Letter-to-Phoneme Conversion for a German Text-to-Speech System

Vera Demberg

Diplomarbeit Nr. 47

Betreuer:	Dr. Helmut Schmid
	Dr. Gregor Möhler
Prüfer:	Prof. Hinrich Schütze, PhD
	Dr. Helmut Schmid
Begin:	December 01, 2005
End:	May 31, 2006

Universität Stuttgart
Institut für Maschinelle Sprachverarbeitung
Azenbergstr. 12
70174 Stuttgart
Germany

Abstract

This thesis deals with the conversion from letters to phonemes, syllabification and word stress assignment for a German text-to-speech system.

In the first part of the thesis (chapter 5), several alternative approaches for morphological segmentation are analysed and the benefit of such a morphological preprocessing component is evaluated with respect to the grapheme-to-phoneme conversion algorithm used.

In the second part (chapters 6 - 8), a pipeline architecture in which syllabification, stress assignment and letter-to-phoneme conversion are tackled in separate modules is compared with a design that integrates all tasks into one step. Alternative approaches to the problems are discussed and evaluated.

Best results are obtained with a Hidden Markov Model that does letter-to-phoneme conversion, syllabification and stress assignment simultaneously and incorporates some very simple and language-independent phonological constraints regarding syllable structure and word stress. This model achieves a 50% reduction in word error rate relative to the AWT decision tree.

The Hidden Markov Model is shown to compare well to state-of-the-art approaches to grapheme-to-phoneme conversion in English and French.

Acknowledgements

First of all, I want to thank my supervisor Helmut Schmid,
who always had an open door for me to bump in,
for very interesting discussions and helpful advice.

I am also grateful to Gregor Möhler,
my supervisor at IBM.

Special honours and my deepest gratitude to
Ralf Jankowitsch for proof-reading the thesis
and very competent advice on using Latex.
I also want to thank Mustafa Karakoc
for helpful comments on my thesis.

I am thankful to
Delphine Bernhard, Stefan Bordag,
Samarth Keshava and Emily Pitler
for allowing me to use the morphological segmentations by their algorithms
and fruitful discussion.

The author would furthermore like to thank the following
for discussion, help and advice:
Ruth Fuchss,
Bernd Möbius, Hinrich Schütze,
Yannick Marchand, Bob Damper,
Stanley Chen and John Goldsmith.

Personally, I want to say thanks to
Iglika Misheva, Sarah Böck, Sonja Lohrengel
Stefan Andreas Link, Jan-Erek Thiede, Juri Ganitkevitch,
Andreas Pabst, Martin Strenge and Daniel Nüsse,
who were students at IBM at the same period for a great time,
and Ines Flechsig, Manuel Kountz, Andreas Madsack,
Ralf Jankowitsch and Nicolai Hezinger
for excellent lunch company.

The author took part in the 2nd Pascal Challenges Workshop
in Venice, Italy from 10th to 12th April 2006.
Participation was partly financed
by IBM Germany Research and Development GmbH and
by Evangelisches Studienwerk e.V. Villigst.

I very gratefully acknowledge support by
Evangelisches Studienwerk e.V. Villigst throughout my studies.

Last and least, I would like to thank Lenchen for the chocolate.

Declaration

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst habe und dabei keine andere als die angegebene Literatur verwendet habe.
Alle Zitate und sinngemäßen Entlehnungen sind als solche unter genauer Angabe der Quelle gekennzeichnet.

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text.

(*Vera Demberg*)

Table of Contents

Chapter 1

Introduction

Letter-to-phoneme conversion is often also referred to as “letter-to-sound conversion” or “grapheme-to-phoneme conversion” (commonly abbreviated as g2p conversion). I therefore want to define very briefly the notions of “letter”, “grapheme”, “phoneme” and “sound”.

A *letter* is an element of the alphabet of a language (I am only considering languages with alphabets here).

There are different definitions for a *grapheme*. In this thesis, I refer to a grapheme as a string consisting of one or more letters that together correspond to exactly one phoneme.

A *phoneme* is an entity that is a transcription of a sound which is nearer to the pronunciation of a word than its orthographical representation. It defined by its ability to distinguish relevantly different sounds in a language. Phonemes differ from phones in that they are relative entities that describe differences which are significant within a language. In German, there is a phonemes `/P/` (as in Pass) that subsumes the phone [p] and the phone [p^h] (aspirated [p]), which is not relevant for German lexical semantics (i.e. [p^{h}as] would be assigned the same semantics by a German speaker as [pas]). But there is a separate phoneme `/B/` for the phone [b]. `/B/` this is needed to make the difference between the words “Pass” and “Bass”. The phoneme inventory I used in this thesis is the IBM Native Phonology for TTS (see section 3.6).

A *sound* is an acoustic signal. In the context of a TTS system, sounds are generated by the synthesizing component but not by the grapheme-to-phoneme conversion component.

Although “letter to phoneme conversion” describes best what I have been doing in this thesis, I will use the abbreviation g2p interchangeably.

1.1 Text-to-Speech Synthesis Systems

The conversion from text to speech is a complex procedure that involves several subtasks. The processing steps typically performed in TTS systems are depicted in figure 1.1.

The first obligatory step in TTS is to segment and tokenize a text. In this step, the input text is split up into sentences and words. This task is not as trivial as it may seem because words are not always well-defined, and punctuation marks can be ambiguous.

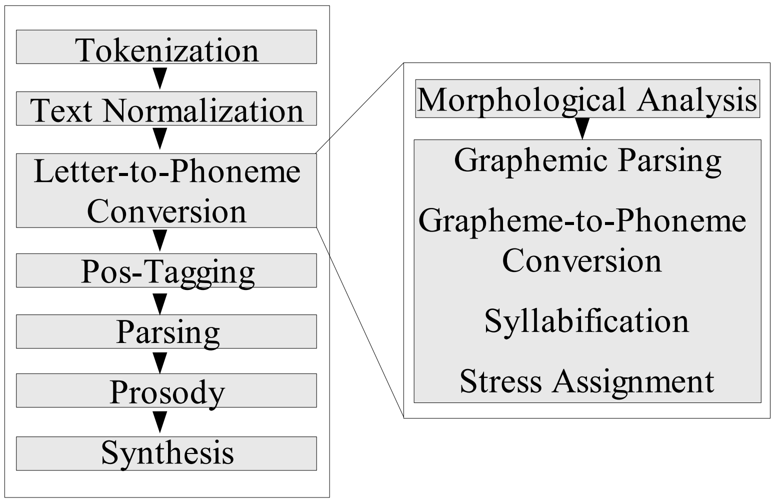

Figure 1.1: Typical Architecture of a Text-to-Speech System.

