2,P2            I2,P2 kndiscount gtmin 1000000 combine mean
  W1,I1,P1,P2               P2 kndiscount gtmin 4 kn-count-parent W1,I1,P1,W2,I2,P2
  W1,I1,P1,I2               I2 kndiscount gtmin 2 kn-count-parent W1,I1,P1,W2,I2,P2
  W1,I1,P1                  W1 kndiscount gtmin 2 interpolate
  I1,P1                     I1,P1 kndiscount gtmin 1000000 combine max strategy bog_node_prob
  P1                        P1 kndiscount gtmin 3 kn-count-parent W1,I1,P1
  I1                        I1 kndiscount gtmin 1 kn-count-parent W1,I1,P1
  0                         0  kndiscount gtmin 1 kn-count-parent W1,I1,P1
```

Figure 4.4. Topologies of the German FLMs using the format specifications of the SRILM-FLM extensions (**W**: word; **L**: lexeme; **I**: class-index; **P**: POS-tag).



(a) $GR–FLM_1$ (b) $GR–FLM_2$ (c) $GR–FLM_{3,4}$

(d) $GR–FLM_{5,6}$ (e) $GR–FLM_7$

Figure 4.5. Backoff graphs for $GR–FLM_{1:7}$, detailed topologies are given in Figure 4.4 (**W**: word; **L**: lexeme; **I**: class-index; **P**: POS-tag).

Table 4.3. Recognition WERs [%] on German corpora using different factored LMs (N-best size = 1000; **word-based**: 100k full-words, OOV rate = [gr-dev09: 5.0%, gr-eval09: 4.8%], N-best error rate (NER) = [gr-dev09: 23.6%, gr-eval09: 21.4%]; **morpheme-based**: 5k full-words + 95k morphemes, OOV rate = [gr-dev09: 1.5%, gr-eval09: 1.4%], N-best error rate (NER) = [gr-dev09: 20.0%, gr-eval09: 18.8%]).

	word-based		**morpheme-based**	
LM	**gr-dev09**	**gr-eval09**	**gr-dev09**	**gr-eval09**
before rescoring:				
4-gram	33.9	29.7	31.7	28.5
N-best rescoring:				
$GR{-}FLM_2$	34.1	29.5	31.8	28.5
$GR{-}FLM_3$	34.0	29.6	31.7	28.4
$GR{-}FLM_4$	34.0	29.6	31.7	28.5
$GR{-}FLM_5$	**33.8**	**29.4**	**31.4**	**28.4**
$GR{-}FLM_6$	33.9	29.5	31.5	28.4
$GR{-}FLM_7$	34.0	29.6	31.7	28.4

combination is used. In all cases, the results are compared to the traditional 4-gram LM lattice rescoring.

Table 4.4. Recognition WERs [%] on Arabic ar-dev07 corpus using stream- and class-based LMs built over words and morphemes (N-best size = 1000; **word-based**: 70k full-words, OOV rate = 3.7%, N-best error rate (NER) = 9.5%; **morpheme-based**: 20k full-words + 50k morphemes, OOV rate = 1.4%, N-best error rate (NER) = 8.2%).

		ar-dev07	
LM	**classes**	**word-based**	**morpheme-based**
traditional 4-gram	-	16.2	14.5
stream-based LM	lexeme	**16.0**	**14.5**
	morph	16.4	15.0
	pattern	16.6	14.8
class-based LM	lexeme	**16.1**	**14.5**
	morph	16.1	14.6
	pattern	16.2	14.9
linear interpolation: traditional 4-gram + class-based LM	lexeme	**15.8**	**14.2**
	morph	15.9	14.3
	pattern	15.9	14.3
log-linear score combination: traditional 4-gram + stream-based LM	lexeme	**15.9**	**14.4**
	morph	16.3	15.0
	pattern	16.5	14.7

Table 4.4 shows that the performance of class-based LMs is in general better than the performance of stream-based LMs. The best results are obtained using interpolated class-based LMs with traditional 4-gram LMs. Combining the scores of stream-based LMs log-linearly with the scores of traditional 4-gram LMs slightly improves the recognition performance compared to stream-based LMs alone. However, it does not succeed to outperform the interpolated class-based with traditional 4-gram LMs. Therefore, in our further experiments, class-based LMs are always interpolated with traditional 4-gram LMs.

Table 4.5 shows the results of recognition experiments performed on Arabic corpora using different word-based LMs that utilize word-level classes. The recognition system uses a vocabulary of 750k full-words. All class-based LMs are linearly interpolated with traditional word-based LMs. Both conventional and hierarchical Pitman-Yor models are used in the estimation of word- and class-based LMs. The factored LM uses the optimized topology of $AR{-}FLM_4$ shown in Section 4.7.1, Figures 4.2 and 4.3.

Table 4.5 shows that the best recognition results are obtained using an extended interpolation of word-based and class-based LMs over lexeme, morph, and pattern classes. For every LM, two different versions are involved in the linear interpolation; one is estimated using conventional modified Kneser-Ney (MKN) smoothing method, and the other is estimated based on hierarchical Pitman-Yor models. This achieves WER improvements of [ar-dev07: 0.5% absolute (3.4% relative); ar-eval07: 0.4% absolute (2.5% relative)]

Table 4.5. Recognition experiments on Arabic corpora using class-based LMs, factored LM ($AR-FLM_4$), and hierarchical Pitman-Yor LMs built over **full-words** (vocabulary: 750k full-words; OOV rate = [ar-dev07: 0.5%, ar-eval07: 0.7%]; N-best size = 1000; N-best error rate (NER) = [ar-dev07: 7.6%, ar-eval07: 9.1%]).

LM	classes	ar-dev07 WER (ins/del) [%]	ar-dev07 CER (ins/del) [%]	ar-eval07 WER (ins/del) [%]	ar-eval07 CER (ins/del) [%]
4-gram	-	**14.6** (2.7/1.5)	**6.6** (2.5/2.7)	**16.3** (2.3/1.4)	**8.7** (2.2/4.3)
+ HPYLM	-	14.5 (2.7/1.4)	6.7 (2.6/2.7)	16.2 (2.3/1.4)	8.6 (2.1/4.3)
factored LM	lexeme & morph	14.5 (2.2/2.0)	6.7 (2.3/3.1)	16.1 (1.9/1.7)	8.7 (2.0/4.6)
conventional class LM	lexeme	14.2 (2.4/1.5)	6.6 (2.3/2.9)	15.9 (2.1/1.5)	8.6 (2.0/4.6)
	morph	14.2 (2.5/1.6)	6.6 (2.3/2.9)	16.0 (2.1/1.4)	8.6 (2.0/4.5)
	pattern	14.3 (2.5/1.5)	6.6 (2.3/2.9)	16.1 (2.1/1.5)	8.6 (2.0/4.6)
	all	14.2 (2.4/1.6)	6.6 (2.3/3.0)	15.9 (2.0/1.5)	8.6 (2.0/4.6)
conventional + HPY class LM	lexeme	14.1 (2.5/1.6)	6.5 (2.3/2.9)	16.0 (2.2/1.4)	8.5 (2.0/4.4)
	morph	14.2 (2.5/1.6)	6.6 (2.3/2.9)	16.0 (2.0/1.5)	8.5 (2.0/4.6)
	pattern	14.2 (2.5/1.6)	6.6 (2.3/2.9)	16.0 (2.2/1.4)	8.5 (2.0/4.4)
	all	**14.1** (2.5/1.6)	**6.5** (2.3/2.9)	**15.9** (2.0/1.5)	**8.5** (2.0/4.5)

compared to lattice rescoring via a traditional word-based 4-gram LM based on modified Kneser-Ney smoothing. Table 4.6 shows the word- and character-level perplexities for the models listed in Table 4.5.

Table 4.6. Word- and character-level perplexities on Arabic corpora for LMs that utilize word-level classes (**inv**: perplexity for in-vocabulary text excluding the unk symbol; **all**: perplexity for the whole text including the unk symbol).

corpus	LM	word-level PPL inv (#words)	word-level PPL all (#words)	char-level PPL inv (#chars)	char-level PPL all (#chars)
ar-dev07	full-words	**502.6** (18920)	**500.8** (19002)	**3.084** (104489)	**3.075** (105142)
	factored LM	509.8	508.0	3.092	3.083
	class LM	473.3	471.9	3.051	3.043
	HPY LM	467.2	465.7	3.044	3.035
	HPY class LM	**448.3**	**447.1**	**3.021**	**3.013**
ar-eval07	full-words	**679.0** (29249)	**673.9** (29430)	**3.296** (159882)	**3.280** (161384)
	factored LM	667.2	663.2	3.286	3.270
	class LM	631.3	627.2	3.253	3.237
	HPY LM	630.3	620.2	3.252	3.230
	HPY class LM	**603.6**	**600.1**	**3.226**	**3.211**

Similarly, Table 4.7 shows the results of recognition experiments performed on Arabic corpora using different morpheme-based LMs that utilize morpheme-level classes. The recognition system uses a vocabulary of 20k full-words + 236k morphemes. All class-based LMs are linearly interpolated with traditional morpheme-based LMs. Both conventional and hierarchical Pitman-Yor models are used in the estimation of morpheme- and class-based LMs. The factored LM uses the optimized topology of $AR-FLM_4$ shown in Section 4.7.1, Figures 4.2 and 4.3.

Table 4.7 shows that the best recognition results are also obtained using the same extended interpolation of morpheme-based and class-based LMs over lexeme, morph, and pattern classes using conventional and HPY based models. WER improvements of [ar-dev07: 0.4% absolute (2.8% relative); ar-eval07: 0.5% absolute (3.1% relative)] are achieved compared to lattice rescoring via a morpheme-based 4-gram LM based on modified Kneser-Ney smoothing. Table 4.8 shows the morpheme- and character-level perplexities for the models listed in Table 4.7.

Table 4.9 reports the number of lexemes, morphs, and patterns involved in both word- and morpheme-based models.

Table 4.7. Recognition experiments on Arabic corpora using class-based LMs, factored LM ($AR-FLM_4$), and hierarchical Pitman-Yor LMs built over **morphemes** (vocabulary: 20k full-words + 236k morphemes; OOV rate = [ar-dev07: 0.5%, ar-eval07: 0.7%]; N-best size = 1000; N-best error rate (NER) = [ar-dev07: 7.6%, ar-eval07: 8.8%]).

LM	classes	ar-dev07 WER (ins/del) [%]	ar-dev07 CER (ins/del) [%]	ar-eval07 WER (ins/del) [%]	ar-eval07 CER (ins/del) [%]
4-gram	-	**14.1** (2.4/1.6)	**6.5** (2.5/2.7)	**16.1** (2.1/1.5)	**8.6** (2.1/4.4)
+ HPYLM	-	14.0 (2.3/1.5)	6.5 (2.5/2.7)	16.0 (2.1/1.5)	8.6 (2.1/4.4)
factored LM	*lexeme & morph*	13.9 (2.3/1.5)	6.5 (2.4/2.7)	15.9 (1.6/1.8)	8.6 (1.9/4.8)
conventional	lexeme	13.9 (2.0/1.8)	6.6 (2.2/3.1)	15.9 (1.6/1.8)	8.7 (1.9/4.8)
class LM	morph	13.9 (2.1/1.7)	6.6 (2.2/3.0)	16.0 (1.6/1.8)	8.7 (1.9/4.8)
	pattern	13.9 (2.0/1.8)	6.6 (2.2/3.1)	16.0 (1.6/1.8)	8.7 (1.9/4.8)
	all	13.8 (2.0/1.8)	6.6 (2.2/3.0)	15.8 (1.7/1.7)	8.6 (1.9/4.7)
conventional	lexeme	13.8 (2.0/1.8)	6.6 (2.2/3.0)	15.7 (1.6/1.8)	8.5 (1.8/4.7)
+ HPY	morph	13.9 (2.0/1.8)	6.6 (2.2/3.0)	15.7 (1.6/1.7)	8.6 (1.8/4.7)
class LM	pattern	13.9 (2.0/1.8)	6.6 (2.2/3.0)	15.7 (1.6/1.7)	8.6 (1.9/4.7)
	all	**13.7** (2.0/1.8)	**6.6** (2.2/3.0)	**15.6** (1.6/1.8)	**8.5** (1.8/4.7)

Table 4.8. Morpheme- and character-level perplexities on Arabic corpora for LMs that utilize morpheme-level classes (**inv**: perplexity for in-vocabulary text excluding the unk symbol; **all**: perplexity for the whole text including the unk symbol).

corpus	LM	morpheme-level PPL inv (#morphemes)	morpheme-level PPL all (#morphemes)	char-level PPL inv (#chars)	char-level PPL all (#chars)
ar-dev07	morphemes	**392.8** (19600)	**391.7** (19724)	**3.051** (104967)	**3.041** (105869)
	factored LM	408.3	407.2	3.073	3.064
	class LM	373.0	372.2	3.021	3.013
	HPY LM	365.9	365.0	3.011	3.002
	HPY class LM	**353.9**	**353.1**	**2.992**	**2.983**
ar-eval07	morphemes	**513.1** (30538)	**509.7** (30752)	**3.266** (161028)	**3.248** (162719)
	factored LM	518.3	515.6	3.272	3.255
	class LM	482.9	479.9	3.228	3.212
	HPY LM	478.1	475.1	3.222	3.205
	HPY class LM	**364.3**	**333.7**	**2.776**	**2.640**

Table 4.9. Number of instances of every class for Arabic vocabularies.

class	vocabulary: 750k word-based	vocabulary: 256k morpheme-based
lexeme	300,355	146,221
morph	1,794	743
pattern	98,789	37,154

4.7.3 Experiments on German

To compare the performance of stream-based LMs versus class-based LMs, Table 4.10 shows the results of recognition experiments performed on German corpora using the German testing system described in Appendix A. The system performs 2 recognition passes, each of which uses a 4-gram LM. In the second recognition pass, the system produces N-best lists *(N = 1000)* which are rescored using different LMs. The results are shown for both word-based and morpheme-based systems. The word-based system uses 100k full-words, whereas, the morpheme-based system uses 5k full-words + 95k morphemes. The configuration of the morpheme-based system is motivated by the optimization performed in Section 3.5.2. In addition, class-based LMs are linearly interpolated with a traditional word- or morpheme-based LM and used to perform the N-best rescoring. Log-linear score combination is used between stream-based LMs and traditional word- or morpheme-based LMs. In all cases, the results are compared to the traditional 4-gram LM before rescoring.

Table 4.10. Recognition WERs [%] on German corpora using stream- and class-based LMs built over words and morphemes (N-best size = 1000; **word-based**: 100k full-words, OOV rate = [gr-dev09: 5.0%, gr-eval09: 4.8%], N-best error rate (NER) = [gr-dev09: 23.6%, gr-eval09: 21.4%]; **morpheme-based**: 5k full-words + 95k morphemes, OOV rate = [gr-dev09: 1.5%, gr-eval09: 1.4%], N-best error rate (NER) = [gr-dev09: 20.0%, gr-eval09: 18.8%]).

LM	classes	word-based		morpheme-based	
		gr-dev09	**gr-eval09**	**gr-dev09**	**gr-eval09**
before rescoring: traditional 4-gram	-	33.9	29.7	31.7	28.5
stream-based LM	lexeme	34.0	29.6	31.8	28.4
	POS-tag	34.0	29.6	31.8	28.4
	cluster-index	34.1	29.6	31.8	28.5
class-based LM	lexeme	33.9	29.5	31.6	28.3
	POS-tag	33.9	29.5	31.6	28.4
	cluster-index	34.0	29.6	31.7	28.5
linear interpolation: traditional 4-gram + class-based LM	lexeme	33.7	29.4	31.6	28.2
	POS-tag	**33.6**	**29.3**	**31.3**	**28.0**
	cluster-index	33.9	29.5	31.3	28.0
log-linear score combination: traditional 4-gram + stream-based LM	lexeme	33.9	29.5	31.8	28.3
	POS-tag	33.9	29.5	31.7	28.3
	cluster-index	34.0	29.6	31.7	28.4

Going in the same line with Arabic experiments, Table 4.10 shows that the performance of class-based LMs is in general better than the performance of stream-based LMs. The best performance is obtained using interpolated class-based LMs with traditional 4-gram LMs. However, combining the scores of stream-based LMs with the scores of traditional 4-gram LMs does not succeed to outperform the interpolated class-based with traditional 4-gram LMs. Therefore, in our further experiments, class-based LMs are always interpolated with traditional 4-gram LMs.

Table 4.11 shows the results of recognition experiments performed on German corpora using different word-based LMs that utilize word-level classes. The recognition system uses a vocabulary of 750k full-words. All class-based LMs are linearly interpolated with traditional word-based LMs. Both conventional and hierarchical Pitman-Yor models are used in the estimation of word- and class-based LMs. The factored LM uses the optimized topology of $GR{-}FLM_5$ shown in Section 4.7.1, Figures 4.4 and 4.5.

Table 4.11 shows that the best recognition results are obtained using an extended interpolation of word-based and class-based LMs over lexeme, POS-tag, and cluster-index classes. For every LM, two different versions are involved in the linear interpolation; one is estimated using conventional modified Kneser-Ney (MKN) smoothing, and the other is estimated based on hierarchical Pitman-Yor models. This achieves WER improvements of [gr-dev09: 0.5% absolute (1.6% relative); gr-eval09: 0.6% absolute (2.2% relative)] compared to the traditional word-based 4-gram LM. Table 4.12 shows the word- and character-level perplexities for the models listed in Table 4.11.

Table 4.11. Recognition experiments on German corpora using class-based LMs, factored LM ($GR-FLM_5$), and hierarchical Pitman-Yor LMs built over **full-words** (vocabulary: 750k full-words; OOV rate = [gr-dev09: 2.3%, gr-eval09: 2.1%]; N-best size = 1000; N-best error rate (NER) = [gr-dev09: 20.6%, gr-eval09: 18.9%]).

		gr-dev09		**gr-eval09**	
LM	**classes**	**WER** (ins/del) [%]	**CER** (ins/del) [%]	**WER** (ins/del) [%]	**CER** (ins/del) [%]
4-gram	-	**31.3** (**4.6/6.0**)	**14.3** (**3.5/5.9**)	**27.4** (**3.2/6.5**)	**12.8** (**2.7/5.9**)
+ HPYLM	-	31.0 (4.7/5.9)	14.13 (3.5/5.8)	27.1 (3.2/6.2)	12.6 (2.7/5.7)
factored LM	lexeme & POS-tag	31.2 (4.5/6.1)	14.2 (3.5/6.0)	27.2 (3.2/6.3)	12.6 (2.7/5.7)
conventional	lexeme	31.3 (4.6/6.0)	14.2 (3.5/5.9)	27.1 (3.4/5.9)	12.4 (2.7/5.5)
class LM	POS-tag	31.2 (4.6/6.0)	14.2 (3.5/6.0)	27.1 (3.3/6.0)	12.4 (2.7/5.6)
	cluster-index	31.3 (4.4/6.3)	14.3 (3.4/6.1)	27.1 (3.2/6.1)	12.5 (2.7/5.7)
	all	31.1 (4.4/6.3)	14.2 (3.3/6.1)	27.0 (3.2/6.2)	12.5 (2.6/5.7)
conventional	lexeme	31.0 (4.6/5.9)	14.1 (3.5/5.8)	26.9 (3.3/5.9)	12.4 (2.7/5.5)
+ HPY	POS-tag	30.8 (4.6/5.8)	14.0 (3.5/5.8)	26.9 (3.3/5.9)	12.4 (2.7/5.5)
class LM	cluster-index	30.9 (4.6/5.9)	14.1 (3.5/5.8)	26.9 (3.3/5.9)	12.4 (2.7/5.5)
	all	**30.8** (**4.4/6.0**)	**14.1** (**3.3/6.0**)	**26.8** (**3.2/6.0**)	**12.4** (**2.7/5.5**)

Table 4.12. Word- and character-level perplexities on German corpora for LMs that utilize word-level classes (**inv**: perplexity for in-vocabulary text excluding the unk symbol; **all**: perplexity for the whole text including the unk symbol).

corpus	**LM**	**word-level PPL**		**char-level PPL**	
		inv (#words)	all (#words)	inv (#chars)	all (#chars)
gr-dev09	full-words	**509.0** (**69548**)	**490.4** (**71133**)	**2.818** (**418447**)	**2.725** (**439560**)
	factored LM	512.3	495.1	2.821	2.729
	class LM	491.9	475.3	2.802	2.712
	HPY LM	490.9	472.7	2.801	2.709
	HPY class LM	**479.0**	**462.3**	**2.789**	**2.699**
gr-eval09	full-words	**520.0** (**35591**)	**503.3** (**36319**)	**2.793** (**216684**)	**2.713** (**226395**)
	factored LM	527.5	512.0	2.800	2.720
	class LM	502.6	487.8	2.778	2.699
	HPY LM	502.4	486.1	2.778	2.698
	HPY class LM	**490.4**	**475.5**	**2.766**	**2.688**

Table 4.13 shows the results of recognition experiments performed on German corpora using different morpheme-based LMs that utilize morpheme-level classes. The recognition system uses a vocabulary of 5k full-words + 495k morphemes. All class-based LMs are linearly interpolated with traditional morpheme-based LMs. Both conventional and hierarchical Pitman-Yor models are used in the estimation of morpheme- and class-based LMs. The factored LM uses the optimized topology of $GR\text{–}FLM_5$ shown in Section 4.7.1, Figures 4.4 and 4.5.

Table 4.13. Recognition experiments on German corpora using class-based LMs, factored LM ($GR\text{–}FLM_5$), and hierarchical Pitman-Yor LMs built over **morphemes** (vocabulary: 5k full-words + 495k morphemes; OOV rate = [gr-dev09: 0.9%, gr-eval09: 0.7%]; N-best size = 1000; N-best error rate (NER) = [gr-dev09: 19.1%, gr-eval09: 17.3%]).

LM	classes	gr-dev09 WER (ins/del) [%]	gr-dev09 CER (ins/del) [%]	gr-eval09 WER (ins/del) [%]	gr-eval09 CER (ins/del) [%]
4-gram	-	**31.0** (4.4/5.8)	**14.2** (3.5/5.8)	**27.2** (3.1/6.1)	**12.5** (2.7/5.6)
+ HPYLM	-	30.8 (4.5/5.6)	14.1 (3.2/5.7)	27.0 (3.2/6.0)	12.4 (2.7/5.6)
factored LM	lexeme & POS-tag	30.8 (4.5/5.6)	14.1 (3.2/5.7)	27.0 (3.2/6.0)	12.4 (2.7/5.6)
conventional	lexeme	30.9 (4.10/6.36)	14.3 (3.3/6.2)	27.2 (3.1/6.1)	12.5 (2.7/5.6)
class LM	POS-tag	30.8 (4.16/6.13)	14.2 (3.4/6.0)	27.1 (3.1/6.0)	12.4 (2.7/5.5)
	cluster-index	30.8 (4.04/6.35)	14.3 (3.3/6.2)	27.2 (3.1/6.2)	12.5 (2.7/5.5)
	all	30.8 (4.15/6.13)	14.2 (3.3/6.0)	27.1 (3.1/5.9)	12.4 (2.7/5.5)
conventional	lexeme	30.7 (4.3/6.0)	14.1 (3.4/5.9)	26.9 (3.1/6.0)	12.4 (2.7/5.6)
+ HPY	POS-tag	30.7 (4.2/6.2)	14.2 (3.3/6.1)	26.9 (3.1/6.0)	12.4 (2.6/5.6)
class LM	cluster-index	30.7 (4.3/5.9)	14.1 (3.4/5.9)	26.9 (3.2/6.0)	12.4 (2.6/5.6)
	all	**30.6** (4.1/6.2)	**14.1** (3.3/6.1)	**26.9** (3.1/6.0)	**12.4** (2.6/5.6)

Table 4.13 shows that the best recognition results are also obtained using the same extended interpolation of morpheme-based and class-based LMs over lexeme, POS-tag, and cluster-index classes using conventional and HPY based models. WER improvements of [gr-dev09: 0.4% absolute (1.3% relative); gr-eval09: 0.3% absolute (1.1% relative)] are achieved compared to the morpheme-based 4-gram LM based on modified Kneser-Ney smoothing. Table 4.14 shows the morpheme- and character-level perplexities for the models listed in Table 4.13.

Table 4.14. Morpheme- and character-level perplexities on German corpora for LMs that utilize morpheme-level classes (**inv**: perplexity for in-vocabulary text excluding the unk symbol; **all**: perplexity for the whole text including the unk symbol).

corpus	LM	morpheme-level PPL inv (#morphemes)	morpheme-level PPL all (#morphemes)	char-level PPL inv (#chars)	char-level PPL all (#chars)
gr-dev09	morphemes	**403.9** (72391)	**393.2** (73906)	**2.799** (422086)	**2.713** (442333)
	factored LM	409.2	399.5	2.805	2.721
	class LM	395.7	385.8	2.789	2.705
	HPY LM	389.6	379.1	2.782	2.697
	HPY class LM	**383.7**	**373.9**	**2.774**	**2.691**
gr-eval09	morphemes	**403.0** (37151)	**393.8** (37845)	**2.772** (218582)	**2.697** (227921)
	factored LM	412.3	403.7	2.783	2.708
	class LM	395.8	387.2	2.764	2.690
	HPY LM	390.4	381.3	2.757	2.683
	HPY class LM	**385.1**	**376.5**	**2.751**	**2.677**

Table 4.15 reports the number of lexemes, POS-tags, and cluster-indexes involved in both word- and morpheme-based models.

Table 4.15. Number of instances of every class for German vocabularies.

class	vocabulary	
	750k word-based	**500k morpheme-based**
lexeme	550,172	336,347
POS-tag	51	51
cluster-index	250	250

4.7.4 Overview of Experimental Results

From the experimental results shown in the previous sections, it is noted that each one of the introduced approaches has its own positive influence in reducing the recognition WERs. Figure 4.6 presents a comparative summary of the best obtained WERs on Arabic and German corpora using: (1) full-word LMs, (2) morpheme-based LMs, (3) morpheme-based factored LMs, (4) morpheme-based class LMs, and (5) morpheme-based HPY class LMs. The observed WER improvements are found statistically significant using a bootstrap method of significance analysis described in [Bisani & Ney 2004], the probability of improvement (POI_{boot}) ranges between 93% and 97%.