(For a more detailed description of the problem refer to Möbius [2001].) The next step is referred to as “text normalization”. Text normalization comprises:

- expansion of abbreviations
  e.g. expanding *z.B.* to *zum Beispiel* (Engl: for example).

- interpretation of numbers
  e.g. correct expansion of ordinal numbers with respect to case and gender as well as the correct interpretation of symbols like “%”.

- recognition of dates and similar
  e.g. the expansion of 1993 to *neunzehnhundert und dreiundneunzig* (Engl: “nineteen hundred ninety three” instead of expanding it to the German equivalent of “one thousand nine hundred ninety three”), as well as the correct expansion with respect to case: *am 14.3.* should be expanded to *am vierzehnten dritten* or *am vierzehnten März*. The standard nominative case form *vierzehnter* would be incorrect. Similar problems occur for expressions like *Heinrich IV*, which has to be expanded to *Heinrich der Vierte*, *Heinrich des Vierten*, *Heinrich dem Vierten* or *Heinrich den Vierten* (Engl: different grammatical cases of “Henry fourth”) dependent on the context.

- recognition of acronymes
  There is no general rule for acronym pronunciation in German. Some are spelled out e.g. ADAC, IMS or IBM, while others are pronounced as words, e.g. NATO or AIDS.

This thesis focusses on the letter-to-phoneme conversion step, which comprises a number of different subtasks: determining which letters map to one phoneme (also commonly referred to as *alignment* or *graphemic parsing*), syllabification, word stress assignment and g2p conversion.

The advantage of a modular architecture in which each task is performed by a separate module is that problems can be isolated easily and it is straightforward to substitute one module by another one if a better algorithm is found. But stress assignment, syllabification and g2p conversion are dependent on one another – e.g., stress depends on the position and weight of a syllable. But syllable weight can only be determined when the *phonemes* are known – letters are very ambiguous, for example, the letter *e* can be transcribed to unstressable schwa (/E#/), light /AE/ or vowels such as /AE:/ or /E:/ that contribute to make a syllable "heavy". On the other hand, if we know whether a vowel is stressed or not, the correct phoneme can be assigned with higher probability.

It can thus be beneficial to do strongly interdependent tasks in a single step. In current systems, both types of architectures can be found (Möbius [2001]; Black *et al.* [1998]; Pagel *et al.* [1998]; van den Bosch *et al.* [1998]; Marchand and Damper [2005]; Guérillon [2002]). Among the modular systems, some systems do syllabification and stress assignment in a preprocessing step, while other do grapheme-to-phoneme conversion first and then insert syllable boundaries and assign word stress.

Once words have been converted to their phonemic representation, prosody (i.e. intonation and rhythm of speech) has to be added to the sentences, and phrase boundaries have to be inserted. To do this, deeper level linguistic processes such as tagging (determining word types and other grammatical categories) and parsing (to detect chunks and decide on where pauses should be inserted) are applied. In speech synthesis, the linguistically preprocessed material is finally synthesized to speech sounds.

1.2 Applications and their Requirements

An obvious effect of such a pipeline architecture as outlined above is that it causes mistakes to be propagated through the process: errors in early stages cause errors in the following processing steps. This emphasizes the need for good quality conversion results.

Text-to-speech systems are nowadays applied in many domains such as automatically reading text documents to people with impaired vision, or communicating with spoken dialogue systems.

The most important requirement for a TTS system is that quality is good enough to be understandable in order to reach the dialogue goal. But that is no enough: users have been found dislike systems that do not sound natural. Even minor mistakes are relevant.

Requirements to the TTS system vary in function of the target application. If the system is to be run on a mobile device with relatively low storage capacity, it needs to run fast and should consume little memory resources. It may even be necessary to trade quality for efficiency in such systems. On the other hand, if the system is to be run on a bigger system, achieving optimal quality is the prior concern. For all systems, an important requirement is that the system runs in real time – no user would be willing to talk to a system that takes 5 minutes to compute the answer for every dialogue turn.

At the level of system design, it is desirable to have a system that can be easily adapted to other languages or transferred to new domains. With rule-based systems, this property is usually not given, as the systems rely on large expert-designed rule-sets and often also on linguistically annotated lexicons. Rules would have to be rewritten completely, and a new lexicon would have to be designed for the adaptation of the system to a new language. To apply a system to a another domain with special words, a lexicon from this new domain is needed. Data-based approaches do not require expert rule sets and therefore have lower development costs. But large annotated databases are needed for supervised learning. The acquisition of such databases can constitute a bottleneck similar to the rule acquisition problem.

1.3 Project Goals and Execution

The overall goal of this thesis is to reach higher correctness rates in letter-to-phoneme conversion of the IBM TTS system.

The initial approach was to use a newly acquired lexicon, IMSLEX (cf. chapter 3) to build a better morphology module for the TTS system, by using the finite state transducer from SMOR (see also chapter 3) and by taking into account data-based approaches. Morphological information had been found to be relevant for g2p, as syllable structure, word stress and the phonemizations of certain letters depend on word structure. Words that were annotated with morphological structure from the IBM ETI morphology, had shown to reduce word error rate significantly.

Unfortunately, I could not achieve better performance than with the original morphology due to lacking information in IMSLEX. I furthermore found that state-of-the-art data-based approaches do not reach the rule-based morphological systems' performance rates, and therefore do not provide information valuable to the g2p task.

When I analysed the conversion mistakes made by the system, I found stress assignment to constitute a major problem that accounts for the largest proportion of errors. I therefore implemented separate components for syllabification and stress assignment, which lead to reductions in word error rates. In a last step, the statistical model which had shown to return best results for the syllabification and stress assignment problems was adapted to g2p conversion to substitute the IBM decision tree. I experimented on the optimal arrangement of modules and also built one module that performs all three tasks at once.

Learning and evaluation was only performed on word lists – context dependent phenomena (e.g. word stress in context or co-articulation effects across word boundaries) are *not* considered in this work. Also, the word list used does not contain any names, which are very frequent in naturally occurring text and the transcription of which tends to be particularly error-prone (for detailed discussion and specialized systems for name transcription see Jannedy and Möbius [1997]).

1.4 Structure of Thesis

In chapter 2, a brief introduction into the statistical models used in this work (decision trees and Hidden Markov Models) and smoothing methods for n-gram models is given. During my thesis, I used a variety of different corpora and lexica which I briefly present in chapter 3. Chapter 4 addresses the issue of training and test set design and evaluation methodologies. Chapters 5–8 contain the main parts of this thesis. They are concerned with the subtasks in letter-to-phoneme conversion, i.e. morphological decomposition of words, syllabification, word stress assignment and g2p conversion respectively. Each of the chapters summarizes previous work on the respective field, discusses the different approaches followed in this work and presents their results as lone-standing systems and with respect to their impact on the overall conversion task. Finally, I draw conclusions about how to best design the g2p conversion component of a TTS system and suggest future work in chapter 9.

Chapter 2

Mathematical Background

The function of this chapter is to give a very broad overview of the most important probabilistic models that are used and discussed in this thesis, Hidden Markov Models and decision trees, and the most important issues that come up when using these models.

2.1 Hidden Markov Models

A Markov process (named after Andrey Markov) is a stochastic process with the Markov property. The Markov property states that the previous states are irrelevant for subsequent states given the actual state (i.e. subsequent states are conditionally independent from previous states). This property can also be generalized to higher order Markov Models, where a state depends on *k* previous states. A Markov model's only parameters are the transitional probabilities from one state to the next state.

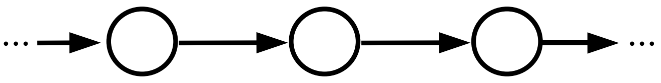

Figure 2.1: A first order Markov Model.

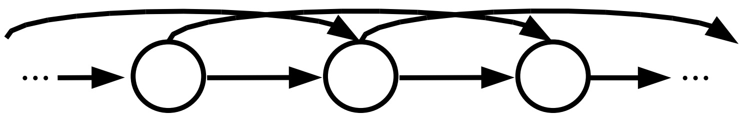

Figure 2.2: A second order Markov Model.

A Hidden Markov Model (HMM) is a statistical model where the system being modelled is assumed to be a Markov process with unknown parameters. The challenge consists of determining the hidden parameters from the observable parameters. Hidden Markov Models are very often used in speech recognition. The observed symbols do not depend on one another directly, but depend from some hidden states, which in turn depend on one another with the Markov property.

A Markov model is defined such that the states are directly visible to the observer. In a Hidden Markov Model, the state is not directly visible, but variables which are influenced by the state are visible. Each state has a probability distribution over the possible output tokens. Thus, the sequence of tokens generated by an HMM gives some information about the sequence of states. In addition to the state transitional probabilities, output probabilities from hidden to observed states have to be estimated in a Hidden Markov Model. For a formal definition and more detailed introduction to HMMs refer to e.g. Manning and Schütze [1999].

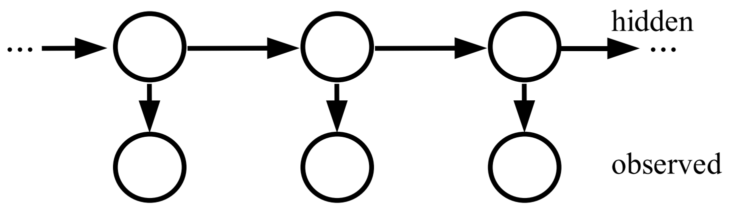

Figure 2.3: A first order Hidden Markov Model.

The models described in chapters 6, 7 and 8 can be thought of as Hidden Markov Models where the hidden states are letter–syllable-boundary, syllable–stress, or letter–phoneme-sequence pairs respectively, and the observed states are letters or syllables. This definition causes the output probability of a hidden state to be always equal to one, which is rather untypical for HMMs. But the model is not a Markov model either, because the paired symbols were not observed. In literature, I found similar models to simply have been called *joint n-gram models*, but I will nevertheless refer to these models as HMMs in this thesis.

2.1.1 The Viterbi Algorithm

The typical goal when using a Hidden Markov Model is to find the most probable complete hidden state sequence. The lattice that is built up during this calculation can be enormous: for a model with t observed symbols and n eligible hidden states, t^n paths have to be calculated and compared. For example, consider a word with 10 letters where each letter maps on average to 12 different phoneme sequences. Calculating the whole trellis would mean to calculate 10^{12} different state sequences.

The Viterbi algorithm is a dynamic programming algorithm for finding the most likely sequence of hidden states. The Viterbi algorithm stores the most probable path for getting to a state and its probability with each state. In the end, the algorithm chooses the state with the highest overall probability and goes backward to the beginning of the lattice to discover the final state sequence. Using the Viterbi algorithm is much more efficient than calculating each path in the whole trellis to the end, and reduces complexity to $O(n^2t)$. For more comprehensive explications of the Viterbi algorithm see e.g. Manning and Schütze [1999], Schöning [2001] or Carstensen *et al.* [2001].

2.2 Smoothing

When we want to estimate the probability of n-grams (i.e. sequences of n letters (or letter-phoneme pairs, or syllables, or words)), there will be some n-grams which we have seen very rarely, and there might be other n-grams which we have not seen at all in training. A statistical model that estimates probabilities of n-grams from data seen in the training set would therefore assign a probability of zero to these sequences. But in language, this is often not what we want: we do not assume to have seen all possible combinations of words, letters or syllables that may occur in our training set, and should therefore redistribute some of the probability mass to these unseen events. To estimate the probability of the unseen events as exactly as possible (because even among the unseen events, there are some which are more probable than others), the probability of an n-gram is usually estimated from the probability of a lower-order n-gram model, and weighted by a backoff-factor to make sure to obtain a probability distribution.
This procedure can be formally expressed as: $p(n\text{–}gram) = backoff\text{-}factor * p(n\text{-}1\text{–}gram)$

In smoothing, two types of models are commonly distinguished: interpolated models and backoff models. A backoff-model only looks at lower-order n-gram models in case the higher-order n-gram was not observed, while an interpolated model always combines the estimates from the the actual and all lower-order n-grams via linear interpolation.

An overview of the most commonly used smoothing methods is given in Chen and Goodman [1996].