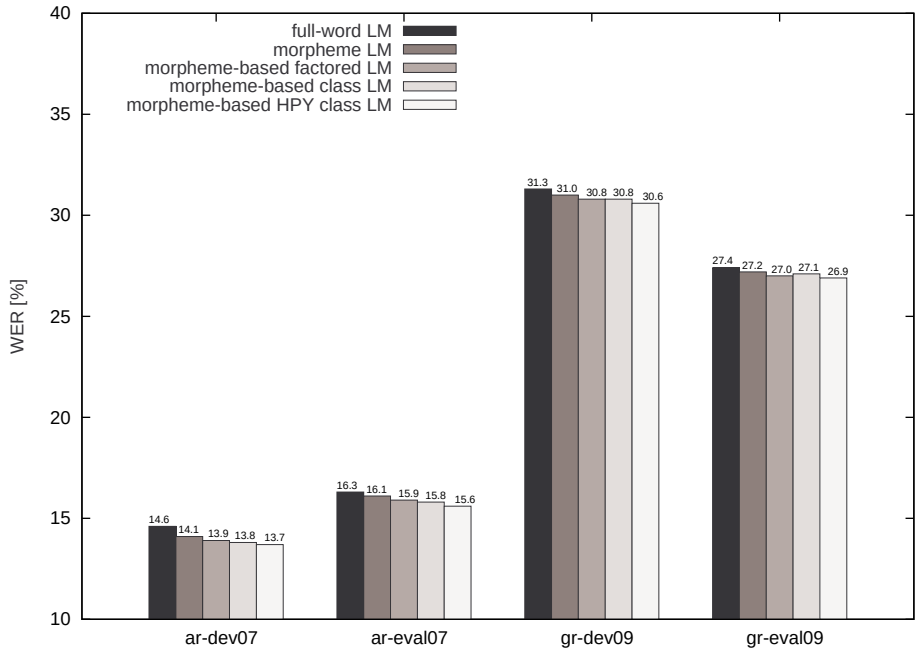
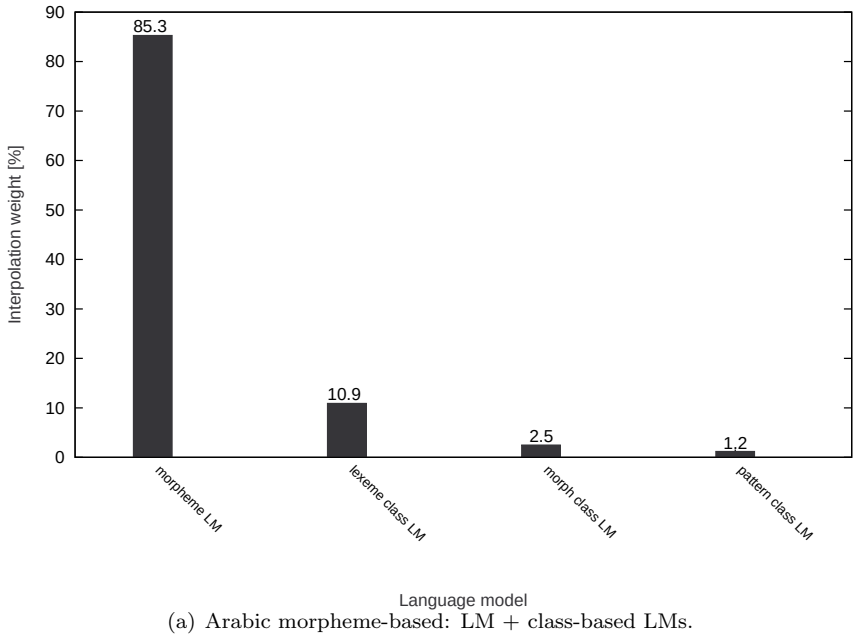
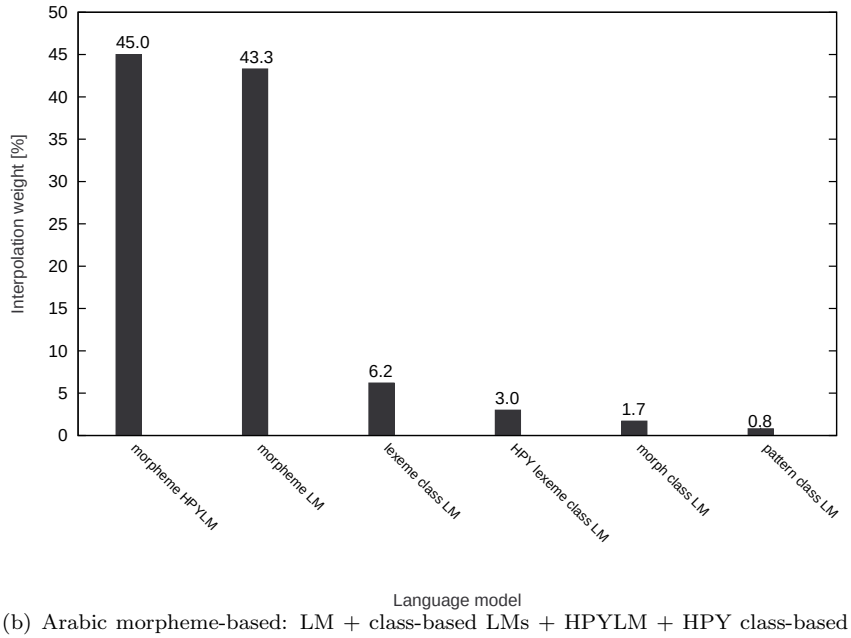


Figure 4.6. Comparison of recognition WERs [%] on Arabic and German corpora using different LMs.

Interpolation weights. In Figure 4.6, the last two bars for every corpus represent the WERs for several models interpolated together. To clarify the relative importance of the individual LMs during linear interpolation, we report the interpolation weights assigned to every LM. Figures 4.7, 4.8 give the interpolation weights as percent values for Arabic and German models. Here, it should be recalled that the interpolation weights are optimized over the development corpus in every case.

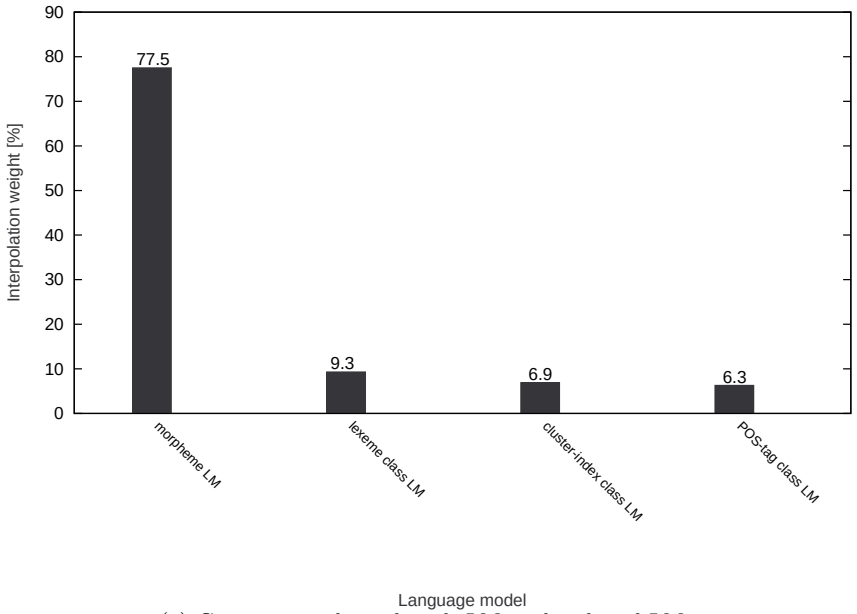


(a) Arabic morpheme-based: LM + class-based LMs.

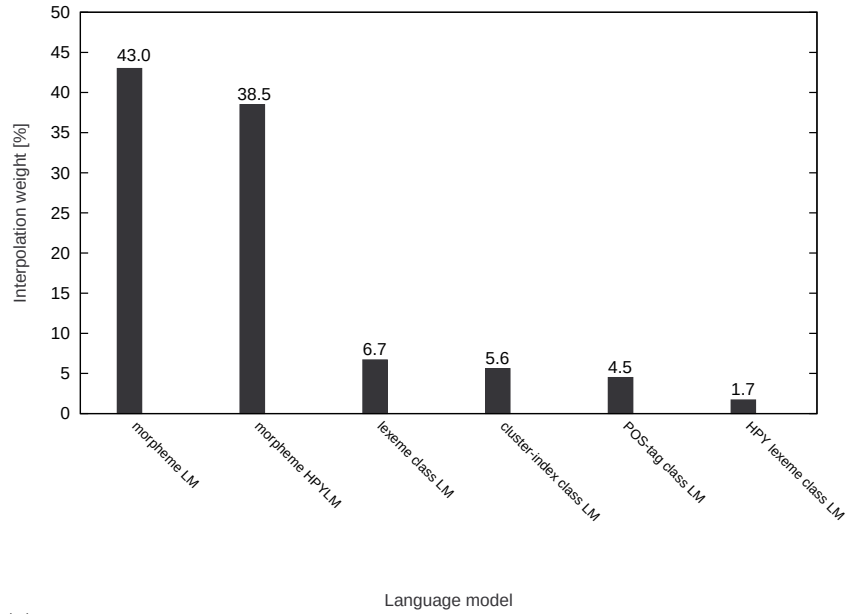


(b) Arabic morpheme-based: LM + class-based LMs + HPYLM + HPY class-based LMs.

Figure 4.7. Interpolation weights of individual Arabic morpheme-based LMs, models with negligible weights are not shown in the figure.



(a) German morpheme-based: LM + class-based LMs.



(b) German morpheme-based: LM + class-based LMs + HPYLM + HPY class-based LMs.

Figure 4.8. Interpolation weights of individual German morpheme-based LMs, models with negligible weights are not shown in the figure.

4.8 Summary

In this chapter, incorporation of morphology-based classes into the LM estimation process has been investigated in order to cope with morphologically rich languages. A novel approach using morpheme-level classes has been proposed instead of the traditional word-level classes. This enables the utilization of classes on top of morpheme-based LVCSR systems. This is a novel combination that preserves the advantages of the morpheme-based modeling along with the benefits of using classes. Morpheme-based models achieve better lexical coverage and reduce the influence of the data sparsity. Whereas, classes help to achieve better generalization to unseen word sequences.

Stream-based, class-based, and factored LMs have been experimented and compared. In addition, the use of hierarchical Pitman-Yor LMs has been investigated to estimate word- and class-based LMs. Experiments have been conducted on Arabic and German tasks using the state-of-the-art LVCSR systems with very large vocabularies that are heavily optimized based on the work of the previous chapter.

Different types of classes have been utilized using MADA Arabic morphological analyzer, Sebawai Arabic root generator, and the German TreeTagger. In addition, a novel word class based on a data-driven word clustering algorithm has been proposed for the German experiments. Experimental results have shown that morpheme-level classes can be used as efficiently as word-level classes.

Experiments on factored LMs have shown that the parameter optimization is a crucial issue. Although the factored LM is a quite general and powerful model, it is very difficult to optimize its parameters to achieve the most efficient performance in a given task. The huge space of the model parameters, including the parent factors; the backoff graph; and the smoothing options, turns the parameter optimization into a complex and time consuming process. The optimum parameters are always heavily dependent on the data and the available classes. Even if automatic search tools are used, it is not guaranteed to come up with the best factored LM topology. Experiments have shown that an interpolation of different class-based LMs built separately over different classes together with a traditional m-gram LM could easily beat a carefully optimized factored LM that uses a complex backoff scheme with the same set of classes.

Experiments on hierarchical Pitman-Yor class-based LMs have shown little improvements in the recognition performance compared to the conventional class-based LMs. However, the obtained improvements are quite persistent and systematically consistent over all the test corpora.

The best recognition results have been obtained using an extended interpolation of word-based and class-based LMs over all the available classes. For every LM, two different versions have been involved in the linear interpolation; one is estimated using conventional modified Kneser-Ney (MKN) smoothing method, and the other is estimated based on hierarchical Pitman-Yor models.

Chapter 5

Deep Neural Network Language Models

One major disadvantage of the backoff m-gram LM is its poor modeling performance in cases of data sparseness. Even when large training corpora are used, still extremely small probabilities can be assigned to many valid m-grams. So far, discrete space LMs that utilize word classes, like class-based LMs and factored LMs, have introduced a partial solution to this problem by utilizing additional word classes in cases of data sparseness. Moreover, the sub-word based LMs have introduced an additional contribution to the solution of this problem by breaking down the words into smaller and more frequent sub-word units, like morphemes or syllables. However, still a decisive solution to the data sparsity problem is not reached. Nevertheless, a better solution to this problem could be found when dealing with the original sources of the problem which are: the method of word representation, and the fundamental construction of the parameter space during the LM estimation.

In fact, the failure to deal with the sparse data domains is an inherent disadvantage of all the LMs that are estimated in a discrete space. The discrete nature of such models makes it difficult to achieve high levels of generalization even after applying the most efficient smoothing techniques, like the modified Kneser-Ney (MKN) smoothing of the backoff m-gram models [Chen & Goodman 1996]. Here, the essential issue is the lack of a notion of similarity among words. Since words are represented in a discrete space (the vocabulary space), it is not possible to perform a true interpolation [1] to approximate the probabilities of the unseen m-grams. For these reasons, going to a continuous representation space is considered a natural promotion of the LM probability estimation.

The basic idea behind the continuous word representation [Bengio & Ducharme 2001; Sarikaya & Afify$^+$ 2009; Schwenk 2007; Schwenk & Gauvain 2005] is to convert the numerical indexes of the vocabulary words into a continuous representation space and to use a probability estimator operating in this space. Since the resulting distributions are smooth functions of the word representation, better generalization to unknown m-grams is expected. The probability estimation and the interpolation in a continuous space are mathematically well understood and numerous powerful algorithms are available that can perform reliable interpolations even when a limited amount of training data is available.

A common type of continuous space LM is the shallow neural network LM (SNNLM) which estimates probabilities of word sequences in a continuous space using a single hidden layer (shallow) feed-forward neural network [Bengio & Ducharme$^+$ 2003; Schwenk 2007; Schwenk & Gauvain 2005]. Recently, deep neural networks (DNNs) with multiple hidden layers have shown the capability to capture higher-level abstract information that are more discriminative to the input features. They have been shown to provide improved performance compared to shallow networks in different tasks [Arisoy & Sainath$^+$ 2012; Bengio 2009; Dahl & Ranzato$^+$ 2010; Mohamed & Dahl$^+$ 2009; Seide & Li$^+$ 2011].

In this chapter, we investigate the use of deep neural network language models (DNNLMs) to estimate sub-word based LMs, namely morpheme-based LMs. Moreover, the input of the DNNs is augmented with morphology-based classes derived on both word and morpheme levels in order to estimate robust models for morphologically rich languages. To reach the highest level of performance, DNNLMs are always interpolated with backoff m-gram LMs. This is a novel approach that combines the advantages of using morpheme-based LMs and morphology-based classes along with the modeling capabilities of the DNNs. The methods developed in this chapter are evaluated on an Egyptian Arabic conversational telephone speech recognition task. These methods have been introduced in [El-Desoky & Kuo$^+$ 2013].

Section 5.1 presents an overview of the state-of-the-art continuous space language modeling techniques. Section 5.2 describes the architectures of the feed-forward neural network based LMs including shallow

[1] Interpolation is a method of constructing new data points within the range of a discrete set of already known data points.

and deep multilayer neural network LMs. In addition, it shows how this type of neural network LM can be enriched with additional morphology-based classes. Section 5.3 gives a description of the back-propagation training algorithm. Section 5.4 briefly describes the state-of-the-art pre-training techniques essentially related to the efficient training of deep neural network architectures. Section 5.5 introduces different common approaches used to speed up the training and recognition processes when a neural network LM is utilized in a typical LVCSR system. Section 5.6 presents some necessary preparations for performing our recognition experiments including the techniques of word decomposition and class derivation for Egyptian Arabic text. Section 5.7 presents the experimental results, and Section 5.8 summarizes the chapter.

5.1 Continuous Space Language Models: An Overview

The idea of estimating the LM probabilities in a continuous space has been explored in several previous researches [Afify & Siohan+ 2007; Arisoy & Sainath+ 2012; Bengio & Ducharme+ 2003; Mikolov & Karafiát+ 2010; Sarikaya & Afify+ 2009; Sarikaya & Emami+ 2010; Schwenk 2007] using various types of probability estimators. The research effort in this direction was started by [Nakamura & Shikano 1989], where a neural network estimator is used to predict word categories. In [Miikkulainen & Dyer 1991], hierarchical modular subnetworks are utilized for natural language processing. In [Xu & Rudnicky 2000], a simple feed-forward neural network with one input word and no hidden layers is used to estimate bigram probabilities. In [Castro & Prat; Castro-Bleda & Polvoreda+ 2001], bigram and trigram probabilities are estimated using multilayer feed-forward neural networks well known as multilayer perceptrons (MLPs).

Shallow neural network LM. In [Bengio & Ducharme 2001; Bengio & Sénécal 2003; Bengio & Ducharme+ 2003], a shallow neural network LM (SNNLM) that uses one hidden layer is developed and used to estimate up to 5-gram LM probabilities. The resulting models have achieved significant reductions in perplexities (PPLs). However, no trial was made in this work to test the models in speech recognition tasks. The first application of a SNNLM to LVCSR was done by [Schwenk 2007; Schwenk & Gauvain 2005] on the DARPA HUB5 conversational telephone speech recognition task. The proposed neural network uses a feed-forward architecture with only one hidden layer. This type of a single hidden layer neural network is called a shallow neural network (SNN), where the word *shallow* is used to distinguish it from other deeper architectures. The input of the neural network is simply the $m-1$ previous words and the output is the conditional probabilities of all the vocabulary words given the input history. The training of the network takes place using a so-called *back-propagation algorithm* [Rumelhart & Hinton+ 1986] with gradient descent optimization. In [Schwenk 2007], several highly efficient training and lattice rescoring algorithms were introduced. The experimental results have shown significant reductions in both PPLs and WERs.

Deep neural network LM. Recently, feed-forward deep neural networks (DNNs) with multiple hidden layers have been found to achieve improved performance across various tasks [Bengio 2009; Dahl & Ranzato+ 2010; Mohamed & Dahl+ 2009] compared to the feed-forward shallow neural networks with only a single hidden layer. A major success have been reported in the application of the DNNs to the acoustic modeling problem of the ASR systems [Mohamed & Dahl+ 2009; Sainath & Kingsbury+ 2012]. The essential reason behind the success of the DNN models is their ability to capture higher-level and abstract information about the input features at the upper layers of the network. Normally, the training of the DNN is performed using the same back-propagation algorithm [Rumelhart & Hinton+ 1986] as in the case of shallow architectures. However, having these many number of hidden layers in the DNN turns the training process into a hard task. The standard learning strategy consisting of randomly initializing the weights of the network and applying gradient descent using back-propagation can easily get stuck in poor local minima or plateaus of the non-convex training criterion [Auer & Herbster+ 1996; Larochelle & Bengio+ 2009]. The breakthrough to effective training strategies for DNNs came in [Hinton & Osindero+ 2006] with the application of a greedy layer-wise unsupervised pre-training followed by a supervised fine-tuning. The unsupervised pre-training strategies may rely on generative models like the restricted Boltzmann machines (RBM) [Hinton & Osindero+ 2006] or on encoding models like the nonlinear auto-encoder (or auto-associator) neural networks (AENNs) [Saund 1989]. Alternatively, in [Seide & Li+ 2011], a supervised discriminative pre-training strategy is introduced that is also greedy and layer-wise. A recent trial to use

the DNNs for estimating LMs is made in [Arisoy & Sainath+ 2012]. Significant reductions in both PPLs and WERs are reported over a small Wall Street Journal (WSJ) English ASR task.

Recurrent neural network LM. In [Kombrink & Mikolov+ 2011; Mikolov 2012; Mikolov & Karafiát+ 2010], a different type of continuous space LM was proposed that uses a recurrent neural network (RNN). Thus, the model is called a recurrent neural network LM (RNNLM). The main difference between the feed-forward and the recurrent architecture is the representation of the history. While in the feed-forward neural network LM, the history is just the $m-1$ previous words, for the RNNLM, an effective representation of history is learned from the data during training. The hidden layer of the RNN represents all the previous history and not only the $m-1$ previous words. In fact, the RNN receives as input the direct predecessor word, and the hidden layer contains recurrent connections by which it implicitly takes into account multiple predecessor words that were presented previously to the network [Sundermeyer & Oparin+ 2013]. Thereby, the model can theoretically represent long contextual patterns found in the training text. This type of network is trained using a so-called *back-propagation through time* [Rumelhart & Hinton+ 1986]. Here, it is worth mentioning that the RNN can be viewed as a folded version of a DNN, where the multiple hidden layers of the DNN have tied weights and folded together in one recurrent layer with self looping input. In an opposite viewpoint, the DNN can be viewed as an unfolded simplification of the RNN with different non-tied weights at every hidden layer.

Long short-term memory neural network LM. The training of the RNNs using the back-propagation through time [Rumelhart & Hinton+ 1986] is a difficult task. The main difficulty lies in the well-known vanishing gradient problem [Bengio & Simard+ 1994]. The gradient that is propagated back through the network either decays or grows exponentially. One approach to improve the training of the RNNs is to use better optimization algorithms like the Hessian-free optimization [Martens & Sutskever 2011]. However, this usually leads to a significant increase in the computational complexity. An alternative type of RNN that is recently used to estimate continuous space LMs is called the long short-term memory (LSTM) RNN [Gers 2001; Graves & Schmidhuber 2005; Hochreiter & Schmidhuber 1997]. The architecture of this network is designed such that the vanishing gradient problem is avoided without a need to change the training algorithm. Normally, when propagating back the gradient of the error function of the RNN through a unit of the network, it gets scaled by a certain factor which leads to an exponential decay or growth of the gradient over time. Thus, the gradient either dominates the next weight adaptation step or effectively gets lost. To avoid this scaling effect, the unit of the RNN is re-designed in the LSTM in such a way that the scaling factor is fixed to one. Since the new unit type has a limited learning capability, it is enriched by several so-called gating units. Moreover, some modifications of the original LSTM unit are proposed in [Gers & Schmidhuber+ 1999; Gers & Schraudolph+ 2002]. Recent researches have shown improvements in WERs and PPLs using LSTMLM compared the other types of networks [Sundermeyer & Schlüter+ 2012; Sundermeyer & Oparin+ 2013].

Tied-mixture LM. Another different and recent approach in continuous space language modeling is investigated in [Afify & Siohan+ 2007; Sarikaya & Afify+ 2009; Sarikaya & Emami+ 2010], where Gaussian mixture models (GMMs) and hidden Markov models (HMMs) are used as the probability estimators in the continuous space. In [Afify & Siohan+ 2007], a set of GMMs are used to estimate the LM probabilities; the resulting model is called *a Gaussian mixture LM (GMLM)*. Later, this model was improved in [Sarikaya & Afify+ 2009] using a set of HMMs that are based on a tied Gaussian mixture model to perform the probability estimation; the resulting model is called *a tied-mixture LM (TMLM)*. In this approach, the discrete words are projected into a continuous space to obtain a set of real valued vectors using a bigram co-occurrence matrix and singular value decomposition (SVD) [Bellegarda 2000; Sarikaya & Afify+ 2009]. The resulting word vectors are called *feature vectors*. Then, the history feature vectors are used to estimate the tied-mixture distribution. Additionally, in [Sarikaya & Emami+ 2010], a trial was made to estimate a TMLM using input feature vectors extracted from the projection layer of a corresponding SNNLM. These approaches are evaluated on a Chinese real-time speech-to-speech translation task. The TMLM with SNNLM based input features (TMLM-NN) outperformed the SNNLM. In addition, the TMLM with bigram co-occurrence based features (TMLM-CO) provided further improvement over the TMLM-NN.

Investigating deep neural network LMs. Up to this moment, the question of which technique of all the above ones is the best selection to estimate the LM probabilities seems to be still an open question. However, although the use of deep neural networks (DNNs) has achieved a great success in application to the acoustic modeling problem of speech recognition, so far, there has been very few researches on applying this type of neural network to the language modeling problem. To the best of our knowledge, until the moment of writing this thesis, only in [Arisoy & Sainath$^+$ 2012], a trial has been made on using DNNLMs for the Wall Street Journal (WSJ) English ASR task, which is a rather small task compared to the real world LVCSR tasks. Therefore, a decision was made in this thesis to investigate the use of DNNLMs to tackle the problem of LVCSR for morphologically rich languages. In addition, more attention is given to the combination of DNNLMs with sub-word based LMs and morphology-based classes.

5.2 Feed-Forward Neural Network Language Models

In this section, we describe different architectures of multilayer feed-forward neural network LMs. This includes shallow and deep neural network LMs. Moreover, we show how to use multilayer feed-forward neural networks to estimate morpheme-based LMs enriched with morphology-based classes to deal with LVCSR tasks of morphologically rich languages.

5.2.1 Shallow Neural Network Language Model

As previously mentioned in Section 5.1, a SNNLM uses a single hidden layer feed-forward neural network that maps words into a continuous space and predicts the probability of a word given the continuous representations of the preceding words of the history. The projection of words into a continuous space is done jointly with the neural network training in a single process. This ensures the learning of the most suitable projection matrix that best fits the probability estimation task. Thereby, words that are semantically or grammatically related are hopefully mapped to similar locations in the continuous space. Thus, the similarity measure among words is defined as being close in the multi-dimensional feature space. The probability estimates are smooth functions of the continuous word representations, a small change in the input features leads to a small change in the probability estimation. This gives the model a built-in smoothing capability that enables it to achieve better generalization to unseen m-grams.