2.3 Decision Trees

A decision tree is an n-branching tree with questions on its inner nodes and decisions on its leaves. A decision tree serves at solving a complex problem through answering questions on partial problems which are dependent on the complex problem. For g2p conversion, the complex problem is how to pronounce a letter, the questions concern letter identity and context of the letter. Breaking down the problem into smaller problems consists of deciding on the pronunciation of a letter in a particular context, which is a less difficult problem. The most commonly used types of decision trees are CART (Breimann [1984]), C4.5 (Quinlan [1993]) and CHAID (Sonquist and Morgan [1964]). These algorithms mainly differ in their pruning strategies and branching factors used.

A statistical decision tree defines a probability distribution over the set of possible choices.

2.3.1 The Entropy Criterion

The questions to be asked in a decision tree depend on the set of entities that are considered at a particular node. The goal in choosing a question is to reduce the entropy[1]

[1]Entropy is defined as a measure of disorder present in a system. In this context, it measures the uncertainty of how to classify the input symbol.

of the decision. This is achieved by choosing the question that yields the highest information gain, i.e. the question that divides the data set into classes such that each class has as low an entropy as possible. That is, the question is chosen such that its answers contribute as much as possible to the decision.

The information gain (IG) criterion is defined as the difference of the entropy (H) of the mother node and the weighted sum of the entropies of the child nodes d1 to dn.

$$\begin{aligned} IG(quest) &= H(mother) - H(mother|quest) \\ &= H(mother) - (p_{d1}H(d1) + p_{d2}H(d2) + ...p_{dn}H(dn)) \end{aligned}$$

where p_{dx} is the proportion of entities that are in branch dx
and $H(set) = -\sum_{x \in set} p(x) log_2 p(x)$.

The intuitive idea here is to choose the question that is most relevant. Information gain measures the reduction of uncertainty. If the question that maximizes information gain is chosen, the uncertainty in classification is reduced maximally.

In principle, the order of questions is irrelevant in an unpruned tree – the solution will always be the same. The entropy criterion is important for the efficiency of decision trees, as is prevents trees to become larger than necessary, and for pruned trees (see below).

2.3.2 Pruning

If no stopping criterion is used, the tree will grow (by asking new questions and thus branching further) until all the data is disambiguated or until there are no more questions to ask (e.g. if all the information the algorithm has access to does not suffice to disambiguate the examples). The tree then does not learn general rules any more but asks for specific properties of concrete instances. Disambiguating to the end improves the performance on the training data (as the training corpus is stored completely in the tree) but typically reduces the tree's ability for generalization, so that performance on unseen data is worse. This effect is commonly referred to as *over-training*, and can be prevented by pruning.

The most widely used pruning strategies are to either define a stopping rule that defines criteria for determining when a node may be divided further, or to first fully grow the tree and cut branches afterwards.

Chapter 3

Resources

This chapter briefly presents the corpora I used for training and testing. For German, these included CELEX, BOMP, the computational morphology SMOR which uses the lexicon IMSLEX, and a large taz newspaper corpus. The grapheme-to-phoneme conversion component was also evaluated on English and French, to assess generalizability of the approach to other languages – the resources used are briefly characterized here. The last section gives an overview of the German phonology I worked with (IBM Native Phonology for TTS).

3.1 CELEX

CELEX is a lexical database created by the Instituut voor Nederlandse Lexicografie (INL) and the Max-Planck-Institute, Nijmegen. It contains data for Dutch, English, and German, which was classified automatically and corrected manually.

CELEX contains information about orthography, orthographic syllable boundaries, phonetic transcriptions of the words that include syllable boundaries and primary and secondary stress markers. If a word has multiple alternative pronunciations or meanings, its orthographic form is included for each of these. There is also a data file for the derivational and compositional morphology of words for the base form lexicon and a second lexicon containing the inflected word forms, from which inflectional morpheme boundaries can be derived. Moreover, CELEX provides information about word classes and word frequencies.

German CELEX contains about 365,000 word forms, thereof 51,000 different stems. I here worked with a subset of approximately 250,000 words. The standard training set I am used contains 239,000 words, and the test set 12,326 words.

3.1.1 Phonetic Transcription

CELEX uses its own system of transcribing phonology, called Disc. The Disc representations were converted to IBM Native Phonology for TTS format (see section 3.6), which I worked with throughout this project. The CELEX data contains quite a lot of inconsistencies, in particular for word stress. For example, the syllable *-tion* is usually stressed in German unless it is followed by an even more stress-attracting suffix such as *-al* or *-ier*. In the CELEX data, only 20% of the *-tion* affixes are stressed. I manually went through the first 100 cases and found that they were all annotated incorrectly. There are some more inconsistencies, such as in 2. person inflections following stems that end in *-sch*, (*wä-*)*schst* is sometimes transcribed as `/S% S T/`, and sometimes as `/S% T/`.

Approximately 0.8% of the words in the test set occur twice (because they are ambiguous, such as *Bank* (Engl: bank or bench)), about one third of these have different phonetic transcriptions (0.3% or the words). Another particularity of CELEX is that some (0.5%) words have two annotated primary word stresses to indicate alternative stressing dependent on the context. These cases are problematic for learning and testing (as the word is correct if it has one of the stresses but not both), but I currently disregard the problem because of its low frequency. The best way to cope with these cases would probably be to manually disambiguate them by choosing one of the analyses. Most of the examples should not be ambiguous according to my intuition. Examples include: *stocksauer*, *weiterhin*, *stinkfaul*, *versprochenermaßen*.

3.1.2 Morphological Information

CELEX also contains manually annotated morphological information. Using good qualtiy morphological information has several benefits.

Firstly, the morphological analysis can be used to determine an upper boundary for the maximal quality that is reachable using morphological pre-processing for the input to the AWT, a syllabification or a stress-assignment step, and to estimate the degradation in quality due to morphological mistakes for each of these processes. Secondly, the analyses can be used to disambiguate SMOR-analyses and to learn a weighted FST based on these disambiguated analyses. Thirdly, these analyses can provide an evaluation standard, against which the output of the ETI-morphology and SMOR can be compared, thus measuring morphological performance by a different and more direct means than by its impact on grapheme-to-phoneme conversion.

In order to obtain enough morphological information to cover the whole training set and test set, some automatic processing had to be performed, as the data has to be retrieved from two separate CELEX data files. One file contains word stems and their morphological decomposition. Mapping was not always trivial, especially for word that are subject to vowel changes:

```
arbeitswillig Arbeit+s+willig
(((arbeit)[V])[N],(s)[A|N.A],(((woll)[V])[N],(ig)[A|N.])[A])[A]

Draufgaenger drauf+Gang+er
((((da)[B],(r)[B|B.P],(auf)[P])[B])[B],((geh)[V])[N],(er)[N|BN.])[N]
```

When consonant doubling occurs, it is not clear either from the lexicon where to insert the boundary. Also, there are some errors /inconsistencies for sequences of three identical letters (see Schiffahrt vs. Brennnessel):

```
abandonnieren Abandon+ier ((Abandon)[N],(ier)[V|N.])[V]
Schiffahrt Schiff+Fahrt ((Schiff)[N],((fahr)[V],(t)[N|V.])[N])[N]
Brennnessel brenn+Nessel ((brenn)[V],(Nessel)[N])[N]
```

Furthermore, not all words were decomposed. Some (the majority of the participles) were annotated with the flag [F], which stands for "Lexicalised flection". I annotated these (altogether more than 1000 entries) semi-automatically, by running them through SMOR and disambiguating them manually (circa 4 hours of work).

```
abfahren ab+fahr ((ab)[V|.V],(fahr)[V])[V]
Abfahrt Abfahrt ((Abfahrt)[F])[N]
dichtgedraengt dicht+gedraengt ((dicht)[A],(gedraengt)[F])[A]
```

Another file contained inflections of the words in the first file, so that automatic processing was necessary to merge the information. In particular, not all of the inflections needed (i.e. all forms that were in the training and test set I was using) were in the second file, so that the annotation of the missing entries had to be deduced from the entries available. This was not a difficult but time-consuming task.

Labels for morphological boundaries were derived from the annotation after each bracket – morphemes followed by [X] were assigned the "stem"-label, [X|.X] the "prefix"-label and [X|X.] the "suffix"-label. All suffixes from the first data file (the lemma file) were annotated as derivational suffixes while all affixes from the full form lexicon were labelled as inflectional suffixes.

3.1.3 Syllable Boundaries

There are two ways of acquiring syllabification information and adding it to the (morphologically annotated) data. Firstly, the syllable boundaries can be projected from the phonetic transcription if an alignment is available. I ran the alignment component of the AWT on both the training and the test data to obtain the syllabification information. The other possibility is to use the orthographic syllabification information available in CELEX. Orthographic and projected syllable boundaries differ in several ways: Because of orthographic conventions in German, the orthographic syllabified string does not always correspond directly to the original string. (Consider e.g. *Schiffahrt* vs. `Schiff-fahrt`[1], *backen* vs. `bak-ken`) and also contained some mistakes due to these conventions (e.g. `Brenn-esseln` instead of `Brenn-nesseln`). Direct mapping from phonemes leads to (on the grapheme level unconventional) syllabification due to the fact that *ng* in the suffix *-ungen* is transcribed as one phoneme. The orthographic convention would be to separate the word *Schulungen* into `Schu-lun-gen`, but the mapping from phonemes cannot do this and would yield `Schu-lu-ngen`. Differences also occur in the case of *st* because there is an orthographic rule forbidding to separate *s* and *t*, even though the natural syllabification is such.

[1]This problem does not occur any longer in new German orthography, where *Schifffahrt* has become the correct form.

3.2 BOMP

BOMP is another machine-readable pronunciation dictionary for German.[2] BOMP contains roughly 140,000 inflected words derived from about 61,000 stems. The overlap with CELEX (same word and same phonemization) is approximately 92,000 word forms.

Unfortunately, it cannot just be appended to CELEX to learn from more data, because this would make the data even more inconsistent. For example, syllabification is sometimes quite unusual:
`/. ? A . B R E) V . J A .1 T S J O: N/` (Abbreviation)
`/.1 ? A . P F A L .2 ? EI . M A#/` (Abfalleimer)

I used BOMP when I discovered that stress information in CELEX is very unreliable. Any data-based algorithm can never become better than the data it learns on. Therefore, I ran the algorithm only on those entries that had the same annotation in BOMP and CELEX (see chapter 7).

3.3 SMOR and IMSLEX

SMOR is a computational morphology which was developed at the Institute for Natural Language Processing (IMS), University of Stuttgart. SMOR contains an extensive lexicon, IMSLEX, which is annotated with many features (e.g. origin, complexity, word class). These features are used to alleviate the problem of overgeneration (which would occur if any affixes and stems were allowed to combine). The rules and the lexicon are compiled into a finite state transducer, using the SFST (Stuttgart Finite-State-Transducer) tools. The lexicon contains the stems of each word, which are marked as <Base_Stems>. If the stem varies in derivation or compounding, separate lexicon entries contain these forms. See below for some example lexicon entries.

```
<Base_Stems>Abasie<NN><base><frei><NFem/Sg>
<Base_Stems>Abbild<NN><base><nativ><NNeut_es_er>
<Base_Stems>Abbildung<NN><base><nativ><NFem-Deriv>
...
<ge><Base_Stems>schaffe:<>n:<><V><base><nativ><VVReg>
<ge><Base_Stems>schalte:<>n:<><V><base><nativ><VVReg>
<NoPref><Base_Stems>rund<VPART><ge>gehe:<>n:<><V><base><nativ><VVPres>
<NoPref><Base_Stems>rckdatiere:<>n:<><V><base><nativ><VVReg>
...
<Deriv_Stems>dedukt<V><deriv><lang>
<Deriv_Stems>deduz<V><deriv><kurz>
...
<Pref_Stems>ego<PREF><V><klassisch>
<Pref_Stems>ein<PREF><V><fremd,klassisch,nativ>
<Pref_Stems>einher<PREF><V><nativ>
...
<Suff_Stems><simplex,suffderiv><frei,fremd,nativ><kompos><V>mig<ADJ><SUFF>
<kompos><nativ>
<Suff_Stems><simplex,suffderiv><frei,gebunden><deriv><ADJ>ial<ADJ><SUFF><base>
<frei><Adj+>
```

[2]For more information consult www.ikp.uni-bonn.de/dt/forsch/phonetik/bomp/README.html.

The quality of the morphological analyses returned by the system is very high, as can be expected from the large amount of linguistic knowledge it contains. But as many rule-based systems, it cannot analyse words if their stem is not contained in the dictionary. In the case of CELEX, 10% of the words do not receive an analysis.