Figure 5.1 shows the architecture of a standard SNNLM. Assuming that the vocabulary size is N, each vocabulary word is represented by a binary N dimensional indication vector having a value of *one* at the index of that word and *zero* elsewhere. This is usually called the 1-of-N encoding. For an m-gram LM, to estimate the probability of a word w_n given the previous $m-1$ history words h_n, where $h_n = w_{n-m+1}^{n-1}$, the input is given to the neural network as the concatenation of all the indication vectors of these $m-1$ history words. Then, a linear projection layer of size P is used to map each word to its continuous representation as a point in the R^P space. This layer uses a linear activation function. The projection matrix is tied for all the $m-1$ history words. The used encoding scheme simplifies the calculation of the projection layer output since it is only needed to copy one row of the $N \times P$ dimensional projection matrix. The $m-1$ continuous feature vectors of the history words, having P values each, are concatenated together to form the input of the hidden layer. Thus, the $m-1$ history gram is now represented as a point in the $R^{(m-1)P}$ space and given as input to the hidden layer. This hidden layer has H hidden units with hyperbolic tangent activation function. This is followed by an output layer with N target units that use the softmax function to produce the posterior probabilities $p(w_n = i|h_n)$. These posteriors make up the LM probabilities of each word in the vocabulary given a specific history h_n.

Let the linear activities of the projection layer be $\mathbf{c} = [c_l]$ with $l = 1, ..., (m-1)P$, $\mathbf{U} = [u_{jl}]$ is the weight matrix between the projection and the hidden layer, $\mathbf{d} = [d_j]$ with $j = 1, ..., H$ are the outputs of the hidden layer, $\mathbf{V} = [v_{ij}]$ is the weight matrix between the hidden and the output layer, $\mathbf{o} = [o_i]$ with $i = 1, ..., N$ are the values at the output layer before applying the activation function, $\mathbf{p} = [p_i]$ with $i = 1, ..., N$ are the outputs of the network after applying the final softmax activation function, b_j and k_i are the biases of the hidden and the output layers respectively, then the operations performed at every layer of the neural network are given by the following set of equations:

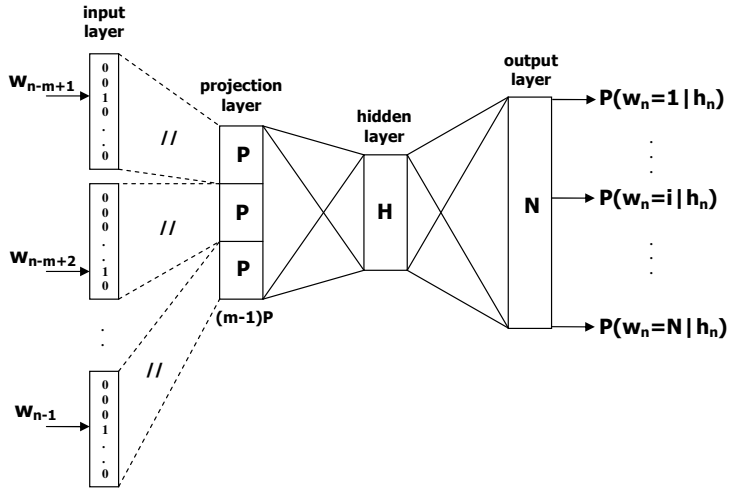


Figure 5.1. Architecture of a shallow NNLM (SNNLM) that estimates the model $p(w_n|w_{n-m+1}^{n-1})$.

$$d_j = tanh\left(\sum_{l=1}^{(m-1)P} u_{jl}c_l + b_j\right) \qquad \forall j = 1,...,H \tag{5.1}$$

$$o_i = \sum_{j=1}^{H} v_{ij}d_j + k_i \qquad \forall i = 1,...,N \tag{5.2}$$

$$p_i = \frac{e^{o_i}}{\sum_{r=1}^{N} e^{o_r}} = p(w_n = i|h_n) \qquad \forall i = 1,...,N \tag{5.3}$$

The importance of the bias terms b_j and k_i used in Equations 5.1 and 5.2 respectively is to allow the neural network to shift the activation function to the left or right, which may be critical to learn complex functions successfully. From the above operations, we can see that the neural network predicts the LM probabilities of all the vocabulary words simultaneously given the input history words. In principle, this model is able to predict the probability of any word given any history, where the history words are viewed together as an interpolated point in the $R^{(m-1)P}$ space. The softmax normalization of Equation 5.3 guarantees the generation of a normalized probability distribution from the neural network. In fact, performing this normalization for each LM probability is very time consuming due to the large size of the output layer that is usually needed in practice, therefore, in [Schwenk 2007], several techniques are developed to enable the usage of this model for the LVCSR tasks. The neural network training will be discussed in Section 5.3. Whereas, the speeding up techniques will be discussed in Section 5.5.

Using matrix/vector notation, the above equations can be rewritten as [Schwenk 2007]:

$$\mathbf{d} = tanh(\mathbf{U} \times \mathbf{c} + \mathbf{b}) \tag{5.4}$$

$$\mathbf{o} = \mathbf{V} \times \mathbf{d} + \mathbf{k} \tag{5.5}$$

$$\mathbf{p} = \frac{exp(\mathbf{o})}{\sum_{r=1}^{N} e^{o_r}} \tag{5.6}$$

where matrices are denoted by upper case bold letters, and vectors are denoted by lower case bold letters. The real functions *tanh* and *exp*, together with the division operation are performed element-wise.

5.2.2 Deep Neural Network Language Model

Figure 5.2 shows the architecture the DNNLM. It is similar to the standard SNNLM, however it employs several hidden layers of nonlinearities instead of a single one. The number of hidden layers is L and every hidden layer has H hidden units with hyperbolic tangent activation function. The operations of Equation 5.1 are repeated for every hidden layer in the network in a cascaded manner such that every layer gives its output to the input of the next one. The main concept behind this architecture is that every hidden layer is supposed to learn a nonlinear transformation that captures the main variations in its input. The level of abstraction increases gradually from the lower to the upper layer. At the topmost layer of the network, the most complex and abstract information is well captured.

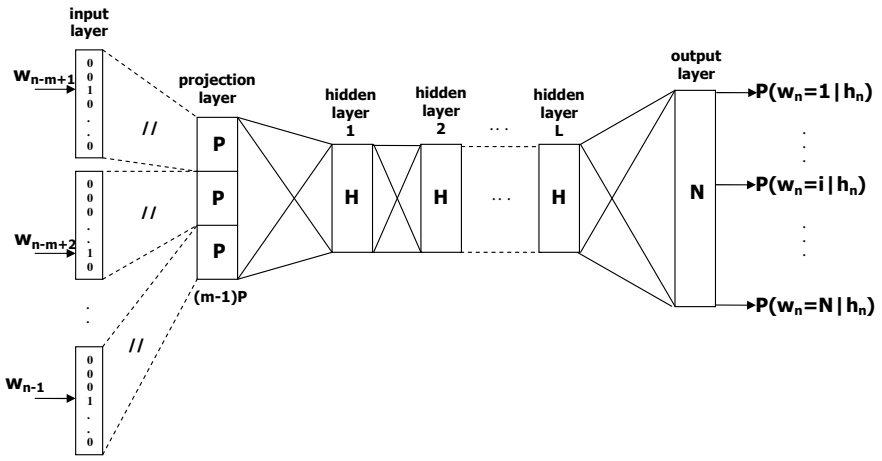


Figure 5.2. Architecture of a deep NNLM (DNNLM) that estimates the model $p(w_n|w_{n-m+1}^{n-1})$.

5.2.3 Deep Neural Network Language Model with Classes

To enrich the DNNLM in such a way that makes it more suitable for morphologically rich languages, we use the so-called *morpheme-based DNNLM with morphology-based classes*. Thus, instead of using the DNN to estimate the LM probabilities over word sequences, it is used to estimate the conditional probabilities over a mixture of word and morpheme sequences presented at the input layer. In addition, morphology-based classes derived on word and morpheme levels are added to the network input in order to estimate more reliable probabilities with the highest level of generalization. The objective of this setup is to combine the advantages of using morpheme-based LMs and morphology-based classes with the advanced modeling capabilities of the DNNs. In order not to lose the benefits of the traditional backoff m-gram LM, interpolation is performed between the DNNLM and the standard backoff m-gram LM.

A related previous work was performed in [Alexandrescu & Kirchhoff 2006], where SNNLMs are used to estimate LMs for modern standard Arabic (MSA) using word-level classes as inputs to the network with a focus on the PPL improvements only. Also, in [Kuo & Mangu$^+$ 2009], syntactic features are utilized with SNNLMs to perform LVCSR for MSA leading to significant reductions in WER. To the best of our knowledge, the approach investigated in this chapter is the first trial to utilize DNNLMs to estimate morpheme-based LMs in combination to the use of morphology-based classes on morpheme-level. It is also the second trial to use the DNNLMs in general (see [Arisoy & Sainath$^+$ 2012]). The proposed models are evaluated on an Egyptian Arabic conversational telephone speech recognition task.

Figure 5.3 shows the architecture of the DNNLM that utilizes morphology-based classes assuming that only one class of input words[2] is added to the network, and that a trigram-like model is to be estimated. In the general case, the model can utilize any number of classes for every history word, and can estimate any m-gram-like model. To add these classes to the input of the network, a unified vocabulary is created

[2]For the sake of generality, the term *word* is used here to refer to a word or a morpheme.

by putting together all the used words and class instances in a single hybrid vocabulary. Thereby, a unified binary indication vector can be used to encode any word or class instance. A separate vector is used for every type of input (word or class). For a given predicted word, its history words are expanded by adding classes. Then, all the words and classes in the history are encoded as binary indication vectors that are used as inputs to the DNN in a similar fashion as previously used for only words. If the word is indicated by w_n and the class is indicated by c_n, then the DNN of Figure 5.3 is now estimating the trigram-like model $p(w_n|w_{n-1}c_{n-1}w_{n-2}c_{n-2})$. In a similar way, more history words and more classes for every history word can be incorporated in the DNNLM.

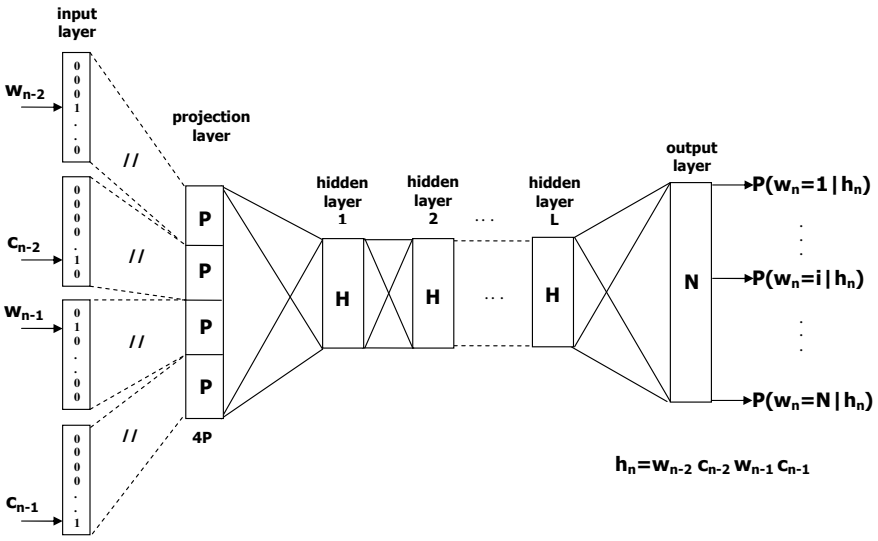


Figure 5.3. Architecture of a deep NNLM (DNNLM) with input classes. The input encoding uses separate vectors for words and their classes for every history position. The network estimates the model $p(w_n|w_{n-1}c_{n-1}w_{n-2}c_{n-2})$.

An alternative input encoding scheme is shown in Figure 5.4 that is similar to the one proposed in [Alexandrescu & Kirchhoff 2006], where only one vector per history position is used to encode the word and its classes at that position. This vector uses multiple values of *one* to indicate the presence of the word and its classes at the input layer of the network. In fact, this can be thought of as a kind of tying for some of the network parameters at the input layer. Although this tying might be useful, the encoding scheme of Figure 5.3 is used in our experiments due to its consistency with the available software.

5.2.4 Lattice Rescoring

It can be seen in all the above neural network models that the generation of the required conditional LM probabilities is very time consuming due to the heavy computations performed by the neural network especially at the output layer (speeding up techniques are to discussed in Section 5.5). Therefore, it is impractical to use this type of LM during the ASR search. Rather, it is used in the rescoring phase.

For a given DNNLM, let w_n be the predicted word, h_n represents the history words, and $\bar{h}_n$ represents the classes of the history words. Assume that the given DNNLM is estimating the probability distribution $p(w_n|h_n, \bar{h}_n)$. For example, in the estimated model of Figure 5.3, $h_n = w_{n-2}w_{n-1}$ and $\bar{h}_n = c_{n-2}c_{n-1}$.

There are two possible approaches to utilize the probabilities of this DNNLM. The first one is to perform N-best rescoring for sentences expanded with classes. In this case, to combine the DNNLM with the standard backoff m-gram LM, we need to perform N-best score combination as discussed previously in Section 4.6.2. This is because a direct interpolation of both models is not possible. The second approach is to perform lattice rescoring. In this chapter, only the second approach is investigated leaving the first one as a future work.

In order to perform lattice rescoring, we need to estimate the word probability $p(w_n|h_n)$ from the

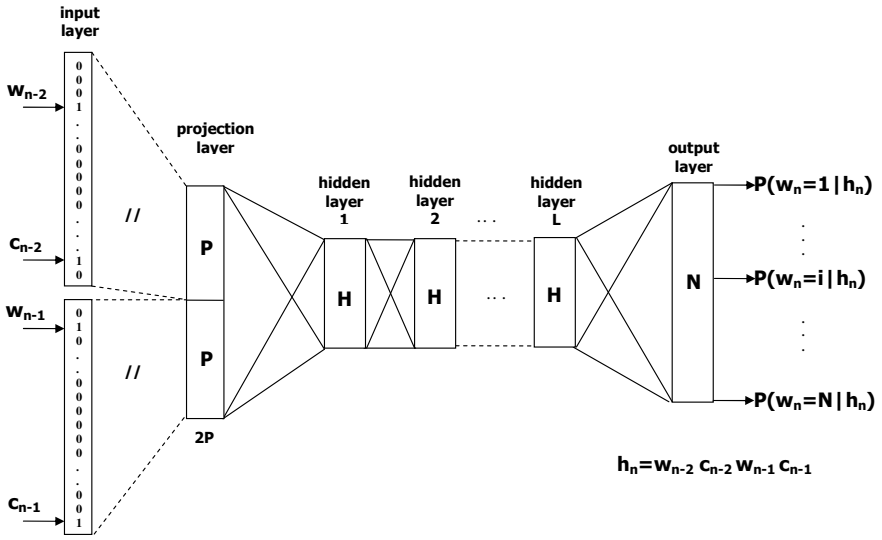


Figure 5.4. Architecture of a deep NNLM (DNNLM) with input classes. The input encoding uses one combined vector for each word and its class for every history position. The network estimates the model $p(w_n|w_{n-1}c_{n-1}w_{n-2}c_{n-2})$.

distribution $p(w_n|h_n, \bar{h}_n)$ provided by the DNNLM. For simplicity of illustration, assume a bigram case with a single class for a single history word. Let w_n be the predicted word, and let $c(w_n)$ denote the set of possible class instances assigned to w_n. Thus, we need to estimate $p(w_n|w_{n-1})$ given $p(w_n|w_{n-1}, c_{n-1})\ \forall\ c_{n-1} \in c(w_{n-1})$. This can be done using the following equation:

$$\begin{aligned} p(w_n|w_{n-1}) &= \sum_{c_{n-1}\in c(w_{n-1})} p(w_n, c_{n-1}|w_{n-1}) \\ &= \sum_{c_{n-1}\in c(w_{n-1})} p(w_n|w_{n-1}, c_{n-1})\ p(c_{n-1}|w_{n-1}) \end{aligned} \tag{5.7}$$

The distribution $p(w_n|w_{n-1}, c_{n-1})$ is obtained from the DNNLM which ensures the normalization property, such that $\sum_{w_n} p(w_n|w_{n-1}, c_{n-1}) = 1$. The probability $p(c_{n-1}|w_{n-1})$ is the probability of assigning some particular class c_{n-1} to a given word w_{n-1}. Therefore, it is called the *class membership probability*. Here, it should be noted that different instances of the class can be assigned to the same word in different positions, this is what we called before a *soft class* (review Section 4.3). For example, the same word can have different POS-tags in different positions of the sentence. It is also necessary that this probability satisfies the normalization constraint, thus $\sum_{c_{n-1}\in c(w_{n-1})} p(c_{n-1}|w_{n-1}) = 1$. The following paragraphs describe different methods to estimate $p(c_{n-1}|w_{n-1})$.

Maximum approximation. To estimate $p(c_{n-1}|w_{n-1})$, we could make an assumption that for a particular value of c_{n-1}, the probability $p(c_{n-1}|w_{n-1})$ is close to *one*, whereas for all other values, it is close to *zero*. Thus, under this assumption, the following maximum approximation can be used to estimate $p(w_n|w_{n-1})$:

$$p(w_n|w_{n-1}) = \max_{c_{n-1}\in c(w_{n-1})} p(w_n|w_{n-1}, c_{n-1}) \tag{5.8}$$

Uniform approximation. An alternative approach is to assume that $p(c_{n-1}|w_{n-1})$ is uniformly distributed over the set of possible classes $c(w_{n-1})$. Thus, if $|c(w_{n-1})|$ is the number of elements in the set $c(w_{n-1})$, then we have:

$$p(c_{n-1}|w_{n-1}) = \frac{1}{|c(w_{n-1})|} \tag{5.9}$$

This leads to the following averaging equation to estimate $p(w_n|w_{n-1})$:

$$p(w_n|w_{n-1}) = \frac{1}{|c(w_{n-1})|} \sum_{c_{n-1} \in c(w_{n-1})} p(w_n|w_{n-1}, c_{n-1}) \tag{5.10}$$

Estimation from the training data. A third approach is to estimate $p(c_{n-1}|w_{n-1})$ from the training corpus, for example:

$$p(c_{n-1}|w_{n-1}) = \frac{N(c_{n-1}, w_{n-1})}{N(w_{n-1})}$$

where the numerator represents the count of occurrences of class c_{n-1} with the word w_{n-1} in the training data, whereas the denominator represents the count of word w_{n-1} in the training data. This leads to the following estimation of $p(w_n|w_{n-1})$:

$$p(w_n|w_{n-1}) = \sum_{c_{n-1} \in c(w_{n-1})} p(w_n|w_{n-1}, c_{n-1}) \frac{N(c_{n-1}, w_{n-1})}{N(w_{n-1})} \tag{5.11}$$

Preliminary empirical results have shown that there is no significant difference in performance among the three approaches described above. However, the uniform approximation performs a little better in practice. Therefore, Equation 5.10 is used in our experiments to estimate $p(w_n|w_{n-1})$.

In order to apply Equation 5.10, all the m-grams of the required length are extracted from the lattices and expanded by adding all possible classes for each history word. The probabilities $p(w_n|w_{n-1}, c_{n-1})$ are extracted from the DNNLM. Then, the averaging of Equation 5.10 is performed. Thereby, the classes at the history side of the conditional probabilities are marginalized out to obtain word conditional probabilities that are used to rescore the lattices. Moreover, these probabilities can be interpolated with the probabilities obtained from a normal DNNLM that uses no classes at the input layer.

5.3 Back-Propagation Training Algorithm

All the different neural network architectures discussed in this chapter belong to the category of multilayer feed-forward neural networks. This type of neural network can be trained using the standard back-propagation algorithm [Rumelhart & Hinton$^+$ 1986]. This algorithm seeks to minimize the value of a given error (or loss) function over the training dataset. The error function is defined as the cross-entropy between the actual neural network output and the target (or desired) output. The cross-entropy error is used as an alternative to the famous mean square error (MSE). This is because the activations of the output neurons represent a probability distribution. Therefore, the cross-entropy error indicates the distance between what the network believes this distribution should be, and what the teacher (the target) says it should be [Plunkett & Elman 1997]. Following a similar formulation as in [Schwenk 2007], we have the error function:

$$E = \sum_{i=1}^{N} t_i log(p_i) + \lambda R \tag{5.12}$$

where t_i denotes the target output of the neural network. At the output layer, a target value of *one* is used for the predicted word in the training example, whereas a value of *zero* is used for all other words. The first part of the error function represents the cross-entropy between the output and the target probability distributions. The second part is called *a weight decay regularization term*. The parameter λ is called *the weight decay coefficient*. Assuming that only one hidden layer is used, the value of R is computed as:

$$R = \sum_{j=1}^{H} \sum_{l=1}^{(m-1)P} u_{jl}^2 + \sum_{i=1}^{N} \sum_{j=1}^{H} v_{ij}^2 \tag{5.13}$$

This is called a *squared ℓ^2 norm regularizer*, which is simply the sum of squared magnitudes of the network weights. If a number of L hidden layers are used, then the weight decay term can be generalized as:

$$R = \sum_{j=1}^{H} \sum_{l=1}^{(m-1)P} u_{jl}^2(1) + \sum_{l=2}^{L} \sum_{i=1}^{H} \sum_{j=1}^{H} u_{ij}^2(l) + \sum_{i=1}^{N} \sum_{j=1}^{H} v_{ij}^2 \tag{5.14}$$

where $u_{ij}(1)$ are the weights of the first hidden layer, and $u_{ij}(l)$ are the weights of the l^{th} hidden layer.

To minimize the error function 5.12, each weight or bias in the neural network is updated by an amount proportional to the partial derivative of E with respect to this particular weight or bias, thus:

$$\tilde{\omega} = \omega - \eta \frac{\partial E}{\partial \omega} \tag{5.15}$$

where ω is one of the weights or biases of the neural network, and η is called *the learning rate* or *the step size* that determines how much an updating step influences the current values of the weights.

5.3.1 Weight Decay Regularization

The objective of the weight decay regularization term in the error function 5.12 is to prevent the neural network from over-fitting the training data. An over-fitted neural network tends to exactly reproduce the finest details of the training data which is not a desired behavior since it badly affects the generalization capabilities of the network. Normally, the over-fitting behavior is related to the existence of large weights in the neural network [Bishop 1996; Hagiwara & Fukumizu 2008]. When the neural network weights are relatively high, the network tends to produce non-smooth outputs that are usually over-fitting the training data. In contrast, when the network weights are relatively small, the network tends to produce smoother and better generalized outputs. For this reason, the weight decay term is introduced in the error function in order to penalize (or discourage) the large weights. This means that the network is prevented from over-fitting by limiting the growth of its weights. For simplicity, imagine that the error function of Equation 5.12 is re-written as:

$$E(\mathbf{w}) = \hat{E}(\mathbf{w}) + \lambda \mathbf{w}^2 \tag{5.16}$$

where $E(\mathbf{w})$ is the regularized error as a function of the network weights $\mathbf{w}$, $\hat{E}(\mathbf{w})$ is the original (non-regularized) cross-entropy error, $\lambda \mathbf{w}^2$ is the weight decay regularization term, and λ is the weight decay coefficient. Applying gradient descent to this error function, we obtain:

$$\tilde{\omega} = \omega - \eta \frac{\partial \hat{E}}{\partial \omega} - \frac{1}{2}\eta\lambda\omega \tag{5.17}$$

Now, it can be seen that the addition of the regularization term to the weight update rule causes the weights to decay in proportion to its size ($-\frac{1}{2}\eta\lambda\omega$). Moreover, the weights are exponentially decayed to zero if no other update is presented by the original error term, i.e. when the derivative of the original error is asymptotically close to *zero*. The weight decay coefficient λ determines how we trade off the original error with the penalization of the large weights. Usually, an optimal λ is experimentally selected using a validation dataset.

5.3.2 Stochastic Back-Propagation

In the classical learning theory, the minimization of the neural network error should be performed for the sum of the error function E over all the training examples. However, this could lead to very slow convergence for large training corpora [Bishop 1996]. Alternatively, the so-called *stochastic back-propagation* is used, where the following steps are applied:

1. prepare the set of training examples used to train the neural network; these are all the m-grams extracted from the training text corpus in the form: $w_1 w_2 ... w_m$.
2. randomly select one m-gram example from the training set, now the neural network should learn $p(w_m = i | w_1, w_2, ..., w_{m-1})$.

3. supposing that the word w_m is the i^{th} word in the vocabulary, set the target outputs $t_j = \delta_{i=j}$, where $j = 1, ..., N$, and $\delta_{i=j}$ is the Kronecker delta function ($\delta_{i=j} = 1.0$ iff $i = j$; otherwise $\delta_{i=j} = 0.0$).
4. calculate the gradient of the error function E with respect to all the network weights.
5. update all the network weights using the update formula 5.15.
6. randomly select a different training example and repeat the above procedure until all the training examples are exhausted.
7. repeat the whole procedure for several epochs[3].

After each epoch, the total error is measured over a held-out set to monitor the convergence of the training process. In the above steps, it is noted that the gradient is back-propagated through the projection layer, which means that the neural network learns the projection matrix that best fits the probability estimation task. Using this training algorithm, it can be empirically shown that the outputs of a neural network converge to the posterior probabilities. Thus, the neural network minimizes the perplexity of the training data under the constraints given by its architecture and the limited computational resources available for optimizing its parameters [Schwenk 2007].

5.3.3 Computational Complexity

In order to calculate one m-gram probability $p(w_n = i|w_{n-m+1}^{n-1})$ using one forward pass through the neural network, the following set of calculations are needed. First, the calculations of the projection layer is achieved by simple table look-up due to the use of the 1-of-N input encoding (review Section 5.2.1), thus the calculation of the projection layer can be neglected from the complexity analysis. Second, the calculation of the hidden and the output layer corresponds to a matrix/vector multiplication followed by applying a non-linear function. The addition of the bias terms can also be neglected since it can be done in one *multiply-and-add* operation. Then, the number of the floating point operations for a single hidden layer neural network can be given as:

$$(m-1)PH + H + HN + N$$

If L hidden layers are used, then the number of the floating point operations becomes:

$$(m-1)PH + H + (L-1)(H^2 + H) + HN + N$$

It can be seen that the use of L hidden layers, which is the case in deeper neural networks, adds a factor of $(L-1)(H^2 + H)$ to the computational complexity. Since N is usually much larger than H, the computational complexity is still dominated by the calculation of the output layer which is the most time consuming part of the calculations. Therefore, it is impractical to train such large networks for typical LVCSR tasks [Schwenk 2007], where the vocabulary might contain up to several hundred thousands of words, and the training examples might reach several millions of examples. In Section 5.5, several techniques are described to reduce the computational effort [Schwenk & Gauvain 2005].