The SMOR version used for this project returns the following analysis for the word *Aufgabenheften*:

```
Aufgabe<>:n<NN>:<><kompos>:<><X>:<>H:heft<NN>:<><base>:<><X>:<><+NN>:<><Neut>:<>
<>:e<Dat>:n<Pl>:<>
Aufgabe<>:n<NN>:<><kompos>:<><X>:<>hefte:<>n:<><V>:<><deriv>:<><X>:<><deriv>:<>
<V>:<><>:e<>:n<NN>:<><SUFF>:<><base>:<><X>:<><+NN>:<><Neut>:<><NDA>:<><Sg>:<>
a:Auf<VPART>:<><V>:<><X>:<>ge:abe:<>n:<><V>:<><deriv>:<><X>:<><deriv>:<><V>:<>
<>:e<>:n<NN>:<><SUFF>:<><kompos>:<><X>:<>H:heft<NN>:<><base>:<><X>:<><+NN>:<>
<Neut>:<><>:e<Dat>:n<Pl>:<>
a:Auf<VPART>:<><V>:<><X>:<>ge:abe:<>n:<><V>:<><deriv>:<><X>:<><deriv>:<><V>:<>
<>:e<>:n<NN>:<><SUFF>:<><kompos>:<><X>:<>hefte:<>n:<><V>:<><deriv>:<><X>:<><deriv>
:<><V>:<><>:e<>:n<NN>:<><SUFF>:<><base>:<><X>:<><+NN>:<><Neut>:<><NDA>:<><Sg>:<>
```

The output of SMOR is then converted to the ETI digit notation:

```
Aufgabenheften 1aufgaben1heft2en 6auf1gab3en1heft2en
```

For a detailed discussion about using SMOR for morphological decomposition to improve g2p conversion quality, see chapter 5.

3.4 taz-Corpus

The taz-corpus (taz is a major German newspaper) comprises newspaper articles from the last 19 years, from 1986 to November 2005. This corresponds to 5.7 GB of data, or approx. 240 million words, 4.13 million tokens. The ratio of tokens that occurred just once is 62% (2.59 million).

- coverage of SMOR on taz
  There are 410,155 (26.5 %) out of the 1,543,466 different words in taz with a count of more than 2 which cannot be analysed by SMOR. Many of these are of course names, although SMOR does have a small processing component for names.

- coverage of CELEX on taz
  1,427,176 words cannot be directly looked up from the CELEX data base, that is 92% of the words that occur more than twice (of course, the more frequent ones tend to occur in CELEX, while the less frequent ones and names do not). Examples of very frequent words that are not contained in CELEX are: “Bundesregierung”, “Angebot”, “beispielsweise”.

It contains approximately 80 million words (2 GB worth of text), and 8 million different tokens (names, inflected word forms, compounds). Out of these, more than half of the tokens occur just once. This phenomenon is often referred to as the lnre-problem (large number of rare events), discussed in Möbius [2001].

In a first step, all meta-information, digits and punctuation marks were removed from the text in order to extract “pure” words (i.e. characters (and hyphens) enclosed by white spaces or punctuation). As I only needed the text to extract word frequencies, the information conveyed by punctuation was not relevant. However, there are a number of issues arising when trying to extract words (see also Möbius [2001]; Manning and Schütze [1999]).

1. abbreviations
   For the sake of simplicity, I also cut off full-stops in abbreviations. Alternatively, it would be possible to keep all those full-stops that are followed by a non-capital letter, make a list of these likely abbreviations, see whether any of these likely abbreviations also occur without being followed by a full-stop, in order to create a reliable list of abbreviations.

2. uppercase and lowercase
   A related issue is the question whether to map capitals and non-capitals all onto non-capitals or whether to keep the capitals, which would give us information about the occurrence frequency of verb-noun homographs, and to count separately common words and proper names (*die grünen* (adj.) *Wiesen* vs. *die Grünen* (political party)). Here again, the use of capitals at the beginning of a sentence is an issue, which can be easily solved for cases that are not ambiguous.

3. compounds and apostrophes
   A problem that generally arises when tokenizing English text is the common use of apostrophes (*it's, don't, Peter's*) and the use of white spaces in compounds (data base, natural language processing). These issues do not arise in German, as German compound nouns are written in a single word (e.g. *Datenbank*, *Sprachverarbeitung*).

4. separable verb prefixes
   The issue of separable verb prefixes is a real problem in this simplistic approach to word counting. In German, verbs with separable prefixes (English examples of which are “work out” or “spit up”) are very common. In German sentences with verb-second position usually have a long-distance relation between the verb and its separable prefix, which will be at the end of the sentence, as in “Das Auto *fährt* regelmäßig das Verkehrsschild beim Kreisel *um*.” (The car regularly knocks over the traffic sign near the round-about.) It is difficult to correctly relate the separable verb prefixes to their verbs, as separable prefixes in general are homographs with prepositions, so that syntactic processing would be necessary to detect that the *um* in the following sentence is not a separable verb prefix to the verb *fahren* (“Das Auto fährt regelmäßig beim Kreisel links um die Ecke.”) Why do we want this information at all? The training and test lists do not contain words that are made of two strings (e.g. “fährt um”) anyway. But it does contain the infinitives and verb-last-position forms “umfahren” and “umfährt”. Both of these are ambiguous with respect to a non-seperable-prefix-verb “umfahren” (to drive round). In a German main clause, the prefix stays with the verb (e.g. “Das Auto umfährt das Verkehrsschild beim Kreisel vorsichtig.” - The car carfully drives round the traffic sign near the round-about.) Reliable frequency counts for both verbs would be desirable for the TTS task, as separable prefix verbs are stressed on the separable verb prefix while verbs with non-separable prefixes are

stressed on the stem.
"...dass er das Schild **um**fährt." (= that he drives over the traffic sign)
"...dass er das Schild um**fährt**." (= that he drives round the traffic sign)
The distinction is meaningful and important for conveying the correct semantics of the sentences. Hence it would be beneficial to estimate occurrence frequencies for both verbs. With my naïve approach of not relating separated prefixes and their respective verbs, I cannot disambiguate verbs and therefore not estimate their frequency. Furthermore I will also underestimate the frequency of the ambiguous "umfahren" verb.