It is worth noting that the computational complexity of the neural network LM increases only linearly with the order of the used m-gram, the size of the vocabulary, and the number of hidden layers. This is a major advantage compared to the backoff m-gram LM whose complexity increases exponentially with the order of the used m-gram. Therefore, it is quite common to use longer span neural network LMs in typical LVCSR tasks. However, it is not common in practice to use higher than 5-gram backoff LMs due to the data sparsity problems.

5.4 Pre-Training Strategies

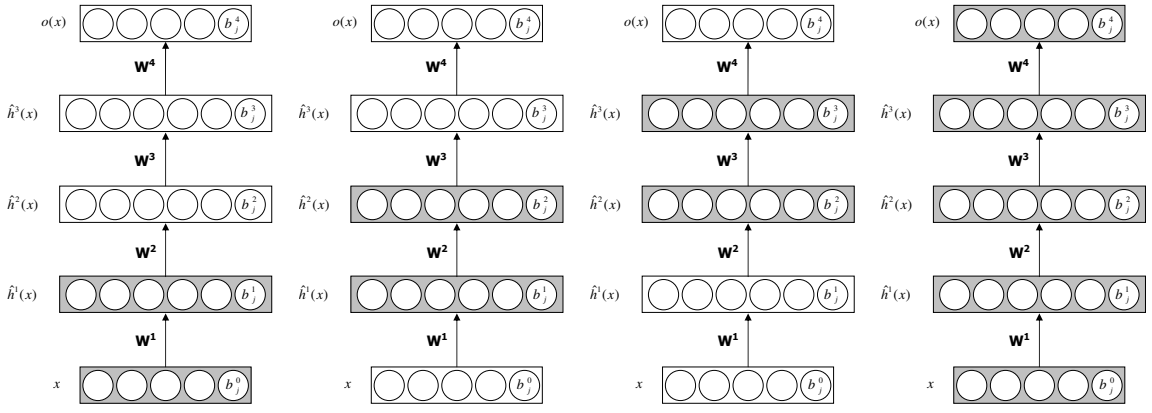
Having many hidden layers of nonlinearities in the DNN architecture makes the training of such DNNs more prone to fall into poor local minima or plateaus of the non-convex training criterion compared to SNNs [Auer & Herbster$^+$ 1996; Larochelle & Bengio$^+$ 2009]. Therefore, it becomes vital to think of more

[3] An epoch is one pass through all the training examples.

efficient techniques to initialize the weights of the neural networks such that better initial solutions are reached. The recently proposed DNN learning algorithms make use of the so-called pre-training strategies. These strategies use the idea of greedily training the network to break down the learning problem into easier steps in order to provide more effective hint about what the hidden units should learn. This is considered as a form of regularization that prevents over-fitting even for deep networks that have many degrees of freedom [Larochelle & Bengio$^+$ 2009].

The main challenge in training DNNs is dealing with the strong dependencies that exist between the parameters across layers. Thus, it is important to adapt the lower layers in order to provide adequate input to the upper layers, and at the same time adapt the upper layers to make good use of the lower layers [Erhan & Bengio$^+$ 2010]. For instance, the greedy layer-wise unsupervised pre-training provides an initialization procedure followed by a fine-tuning step using a global supervised target. Examples of the unsupervised pre-training make use of the restricted Boltzmann machines (RBMs) [Hinton & Osindero$^+$ 2006], and the nonlinear auto-encoder neural networks (AENNs) [Saund 1989]. As illustrated in [Larochelle & Bengio$^+$ 2009], the general paradigm of the greedy layer-wise unsupervised pre-training can be described in the following steps (see Figure 5.5):

1. **initialization:** randomly initialize all the parameters of the neural network.
2. **first phase:** greedily train subsets of the neural network parameters using a layer-wise unsupervised training criterion by repeating the following steps for every layer $l \in \{1, ..., L\}$ using iterations through all the training dataset.
   a) use the current input training sample $\mathbf{x}_t$ to produce representations $\widehat{h}^{l-1}(\mathbf{x}_t)$ and $\widehat{h}^{l}(\mathbf{x}_t)$ for layers $l-1$ and l respectively.
   b) update the biases $\mathbf{b}^{l-1}$ and $\mathbf{b}^{l}$ of layers $l-1$ and l respectively, and update the in-between weight matrix $\mathbf{w}^{l}$ using some unsupervised learning algorithm.
3. **second phase:** fine-tune all the neural network parameters using back-propagation and gradient descent using a global supervised error function.



(a) Pre-training 1^{st} hidden layer. (b) Pre-training 2^{nd} hidden layer. (c) Pre-training 3^{rd} hidden layer. (d) Fine-tuning the whole network.

Figure 5.5. General steps of a greedy layer-wise unsupervised pre-training algorithm.

An alternative pre-training approach presented in [Seide & Li$^+$ 2011] is to perform a supervised layer-wise discriminative pre-training using the target labels at every step, then applying a final supervised global fine-tuning.

In the work of [Arisoy & Sainath$^+$ 2012] on DNNLMs, a supervised discriminative pre-training approach has been experimented following the description in [Seide & Li$^+$ 2011]. However, no consistent gains have been observed. In this thesis, investigations on pre-training strategies of DNNLMs is left as a future work.

5.5 Speeding Up Techniques

In this section, we discuss several techniques used in our experiments to speed up both the training and the recognition processes of the DNNLM. These techniques are originally proposed in [Schwenk 2007] in order to make the neural network LM tractable for typical LVCSR tasks.

5.5.1 Lattice or N-best Rescoring

The use of the neural network LM during the ASR search could lead to a significant increase in the recognition time. Alternatively, the neural network LM can be used to rescore lattices or N-best lists produced by traditional speech decoders using the conventional backoff m-gram LMs. The required m-gram probabilities needed to perform the rescoring can be generated off-line from the neural network and stored in a table which is just looked-up in the rescoring time.

5.5.2 Regrouping Probability Requests

Usually, when requesting m-gram probabilities from a neural network LM, probabilities with the same context h_n are often requested multiple times. To speed up the calculation of the requested probabilities, it is better that all these request are grouped together so as to avoid multiple forward passes on the neural network since all the LM probabilities for the same context are immediately available on the output layer.

5.5.3 Vocabulary Truncation

As previously mentioned in Section 5.3, the computational complexity of the neural network LM is dominated by the calculation of the output layer. Therefore, one approach to speed up the calculation of the m-gram probabilities is to decrease the size of the output layer. Thus, instead of using a full vocabulary size N, the output layer is limited to the s most frequent words, where $s \ll N$. This is called a *short-list*. Assume that the set of words in the short-list is indicated by $\mathcal{S}$, where $|\mathcal{S}| = s$. Let $p_{NN}(w_n|h_n) : w_n \in \mathcal{S}$ denote the LM probabilities of the words in the short-list calculated from the neural network, and $p_{BO}(w_n|h_n) : w_n \notin \mathcal{S}$ denotes the LM probabilities of the words not in the short-list obtained form a standard backoff m-gram LM, then:

$$p(w_n|h_n) = \begin{cases} p_{NN}(w_n|h_n) \cdot \alpha(h_n) & \text{if } w_n \in \mathcal{S} \\ p_{BO}(w_n|h_n) & \text{otherwise} \end{cases} \tag{5.18}$$

where, $\alpha(h_n)$ is a normalization factor used to guarantee that the whole model sums to unity. To find $\alpha(h_n)$, we make the total probability mass of the model equal to *one*, thus:

$$\alpha(h_n) \sum_{w_n \in \mathcal{S}} p_{NN}(w_n|h_n) + \sum_{w_n \notin \mathcal{S}} p_{BO}(w_n|h_n) = 1 \tag{5.19}$$

Since the neural network LM is normalized over the short-list, then we have:

$$\sum_{w_n \in \mathcal{S}} p_{NN}(w_n|h_n) = 1 \tag{5.20}$$

In addition, the backoff LM is normalized over the whole vocabulary, then we can use:

$$\sum_{w_n \notin \mathcal{S}} p_{BO}(w_n|h_n) = 1 - \sum_{w_n \in \mathcal{S}} p_{BO}(w_n|h_n) \tag{5.21}$$

Substituting 5.20 and 5.21 into 5.19, we obtain:

$$\alpha(h_n) = \sum_{w_n \in \mathcal{S}} p_{BO}(w_n|h_n) \tag{5.22}$$

Furthermore, it is possible to use the neural network LM to predict the probability mass of all the out-of-short-list words. This is achieved by using a special output word (usually called *unk*) to group together all the words that are not in the short-list. In this case, the other outputs of the neural network correspond directly to the LM probabilities without any rescaling factors. Let this *unk* word be denoted by u, then the probability of this special word, denoted by $p_{NN}(u|h_n)$, is used to rescale the probabilities obtained from the backoff LM. Therefore, Equation 5.18 can be rewritten as:

$$p(w_n|h_n) = \begin{cases} p_{NN}(w_n|h_n) & \text{if } w_n \in \mathcal{S} \\ \alpha(h_n) \cdot p_{NN}(u|h_n) \cdot p_{BO}(w_n|h_n) & \text{otherwise} \end{cases} \tag{5.23}$$

Since the neural network LM is now normalized over $\mathcal{S} \cup \{u\}$, then we have:

$$\sum_{w_n \in \mathcal{S}} p_{NN}(w_n|h_n) + p_{NN}(u|h_n) = 1 \tag{5.24}$$

Again, taking into account Equation 5.21, we obtain the new normalization factor $\alpha(h_n)$, as:

$$\alpha(h_n) = \frac{1}{1 - \sum_{w_n \in \mathcal{S}} p_{BO}(w_n|h_n)} \tag{5.25}$$

Instead of using short-lists, classing or factorization of the output layer can be used as presented in [Mikolov & Kombrink+ 2011]. In this case, the neural network estimates the probability distribution over classes of the words rather than the words themselves. In the work of this chapter, short-lists are used with Equation 5.23 to calculate the LM probabilities over the complete vocabulary.

5.5.4 Bunch Mode

Another possible reduction in the computational effort of the neural network LM can be achieved by propagating several examples at once to the neural network. This is known as *bunch mode* [Bilmes & Asanovic+ 1997], which leads to the use of matrix/matrix operations rather than matrix/vector operations (see Equations 5.4 and 5.5), thus:

$$\mathbf{D} = tanh(\mathbf{U} \times \mathbf{C} + \mathbf{B}) \tag{5.26}$$

$$\mathbf{O} = \mathbf{V} \times \mathbf{D} + \mathbf{K} \tag{5.27}$$

where $\mathbf{B}$ and $\mathbf{K}$ are the bias matrices obtained by duplicating the bias vectors $\mathbf{b}$ and $\mathbf{k}$ in each line of the corresponding matrix. The real functions *tanh* is performed element-wise. These matrix/matrix operations can be heavily optimized on the current CPU architectures (cf. [Schwenk 2007]).

5.5.5 Resampling the Training Data

One possible approach to speed up the training of the neural network is to avoid performing epochs over the whole training data several times. Instead, a small random subset of the training data can be selected at each epoch. The advantage of this procedure is that any amount of training data can be used. In addition, the change in the training examples after each epoch adds noise to the training process which increases the generalization performance. Moreover, After performing some epochs, it is highly probable that all the training examples are covered. On the other hand, it might be important to resample differently from different corpora. Thus, to use all the examples of a small in-domain corpus, and take only small parts of a large complementary corpus.

5.6 Generating Morphemes and Classes for Egyptian Arabic

As previously mentioned in Section 5.2.3, the experiments of this chapter are performed on an Egyptian Arabic conversational telephone speech recognition task. In order to use morpheme-based DNNLMs with

morphology-based classes, first we need to find an efficient method to perform word decomposition and to produce morpheme-level classes.

In fact, almost all the available Arabic morphological analyzers are specifically designed for modern standard Arabic (MSA). However, one important property about Egyptian Arabic is that it shares a large portion of the written vocabulary with MSA (review Section 1.9.1). This makes it possible to reuse the MSA morphological analyzers to perform morphological decomposition and class derivation for Egyptian Arabic with some acceptable margin of error. In this chapter, we reuse the morphological analyzer and disambiguator for Arabic (MADA) [Habash & Rambow 2005, 2007] previously investigated for MSA (see Sections 3.3.1 and 4.1.1).

5.6.1 Word Decomposition

The available LM training text is completely preprocessed using the MADA tool in order to perform morphological analysis and disambiguation and produce the corresponding set of morphological tags along with the associated word tokenization as described in Section 3.3.1. For the non-MSA words that occur in the Egyptian Arabic text, MADA produces special *unknown* markers to indicate the inability to analyze the words.

To get an idea of how MADA behaves differently with Egyptian Arabic than with MSA, we performed some analysis on the unknown word rate. For a typical MSA text, the unknown word rate is around 1% to 3%. However, for some Egyptian Arabic text, the unknown word rate is around 10% to 12%. In addition, MADA produces some additional errors in the already known MSA words that are used in Egyptian Arabic in a different sense. Such words have deceptively the same surface forms as some well known MSA words but they have completely different meanings and pronunciations in Egyptian Arabic. Given that MADA achieves an accuracy of around 95% for analyzing MSA words as reported in [Habash & Rambow 2005], then this means that Egyptian Arabic words can be processed using MADA with an accuracy of around 80% to 85%.

Based on the produced MADA tokenization, we produce decomposed words in the form of *" prefix+ stem +suffix"*. For more details about the decomposition process, see Section 3.3.1.

5.6.2 Class Derivation

Starting from the set of MADA morphological tags along with the generated decomposition, we derive two different classes, namely *lexeme* and *morph* as previously explained in Section 4.1.1. The LM training text is rewritten so that every word is replaced by a vector of classes as in the form: *W-<word>:L-<lexeme>:M-<morph>*. The same classes are similarly defined for morphemes as well as for words. For more details and examples, see Section 4.1.1.

5.7 Experimental Results

As described in Appendix A, the LM training text have around 7M running full-words including the following corpora: acoustic data transcriptions (140k words), web text (5M words), extra sources (1.5M words). Each corpus of these three is used to estimate a separate backoff trigram LM, these LMs are then linearly interpolated together to create a single background backoff trigram LM. The interpolation weights are chosen so as minimize the perplexity over the development *eca-dev* corpus. The used interpolation weights reveal that the LM built over the last corpus (1.5M words of extra sources) is the least influential to the final model. Therefore, Only the first two corpora are used to train separate trigram neural network LMs (NNLMs) in order to speed up the training process. The two NNLMs are also interpolated together with the background backoff trigram LM to create a single model.

The parameterization of the neural network architecture follows the best reported settings in [Arisoy & Sainath$^+$ 2012]. Thus, if P is the feature dimension at the projection layer, H is the number of hidden units in each hidden layer, L is the number of hidden layers, and N is the size of the output layer (length of short-list). Then, we use the following settings: [P = 120; H = 500; L = 1,...,4; N = 10k to 20k].

To incorporate morphology-based classes into NNLMs, we create a similar setup that estimates the model $p(w_n|w_{n-1}, l_{n-1}, m_{n-1}, w_{n-2}, l_{n-2}, m_{n-2})$, where w is the word, l is the lexeme, and m is the morph. To use this model for lattice rescoring, we follow the procedure described in Section 5.2.4.

It is worth noting that all possible types of parameter tuning are performed on *eca-dev* corpus, like the optimization of the interpolation wights, the neural network learning rates, and the weight decay coefficients. Whereas, the recognition results are only reported on the *eca-eval* corpus. Details about the development and evaluation corpora are given in Appendix A.

Table 5.1 shows the results of recognition experiments performed on CallHome Egyptian colloquial Arabic (ECA) evaluation corpus *eca-eval* for a word-based system using trigram NNLMs with different number of hidden layers interpolated with a traditional backoff trigram LM. The backoff LM is used alone for the baseline experiment. Two different versions of NNLMs are experimented with and without the incorporation of lexeme and morph classes. We follow the 2 passes recognition setup of the Egyptian Arabic testing system described in Appendix A, where a standard backoff trigram LM is used to produce lattices which are then rescored using different models. The system uses a vocabulary of 350k full-words that represent almost all the available distinct words in the training text.

Table 5.1. Recognition experiments on CallHome Egyptian colloquial Arabic (ECA) evaluation corpus *eca-eval* using word-based neural network LMs (NNLMs) for lattice rescoring. vocabulary: 350k full-words, OOV rate = 1.4%, graph (lattice) error rate (GER) = 37.2%.

LM	hidden layers	classes	WER (ins/del) [%]	CER (ins/del) [%]
3-gram backoff	-	-	**59.7** (2.8/24.4)	**44.9** (8.5/26.1)
+ NNLM	1	-	59.2 (2.8/24.3)	44.2 (8.3/26.0)
	2		59.4 (2.8/24.3)	44.3 (8.6/26.2)
	3		**59.1** (2.8/24.2)	**44.2** (8.3/26.0)
	4		59.2 (2.8/24.3)	44.3 (8.4/26.1)
+ NNLM	1	lexeme & morph	59.2 (2.7/24.3)	44.3 (8.4/26.1)
	2		59.2 (2.7/24.3)	44.3 (8.4/26.1)
	3		**59.0** (2.7/24.2)	**44.2** (8.3/26.0)
	4		59.1 (2.7/24.3)	44.2 (8.3/26.0)

It can be seen that the use of a shallow NNLM improves the WER by [0.5% absolute (0.8% relative)] compared to the traditional backoff LM. In addition, using a deeper NNLM with 3 hidden layers improves the WER a little further by [0.6% absolute (1.0% relative)] compared to the traditional backoff LM. Moreover, using lexeme and morph classes with a 3 hidden layer NNLM also improves the WER a little further by [0.7% absolute (1.2% relative)] also compared to the traditional backoff LM.

Similarly, Table 5.2 shows the results of recognition experiments performed on *eca-eval* corpus for a morpheme-based system using trigram NNLMs with and without lexeme and morph classes. In addition, interpolation is performed between both types of NNLMs. The system uses a 250k vocabulary having *5k full-words + 245k morphemes* which also represent almost all the available distinct words/morphemes in the decomposed training text. The 5k full-words are the most frequent decomposable full-words in the training data (see decomposition constraint (2) in Section 3.3.1). The choice of keeping exactly 5k most frequent decomposable full-words in the recognition vocabulary is made after performing a series of recognition experiments to minimize the WER over the development corpus *eca-dev* using gradually increased number of full-words (see Figure 5.6).

It can be seen that initially going to a morpheme-based system improves the WER by [2.9% absolute (4.9% relative)] compared to the traditional word-based backoff LM shown in Table 5.1. Using a NNLM with 2 hidden layers significantly improves the WER by [0.8% absolute (1.4% relative)] compared to the conventional morpheme-based backoff LM. Using a NNLM with lexeme and morph classes alone does not lead to further improvement in the WER. However, interpolating this NNLM with the previous NNLM that uses no classes achieves a little further improvement in the WER by [1.0% absolute (1.8% relative)] compared to the conventional morpheme-based backoff LM.

Here, it is worth noting that the optimum number of hidden layers is 3 layers for the word-based system and 2 layers for the morpheme-based system. This is because the degree of variation in the word domain

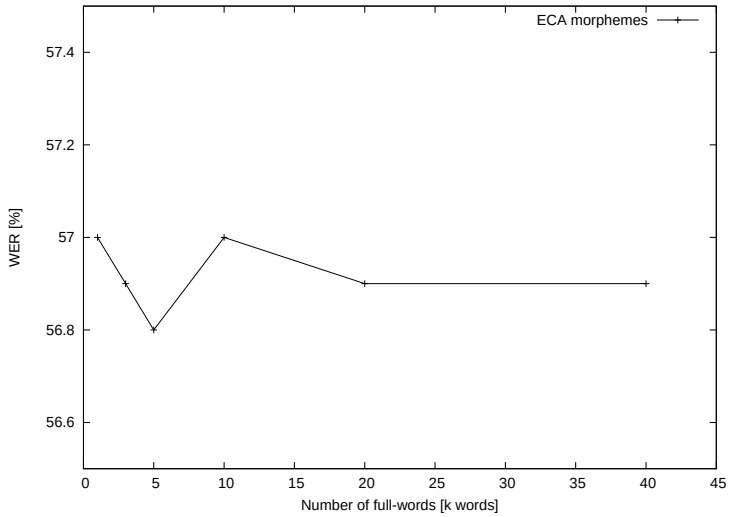


Figure 5.6. Optimization of the number of decomposable full-words retained in the morpheme-based vocabulary performed over *eca-dev* corpus using overall vocabulary size of 250k (best WER = 56.8% with 5k full-words). Baseline WER on *eca-dev* using 350k full-words vocabulary = 56.9%.

Table 5.2. Recognition experiments on CallHome Egyptian colloquial Arabic (ECA) evaluation corpus *eca-eval* using morpheme-based neural network LMs (NNLMs) for lattice rescoring. vocabulary: 250k (5k words + 245k morphemes), OOV rate = 0.9%, graph (lattice) error rate (GER) = 32.3%.

LM	**hidden layers**	**classes**	**WER** (ins/del) [%]	**CER** (ins/del) [%]
3-gram backoff	-	-	**56.8** (2.9/19.1)	**40.4** (9.0/20.6)
+ NNLM	1	-	56.1 (2.8/19.1)	40.05 (8.93/20.40)
	2		**56.0** (2.8/19.1)	**40.0** (8.9/20.4)
	3		56.2 (2.8/19.1)	40.1 (9.0/20.4)
	4		56.0 (2.8/19.1)	40.0 (8.9/20.4)
+ NNLM	1	lexeme & morph	56.0 (2.8/19.1)	40.0 (8.9/20.4)
	2		**56.0** (2.8/19.1)	**40.0** (8.9/20.3)
	3		56.0 (2.8/19.1)	40.0 (8.9/20.4)
	4		56.1 (2.8/19.1)	40.0 (8.9/20.4)
+ NNLM		-		
+ NNLM		lexeme & morph		
	1		56.0 (2.8/19.1)	40.0 (8.9/20.3)
	2		**55.8** (2.7/19.2)	**39.9** (8.9/20.4)
	3		56.0 (2.8/19.1)	39.97 (8.9/20.3)
	4		55.9 (2.7/19.1)	39.97 (8.9/20.3)

is more than that in the morpheme domain. This means that, when using morpheme-based systems, simpler NNLMs with less number of hidden layers are required and can even achieve better performance compared to more complex NNLMs operating on full-words.

The final improvement in the WER compared to the traditional word-based backoff LM is [3.9% absolute (6.5% relative)] using a threefold interpolation of: morpheme-based backoff LM + deep NNLM + deep NNLM with lexeme and morph classes. Table 5.3 shows the word- and character-level perplexities for the models listed in Tables 5.1 and 5.2.

Table 5.3. Word-/morpheme-level and character-level perplexities on CallHome Egyptian colloquial Arabic (ECA) evaluation corpus *eca-eval* for different LMs (**inv**: perplexity for in-vocabulary text excluding the unk symbol; **all**: perplexity for the whole text including the unk symbol; **units**: words or morphemes).

LM	hidden layers	classes	word-/morpheme-level PPL		char-level PPL	
			inv (#units)	all (#units)	inv (#chars)	all (#chars)
full-words backoff	-	-	**336.1** (17269)	**330.2** (17514)	**4.070** (71568)	**3.991** (73396)
+ shallow NNLM	1	-	321.5	316.3	4.027	3.950
+ deep NNLM	3	-	323.9	318.2	4.034	3.956
+ deep NNLM	3	lexeme & morph	**312.8**	**307.2**	**4.000**	**3.922**
morphemes backoff	-	-	**312.9** (17939)	**308.4** (18093)	**4.120** (72801)	**4.063** (73971)
+ shallow NNLM	1	-	291.1	286.4	4.047	3.990
+ deep NNLM	2	-	290.5	285.3	4.045	3.986
+ deep NNLM	2	lexeme & morph	286.4	281.2	4.031	3.972
+ deep NNLM	2	-				
+ deep NNLM	2	lexeme & morph	**285.3**	**280.0**	**4.027**	**3.968**

Figure 5.7 presents a comparative summary of the WERs obtained on *eca-eval* corpus using the models introduced in this chapter. The figure illustrates the gradual improvement in the WER achieved by each proposed step. The WER improvements are found statistically significant using a bootstrap method of significance analysis described in [Bisani & Ney 2004], the probability of improvement (POI_{boot}) ranges between 95% and 97%.

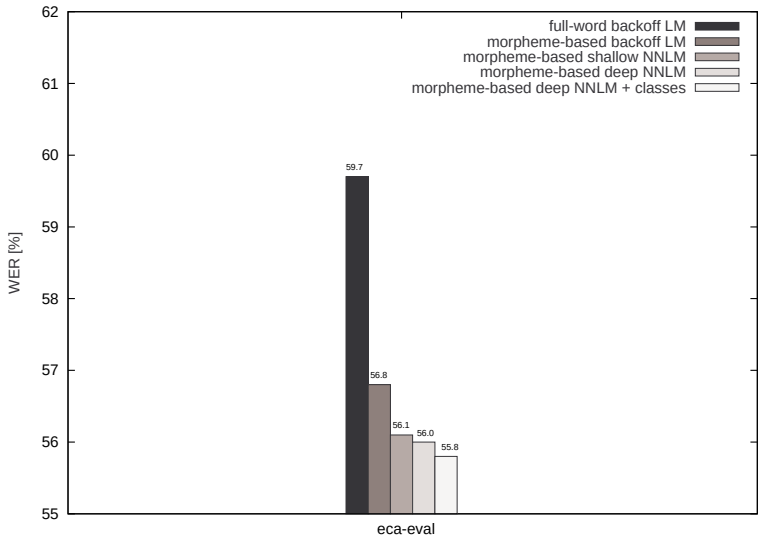
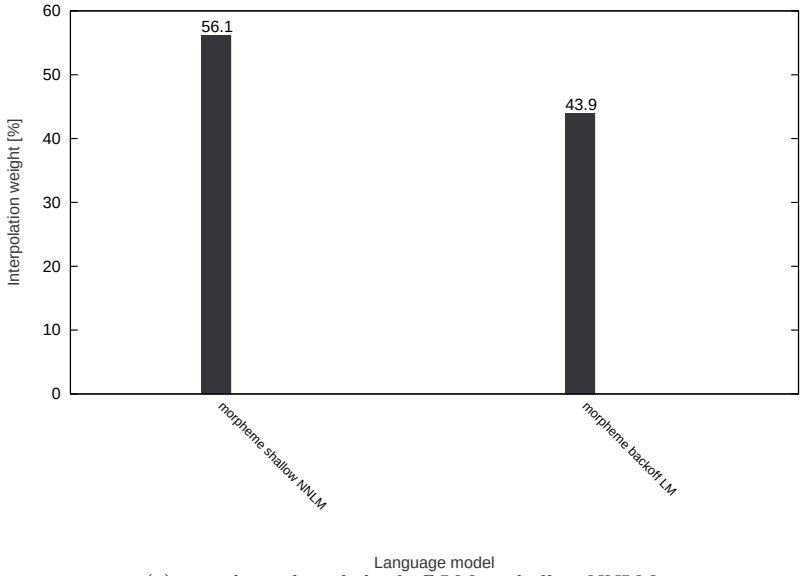
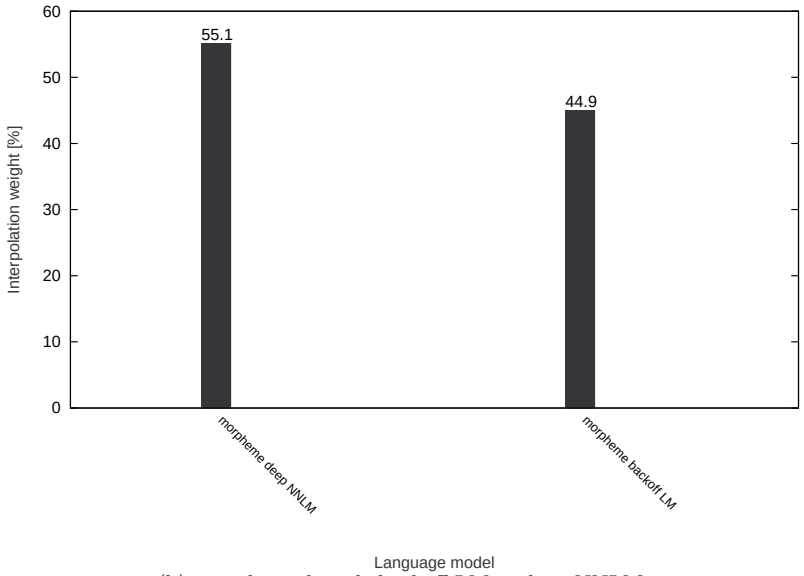


Figure 5.7. Comparison of recognition WERs [%] on Egyptian Arabic *eca-eval* corpus using different LMs.

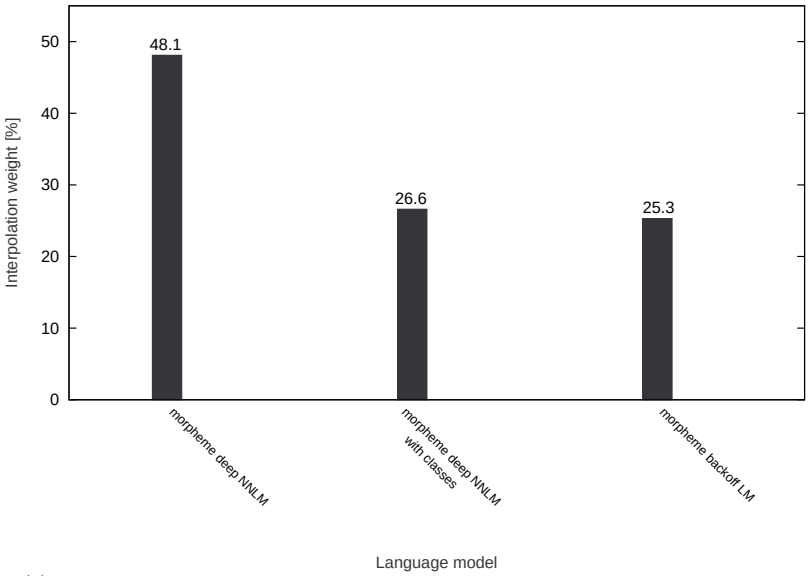
Interpolation weights. In Figure 5.7, the last three bars represent the WERs for several models interpolated together. To clarify the relative importance of the individual LMs during linear interpolation, we report the interpolation weights assigned to every LM. Figure 5.8 gives the interpolation weights as percent values for the interpolated models. These weights are optimized over the *eca-dev* corpus.



(a) morpheme-based: backoff LM + shallow NNLM.



(b) morpheme-based: backoff LM + deep NNLM.



(c) morpheme-based: backoff LM + deep NNLM + deep NNLM with classes.

Figure 5.8. Interpolation weights of individual morpheme-based LMs.

5.8 Summary

In this chapter, feed-forward deep neural network LMs (DNNLMs) with several hidden layers of nonlinearities have been investigated to perform LVCSR for Egyptian colloquial Arabic (ECA) conversational telephone speech. The deep neural networks (DNNs) have been utilized to estimate morpheme-based LMs, where a mixture of words and morphemes have been presented as inputs to the neural network. To enhance the generalization of the models, morphology-based classes derived on word and morpheme levels have been added to the network input. This is a novel approach that aims at combining the advantages of using morpheme-based LMs and morphology-based classes with the advanced modeling of the DNNs.

The morpheme-based modeling aims at achieving better lexical coverage and reducing the severity of the data sparsity problem. The use of morphology-based classes helps to promote the generalization of the LMs. Whereas, the use of DNNLMs allows for improved smoothness and higher discrimination in continuous space. The neural network LMs have been used to perform lattice rescoring after running the ASR search with a traditional backoff LM. Recognition experiments have been conducted using the state-of-the-art LVCSR systems.

The morphological analyzer and disambiguator tool (MADA) designed for modern standard Arabic (MSA) has been reused to perform word decomposition and class generation for Egyptian Arabic. Experimental results have shown that the most influential step on the recognition performance is the use of morpheme-based LMs. The second most influential step is the use of DNNLMs. Therein, the largest improvement has been obtained by the first hidden layer, whereas a little additional improvement has been acquired by using deeper neural networks. The third most influential step is the use of morphology-based classes, which introduces a little further improvement. In all cases, linear interpolation of neural network models with conventional backoff models has been a vital process to obtain the observed improvements.

The best recognition performance has been obtained by interpolating the following models: morpheme-based backoff LM, morpheme-based DNNLM, and morpheme-based DNNLM with classes. Proper tests have shown the statistical significance of the achieved WER improvements.

Future work. In order to improve the proposed methodologies presented in this chapter, the following set of investigations are recommended as a future work:

1. Increasing the context length of the neural network models (i.e. going to 4, 5, or 7-gram LMs).
2. Optimizing the size of each layer of the neural network including the sizes of: the projection, the hidden, and the output layers.
3. Using a different input encoding scheme for the DNNLMs with classes (see Figure 5.4).
4. Investigating the effect of different pre-training strategies on the DNNLMs (see Section 5.4).
5. Using different techniques to reduce the size of the output layer, such as classing or factorization of the output layer, instead of simply using *short-lists* (see Section 5.5).
6. Investigating the use of N-best rescoring instead of lattice rescoring.
7. Using a morphological analysis tool designed for Egyptian Arabic rather than using MADA, which is dedicated for MSA.

Chapter 6

Scientific Contributions

The goal of this thesis has been to develop improved language modeling approaches that are intended to be utilized for performing efficient large vocabulary continuous speech recognition (LVCSR) for morphologically rich languages. The work of this thesis has been focused in three major directions that are explored in parallel during the creation of the language models (LMs): the first is to investigate the use of different types of sub-word units, the second is to incorporate various morphology-based classes in the estimation of the LMs, the third is to explore the use of recent state-of-the-art modeling and parameter estimation techniques. To enhance the performance of the proposed models, special attention has been paid to combine these approaches together. Experiments have been conducted on modern standard Arabic (MSA), Egyptian colloquial Arabic (ECA), German, and Polish LVCSR tasks. This work contains the following contributions which address the aforementioned research directions:

Development of optimized sub-word based language models. Sub-word based language models have been investigated using different types of graphemic sub-words in combination to full-words, like morphemes and syllables. Different approaches have been used to break down the full-words into the required types of sub-word units including: supervised and unsupervised word decomposition approaches, and rule based syllabification. It has been shown that it is very influential to keep some number of the most frequent full-words without decomposition as a part of the recognition vocabulary. A careful optimization of the number of full-words has been found to have a high impact on the recognition performance. Very large vocabularies have been employed in the recognition systems in order to discover the actual potential of the sub-word based models compared to full-word models. This approach has lead to a significant increase in the overall lexical coverage indicated by a considerable reduction in the out-of-vocabulary (OOV) rates measured on the development and evaluation datasets. This has introduced a one step towards the solution of the data sparsity and the poor lexical coverage problems. As a result, significant improvements in the recognition performance have been achieved compared to the traditional full-word based LMs.

Joint language and pronunciation models. Joint language and pronunciation models have been investigated via incorporating sub-word pronunciations into LMs. This has been achieved by using a novel type of sub-word unit called *graphone* which is a joint unit of a sub-word with its context dependent pronunciation. In order to combine this approach with the previous graphemic sub-lexical approach, novel types of graphones have been explored based on different types of sub-words, like morpheme-based and syllable-based graphones in addition to the previously known fragment-based graphones. These graphone models have been derived from grapheme-to-phoneme (G2P) conversion models. They have been essentially utilized to cope with high OOV rates by spelling new words as sequences of graphones. For Arabic LMs, a special type of units called *diacritized sub-words* have been used in place of graphones. This approach has introduced improved models to deal with OOV words. As a result, improvements in the recognition performance have been observed compared to the traditional full-word based LMs.

Development of extended hybrid language models. Extended hybrid LMs comprising mixed types of lexical and sub-lexical units have been investigated. For example, these units have included: full-words, morphemes, and morpheme-based graphones; or full-words, syllables, and syllable-based graphones in one hybrid lexicon and LM. Mainly, full-words have been used to model the most frequent tokens in the training data. Whereas, morphemes or syllables have been used to model the moderately frequent tokens. However, morpheme-based graphones or syllable-based graphones have been used to model the least frequent tokens in the training data. This mixture of multiple types of units have been used to perform open vocabulary

LVCSR tasks, where systems operate over open and constantly changing vocabularies. To achieve the best performance, an optimization has been performed on the numbers of the used vocabulary items of each type of unit. Using such models, improvements in recognition performance have been achieved over the traditional full-word based LMs.

Investigations on using morphology-based classes in language models. Various morphology-based classes have been incorporated into the estimation of the LMs in order to overcome the data sparseness and achieve higher levels of generalization. In this context, stream- and class-based LMs have been explored. These models have exploited different morphology-based classes. This approach has been found to yield better smoothing and better generalization with regard to unseen word sequences. In stream- and class-based LMs, every class stream has been treated separately without considering any interaction with the other classes during the backoff. Therefore, separate models have been built over every individual class. In addition, linear interpolation have been used to combine multiple models together. This approach has taken a step further towards the solution of the data sparsity problem by supporting sparse word sequences with equivalence classes which are naturally more frequent. This method has shown significant improvements in recognition performance over the use of traditional word-based models.

In addition, factored LMs (FLMs) have been investigated, where both words and their classes have been viewed as generic factors. Every word has been considered as a vector of parallel factors over which the probability estimation has been performed. Class streams have been jointly handled during the backoff using a complex backoff mechanism defined by a so-called *backoff graph*. It has been shown that the optimization of the FLM parameters is a crucial issue that needs a careful handling because it affects the performance significantly. Although the FLM is a quite general and powerful model, it has been found that it is very difficult to optimize its parameters for a given task. This is due to the huge space of the FLM parameters. These parameters have been optimized using both manual and automatic optimization techniques. Trials have been made to combine the scores of the FLMs with the scores of other models, like the class-based LMs, during the N-best rescoring. Experimental results have shown improvements using FLMs over the traditional word-based models.

Combining the benefits of sub-word based LMs and morphology-based classes. A novel approach have been introduced to use morphology-based classes for sub-word based LMs, rather than for word-based LMs. Thus, investigations have been performed to build stream-based, class-based, and factored LMs over sequences of morphemes and their classes. These models have been shown to effectively retain the benefits of the sub-word based LMs along with the advantages of using morphology-based classes. Sub-words have been used to achieve better lexical coverage and reduce the influence of the data sparsity. Whereas, the morphology-based classes have been used to achieve better generalization to unseen word sequences. To estimate such models, classes have been derived on morpheme level. Different classes have been utilized, like the morphology-based classes derived from the MADA Arabic morphological analyzer and the German TreeTagger. An additional data-driven class generated based on a data-driven word clustering algorithm has been used in the German LVCSR experiments. Experimental results have shown similar significant improvements in recognition performance when using morphology-based classes on sub-word level compared to the use of morphology-based classes on word level.

Investigations on hierarchical Pitman-Yor language models. Investigations have been made to combine the use of sub-word based LMs and morphology-based classes with the recent state-of-the-art modeling techniques. From among those techniques, hierarchical Pitman-Yor LMs (HPYLMs) have been investigated. The HPYLM is a type of hierarchical Bayesian LM based on a coherent Bayesian probabilistic model. It relies on the Pitman-Yor (PY) process, a generalization of the Dirichlet distribution. In fact, it is considered as a direct generalization of the hierarchical Dirichlet LM. The HPYLM produces *power-law* distributions over word frequencies, which has been found to be an important statistical property of natural languages. The HPYLM has been utilized to estimate class-based LMs on morpheme level. The resulting models are called *hierarchical Pitman-Yor class-based LMs (HPYCLMs)*. Using such models, it has been shown that all the benefits of the sub-word modeling, morphology-based classes, and the efficiency of the HPYLM can be combined together in a single model. The sub-word models achieve

better lexical coverage and reduce the data sparsity. The use of morphology-based classes helps to achieve better generalization to unseen word sequences. On the other hand, the use of HPY models improves the smoothness of the m-gram probabilities over the conventional modified Kneser-Ney (MKN) smoothing for both normal word-based and class-based models.

The best results have been achieved using an extended interpolation of: modified Kneser-Ney and hierarchical Pitman-Yor based models. All the models are morpheme-based with different class-based LMs involved in the interpolation using all the available classes. The experimental results have shown systematic improvements in recognition performance over all the test corpora.

Investigations on continuous space language models using feed-forward deep neural networks. Estimating the LMs in a discrete space suffers from inherent problems. The discrete nature of such models makes it difficult to achieve high levels of generalization even after applying the most efficient smoothing techniques, like the modified Kneser-Ney (MKN) smoothing of the backoff m-gram models. The discrete LMs have particularly poor performance in cases of data sparseness due to the lack of a notion of similarity among words. Since words are represented in a discrete space, it is not possible to perform a true interpolation to approximate the probabilities of the unseen m-grams. As a result, still extremely small probabilities are assigned to many valid word sequences even if large training corpora are used.

To avoid such problems, investigations have been made to estimate LMs in continuous space using feed-forward deep multilayer neural networks, shortly known as deep neural networks (DNNs). These models have been shown to be able to capture complex relationships from the input patterns. The resulting LMs are called *deep neural network LMs (DNNLMs)*. The basic idea is to convert the numerical indexes of the words into a continuous representation and to use DNNs as probability estimators to estimate conditional probabilities in continuous space. The DNNs have been trained via the stochastic back-propagation algorithm to predict the next word at the output layer given the history words at the input layer. Since the resulting distributions are smooth functions of the word representation, better generalization to unknown m-grams has been achieved.

In addition, DNNs have been used to estimate sub-word based rather than full-word based LMs. Moreover, word- and morpheme-level classes have been fed into the inputs of the DNNs in order to estimate robust probabilities for morphologically rich languages. To achieve the highest level of performance, DNNLMs have been interpolated with the backoff m-gram LMs. Furthermore, interpolations have been performed among: backoff LMs, normal DNNLMs without classes, and DNNLMs with input classes. The interpolated models have been used to perform lattice rescoring after a traditional recognition pass. This is a novel approach that combines the benefits of sub-word modeling and morphology-based classes, with the improved state-of-the-art performance of the DNNLMs. Using this combination, the experimental results have shown significant improvements in recognition performance.

Chapter 7

Outlook

In this thesis, efficient language modeling approaches have been proposed to perform LVCSR tasks for morphologically rich languages. In particular, several techniques have been investigated to build different types of sub-word based LMs. In addition, investigations have been made to integrate various morphology-based classes into the LM estimation process. Moreover, advanced state-of-the-art language modeling techniques have been visited. A special attention has been given to combine all these approaches together in order to achieve the optimal recognition performance. The presented approaches have succeeded to improve over the optimized traditional approaches by up to 7% relative. However, the following theoretical and experimental questions remain open and may serve as a starting point for further research:

Sub-word based language modeling.

- The simple approach of using sub-word based LMs comprising a set of morphemic sub-words and some fraction of the high frequent full-words performs very well and is hardly outperformed by more sophisticated approaches. The question is: is it possible to beat this approach by using different ideas, like the use of an additional character-based or phoneme-based LM component that drags the OOV rates to zero? or by using a hierarchical language modeling setup, where every region of words is handeled by a different LM that uses its own type of sub-word unit? It could be easier to accomplish this by using a weighted finite state transducer (WFST) based ASR search rather than the traditional tree-based search.
- All the investigated sub-word based approaches are using flat models, where a single lexicon contains all the different types of recognition units. During the search, the competition between different units to recognize the same word is rather limited. The question is: is it possible to build a LM that allows different types of units (e.g. fragments of arbitrary lengths) to compete together in recognizing words? In such a way, the search process itself is responsible for selecting the best combination of units to recognize a given sentence.
- One of the tedious and time consuming tasks that are highly effective in performing efficient recognition using sub-word based LMs is to optimize the number of vocabulary items used from each unit type. This is usually heavily dependent on the underlying language, corpus, and the used types of units. Is there some technique that enables the automation of this process? or is there some intuition for increasing or decreasing the number of items of a given type of unit? This is an open question that needs to be investigated.

Language modeling with morphology-based classes.

- The LM techniques investigated in this thesis are heavily dependent on the quality of the used classes. In this work, mainly morphology-based classes are utilized. Can those models be improved by introducing different types of classes? For example data-driven, syntactic, or semantic classes. If this is possible, how can we efficiently generate high quality classes of these types? A possible way to do that is to use classes derived from syntactic parse trees generated by syntactic parsers, or semantic classes generated by semantic annotators.
- The huge space of the factored language model (FLM) parameters turns the parameter optimization into a complex and time consuming process. The optimum parameters are always heavily dependent on the data and the available classes. Even if automatic search tools are used, it is not guaranteed to come up with the best FLM topology. The existing algorithms for automatic selection of the FLM parameters are not efficient enough and not all the possible values of the parameters are

representable. This causes a drop in the performance of such models compared to simpler models. The question is: can we come up with a better algorithm to efficiently optimize the parameters of the FLM such that all the possible values of the parameters can be represented? This could help so much to discover the potential of the FLMs.

- Since FLMs are quite general models of backoff, it could be interesting to think of extending them, for example by introducing different techniques of estimating probabilities at every node of the backoff graphs. One possible way is to use log-linear models or even neural network models for probability estimation. The question is: can these extensions be implemented efficiently within reasonable computational complexity? and how much improvement can they offer over the originally existing models?

Improved language modeling techniques.

- So far, we have investigated the use of hierarchical Pitman-Yor LMs (HPYLMs) and feed-forward deep neural network LMs (DNNLMs) as advanced state-of-the-art techniques that can improve over the traditional backoff m-gram LMs. These models are used in estimating sub-word LMs enriched with morphology-based classes. However, there are many other advanced techniques need to be explored. For example, how much improvement can be gained if a maximum entropy LM (MaxEntLM) is used? or alternatively, a random forest LM (RFLM)? or even if a structured LM is used?
- The deep neural network (DNN) approach to language modeling is only at its beginning. Therefore, there are a lot of open questions need to be answered. For example, what is the relationship between increasing the depth of the network and increasing the size of the projection or hidden layer? What is the effect of selecting a certain size of the short-list at the output layer compared to the effect of classing or factorizing the output layer? What are the differences between the methods of pre-training? Is pre-training still effective even if many hidden layers are used? Is it very influential to tie the projection matrices used to project different history words in continuous space? To which extent this tying is important? What if a fully connected input layer is used instead?
- Continuous space LMs are now getting increasingly popular. Among those continuous space LMs, there are three major approaches that have shown the best performance in typical LVCSR tasks. These are: the feed-forward deep neural network LM (DNNLM), the recurrent neural network LM (RNNLM), and the long short-term memory LM (LSTMLM). We still lack a comprehensive comparison among those approaches in order to well understand the following: what are the strengths, weaknesses, and similarities among them? What is the effect of changing different parameters here and there? Is a sufficiently deep DNNLM equivalent to a RNNLM? and to which extent they are equivalent?
- It was reported in few literature that a tied-mixture LM (TMLM) can outperform a shallow neural network LM (SNNLM) in some ASR tasks. Nevertheless, there are no known trials that have been made to compare the TMLM with the other types of continuous space LMs, like the DNNLM, RNNLM, and LSTMLM. Therefore, the following questions are raised: under which constraints the TMLM can outperform the SNNLM? Can the TMLM outperform those other types of continuous space LMs as well? Is it possible to improve the TMLM approach such that it outperforms the other approaches?

Appendix A

Corpora and Systems

The experiments of this thesis have been conducted using four system setups. The first system has been developed as a part of the modern standard Arabic (MSA) track of the GALE[1] project. The second system has been developed for the German language as a part of the Quaero[2] project. The third system has been developed for the Polish language also as a part of the Quaero project. The fourth system has been developed for Egyptian colloquial Arabic (ECA) as a preparatory step for developing Egyptian Arabic LVCSR systems within the framework of the BOLT[3] project. The first three systems; built for MSA, German and Polish languages; are developed at RWTH using the RASR toolkit[4] [Rybach & Gollan$^+$ 2009; Rybach & Hahn$^+$ 2011]. Whereas, the fourth system, built for ECA, is developed at IBM[5] using the IBM Attila speech recognition toolkit [Soltau & Saon$^+$ 2010].

A.1 Development and Evaluation Corpora

Table A.1 summarizes the development and evaluation corpora for all languages. The MSA corpora are provided within the GALE project, and they consist of audio material taken from broadcast news (BN) and broadcast conversation (BC) domains. The German corpora contain audio material from European parliament plenary sessions (EPPS), BN, and podcast sources. However, the Polish corpora contain only BN and podcast material. The German and Polish corpora are distributed within the Quaero project. The ECA corpora consist of spontaneous telephone conversations (TC), and they are publicly provided by LDC[6] under the name: CallHome Egyptian Arabic corpora[7].

Table A.1. Experimental corpora for: modern standard Arabic, German, Polish, and Egyptian colloquial Arabic. **BN**: broadcast news; **BC**: broadcast conversation; **EPPS**: European parliament plenary sessions; **PC**: Podcast; **TC**: Telephone Conversations.

language	corpus	project	domain	#sentences	#words	#chars	#hours
Modern standard Arabic	ar-dev07	GALE	BN+BC	880	19,002	105,142	2.5
	ar-eval07			1,521	29,430	161,384	4.0
	ar-tune07			229	4,739	26,582	0.7
German	gr-dev09	QUAERO	EPPS+BN	2,600	71,133	439,560	7.5
	gr-eval09		+PC	1,039	36,319	226,395	3.8
Polish	pl-dev10	QUAERO	BN+PC	2,750	31,029	194,063	3.2
	pl-eval10			2,720	31,771	192,722	3.5
Egyptian Arabic	eca-dev	Public	CallHome	6,415	37,197	14,6533	3.6
	eca-eval		TC	3,044	17,514	73,396	1.7

[1]GALE: Global Autonomous Language Exploitation.
[2]http://www.quaero.org
[3]BOLT: Broad Operational Language Translation.
[4]RASR: The RWTH Aachen University Speech Recognition System.
[5]While the author was visiting IBM T. J. Watson Research Center, NY, USA.
[6]http://ldc.upenn.edu
[7]LDC catalog numbers: LDC97S45, LDC97T19, LDC2002S37, and LDC2002T38.

A.2 Modern Standard Arabic Testing System

Acoustic front end. In this system, the acoustic front end consists of 16 mel-frequency cepstral coefficients (MFCCs) derived from a bank of 20 filters. The MFCCs are normalized using cepstral mean and variance normalization and augmented with a voicedness feature [Zolnay & Schlüter$^+$ 2002]. A linear discriminant analysis (LDA) matrix is used to project the concatenation of 9 consecutive feature vectors in a sliding window to 45 components. This reduced feature vector is augmented with phoneme posterior features estimated by a multilayer perceptron (MLP) neural network [Hoffmeister & Plahl$^+$ 2007].

Acoustic models. The acoustic models consist of within-word and across-word triphone models trained using 1100h of transcribed audio material taken from broadcast news (BN) and broadcast conversation (BC) domains. Some parts of the transcripts are derived automatically or are quick transcriptions. The basic acoustic models are trained based on maximum likelihood (ML) training. Speaker variations are compensated by applying vocal tract length normalization (VTLN) to the MFCC filterbank and speaker adaptive training (SAT) based on constrained maximum likelihood linear regression (CMLLR) (also known as feature space MLLR (fMLLR)). The models are enhanced by performing discriminative training (DT) based on minimum phone error (MPE) criterion [Povey & Woodland 2002b]. More details about the acoustic models are found in [Rybach & Hahn$^+$ 2007; Vergyri & Mandal$^+$ 2008].

Language models, vocabularies, and lexicons. The LM training corpora consist of around 206 Million running words including text data from Agile Arab text, FBIS, TDT4 and GALE BN and BC data. The recognition vocabularies are selected out of the text corpora using the ML approach as described in [Venkataraman & Wang 2003], where we seek to maximized the probability over some in-domain held-out text. The same text corpora are used to estimate backoff m-gram LMs with the modified Kneser-Ney (MKN) smoothing using the SRILM toolkit [Stolcke 2002]. The MSA lexicons contain around 4 pronunciations per word. The missing pronunciations are generated using the grapheme-to-phoneme (G2P) conversion approach described in [Bisani & Ney 2008].