Based on these words extracted from the newspaper texts, I recorded word frequency information and stored each token and its corresponding frequency in a list, on which all further algorithms operated for efficiency reasons (the frequency list has a size of only approx. 80 MB). Frequency information is needed to estimate the relevance of errors made. Using frequency information from taz instead of CELEX allows to more reliably estimate how well the data performs on general text. Using CELEX frequency information would have biased the results.

The idea in doing a frequency-weighted evaluation is that it is obviously much worse to pronounce very frequent words incorrectly than to incorrectly pronounce a rarely occurring word. The simple word list on which evaluation is performed does not provide such information and does therefore not give a realistic estimate as to how well the system would work on real text. (This effect is also strengthened by the way of constructing the test set, which does not contain the same stems as the training set.)

I also used the taz words and word counts to train the RePortS algorithm and the Morfessor algorithm for unsupervised acquisition of morphemes, see section 5.4.3. Using taz gave results with much better quality than running the algorithm on CELEX alone. Moreover, I used the frequencies from the taz corpus when I ran the algorithms on CELEX data only.

3.5 English and French Resources

The lexica beep, Nettalk, The Teacher's Word Book and Brulex were downloaded from the Pascal challenge web page[3]. All of these corpora were provided with a letter-phoneme alignment.

3.5.1 English CELEX

For English CELEX, the same type of information is available as for German CELEX. The data set used contains approximately 72,000 words, 66,623 in the training set and 5,664 in the test set. See below the data format used. Letters and phonemes were aligned manually, each letter maps onto 0–3 phonemes.

```
abandon .' @ b .' { n d . @ n
abandoned .' @ b .' { n d . @ n d
abandoning .' @ b .' { n d . @ n . I N
```

[3]http://www.pascal-network.org/Challenges/PRONALSYL/Datasets/

3.5.2 NetTalk

The NetTalk corpus is a publicly accessible database containing 20,008 words from Webster's dictionary. It contains information about alignment, syllabification and stress. The annotated corpus was provided by Sejnowski and Rosenberg [1987] in the scope of their work on making neural networks pronounce English text. Their work received a lot of attention at the time and is claimed to be one of the most widely known applications of neural networks and data-based TTS systems.

Many data-based approaches to grapheme-to-phoneme conversion have also been evaluated on the NetTalk corpus, so that it is a good resource for comparing results to other's work despite its small size.

The data format of the NetTalk corpus is:

```
abbreviation    xb-rivieS-xn    0<>>2>01>0<<
```

Each letter is mapped onto one or zero (marked by the hyphen) phonemes. The third string contains syllabification and stress information and is also aligned with the word. Syllable boundaries can be inferred from the notation ('|' marks a syllable boundary):

1. $[<>] \rightarrow [<|>]$
2. $[< digit] \rightarrow [< |digit]$
3. $[digit >] \rightarrow [digit| >]$
4. $[digitdigit] \rightarrow [digit|digit]$

This leads to a syllabification of ab|bre|vi|a|tion or xb|-ri|vi|e|S-xn. As can be seen in this example, the syllabification is sometimes questionable – as x|b-ri|vi|e|S-xn would perhaps be preferable. The digits mark stress status of a syllable: 1 stands for primary stress, 2 for secondary stress and 0 for no stress.

3.5.3 beep

Beep is a large English dictionary containing almost 200,000 words (approx. 178,000 were used for training and 19,863 for testing). It was aligned automatically using the EM algorithm (cf. Damper *et al.* [2004]). Each letter is aligned with two phonemic characters. For an example alignment see the excerpt below.

```
ABANDONDED AX_BAE_N_DAX_N_DIH_D
ABANDONED AX_BAE_N_DAX_N_-_D
ABANDONEE AX_BAE_N_DOW_N_-IY
ABANDONER AX_BAE_N_DAX_NAX_-
```

3.5.4 Teacher's Word Book

The Teacher's Word Book is the smallest data set – it contains only about 16,000 words, and has a 1:0..1 alignment (dots stand for "no corresponding phoneme"). The problematic letter *x*, which maps onto two phonemes in most phoneme sets, is represented by just one symbol here (/n/ or /1/ respectively).

```
RETRACE     RgTRhS.
TOWER     Ti.A.
PIKE     PjK.
EXTRICATE     EnTRAKhT.
EXAMINE     I1cMIN.
MAXIMUM     McnIMAM
```

3.5.5 Brulex

Brulex is a rather small French dictionary, containing about 27,500 words. It does not contain verbal inflections. The alignment is done such that both the orthographic and the phonemic strings can contain empty elements, which are marked as '_'.

```
abandonner ab_`do_n_1
abaque aba__k
abasourdir abaz_uRdiR
_x1nophile  ks1nof_il_
```

3.6 IBM Native Phonology for TTS

The phonological symbol set used for the work on the German system in this project was the *IBM Native Phonology for TTS*. It contains 48 phonemes, a syllable marker and a stress marker. The complete set of symbols are listed below.

?	be_inhalten	L	Linie, Klavier
A	Altweiberfastnacht	M	Mutter, Pampa
A:	Bardame, Mahnmal	N	Nudel, Panne
A#	untermauerte, Werk	N)	Achtung, lenken
AE	exzellenter, Tendenz	O	Otter, Pathos
AE:	bärenmäßig	O:	Rose, Autos, Telefon
AN	Arrangeure, avangardistisch	OE	Öffnung, Gesöff
AU	auftaut, Rowdy	OE:	Öse, hören
B	Beschreibung	ON	Raison, Salon
CH1	Nachzucht, achtfach	P	Pause, Apfel
CH2	Chemie, Ungerechtigkeit	R	zurückkriegt, richtig
D	Rhododendron	R#	forderst, Kurve
E:	telegen, Allée	S	Spaß, Masse
E#	abgerissenerem, Bedenken	S%	Stiel, Masche
EI	Beichtgeheimnis	T	Rad, Rat, Stadt
EU	leutescheuer, Euter	U	Mutter, unter
EY	Playback, native (engl.)	U:	Mut, Utopie
F	vorwurfsfrei	V	Wolle, erwarten
G	gleichgültige	Y	üppig, Krücke
H	Höllenhund	Y:	übrig, übertrüge
I	wichtigtuerisch, Mitte	Z	Sonne, Vase
I:	zivilisiert, Miete	ZH	Journal, Passage
IN	Bassin, Teint	.	syllable boundary
J	Jahr, Yak, Majoran	.1	syll. bound., next syll. stressed
K	kleinkriegtet		

Chapter 4

Evaluation

Evaluation is important for comparing different methods and making reliable estimates about what method works best and why. I devote this section to evaluation because there are a number of different issues I would like to discuss. Evaluation results depend a lot on training and test set design, and in order to judge the reliability of results, it is necessary to think about the evaluation methods and the data used for evaluations. In the last section of this discussion, I will briefly explain the evaluation *measures* I used for this work.