Speech decoder. The speech decoder works in 3 passes. In the first pass, within-word acoustic models are used without speaker adaptation. The second pass uses across-word speaker adapted models via constrained maximum likelihood linear regression (CMLLR). Then, a third pass with additional maximum likelihood linear regression (MLLR) adaptation is performed. In each of the three passes, a bigram LM is used to produce lattices which are rescored using higher order backoff m-gram LMs. Optionally, at the third pass, N-best lists are produced and rescored using advanced LMs. The development and evaluation corpora are shown in Table A.1.

A.3 German Testing System

Acoustic front end. This system uses an acoustic front end that is similar to the MSA system front end described in Section A.2.

Acoustic models. The acoustic models are across-word triphone models trained using about 343h of transcribed audio material taken from BN, BC, European parliament plenary sessions (EPPS), read articles, dialogs, and web data. The acoustic models are trained based on ML training method. Similar to the MSA setup, speaker variations are compensated by applying VTLN and CMLLR based SAT [Sundermeyer & Nußbaum-Thom$^+$ 2011].

Language models, vocabularies, and lexicons. The LM training corpora consist of around 306 Million running words including data from TAZ newspaper, and web collected German news articles. The vocabularies are selected out of the text corpora by choosing some number of the top most frequent words. The same text corpora are used to estimate backoff m-gram LMs smoothed with the MKN smoothing via the

SRILM toolkit [Stolcke 2002]. The German lexicons contain around 1.2 pronunciations per word. The missing pronunciations are generated using the G2P conversion approach shown in [Bisani & Ney 2008].

Speech decoder. The speech decoder works in 2 passes. In the first pass, across-word acoustic models are used without speaker adaptation. The second pass uses the same acoustic models with speaker adaptation based on both CMLLR and MLLR. In each pass, a 4-gram backoff LM is used to construct the search space. No lattice or N-best rescoring is performed in the first pass. However, at the second pass, N-best lists are produced and rescored with advanced LMs. The development and evaluation corpora are also shown in Table A.1.

A.4 Polish Testing System

Acoustic front end. In this system, the acoustic front end uses VTLN normalized MFCC features. The VTLN warping factors are estimated using a Gaussian mixture model (GMM) classifier. The front end is completed by performing cepstral mean normalization and LDA over a window of 7 consecutive frames. This results in a 45 dimensional feature vector [Lööf & Gollan$^+$ 2009].

Acoustic models. A Spanish acoustic model is ported to Polish, through the use of a manually constructed phoneme mapping. The use of acoustic models from one language as a starting point for training acoustic models for a different language is called *cross-language bootstrapping* [Schultz & Waibel 2001]. This initial model is refined through iterative recognition and retraining of the un-transcribed audio data. The unsupervised retraining is performed on about 128h of un-transcribed recordings from EPPS. The training process includes the use of a maximum a-posteriori (MAP) adaptation and SAT based on CMLLR. The CMLLR matrices for SAT were estimated using confidence measures. The details of the training process are described in [Lööf & Gollan$^+$ 2009].

Language models, vocabularies, and lexicons. The LM training corpora consist of around 630 Million running words including data from EPPS, Kurier Lubelski, Nowosci, in addition to the official data provided by the Quaero project which are mainly taken from news and blogs. The vocabularies are selected out of the text corpora by choosing some number of the top most frequent words. Also, the same text corpora are used to estimate backoff m-gram LMs smoothed via the MKN smoothing using the SRILM toolkit [Stolcke 2002]. The Polish lexicons contain around 1.0 pronunciation per word. The missing pronunciations are generated using the G2P conversion approach described in [Bisani & Ney 2008].

Speech decoder. The speech decoder works in 3 passes. In the first pass, across-word acoustic models are used without speaker adaptation. The second pass applies speaker adaptation based on CMLLR. The third pass applies additional MLLR adaptation. A 5-gram LM is used to construct the search space of each pass. No lattice or N-best rescoring is performed. The development and evaluation corpora are shown in Table A.1.

A.5 Egyptian Colloquial Arabic Testing system

Acoustic front end. In this system, the acoustic front end uses VTLN normalized Perceptual linear predictive (PLP) features. The VTLN warping is performed using a set of warping functions that are chosen to maximize the likelihood of frames that align to speech under a model that uses a single, full-covariance Gaussian per context dependent state [Soltau & Kingsbury$^+$ 2005b]. The features are normalized using speaker-based cepstral mean and variance normalization. An LDA transform is used over a window of 9 consecutive frames in order to reduce the feature dimensionality to 40.

Acoustic models. The acoustic models are quinphone across-word models trained using about 16h of transcribed telephone conversational speech from the CallHome Egyptian Arabic speech corpora. The models are initialized using flat-start training and then refined with context dependent maximum likelihood (ML) and discriminative training (DT) based on boosted maximum mutual information (bMMI) criterion. The ML training is interleaved with the estimation of a global semi-tied covariance (STC) transform. Also, CMLLR transforms are applied in training. The detailed general training recipe is similar to the one described in [Soltau & Saon$^+$ 2010].

Language models, vocabularies, and lexicons. The LM training corpora have around 7 Million running words including: acoustic transcriptions (140k words), web text (5M words), extra sources (1.5M words). The vocabularies are selected out of the text corpora using the ML approach described in [Venkataraman & Wang 2003]. The same text corpora are used to estimate backoff m-gram LMs with the MKN smoothing via the SRILM toolkit [Stolcke 2002]. A grapheme-based lexicon is used where the pronunciations are the same as the word orthographies.

Speech decoder. The speech decoder works in 2 passes. In the first pass, CMLLR adaptation is performed. The second pass uses MLLR adaptation in a form of regression trees. In each pass, a trigram backoff m-gram LM smoothed using the MKN smoothing is used to construct the search space and to produce lattices which are then rescored using different advanced LMs.

Appendix B

Symbols and Acronyms

In this appendix, all relevant mathematical symbols and acronyms which are used in this thesis are defined for convenience. Detailed explanations are given in the corresponding chapters.

B.1 Mathematical Symbols

x_1^T	sequence of acoustic observations, $x_1^T := x_1 x_2 ... x_T$
w_1^N	word sequence, $w_1^N := w_1 w_2 w_N$
$p(w_1^N \| x_1^T)$	posterior probability of a word sequence w_1^N given the acoustic observations x_1^T
$p(x_1^T \| w_1^N)$	acoustic model, probability of a sequence of acoustic observations x_1^T given a word sequence w_1^N
$p(w_1^N)$	language model, prior probability of a word sequence w_1^N
$p(a_1^L \| w_1^N)$	pronunciation model, probability of a sequences of sub-word units a_1^L given a sequence of words w_1^N
s_1^T	sequence of states in a hidden Markov model
$p(x_1^T, s_1^T \| w_1^N)$	joint probability of observing the sequence x_1^T of acoustic feature vectors and the state sequence s_1^T given a hidden Markov model representing the word sequence w_1^N
$p(x_t \| s_t; w_1^N)$	emission probability of observing a feature vector x_t given state s_t in a hidden Markov model representing the word sequence w_1^N
$\mathcal{N}(\mu, \sigma^2)$	Gaussian distribution with mean μ and variance σ^2
$x \sim \mathcal{N}(\mu, \sigma^2)$	Gaussian distributed random variable x with mean μ and variance σ^2
$p(w_n \| w_{n-m+1}^{n-1})$	conditional m-gram probability of a word w_n given $m-1$ history words w_{n-m+1}^{n-1}
$H_p(T)$	cross-entropy of a language model p on a dataset T
$PP_p(T)$	perplexity of a language model p on a dataset T
$\mathrm{GER}(\mathbf{L})$	graph error rate of a lattice $\mathbf{L}$
$\mathrm{Lev}(w_1^N, \tilde{w}_1^{\tilde{N}})$	Levenshtein distance between a hypothesized word sequence w_1^N and a reference word sequence $\tilde{w}_1^{\tilde{N}}$
$\mathrm{GD}(\mathbf{L})$	graph density of a lattice $\mathbf{L}$
$E(\mathbf{L})$	set of arcs in a lattice $\mathbf{L}$
$\mathrm{NER}(\mathbf{B})$	N-best error rate for a set of N-best sentences $\mathbf{B}$
ϵ	the empty word
G, Φ	sets of letters and phonemes in a *G2P* model
G^*, Φ^*	sets of all possible orthographies and pronunciations that could be used by a *G2P* model
g, φ	word orthographic form and its corresponding pronunciation in a *G2P* model
(g, φ)	graphone, a joint pair of an orthographic form g and a pronunciation φ
q_1^N	sequence of N graphones $q_1^N := q_1 q_2 ... q_N$, where $q_k = (g_k, \varphi_k)$
Q	graphone inventory in a *G2P* model

$S(g, \varphi)$	set of all joint segmentations of an orthographic form g and a pronunciation φ
w_n, c_n	word and assigned class at time (position) n
$\hat{c}(\cdot)$	deterministic mapping function that maps the word to its corresponding class
$N(\cdot)$	count of occurrences of some event in a given dataset, ex: $N(w_n, c_n)$ = count of occurrences of word w_n in class c_n
$\delta_{i=j}$	Kronecker delta function with a boolean subscript parameter, equals one for $i = j$, and zero otherwise
λ_i	weight assigned to model i during linear interpolation or model combination
Λ	weights of a set of models within model combination, $\Lambda = (\lambda_1, ..., \lambda_K)$ for K models $(p_1, ..., p_K)$
$f_n^{1:K}$	bundle of K parallel factors used in a factored LM for a word at position n, $f_n^{1:K} = f_n^1, f_n^2, ..., f_n^K$
$g(f, f_1, f_2, ..., f_M)$	non-negative generalized backoff function in a factored LM with $M + 1$ variables, 1 child variable f and M parent variables $f_1, f_2, ..., f_M$
$\|s\|$	number of items in a set s
$PY(d, \theta, G_0)$	PY process with a discount parameter $0 \leq d < 1$, a strength parameter $\theta > -d$, and a prior mean vector G_0
h_n	context (or history) of a word w_n in an m-gram LM that consists of $m - 1$ previous words, $h_n = w_{n-m+1}^{n-1}$
$\bar{h}_n$	features of the history words h_n; for every word in h_n, there could be one or more corresponding features in $\bar{h}_n$
P	number of units in a projection layer of a neural network LM
H	number of units in a hidden layer of a neural network LM
N	number of units in the output layer of a neural network LM, might be the size of a short-list of the input vocabulary
c_l	activity of a unit l in the projection layer of a neural network LM
d_j	activity of a unit j in the hidden layer of a neural network LM
o_i	activity of a unit i in the output layer of a neural network LM before applying the activation function
p_i	activity of a unit i in the output layer of a neural network LM after applying the activation function, usually softmax activation
t_i	target output of a unit i in a neural network LM
b_j, k_i	biases of the hidden and the output layers of a neural network LM
u_{jl}	neural network weight linking the unit l of the projection layer to the unit j of the hidden layer
v_{ij}	neural network weight linking the unit j of the hidden layer to the unit i of the output layer
E	regularized error function used in a neural network training
$\hat{E}$	non-regularized error function used in a neural network training
λR	weight decay regularization term with a weight decay coefficient λ used in a regularized error function of a neural network training
η	learning rate used in a neural network training
ω	abstract parameter of a neural network, usually represents a weight or a bias
$\widehat{h}(\mathbf{x})$	hidden representation of input $\mathbf{x}$ generated using an unsupervised neural network pre-training procedure

B.2 Acronyms

AENN	**A**uto-**E**ncoder **N**eural **N**etwork
ASR	**A**utomatic **S**peech **R**ecognition
BAMA	**B**uckwalter **A**rabic **M**orphological **A**nalyzer
BC	**B**roadcast **C**onversation
BIC	**B**ayesian **I**nformation **C**riterion
bMMI	**b**oosted **M**aximum **M**utual **I**nformation
BN	**B**roadcast **N**ews
CD	**C**ontrastive **D**ivergence
CER	**C**haracter **E**rror **R**ate
CLM	**C**lass-based **L**anguage **M**odel
CMLLR	**C**onstrained **M**aximum **L**ikelihood **L**inear **R**egression
CN	**C**onfusion **N**etwork
CRP	**C**hinese **R**estaurant **P**rocess
DARPA	**D**efense **A**dvanced **R**esearch **P**rojects **A**gency
DAT	**D**ialog **A**ct **T**agging
DMC	**D**iscriminative **M**odel **C**ombination
DNN	**D**eep **N**eural **N**etwork
DNNLM	**D**eep **N**eural **N**etwork **L**anguage **M**odel
DP	**D**ynamic **P**rogramming
DT	**D**iscriminative **T**raining
EA	**E**volutionary **A**lgorithms
ECA	**E**gyptian **C**olloquial **A**rabic
EM	**E**xpectation **M**aximization
EPPS	**E**uropean **P**arliament **P**lenary **S**essions
FFT	**F**ast **F**ourier **T**ransform
FLM	**F**actored **L**anguage **M**odel
fMLLR	**f**eature space **M**aximum **L**ikelihood **L**inear **R**egression
G2P	**G**rapheme-to-**P**honeme Conversion
GA	**G**enetic **A**lgorithm
GD	**G**raph **D**ensity
GER	**G**raph **E**rror **R**ate
GMLM	**G**aussian **M**ixture **L**anguage **M**odel
GMM	**G**aussian **M**ixture **M**odel
GPB	**G**eneralized **P**arallel **B**ackoff
GT	**G**amma**t**one filter
HCRP	**H**ierarchical **C**hinese **R**estaurant **P**rocess
HLDA	**H**eteroscedastic **L**inear **D**iscriminant **A**nalysis
HMM	**H**idden **M**arkov **M**odel
HPY	**H**ierarchical **P**itman-**Y**or
HPYCLM	**H**ierarchical **P**itman-**Y**or **C**lass-based **L**anguage **M**odel
HPYLM	**H**ierarchical **P**itman-**Y**or **L**anguage **M**odel
IBM	**I**nternational **B**usiness **M**achines Corporation

IKN	**I**nterpolated **K**neser-**N**ey Smoothing
KN	**K**neser-**N**ey Smoothing
LDA	**L**inear **D**iscriminant **A**nalysis
LM	**L**anguage **M**odel
LSA	**L**atent **S**emantic **A**nalysis
LSTM	**L**ong **S**hort-**T**erm **M**emory Neural Network
LSTMLM	**L**ong **S**hort-**T**erm **M**emory **L**anguage **M**odel
LVCSR	**L**arge **V**ocabulary **C**ontinuous **S**peech **R**ecognition
MADA	**M**orphological **A**nalyzer and **D**isambiguator tool for **A**rabic
MAP	**M**aximum **A**-posterior
MDL	**M**inimum **D**escription **L**ength
MFCC	**M**el-**F**requency **C**epstral **C**oefficients
MKN	**M**odified **K**neser-**N**ey
ML	**M**aximum **L**ikelihood
MLLR	**M**aximum **L**ikelihood **L**inear **R**egression
MLP	**M**ultilayer **P**erceptron Neural Network
MPE	**M**inimum **P**hone **E**rror
MSA	**M**odern **S**tandard **A**rabic
MSE	**M**ean **S**quare **E**rror
MT	**M**achine **T**ranslation
NER	**N**-best **E**rror **R**ate
NNLM	**N**eural **N**etwork **L**anguage **M**odel
OOV	**O**ut-**O**f-**V**ocabulary
PER	**P**honeme **E**rror **R**ate
PLP	**P**erceptual **L**inear **P**redictive features
POI	**P**robability **O**f **I**mprovement
POS	**P**art-**O**f-**S**peech
PPL	**P**er**pl**exity
PY	**P**itman-**Y**or
RBM	**R**estricted **B**oltzmann **M**achines
RNN	**R**ecurrent **N**eural **N**etwork
RNNLM	**R**ecurrent **N**eural **N**etwork **L**anguage **M**odel
RWTH	**R**heinisch **W**estfälische **T**echnische **H**ochschule
SAT	**S**peaker **A**daptive **T**raining
SNN	**S**hallow **N**eural **N**etwork
SNNLM	**S**hallow **N**eural **N**etwork **L**anguage **M**odel
SRILM	**SRI L**anguage **M**odeling Toolkit
STC	**S**emi-**T**ied **C**ovariance
SVD	**S**ingular **V**alue **D**ecomposition
TC	**T**elephone **C**onversations
TDP	**T**ime **D**istortion **P**enalty
TMLM	**T**ied-**M**ixture **L**anguage **M**odel
TMLM-CO	**T**ied-**M**ixture **L**anguage **M**odel with bigram **CO**-occurrence based features
TMLM-NN	**T**ied-**M**ixture **L**anguage **M**odel with **N**eural **N**etwork based features

VTLN	**V**ocal **T**ract **L**ength **N**ormalization
WER	**W**ord **E**rror **R**ate
WFST	**W**eighted **F**inite **S**tate **T**ransducer
WSJ	**W**all **S**treet **J**ournal

List of Figures

1.1 Basic architecture of a statistical automatic speech recognition system according to [Ney 1990]. . . . 3
1.2 6-state hidden Markov model in Bakis topology for the triphone ${}_{s}eh_{v}$ in the word "seven" and the resulting trellis for a time alignment. The HMM segments are denoted by <1>, <2>, and <3>. . . . 6
1.3 An example of a word lattice (taken from [Schwenk 2007]). The lattice is produced using a trigram LM, where each word has a unique bigram context. For simplicity, acoustic and language model scores are not shown on arcs ([**fw**]: filler word; [**breath**]: breath noise). . . . 11
1.4 An example of a confusion network (CN) derived from a lattice. The figure shows: the original lattice, a derived CN, and an intermediate lattice in which all paths have the same length. The positions for the insertions of the ϵ-arcs are derived from the CN according to the algorithm described in [Hoffmeister 2011]. The number that appears on each arc corresponds to the CN slot to which the arc is assigned. . . . 12

3.1 Optimization of the number of full-words retained in the sub-word based vocabularies. . . . 42
3.2 Optimization of the overall vocabulary sizes for full-word and sub-word based experiments. 43
3.3 The best sub-word based experiments compared to the best full-word based experiments on Arabic, German, and Polish corpora. . . . 44

4.1 (a) An example of a general backoff graph showing all possible backoff paths from top to bottom. (b) An example of a backoff graph where only a subset of the possible backoff paths are allowed. . . . 57
4.2 Topologies of the Arabic FLMs using the format specifications of the SRILM-FLM extensions (**W**: word; **M**: morph; **L**: lexeme; **P**: pattern). . . . 62
4.3 Backoff graphs for $AR\text{–}FLM_{1:5}$, detailed topologies are given in Figure 4.2 (**W**: word; **M**: morph; **L**: lexeme; **P**: pattern). . . . 63
4.4 Topologies of the German FLMs using the format specifications of the SRILM-FLM extensions (**W**: word; **L**: lexeme; **I**: class-index; **P**: POS-tag). . . . 65
4.5 Backoff graphs for $GR\text{–}FLM_{1:7}$, detailed topologies are given in Figure 4.4 (**W**: word; **L**: lexeme; **I**: class-index; **P**: POS-tag). . . . 66
4.6 Comparison of recognition WERs [%] on Arabic and German corpora using different LMs. 73
4.7 Interpolation weights of individual Arabic morpheme-based LMs, models with negligible weights are not shown in the figure. . . . 74
4.8 Interpolation weights of individual German morpheme-based LMs, models with negligible weights are not shown in the figure. . . . 75

5.1 Architecture of a shallow NNLM (SNNLM) that estimates the model $p(w_n|w_{n-m+1}^{n-1})$. . . . 81
5.2 Architecture of a deep NNLM (DNNLM) that estimates the model $p(w_n|w_{n-m+1}^{n-1})$. . . . 82
5.3 Architecture of a deep NNLM (DNNLM) with input classes. The input encoding uses separate vectors for words and their classes for every history position. The network estimates the model $p(w_n|w_{n-1}c_{n-1}w_{n-2}c_{n-2})$. . . . 83
5.4 Architecture of a deep NNLM (DNNLM) with input classes. The input encoding uses one combined vector for each word and its class for every history position. The network estimates the model $p(w_n|w_{n-1}c_{n-1}w_{n-2}c_{n-2})$. . . . 84
5.5 General steps of a greedy layer-wise unsupervised pre-training algorithm. . . . 88

5.6 Optimization of the number of decomposable full-words retained in the morpheme-based vocabulary performed over *eca-dev* corpus using overall vocabulary size of 250k (best WER = 56.8% with 5k full-words). Baseline WER on *eca-dev* using 350k full-words vocabulary = 56.9%. 93
5.7 Comparison of recognition WERs [%] on Egyptian Arabic *eca-eval* corpus using different LMs. 94
5.8 Interpolation weights of individual morpheme-based LMs. 95

List of Tables

1.1 Different Arabic words derived from the same root **"ktb"**. 15

3.1 Arabic solar and lunar consonants (**bw**: using Buckwalter transliteration; **ar**: using Arabic script). 31
3.2 An example of the alignment process for the word-pronunciation pair *(phase,feɪz)*. 35
3.3 Recognition experiments on Arabic corpora using morpheme-based LMs with 70k vocabularies. 35
3.4 Recognition experiments on Arabic corpora using full-words, morphemes, and diacritized morphemes for LMs with very large vocabularies. 36
3.5 word- and character-level perplexities for full-word and sub-word based LMs on Arabic corpora (**inv**: perplexity for in-vocabulary text excluding the unk symbol; **all**: perplexity for the whole text including the unk symbol). 37
3.6 Recognition experiments on German corpora using morpheme-based LMs with 100k vocabularies. 37
3.7 Recognition experiments on German corpora using 100k full-words as a baseline vocabulary and adding different fragment-based and morpheme-based graphones. 38
3.8 Recognition experiments on German corpora using full-words, morphemes, and morphemic graphones for LMs with very large vocabularies. 39
3.9 word- and character-level perplexities for full-word and sub-word based LMs on German corpora (**inv**: perplexity for in-vocabulary text excluding the unk symbol; **all**: perplexity for the whole text including the unk symbol). 39
3.10 Recognition experiments on Polish corpora using morpheme- and syllable-based LMs with 300k vocabularies. 40
3.11 Recognition experiments on Polish corpora using full-words, syllables, and syllabic graphones for LMs with very large vocabularies. 41
3.12 word- and character-level perplexities for full-word and sub-word based LMs on Polish corpora (**inv**: perplexity for in-vocabulary text excluding the unk symbol; **all**: perplexity for the whole text including the unk symbol). 41
3.13 Analysis of improvements in the best sub-word based system compared to the best full-word based system for Arabic, German, and Polish corpora. Amount of reduction in WER is divided into (**ins**: reduction in insertion rate; **OOV del/sub**: reduction in deletion/-substitution rate of OOV words; **INV del/sub**: reduction in deletion/substitution rate of INV words). Note: a negative reduction means an increase. 45
3.14 Examples of words for which recognition is improved using the best sub-word based systems. 45
3.15 List of participants in different evaluation campaigns. 46
3.16 Quaero German ASR evaluation 2010. 46
3.17 Quaero German ASR evaluation 2011. 47
3.18 Quaero German ASR evaluation 2012. 47
3.19 Quaero Polish ASR evaluation 2012. 47
3.20 Quaero German ASR evaluation 2013. 47
3.21 IWSLT German ASR evaluation 2013. 48
3.22 OpenHaRT Arabic handwriting recognition evaluation 2013. 48

4.1 Recognition experiments on Arabic ar-tune07 corpus using different factored LMs (**vocabulary**: 70k full-words, OOV rate = 3.6%, N-best size = 1000, **N-best error rate (NER)** = 7.3%; **W**: word; **M**: morph; **L**: lexeme; **P**: pattern). 63

4.2 Perplexities for the German FLMs $GR-FLM_{1:7}$ measured on the German gr-dev09 corpus. Exact FLM topologies are given in Figures 4.4 and 4.5 (**word-based**: 100k full-words vocab; **morpheme-based**: 100k morpheme-based vocab with 5k full-words + 95k morphemes; **W**: word; **L**: lexeme; **I**: class-index; **P**: POS-tag). 64

4.3 Recognition WERs [%] on German corpora using different factored LMs (N-best size = 1000; **word-based**: 100k full-words, OOV rate = [gr-dev09: 5.0%, gr-eval09: 4.8%], N-best error rate (NER) = [gr-dev09: 23.6%, gr-eval09: 21.4%]; **morpheme-based**: 5k full-words + 95k morphemes, OOV rate = [gr-dev09: 1.5%, gr-eval09: 1.4%], N-best error rate (NER) = [gr-dev09: 20.0%, gr-eval09: 18.8%]). 67

4.4 Recognition WERs [%] on Arabic ar-dev07 corpus using stream- and class-based LMs built over words and morphemes (N-best size = 1000; **word-based**: 70k full-words, OOV rate = 3.7%, N-best error rate (NER) = 9.5%; **morpheme-based**: 20k full-words + 50k morphemes, OOV rate = 1.4%, N-best error rate (NER) = 8.2%). 67

4.5 Recognition experiments on Arabic corpora using class-based LMs, factored LM ($AR-FLM_4$), and hierarchical Pitman-Yor LMs built over **full-words** (vocabulary: 750k full-words; OOV rate = [ar-dev07: 0.5%, ar-eval07: 0.7%]; N-best size = 1000; N-best error rate (NER) = [ar-dev07: 7.6%, ar-eval07: 9.1%]). 68

4.6 Word- and character-level perplexities on Arabic corpora for LMs that utilize word-level classes (**inv**: perplexity for in-vocabulary text excluding the unk symbol; **all**: perplexity for the whole text including the unk symbol). 68

4.7 Recognition experiments on Arabic corpora using class-based LMs, factored LM ($AR-FLM_4$), and hierarchical Pitman-Yor LMs built over **morphemes** (vocabulary: 20k full-words + 236k morphemes; OOV rate = [ar-dev07: 0.5%, ar-eval07: 0.7%]; N-best size = 1000; N-best error rate (NER) = [ar-dev07: 7.6%, ar-eval07: 8.8%]). 69

4.8 Morpheme- and character-level perplexities on Arabic corpora for LMs that utilize morpheme-level classes (**inv**: perplexity for in-vocabulary text excluding the unk symbol; **all**: perplexity for the whole text including the unk symbol). 69

4.9 Number of instances of every class for Arabic vocabularies. 69

4.10 Recognition WERs [%] on German corpora using stream- and class-based LMs built over words and morphemes (N-best size = 1000; **word-based**: 100k full-words, OOV rate = [gr-dev09: 5.0%, gr-eval09: 4.8%], N-best error rate (NER) = [gr-dev09: 23.6%, gr-eval09: 21.4%]; **morpheme-based**: 5k full-words + 95k morphemes, OOV rate = [gr-dev09: 1.5%, gr-eval09: 1.4%], N-best error rate (NER) = [gr-dev09: 20.0%, gr-eval09: 18.8%]). 70

4.11 Recognition experiments on German corpora using class-based LMs, factored LM ($GR-FLM_5$), and hierarchical Pitman-Yor LMs built over **full-words** (vocabulary: 750k full-words; OOV rate = [gr-dev09: 2.3%, gr-eval09: 2.1%]; N-best size = 1000; N-best error rate (NER) = [gr-dev09: 20.6%, gr-eval09: 18.9%]). 71

4.12 Word- and character-level perplexities on German corpora for LMs that utilize word-level classes (**inv**: perplexity for in-vocabulary text excluding the unk symbol; **all**: perplexity for the whole text including the unk symbol). 71

4.13 Recognition experiments on German corpora using class-based LMs, factored LM ($GR-FLM_5$), and hierarchical Pitman-Yor LMs built over **morphemes** (vocabulary: 5k full-words + 495k morphemes; OOV rate = [gr-dev09: 0.9%, gr-eval09: 0.7%]; N-best size = 1000; N-best error rate (NER) = [gr-dev09: 19.1%, gr-eval09: 17.3%]). 72

4.14 Morpheme- and character-level perplexities on German corpora for LMs that utilize morpheme-level classes (**inv**: perplexity for in-vocabulary text excluding the unk symbol; **all**: perplexity for the whole text including the unk symbol). 72

4.15 Number of instances of every class for German vocabularies. 73

5.1 Recognition experiments on CallHome Egyptian colloquial Arabic (ECA) evaluation corpus *eca-eval* using word-based neural network LMs (NNLMs) for lattice rescoring. vocabulary: 350k full-words, OOV rate = 1.4%, graph (lattice) error rate (GER) = 37.2%. 92

5.2 Recognition experiments on CallHome Egyptian colloquial Arabic (ECA) evaluation corpus *eca-eval* using morpheme-based neural network LMs (NNLMs) for lattice rescoring. vocabulary: 250k (5k words + 245k morphemes), OOV rate = 0.9%, graph (lattice) error rate (GER) = 32.3%. 93
5.3 Word-/morpheme-level and character-level perplexities on CallHome Egyptian colloquial Arabic (ECA) evaluation corpus *eca-eval* for different LMs (**inv**: perplexity for in-vocabulary text excluding the unk symbol; **all**: perplexity for the whole text including the unk symbol; **units**: words or morphemes). 94

A.1 Experimental corpora for: modern standard Arabic, German, Polish, and Egyptian colloquial Arabic. **BN**: broadcast news; **BC**: broadcast conversation; **EPPS**: European parliament plenary sessions; **PC**: Podcast; **TC**: Telephone Conversations. 103

Bibliography

M. Adda-Decker. A corpus-based decompounding algorithm for German lexical modeling in LVCSR. In *Proc. European Conf. on Speech Communication and Technology*, pages 257 – 260, Geneva, Switzerland, September 2003.

M. Adda-Decker and G. Adda. Morphological decomposition for ASR in German. In *Workshop on Phonetics and Phonology in ASR*, pages 129 – 143, Saarbrücken, Germany, March 2000.

A. M. H. J. Aertsen, P. I. M. Johannesma, and D. J. Hermes. Spectro-temporal receptive fields of auditory neurons in the grassfrog. *Biological Cybernetics*, 38:235 – 248, November 1980.

M. Afify, L. Nguyen, B. Xiang, S. Abdou, and J. Makhoul. Recent progress in Arabic broadcast news transcription at BBN. In *Proc. European Conf. on Speech Communication and Technology*, volume 1, pages 1637 – 1640, Lisbon, Portugal, September 2005.

M. Afify, R. Sarikaya, H-K J. Kuo, L. Besacier, and Y. Gao. On the use of morphological analysis for dialectal Arabic speech recognition. In *Interspeech*, volume 1, pages 277 – 280, Pittsburgh, PA, USA, September 2006.

M. Afify, O. Siohan, and R. Sarikaya. Gaussian mixture language models for speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 4, pages IV–29 – IV–32, Honululu, HI, USA, April 2007.

A. Alexandrescu and K. Kirchhoff. Factored neural language models. In *Proc. Human Language Technology Conf. of the North American Chapter of the ACL*, NAACL-Short '06, pages 1 – 4, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

C. Allauzen, M. Mohri, B. Roark, and M. Riley. A generalized construction of integrated speech recognition transducers. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Montreal, Canada, May 2004.

P. Alleva, X. D. Huang, and M. Y. Hwang. Improvements on the pronunciation prefix tree search organization. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 1, pages 133 – 136, Atlanta, GA, USA, May 1996.

E. Arisoy, T. Sainath, B. Kingsbury, and B. Ramabhadran. Deep neural network language models. In *NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, pages 20 – 28, Montreal, Canada, June 2012.

P. Auer, M. Herbster, and M. K. Warmuth. Exponentially many local minima for single neurons. In David S. Touretzky, Michael Mozer, and Michael E. Hasselmo, editors, *Proc. Neural Information Processing Systems (NIPS) Foundation*, pages 316 – 322. MIT Press, 1996.

L. R. Bahl, F. Jelinek, and R. L. Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5:179 – 190, March 1983.

L. R. Bahl, P. F. Brown, P. V. de Souza, and R. L. Mercer. Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 49 – 52, Tokyo, Japan, May 1986.

L. R. Bahl, M. Padmanabhan, D. Nahamoo, and P. S. Gopalakrishnan. Discriminative training of Gaussian mixture models for large vocabulary speech recognition systems. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 613 – 616, Atlanta, GA, USA, May 1996.

J. K. Baker. Stochastic modeling for automatic speech understanding. In D. R. Reddy, editor, *Speech Recognition*, pages 512 – 542. Academic Press, New York, NY, USA, 1975.

R. Bakis. Continuous speech word recognition via centisecond acoustic states. In *ASA Meeting*, Washington, DC, USA, April 1976.

M. C. Bateson. *Arabic language handbook*. Georgetown Classics in Arabic Language and Linguistics Series. Georgetown University Press, Portland, OR, USA, 2003.

L. E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. In O. Shisha, editor, *Inequalities*, volume 3, pages 1 – 8. Academic Press, New York, NY, 1972.

T. Bayes. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 53:370 – 418, 1763. Reprinted in *Biometrika*, vol. 45, no. 3/4, pp. 293–315, December 1958.

I. Bazzi and J. R. Glass. Modeling out-of-vocabulary words for robust speech recognition. In *Proc. Int. Conf. on Spoken Language Processing*, Beijing, China, October 2000.

T. C. Bell, J. G. Cleary, and I. H. Witten. *Text compression*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990. ISBN 0-13-911991-4.

J. Bellegarda. Large vocabulary speech recognition with multispan language models. *IEEE Transactions on Speech and Audio Processing*, 8(1):76 – 84, 2000.

R. E. Bellman. Dynamic programming. Princeton University Press, Princeton, NJ, USA, 1957.

Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1 – 127, January 2009.

Y. Bengio and R. Ducharme. A neural probabilistic language model. In *Advances in Neural Information Processing Systems*, volume 13, pages 932 – 938, 2001.

Y. Bengio and J.-S Sénécal. Quick training of probabilistic neural nets by importance sampling. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2003.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157 – 166, 1994.

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137 – 1155, March 2003.

A. Berton, P. Fetter, and P. Regal-Brietzmann. Compound words in large-vocabulary German speech recognition systems. In *Proc. Int. Conf. on Spoken Language Processing*, volume 2, pages 1165 – 1168, Philadelphia, PA, USA, October 1996.

K. Beulen, S. Ortmanns, and C. Elting. Dynamic programming search techniques for across-word modeling in speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 609 – 612, Phoenix, AZ, March 1999.

P. Beyerlein. Discriminative model combination. In *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, pages 238 – 245, Santa Barbara, CA, USA, December 1997.

P. Beyerlein. Discriminative model combination. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 481 – 484, Seattle, WA, USA, May 1998.

J. Bilmes and K. Kirchhoff. Factored language models and generalized parallel backoff. In *Proc. Human Language Technology Conf. of the North American Chapter of the ACL*, volume 2, pages 4 – 6, Edmonton, Canada, May 2003.

J. Bilmes, K. Asanovic, C. Chee-Whye, and J. Demmel. Using PHiPAC to speed error back-propagation learning. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 5, pages 4153 – 4156, Munich, Germany, 1997.

M. Bisani and H. Ney. Multigram-based grapheme-to-phoneme conversion for LVCSR. In *Interspeech*, pages 933 – 936, Geneva, Switzerland, September 2003.

M. Bisani and H. Ney. Bootstrap estimates for confidence intervals in ASR performance evaluation. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 1, pages 409 – 412, Montreal, Canada, May 2004.

M. Bisani and H. Ney. Open vocabulary speech recognition with flat hybrid models. In *Interspeech*, pages 725 – 728, Lisbon, Portugal, September 2005.

M. Bisani and H. Ney. Joint-sequence models for grapheme-to-phoneme conversion. *Speech Communication*, 50(5):434 – 451, May 2008.

C. M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, USA, 1 edition, January 1996. ISBN 9780198538646.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993 – 1022, 2003.

M. Brand. Structure learning in conditional probability models via an Entropic prior and parameter extinction. *Neural Computation*, 11(5):1155 – 1182, 1999.

P. Brown, P. deSouza, R. Mercer, V. Della Pietra, and J. Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18:467 – 479, 1992.

T. Buckwalter. *Buckwalter Arabic Morphological Analyzer Version 2.0*. Number LDC2004L02. Linguistic Data Consortium (LDC) catalogue, 2004. ISBN 1-58563-324-0.

W. Byrne, J. Hajič, P. Ircing, P. Krbec, and J. Psutka. Morpheme based language models for speech recognition of Czech. In *Text, Speech and Dialogue*, volume 1902 of *Lecture Notes in Computer Science*, pages 139 –162. 2000.

M. Castro and F. Prat. New directions in connectionist language modeling. 2686:598 – 605.

M. J. Castro-Bleda, V. Polvoreda, and F. Prat. Connectionist n-gram models by using mlps. In *Proceedings of the Second Workshop on Natural Language Processing and Neural Networks*, pages 16 – 22, Tokyo, Japan, November 2001.

K. Çarki, P. Geutner, and T. Schultz. Turkish LVCSR: towards better speech recognition for agglutinative languages. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 1, pages 3688 – 3691, Istanbul, Turkey, June 2000.

B. Chen, Q. Zhu, and N. Morgan. Learning long-term temporal features in LVCSR using neural networks. In *Interspeech*, Jeju Island, Korea, October 2004.

S. F. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. In *Proc. Annual Meeting of the Association for Computational Linguistics*, ACL '96, pages 310 – 318, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics.

S. F. Chen and R. Rosenfeld. A survey of smoothing techniques for ME models. *Speech and Audio Processing, IEEE Transactions on*, 8(1):37 – 50, jan 2000.

S. S. Chenand and P. S. Gopalakrishnan. Speaker, environment and channel change detection and clustering via the Bayesian information criterion. In *DARPA Broadcast News Transcription and Understanding Workshop*, pages 127 – 132, February 1998.

G. Choueiter, D. Povey, S. F. Chen, and G. Zweig. Morpheme-based language modeling for Arabic LVCSR. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 1, pages 1053 – 1056, Toulouse, France, May 2006.

P. Clarkson and T. Robinson. Towards improved language model evaluation measures. In *Proc. European Conf. on Speech Communication and Technology*, pages 1927 – 1930, Budapest, Hungary, September 1999.

P. Clarkson and T. Robinson. Improved language modelling through better language model evaluation measures. *Computer Speech & Language*, 15(1):39 – 53, 2001.

T. M. Cover and J. A. Thomas. *Elements of information theory.* Wiley-Interscience, 99th edition, August 1991. ISBN 0471062596.

M. Creutz. Unsupervised segmentation of words using prior distributions of morph length and frequency. In *Proc. Annual Meeting of the Association for Computational Linguistics*, pages 280 – 287, Sapporo, Japan, July 2003.

M. Creutz. *Induction of the morphology of natural language: Unsupervised morpheme segmentation with application to automatic speech recognition.* PhD thesis, Helsinki University of Technology, Finland, 2006.

M. Creutz and K. Lagus. Unsupervised discovery of morphemes. In *Workshop on Morphological and Phonological Learning of ACL*, pages 21 – 30, Philadelphia, PA, USA, July 2002.

M. Creutz and K. Lagus. Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. Technical report, Computer and Information Science Helsinki University of Technology, Finland, March 2005.

M. Creutz, T. Hirsimäki, M. Kurimo, A. Puurula, J. Pylkkönen, V. Siivola, M. Varjokallio, E. Arisoy, M. Saraclar, and A. Stolcke. Morph-based speech recognition and modeling of out-of-vocabulary words across languages. *ACM Transactions on Speech and Language Processing*, 5(1), December 2007.

G. Dahl, M. Ranzato, A. Mohamed, and G. E. Hinton. Phone recognition with the mean-covariance restricted Boltzmann machine. In *Proc. Neural Information Processing Systems (NIPS) Foundation*, pages 469 – 477, Vancouver, BC, Canada, 2010.

R. I. Damper, Y. Marchand, J. D. Marsters, and A. Bazin. Aligning letters and phonemes for speech synthesis. In *5th ISCA Speech Synthesis Workshop*, pages 209 – 214, Pittsburg, PA, USA, June 2004.

K. Darwish. Building a shallow Arabic morphological analyzer in one day. In *ACL Workshop on Computational Approaches to Semitic Languages*, Philadelphia, PA, USA, July 2002.

S. B. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-28(4):357 – 366, August 1980.

S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(B):1 – 38, 1977.

Anne-Marie Derouault and Bernard Merialdo. Natural language modeling for phoneme-to-text transcription. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):742 – 749, 1986.

E. W. Dijkstra. A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269 – 271, 1959.

G. R. Doddington, M. A. Przybocki, A. F. Martin, and D. A. Reynolds. The NIST speaker recognition evaluation – overview, methodology, systems, results, perspective. *Speech Communication*, 31(2 – 3): 225 – 254, June 2000.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. John Wiley & Sons, New York, NY, USA, 2001.

K. Duh and K. Kirchhoff. Automatic learning of language model structure. In *the 20th International Conference on Computational Linguistics (COLING)*, pages 148 – 154, Geneva, Switzerland, August 2004.

A. El-Desoky, C. Gollan, D. Rybach, R. Schlüter, and H. Ney. Investigating the use of morphological decomposition and diacritization for improving Arabic LVCSR. In *Interspeech*, pages 2679 – 2682, Brighton, UK, September 2009.

A. El-Desoky, R. Schlüter, and H. Ney. A hybrid morphologically decomposed factored language models for Arabic LVCSR. In *Proc. Human Language Technology Conf. of the North American Chapter of the ACL*, pages 701 – 704, Los Angeles, CA, USA, June 2010.

A. El-Desoky, M. Shaik, R. Schlüter, and H. Ney. Sub-lexical language models for German LVCSR. In *IEEE Workshop on Spoken Language Technology*, pages 159 – 164, Berkeley, CA, USA, December 2010.

A. El-Desoky, M. Shaik, R. Schlüter, and H. Ney. Morpheme based factored language models for German LVCSR. In *Interspeech*, pages 1445 – 1448, Florence, Italy, August 2011.

A. El-Desoky, R. Schlüter, and H. Ney. Investigations on the use of morpheme level features in language models for Arabic LVCSR. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 5021 – 5024, Kyoto, Japan, March 2012.

A. El-Desoky, M. Shaik, R. Schlüter, and H. Ney. Morpheme level feature-based language models for German LVCSR. In *Interspeech*, Portland, OR, USA, September 2012.

A. El-Desoky, H.-K. J. Kuo, L. Mangu, and H. Soltau. Morpheme-based feature-rich language models using deep neural networks for LVCSR of Egyptian Arabic. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Vancouver, Canada, May 2013.

A. El-Desoky, M. Shaik, R. Schlüter, and H. Ney. Morpheme level hierarchical Pitman-Yor class-based language models for LVCSR of morphologically rich languages. In *Interspeech*, Lyon, France, August 2013.

A. Emami, K. Papineni, and J. Sorenson. Large-scale distributed language modeling. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 37 – 40, Honolulu, HI, USA, April 2007.

D. Erhan, Y. Bengio, A. Courville, P.-A Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11:625 – 660, March 2010.

G. Evermann and P. Woodland. Posterior probability decoding, confidence estimation and system combination. In *NIST Speech Transcription Workshop*, College Park, MD, USA, 2000.

G. Evermann, H. Y. Chan, M. J. F. Gales, T. Hain, X. Liu, L. Wang D. Mrva, and P. C. Woodland. Development of the 2003 CU-HTK conversational telephone speech transcription system. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 261 – 264, Montreal, Canada, May 2003.

J. G. Fiscus. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER). In *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, pages 347 – 354, Santa Barbara, CA, USA, December 1997.

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(179 – 188), 1936.

A. Fox. *The structure of German*. Oxford University Press, New York, NY, USA, 2005.

M. J. E. Gales and P. C. Woodland. Mean and variance adaptation within the MLLR framework. *Computer Speech and Language*, 10(4):249 – 264, 1996.

L. Galescu. Recognition of out-of-vocabulary words with sub-lexical language models. In *Proc. European Conf. on Speech Communication and Technology*, pages 249 – 252, Geneva, Switzerland, September 2003.

J. Gao, J. T. Goodman, and J. Miao. The use of clustering techniques for language modeling – Application to Asian languages. *Computational Linguistics and Chinese Language Processing*, 6(1):27 – 60, 2001.

A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 2 edition, July 2003. ISBN 158488388X.

M. Generet, H. Ney, and F. Wessel. Extensions to absolute discounting for language modeling. In *Proc. European Conf. on Speech Communication and Technology*, volume 2, pages 1245 – 1248, Madrid, Spain, September 1995.

F. A. Gers. *Long short-term memory in recurrent neural networks*. PhD thesis, Department of Computer Science, Swiss Federal Institute of Technology, Lausanne, EPFL, Switzerland, 2001.

F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: continual prediction with LSTM. In *Artificial Neural Networks, 1999. ICANN 99. 9th International Conference on*, volume 2, pages 850 – 855, 1999.

F. A. Gers, N. N. Schraudolph, and J. Schmidhuber. Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research*, 3:115 – 143, 2002.

S. Goldwater, T. L. Griffiths, and M. Johnson. Interpolating between types and tokens by estimating power-law generators. In *In Advances in Neural Information Processing Systems*, volume 18, 2006.

S. Goldwater, T. L. Griffiths, and M. Johnson. Producing power-law distributions and damping word frequencies with two-stage language models. *Journal of Machine Learning Research*, 12:2335 – 2382, 2011.

J. Goodman. Exponential priors for maximum Entropy models. In *Proc. Human Language Technology Conf. of the North American Chapter of the ACL*, pages 305 – 312, Boston, MA, USA, May 2004. Association for Computational Linguistics.

A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602 – 610, 2005.

D. Guiliani and F. Brugnara. Acoustic model adaptation with multiple supervisions. In *Proc.* TC-Star *Workshop on Speech-to-Speech Translation*, pages 151 – 154, Barcelona, Spain, June 2006.

D. Guiliani and F. Brugnara. Experiments on cross-system acoustic model adaptation. In *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, pages 117 – 122, Kyoto, Japan, December 2007.

R. Häb-Umbach and H. Ney. Improvements in beam search for 10000-word continuous-speech recognition. *IEEE Transactions on Speech and Audio Processing*, 2(2):353 – 356, April 1994.

N. Habash and O. Rambow. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proc. Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 573 – 580, University of Michigan, USA, June 2005.

N. Habash and O. Rambow. Arabic diacritization through full morphological tagging. In *Proc. Human Language Technology Conf. of the North American Chapter of the ACL*, volume Companion, pages 53 – 56, Rochester, NY, USA, April 2007.

N. Habash and F. Sadat. Arabic preprocessing schemes for statistical machine translation. In *Proc. Human Language Technology Conf. of the North American Chapter of the ACL*, volume 1, pages 49 – 52, New York, USA, June 2006.

R. Haeb-Umbach, X. Aubert, P. Beyerlein, D. Klaskow, M. Ullrich, A. Wendemuth, and P. Wilcox. Acoustic modeling in the Philips Hub-4 continous-speech recognition system. In *DARPA Broadcast News Transcription and Understanding Workshop*, February 1998.

K. Hagiwara and K. Fukumizu. Relation between weight size and degree of over-fitting in neural network regression. *Neural Networks*, 21(1):48 – 58, 2008.

D. Hakkani and G. Riccardi. A general algorithm for word graph matrix decomposition. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 596 – 599, Hong Kong, April 2003.

Mahdi Hamdani, Patrick Doetsch, Michal Kozielski, Amr El-Desoky, and Hermann Ney. The RWTH large vocabulary arabic handwriting recognition system. In *International Workshop on Document Analysis Systems (DAS)*, Tours, Loire Valley, France, April 2014.

H. Hermansky. Perceptual linear predictive (PLP) analysis of speech. *Journal of the Acoustical Society of America*, 87(4):1738 – 1752, June 1990.

H. Hermansky, D.P.W. Ellis, and S. Sharma. Tandem connectionist feature stream extraction for conventional HMM systems. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 1635–1638, Istanbul, Turkey, June 2000.

D. Hillard, B. Hoffmeister, M. Ostendorf, R. Schlüter, and H. Ney. iROVER: Improving system combination with classification. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 65 – 68, Rochester, New York, April 2007.

G. E. Hinton, S. Osindero, and Y.-W Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527 – 1554, July 2006.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735 – 1780, 1997.

B. Hoffmeister. *Bayes Risk Decoding and its Application to System Combination.* PhD thesis, RWTH Aachen University, 2011.

B. Hoffmeister, C. Plahland P. Fritz, G. Heigold, J. Lööf, R. Schlüter, and H. Ney. Development of the 2007 RWTH mandarin LVCSR system. In *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, Kyoto, Japan, December 2007.

B. Hoffmeister, C. Plahl, P. Fritz, G. Heigold, J. Lööf, R. Schlüter, and H. Ney. Development of the 2007 RWTH Mandarin GALE LVCSR system. In *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, pages 455 – 460, Kyoto, Japan, December 2007.

B. Hoffmeister, R. Schlüter, and H. Ney. iCNC and iROVER: The limits of improving system combination with classification? In *Interspeech*, pages 232 – 235, Brisbane, Australia, September 2008.

B. Hoffmeister, R. Liang, R. Schlüter, and H. Ney. Log-linear model combination with word-dependent scaling factors. In *Interspeech*, pages 248 – 251, Brighton, UK, September 2009.

B. Hoffmeister, R. Schlüter, and H. Ney. Bayes risk approximations using time overlap with an application to system combination. In *Interspeech*, pages 1191 – 1194, Brighton, UK, September 2009.

S. Huang and S. Renals. Hierarchical Pitman-Yor language models for ASR in meetings. In *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, pages 124 – 129, Kyoto, Japan, December 2007.

X. Huang, M. Belin, F. Alleva, and M. Hwang. Unified stochastic engine (USE) for speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 2, pages 636 – 639, Minneapolis, MN, USA, April 1993.

X. D. Huang and M. A. Jack. Semi-continuous hidden Markov models for speech signals. *Computer Speech and Language*, 3(3):329 – 252, 1989.

H. Ishwaran and L. F. James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96(453):161 – 173, 2001.

F. Jelinek. A fast sequential decoding algorithm using a stack. *IBM Journal of Research and Development*, 13:675 – 685, November 1969.

F. Jelinek. Continuous speech recognition by statistical methods. *Proceedings of the IEEE*, 64(4):532 – 556, April 1976.

F. Jelinek and R. L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, Amsterdam, The Netherlands, May 1980.

N. Jennequin and J.-L. Gauvain. Modeling duration via lattice rescoring. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Honolulu, HI, USA, April 2007.

B.-H. Juang and S. Katagiri. Discriminative learning for minimum error classification. *IEEE Transactions on Signal Processing*, 40(12):3043 – 3054, 1992.

J. Kaiser, B. Horvat, and Z. Kacic. A novel loss function for the overall risk criterion based discriminative training of HMM models. In *Interspeech*, volume 2, pages 887 – 890, Bejing, China, October 2000.

S. Kanthak, K. Schütz, and H. Ney. Using SIMD instructions for fast likelihood calculation in LVCSR. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 1531 – 1534, Istanbul, Turkey, June 2000.

S. Kanthak, H. Ney, M. Riley, and M. Mohri. A comparison of two LVR search optimization techniques. In *Proc. Int. Conf. on Spoken Language Processing*, pages 1309 – 1312, Denver, CO, USA, September 2002.

S. M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Speech and Audio Processing*, 35:400 – 401, March 1987.

T. Kemp and A. Jusek. Modelling unknown words in spontaneous speech. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 1, pages 530 – 533, Atlanta, GA, USA, 1996.

K. Kirchhoff, J. Bilmes, J. Henderson, R. Schwartz, M. Noamany, P. Schone, G. Ji, S. Das, M. Egan, F. He, D. Vergyri, D. Liu, and N. Duta. Novel speech recognition models for Arabic. Technical report, Johns-Hopkins University Summer Research Workshop, Baltimore, Maryland, USA, July 2002.

K. Kirchhoff, D. Vergyri, J. Bilmes, K. Duh, and A. Stolcke. Morphology-based language modeling for conversational Arabic speech recognition. *Computer Speech and Language*, 20(4):589 – 608, October 2006.

K. Kirchhoff, J. Bilmes, and K. Duh. Factored language model tutorial. Technical report, Department of Electrical Engineering, University of Washington, Seattle, Washington, USA, February 2008.

D. Klakow and J. Peters. Testing the correlation of word error rate and perplexity. *Speech Communication*, 38(1):19–28, September 2002.

D. Klakow, G. Rose, and X. Aubert. OOV-detection in large vocabulary system using automatically defined word-fragments as fillers. In *Proc. European Conf. on Speech Communication and Technology*, volume 1, pages 49 – 52, Budapest, Hungary, September 1999.

R. Kneser and H. Ney. Improved clustering techniques for class-based statistical language modeling. In *Proc. European Conf. on Speech Communication and Technology*, volume 2, pages 973 – 976, Berlin, Germany, September 1993a.

R. Kneser and H. Ney. Improved backing-off for M-gram language modeling. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 1, pages 181 – 184, Detroit, Michigan, USA, May 1995.

Reinhard Kneser and Hermann Ney. Forming word classes by statistical clustering for statistical language modeling. In *First Quantitative Linguistics Conference (QUALICO)*, pages 221 – 226, Trier, Germany, September 1991.

Reinhard Kneser and Hermann Ney. Improved clustering techniques for class-based statistical language modeling. In *EUROSPEECH*, pages 973 – 976, Berlin, Germany, September 1993b.

D. Kocharov, A. Zolnay, R. Schlüter, and H. Ney. Articulatory motivated acoustic features for speech recognition. In *Interspeech*, pages 1101 – 1104, Lisbon, Portugal, September 2005.

S. Kombrink, T. Mikolov, M. Karafiát, and L. Burget. Recurrent neural network based language modeling in meeting recognition. In *Interspeech*, pages 2877 – 2880, Florence, Italy, August 2011.

N. Kumar and A. G. Andreou. Heteroscedastic discriminant analysis and reduced rank HMMs for improved speech recognition. *Speech Communication*, 26(4):283 – 297, December 1998.

H.-K. J. Kuo, L. Mangu, A. Emami, I. Zitouni, and Y.-S. Lee. Syntactic features for Arabic speech recognition. In *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, pages 327 – 332, Merano, Italy, December 2009.

L. Lamel, A. Messaoudi, and J.-L. Gauvain. Investigating morphological decomposition for transcription of Arabic broadcast news and broadcast conversation data. In *Interspeech*, volume 1, pages 1429 – 1432, Brisbane, Australia, September 2008.

H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin. Exploring strategies for training deep neural networks. *Journal of Machine Learning Research*, 10:1 – 40, June 2009.

M. Larson, D. Willett, J. Köhler, and R. Rigoll. Compound splitting and lexical unit recombination for improved performance of a speech recognition system for German parliamentary speeches. In *Proc. Int. Conf. on Spoken Language Processing*, Beijing, China, October 2000.

L. Lee and R. Rose. Speaker normalization using efficient frequency warping procedures. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 353 – 356, Atlanta, GA, USA, May 1996.

Y. Lee, K. Papineni, S. Roukos, O. Emam, and H. Hassan. Language model based Arabic word segmentation. In *Proc. Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 399 – 406, Sapporo, Japan, July 2003.

C. J. Leggetter and P. C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. *Computer Speech and Language*, 9(2):171–185, 1995.

S. E. Levinson, L. R. Rabiner, and M. M. Sondhi. An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition. *Bell System Technical Journal*, 62(4):1035–1074, April 1983.

A. Ljolje, F. Pereira, and M. Riley. Efficient general lattice generation and rescoring. In *Proc. European Conf. on Speech Communication and Technology*, pages 1251 – 1254, Budapest, Hungary, September 1999.

J. Lööf, C. Gollan, and H. Ney. Cross-language bootstrapping for unsupervised acoustic model training: Rapid development of a Polish speech recognition system. In *Interspeech*, pages 88 – 91, Brighton, UK, September 2009.

B. Lowerre. *A comparative performance analysis of speech understanding systems.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1976.

D. J. C. MacKay and L. C. B. Peto. A hierarchical dirichlet language model. *Natural Language Engineering*, 1:1 – 19, 1994.

G. Maltese, P. Bravetti, H. Crépy, B. Grainger, M. Herzog, and F. Palou. Combining word- and class-based language models: A comparative study in several languages using automatic and manual word-clustering techniques. In *Proc. European Conf. on Speech Communication and Technology*, pages 21 – 24, Aalborg, Denmark, September 2001.

L. Mangu. *Finding Consensus in Speech Recognition.* PhD thesis, Johns Hopkins University, Baltimore, Maryland, USA, April 2000.

L. Mangu, E. Brill, and A. Stolcke. Finding consensus in speech recognition: word error minimization and other applications of confusion networks. *Computer Speech and Language*, 14:373–400, 2000.

J. Martens and I. Sutskever. Learning recurrent neural networks with Hessian-Free optimization. In *the 28th International Conference on Machine Learning*, Bellevue, WA, USA, 2011.

S. C. Martin. *Statistische Auswahl von Wortabhängigkeiten in der automatischen Spracherkennung.* PhD thesis, RWTH Aachen University, Aachen, Germany, February 2000.

Sven C. Martin, Jörg Liermann, and Hermann Ney. Algorithms for bigram and trigram word clustering. *Speech Communication*, 24(1):19 – 37, 1998.

T. Matsuzaki, Y. Miyao, and J. Tsujii. An efficient clustering algorithm for class-based language models. In *Proc. Human Language Technology Conf. of the North American Chapter of the ACL*, volume 4, pages 119 – 126, Edmonton, Canada, May 2003.

E. McDermott and S. Katagiri. Minimum classification error for large scale speech recognition tasks using weighted finite state transducers. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Philadelphia, PA, USA, April 2005.

A. Messaoudi, J.-L. Gauvain, and L. Lamel. Arabic broadcast news transcription using a one million word vocalized vocabulary. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 1, pages 1093 –1096, Toulouse, France, May 2006.

F. Metze and A. Waibel. A flexible stream architecture for asr using articulatory features. In *Proc. Int. Conf. on Spoken Language Processing*, pages 2133 – 2136, Denver, CO, USA, September 2002a.

F. Metze and A. Waibel. Auditory-based acoustic distinctive features and spectral cues for automatic speech recognition using a multi-stream paradigm. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 837 – 840, Orlando, FL, USA, May 2002b.

R. Miikkulainen and M. G. Dyer. Natural language processing with modular PDP networks and distributed lexicon. *Cognitive Science*, 15(3):343 – 399, 1991.

T. Mikolov. *Statistical Language Models based on Neural Networks.* PhD thesis, Brno University of Technology, 2012.

T. Mikolov, M. Karafiát, L. Burget, J. Cernocky, and S. Khudanpur. Recurrent neural network based language model. In *Interspeech*, pages 1045 – 1048, Makuhari, Japan, September 2010.

T. Mikolov, S. Kombrink, L. Burget, J. Cernocký, and S. Khudanpur. Extensions of recurrent neural network language model. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 5528 – 5531, Prague, Czech Republic, May 2011.

B. Möbius. Word and syllable models for German text-to-speech synthesis. In *Proc. of the Third ESCA Workshop on Speech Synthesis*, pages 59 – 64, NSW, Australia, November 1998.

A. Mohamed, G. Dahl, and G. E. Hinton. Deep belief networks for phone recognition. In *NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, Whistler, BC, Canada, 2009.

M. Mohri and M. Riley. Weighted determinization and minimization for large vocabulary speech recognition. In *Proc. European Conf. on Speech Communication and Technology*, Rhodes, Greece, September 1997.

S. Molau. *Normalization in the acoustic feature space for improved speech recognition*. PhD thesis, RWTH Aachen, Aachen, Germany, 2003.

H. Murveit, J. Butzberger, V. Digalakis, and M. Weintraub. Progressive-search algorithms for large-vocabulary speech recognition. In *HLT '93: Proceedings of the workshop on Human Language Technology*, pages 87 – 90, Morristown, NJ, USA, 1993. Association for Computational Linguistics.

M. Nakamura and K. Shikano. A study of English word category prediction based on neural networks. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 2, pages 731 – 734, Glasgow, Scotland, May 1989.

R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Dept. of Computer Science, University of Toronto, September 1993.

S. B. Needleman and C. D. Wunsch. An efficient method applicable to the search for similarities in the amino acid sequences of two proteins. *Journal of Molecular Biology*, 48(3):444 – 453, March 1970.

H. Ney. The use of a one-stage dynamic programming algorithm for connected word recognition. *IEEE Transactions on Speech and Audio Processing*, 32(2):263 – 271, April 1984.

H. Ney. Acoustic modeling of phoneme units for continuous speech recognition. In L. Torres, E. Masgrau, and M. A. Lagunas, editors, *Signal Processing V: Theories and Applications, Fifth European Signal Processing Conference*, pages 65 – 72. Elsevier Science Publishers B. V., Barcelona, Spain, 1990.

H. Ney and X. Aubert. A word graph algorithm for large vocabulary continuous speech recognition. In *Proc. Int. Conf. on Spoken Language Processing*, volume 3, pages 1355 – 1358, Yokohama, Japan, September 1994.

H. Ney, D. Mergel, A. Noll, and A. Paeseler. A data-driven organization of the dynamic programming beam search for continuous speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 833 – 836, Dallas, TX, USA, April 1987.

H. Ney, R. Häb-Umbach, B.-H. Tran, and M. Oerder. Improvements in beam search for 10000-word continuous speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 1, pages 9 – 12, San Francisco, CA, March 1992.

H. Ney, U. Essen, and R. Kneser. On structuring probabilistic dependencies in language modeling. *Computer Speech and Language*, 2(8):1 – 38, 1994.

H. Ney, S. C. Martin, and F. Wessel. Statistical language modeling using leaving-one-out. In S. Young and G. Bloothooft, editors, *Corpus Based Methods in Language and Speech Processing*, pages 1 – 26. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997.

Y. Normandin, R. Lacouture, and R. Cardin. MMIE training for large vocabulary continuous speech recognition. In *International Conference on Spoken Language Processing*, pages 1367 – 1370, Yokohama, Japan, September 1994.

R. Ordelman, A. V. Hassen, and F. D. Jong. Compound decomposition in Dutch large vocabulary speech recognition. In *Proc. European Conf. on Speech Communication and Technology*, pages 225 – 228, Geneva, Switzerland, September 2003.

S. Ortmanns and H. Ney. An experimental study of the search space for 20000-word speech recognition. In *Proc. European Conf. on Speech Communication and Technology*, volume 2, pages 901 – 904, Madrid, Spain, September 1995.

S. Ortmanns, H. Ney, and A. Eiden. Language-model look-ahead for large vocabulary speech recognition. In *Proc. Int. Conf. on Spoken Language Processing*, volume 4, pages 2095 – 2098, Philadelphia, PA, October 1996.

S. Ortmanns, H. Ney, and X. Aubert. A word graph algorithm for large vocabulary continuous speech recognition. *Computer Speech and Language*, 11(1):43 – 72, January 1997.

S. Ortmanns, A. Eiden, and H. Ney. Improved lexical tree search for large vocabulary recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 817 – 820, Seattle, WA, USA, May 1998.

M. Ostendorf, A. Kannan, S. Austin, O. Kimball, R. Schwartz, and J. R. Rohlicek. Integration of diverse recognition methodologies through reevaluation of N-best sentence hypotheses. In *DARPA Speech and Natural Language Processing Workshop*, pages 83 – 87, Pacific Grove, CA, USA, 1991.

N. Parihar, R. Schlüter, D. Rybach, and E. A. Hansen. Parallel fast likelihood computation for LVCSR using mixture decomposition. In *Interspeech*, Brighton, UK, September 2009.

D. B. Paul. Algorithms for an optimal A^* search and linearizing the search in the stack decoder. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 1, pages 693 – 696, Toronto, Canada, May 1991.

M. Piotr. Syllable based language model for large vocabulary continuous speech recognition of Polish. In *Text, Speech and Dialogue*, volume 5246 of *Lecture Notes in Computer Science*, pages 397 – 401. 2008.

J. Pitman. Exchangeable and partially exchangeable random partitions. *Probability Theory and Related Fields*, 102(2):145 – 158, 1995.

J. Pitman. Combinatorial stochastic processes. Technical Report 621, UC Berkeley Dept. of Statistics, 2002.

J. Pitman and M. Yor. The Two-Parameter Poisson-Dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, 25(2):855 – 900, 1997.

M. Pitz. *Investigations on linear transformations for speaker adaptation and normalization.* PhD thesis, RWTH Aachen University, 2005.

K. Plunkett and J. Elman. *Exercises in rethinking innateness: A handbook for connectionist simulations.* Neural Network Modeling and Connectionism series. MIT Press/Bradford Books, May 1997. ISBN 0-262-66105-5.

D. Povey and P. C. Woodland. Minimum phone error and I-smoothing for improved discriminative training. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 1, pages 105 – 108, Orlando, FL, May 2002a.

D. Povey and P. C. Woodland. Minimum phone error and i-smoothing for improved discriminative training. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 1, pages 105 – 108, Orlando, FL, USA, May 2002b.

R. Prasad, S. Matsoukas, C. L. Kao, J. Z. Ma, D. X. Xu, T. Colthurst, O. Kimball, R. Schwartz, J.-L. Gauvain, L. Lamel, H. Schwenk, G. Adda, and F. Lefevre. The 2004 BBN/LIMSI 20xRT English conversational telephone speech recognition system. In *Interspeech*, Lisbon, Portugal, September 2005.

W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, 1988.

L. Rabiner and B.-H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3(1):4 – 16, 1986.

L. R. Rabiner and R. W. Schafer. *Digital processing of speech signals.* Prentice-Hall Signal Processing Series, Englewood Cliffs, NJ, 1979.

V. Ramasubramansian and K. K. Paliwal. Fast k-dimensional tree algorithms for nearest neighbor search with application to vector quantization encoding. *IEEE Transactions on Speech and Audio Processing*, 40(3):518 – 528, March 1992.

T. Rotovnik, M. S. Maučec, and Z. Kačič. Large vocabulary continuous speech recognition of an inflected language using stems and endings. *Speech Communication*, 49(6):537 – 452, June 2007.

J. Rubach and G. Booij. Syllable structure assignment in Polish. *Phonology*, 7:121 – 158, October 1990.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, (323):533 – 536, 1986.

D. Rybach, S. Hahn, C. Gollan, R. Schlüter, and H. Ney. Advances in Arabic broadcast news transcription at RWTH. In *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, pages 449 – 454, Kyoto, Japan, December 2007.

D. Rybach, C. Gollan, G. Heigold, B. Hoffmeister, J. Lööf, R. Schlüter, and H. Ney. The RWTH Aachen University open source speech recognition system. In *Interspeech*, pages 2111 – 2114, Brighton, UK, September 2009.

D. Rybach, S. Hahn, P. Lehnen, D. Nolden, M. Sundermeyer, Z. Tüske, S. Wiesler, R. Schlüter, and H. Ney. RASR - the RWTH Aachen University open source speech recognition toolkit. In *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, Honolulu, Hawaii, USA, December 2011.

T. Sainath, B. Kingsbury, and B. Ramabhadran. Improvements in using deep belief networks for large vocabulary continuous speech recognition. Technical report, IBM, Speech and Language Algorithms Group, 2012.

H. Sakoe. Two-level DP-matching - A dynamic programming-based pattern matching algorithm for connected word recognition. *IEEE Transactions on Speech and Audio Processing*, 27:588 – 595, December 1979.

C. Samuelsson and W. Reichl. A class-based language model for large-vocabulary speech recognition extracted from part-of-speech statistics. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 1, pages 537 – 540, Phoenix, AZ, USA, March 1999.

R. Sarikaya, M. Afify, and B. Kingsbury. Tied-mixture language modeling in continuous space. In *Proc. Human Language Technology Conf. of the North American Chapter of the ACL*, pages 459 – 467, Boulder, CO, USA, June 2009.

R. Sarikaya, A. Emami, M. Afify, and B. Ramabhadran. Continuous space language modeling techniques. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 5186 – 5189, Dallas, Texas, USA, 2010.

E. Saund. Dimensionality-reduction using connectionist networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(3):304 – 314, March 1989.

R. Schlüter. *Investigations on discriminative training criteria.* PhD thesis, RWTH Aachen University, Aachen, Germany, September 2000.

R. Schlüter, A. Zolnay, and H. Ney. Feature combination using linear discriminant analysis and its pitfalls. In *Proc. Int. Conf. on Spoken Language Processing*, pages 345 – 348, Pittsburgh, PA, USA, September 2006.

R. Schlüter, I. Bezrukov, H. Wagner, and H. Ney. Gammatone features and feature combination for large vocabulary speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Honululu, HI, USA, April 2007.

H. Schmid. Probabilistic part-of-speech tagging using decision trees. In *Conference on New Methods in Language Processing*, Manchester, UK, September 1994.

H. Schmid. Improvements in part-of-speech tagging with an application to German. In *Proc. of the the ACL SIGDAT-Workshop*, pages 47 – 50, Dublin, Ireland, March 1995.

C. Schrumpf, M. Larson, and S. Eickeler. Syllable-based language models in speech recognition for English spoken document retrieval. In *Proc. of the 7th International Workshop of the EU Network of Excellence DELOS on AVIVDiLib*, pages 196 – 205, Cortona, Italy, May 2005.

T. Schultz and A. Waibel. Experiments on cross-language acoustic modeling. In *Proc. European Conf. on Speech Communication and Technology*, pages 2721 – 2724, Aalborg, Denmark, September 2001.

R. Schwartz and Y.-L. Chow. The N-best algorithm: An efficient and exact procedure for finding the N most likely sentence hypotheses. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 81 – 84, Albuquerque, NM, April 1990.

H. Schwenk. Continuous space language models. *Computer Speech and Language*, 21(3):492 – 518, July 2007.

H. Schwenk and J.-L Gauvain. Training neural network language models on very large corpora. In *Proc. of the Conf. on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 201 – 208, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

F. Seide, G. Li, X. Chen, and D. Yu. Feature engineering in context-dependent deep neural networks for conversational speech transcription. In *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, pages 24 – 29, Honolulu, Hawaii, USA, December 2011.

M. Shaik, A. El-Desoky, R. Schlüter, and H. Ney. Hybrid language models using mixed types of sub-lexical units for open vocabulary German LVCSR. In *Interspeech*, pages 1441 – 1444, Florence, Italy, August 2011a.

M. Shaik, A. El-Desoky, R. Schlüter, and H. Ney. Using morpheme and syllable based sub-words for Polish LVCSR. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 4680 – 4683, Prague, Czech Republic, May 2011b.

M. Shaik, Zoltan Tüske, Simon Wiesler, Markus Nussbaum-Thom, Stephan Peitz, ralf Schlüter, and Hermann Ney. The RWTH aachen german and english LVCSR systems for IWSLT-2013. In *International Workshop on Spoken Language Translation (IWSLT)*, Heidelberg, Germany, December 2013.

A. Sixtus. *Across-word phoneme models for large vocabulary continuous speech recognition*. PhD thesis, RWTH Aachen, January 2003.

H. Soltau, B. Kingsbury, L. Mangu, D. Povey, G. Saon, and G. Zweig. The IBM 2004 conversational telephony system for rich transcription. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 205–208, Philadelphia, PA, USA, March 2005a.

H. Soltau, B. Kingsbury, L. Mangu, D. Povey, G. Saon, and G. Zweig. The IBM 2004 conversational telephony system for rich transcription. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 205 – 208, Philadelphia, PA, USA, March 2005b.

H. Soltau, G. Saon, and B. Kingsbury. The IBM Attila speech recognition toolkit. In Dilek Hakkani-Tür and Mari Ostendorf, editors, *IEEE Workshop on Spoken Language Technology*, pages 97 – 102, Berkeley, CA, USA, December 2010.

V. Steinbiss, H. Ney, R. Häb-Umbach, B. H. Tran, U. Essen, R. Kneser, M. Oerder, H. G. Meier, X. Aubert, C. Dugast, and D. Geller. The Philips research system for large-vocabulary continuous-speech recognition. In *Proc. European Conf. on Speech Communication and Technology*, pages 2125 – 2128, Berlin, Germany, September 1993.

A. Stolcke. SRILM - an extensible language modeling toolkit. In *Proc. Int. Conf. on Spoken Language Processing*, volume 2, pages 901 – 904, Denver, Colorado, USA, September 2002.

A. Stolcke, H. Bratt, J. Butzberger, H. Franco, V. R. Rao Gadde, M. Plauche, C. Richey, E. Shriberg, K. Sönmez, F. Weng, and J. Zheng. The SRI March 2000 Hub-5 conversational speech transcription system. In *NIST Speech Transcription Workshop*, College Park, MD, USA, May 2000.

S. Stüker, C. Fügen, S. Burger, and M. Wölfel. Cross-system adaptation and combination for continuous speech recognition: the influence of phoneme set and acoustic front-end. In *Interspeech*, Pittsburgh, PA, USA, September 2006.

M. Sundermeyer, M. Nußbaum-Thom, S. Wiesler, C. Plahl, A. El-Desoky, S. Hahn, D. Nolden, R. Schlüter, and H. Ney. The RWTH 2010 Quaero ASR evaluation system for English, French, and German. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 2212 – 2215, Prague, Czech Republic, May 2011.

M. Sundermeyer, R. Schlüter, and H. Ney. LSTM neural networks for language modeling. In *Interspeech*, Portland, OR, USA, September 2012.

M. Sundermeyer, I. Oparin, J.-L. Gauvain, B. Freiberg, R. Schlüter, and H. Ney. Comparison of feedforward and recurrent neural network language models. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, Vancouver, Canada, May 2013.

O. E. Swan. *A grammar of contemporary Polish*. Slavica, Bloomington, IN, USA, 2002.

M. Y. Tachbelie. *Morphology-Based Language Modeling for Amharic*. PhD thesis, Universität Hamburg, Von-Melle-Park 3, 20146 Hamburg, 2010.

M. Y. Tachbelie, S. T. Abate, and W. Menzel. Morpheme-based and factored language modeling for Amharic speech recognition. In Zygmunt Vetulani, editor, *Human Language Technology. Challenges for Computer Science and Linguistics*, volume 6562 of *Lecture Notes in Computer Science*, pages 82 – 93. Springer Berlin Heidelberg, 2011.

Y. W. Teh. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proc. Annual Meeting of the Association for Computational Linguistics*, pages 985 – 992, Sydney, Australia, July 2006a. Association for Computational Linguistics.

Y. W. Teh. A Bayesian interpretation of interpolated Kneser-Ney. Technical Report TRA2/06, School of Computing, National University of Singapore, 2006b.

A. Tritschler and R. A. Gopinath. Improved speaker segmentation and segments clustering using the Bayesian information criterion. In *Proc. European Conf. on Speech Communication and Technology*, pages 679 – 682, Budapest, Hungary, September 1999.

F. Valente, J. Vepa, C. Plahl, C. Gollan, H. Hermansky, and R. Schlüter. Hierarchical neural networks feature extraction for LVCSR system. In *Interspeech*, Antwerp, Belgium, August 2007.

A. Venkataraman and W. Wang. Techniques for effective vocabulary selection in domain specific speech recognition. In *Proc. European Conf. on Speech Communication and Technology*, volume 1, pages 245 – 248, Geneva, Switzerland, September 2003.

D. Vergyri. *Integration of multiple knowledge sources in speech recognition using minimum error training*. PhD thesis, Johns Hopkins University, Baltimore, Maryland, USA, 2000.

D. Vergyri, S. Tsakalidis, and W. Byrne. Minimum risk acoustic clustering for multilingual acoustic model combination. In *Proc. Int. Conf. on Spoken Language Processing*, pages 873 – 876, Beijing, China, October 2000.

D. Vergyri, A. Mandal, A. Stolcke, J. Zheng, M. Graciarena, D. Rybach, C. Gollan, R. Schlüter, K. Kirchhoff, A. Faria, and N. Morgan. Development of the SRI/Nightingale Arabic ASR system. In *Interspeech*, volume 1, pages 1437 – 1440, Brisbane, Australia, September 2008.

T. K. Vintsyuk. Elementwise recognition of continuous speech composed of words from a specified dictionary. *Kibernetika*, 7:133 – 143, March 1971.

A. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, 13:260 – 269, 1967.

F. Wessel, R. Schlüter, K. Macherey, and H. Ney. Confidence measures for large vocabulary continuous speech recognition. *IEEE Transactions on Speech and Audio Processing*, 9(3):288 – 298, March 2001.

P. C. Woodland and D. Povey. Large scale discriminative training of hidden Markov models for speech recognition. *Computer Speech and Language*, 16(1):25 – 48, 2002.

B. Xiang, K. Nguyen, L. Nguyen, R. Schwartz, and J. Makhoul. Morphological decomposition for Arabic broadcast news transcription. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 1, pages 1089 – 1092, Toulouse, France, May 2006.

B. Xu, B. Ma, S. Zhang, F. Qu, and T. Huang. Speaker-independent dictation of Chinese speech with 32k vocabulary. volume 4, pages 2320 – 2323, Philadelphia, PA , USA, October 1996.

W. Xu and A. Rudnicky. Can artificial neural networks learn language models? In *Interspeech*, pages 202 – 205, Beijing, China, 2000. ISCA.

J. Xue and Y. Zhao. Improved confusion network algorithm and shortest path search from word lattice. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 853 – 856, Philadelphia, PA, USA, March 2005.

R. Zhang and A. Rudnicky. Investigations of issues for using multiple acoustic models to improve continuous speech recognition. In *Proc. Int. Conf. on Spoken Language Processing*, Pittsburgh, PA, USA, September 2006.

A. Zolnay. *Acoustic Feature Combination for Speech Recognition.* PhD thesis, RWTH Aachen University, Aachen, Germany, August 2006.

A. Zolnay, R. Schlüter, and H. Ney. Robust speech recognition using a voiced-unvoiced feature. In *Proc. Int. Conf. on Spoken Language Processing*, volume 2, pages 1065 – 1068, Denver, CO, USA, September 2002.

A. Zolnay, R. Schlüter, and H. Ney. Acoustic feature combination for robust speech recognition. In *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, volume 1, pages 457 – 460, Philadelphia, PA, USA, March 2005.