4.1 Training and Test Set Design

The requirements for training and test data depend on the method used. When working with rule-based systems, no training is needed, so all of the data can be used for evaluation.

The situation is similar for unsupervised learning – the algorithm can be run and evaluated on the whole data set, (although it may be better to use a separate test set to make sure the algorithm was not tuned to the data set).

In supervised learning, the data set needs to be split up into a training set and a disjunct test set. If training and test sets overlap, or if the test set is also used for learning, it is not possible to estimate the algorithm's performance on unseen data. When algorithms that require the optimization of external parameters are used, the data should not only be split into a training and test set, but also a development set for estimate these values.

In general, the test set should be designed such that it is representative for the unseen data that we expect the model to encounter later. This is quite a difficult issue – in g2p conversion, experts have agreed that the fastest and most reliable method to determine the pronunciation of a word is to do a lexicon lookup and use compositional methods only for out-of-vocabulary (OOV) words.

But what are the typical OOV words? All words we can evaluate on are words that have actually been seen. When taking these words out of the training set, we *pretend* not to have seen these words. Common methods simply take every 10th word out of the vocabulary list, regardless of their frequency. This may bias results because the pronunciation of common words has not the same degree of reliability as pronunciation of rare words. Furthermore, the unseen words can be expected to contain a higher ratio of foreign words than the known words.

Perhaps it would therefore be good to put into the test set all words that occurred just once. Luckily, these issues are not relevant for comparing alternative algorithms.

As already mentioned above, the “standard” method in g2p is to randomly divide a data set into a training and a test set (cf. Taylor [2005]) by e.g. choosing 10% of the data set to be the test set and making the remaining 90% the training set. If inflected corpora are used, this method is problematic because the training and test set are not really independent of one another any more – especially in highly inflecting languages such as German. Every word in the test set will then consist of an already known stem and a commonly occurring affix.

Among the new words that will be encountered later there will be also words that contain previously unseen stems. Therefore, such a design does not model the properties of unseen words well – and one cannot reliably estimate the model’s ability for generalizing to new data.

There are two risks that one should be aware of when choosing such a test set design. Firstly, this can lead to over-training, because parameters (e.g. for pruning or smoothing) are estimated wrongly. Secondly, some problems, in particular those related to data sparsity, may not occur and therefore problems with them may not be recognizable.

An alternative strategy is to choose the test set such that the training and test set do not contain common stems other than in compounds. This can e.g. be achieved by randomly choosing words from the data set and taking out all words that differ from that word only by their ending. In the case of CELEX, this can be done very reliably as all inflected words point to their stem, and can therefore be separated cleanly from the other words. The German CELEX test and training set used in this project were designed according to this strategy. As can be seen in evaluations in chapter 6, word error rates are higher, but I believe that this is the better way to estimate the performance of an algorithm.

4.2 Evaluation Methods

When the data set size is small, the evaluation method becomes an issue – it is crucial for the algorithm to be trained on as much data as possible. On the other hand, evaluation results are less reliable when small test sets are used. It is then difficult to say whether difference in performance are just random deviation or a sign showing that one method is better than the other one.

This is why significance tests are performed. A significance test calculates the probability p that the observed number of errors originates from the same distribution as the expected error ratio. The formula for a two-tailed binomial test, which I used in this thesis is:

$$z = \frac{\frac{X}{n} - p}{\sqrt{\frac{pq}{n}}}$$

where X is the number of hits, n is the total number of events and p is the expected ratio of hits. q is the expected ratio of non-hits. Significance rates for the z-score can be looked up in tables. (e.g. http://www.math2.org/math/stat/distributions/z-dist.htm for one-tailed binary tests.)

The data for testing can be maximized by using 10-fold cross-validation (which is what I did for the small English data sets (NetTalk and Teacher's WB) and for test sets with small error rates resulting from using the 90/10 splitting). In 10-fold cross-validation, the data set is split into 10 parts. In 10 iterations, the model is then trained on 9 parts and evaluated on the remaining section. In the end, the average error rate of these 10 test sets is calculated, and this measure is then based on a higher number of words (the whole data set) and is thus less subject to random variation.

4.3 Evaluation Data

Another issue in evaluation is what data to evaluate against. When running a morphological segmentation algorithm, the results can either be evaluated against some gold standard for morphological segmentation (e.g. manually annotated data), or with respect to its influence on overall g2p conversion task performance.

Very importantly, an improvement on one value does not always correspond to an improvement on the other value. The alternative morphological algorithms presented in chapter 5 were evaluated against manually annotated morphological boundaries from CELEX. Best matches were obtained by using SMOR and a special disambiguation heuristic, but this higher similarity to the gold standard did not mirror in correspondingly better performance on the g2p task. It is thus always important to not only evaluate a method by itself but also within the context of the task to be tackled. This has also be found for other fields of computational linguistic (e.g. machine translation, cf. Fraser and Marcu [2006]).

4.4 Evaluation Measures

4.4.1 Error Rates

In this thesis, I evaluated syllabification, stress assignment and g2p conversion by reporting phoneme error rates (PER) or word error rates (WER). For phoneme error rates, the number of insertions, deletions and substitutions that would be necessary to obtain the target phonemic representation are counted and divided by the number of phonemes.

Word error rates are obtained by dividing the number of words that contain an error by the number of all words. For the word error measure, it does therefore not make a difference whether a word contains one mistake or ten mistakes. However, for intuitive estimation of phoneme string quality, the word error rate is perhaps more meaningful: getting 97% of the phonemes right seems to be rather nice, while a word error rate of 20% is actually quite high.

Another comment I want to make is that the reported word error rates reflect the occurrence of words in the word list. They are not representative for real text, where some words occur very often and thus contribute much more to the perceived amount of mistakes than some very rarely occurring words. For example, an unweighted word error rate of 23.3% corresponded to a weighted word error rate of 3.9% (for the AddWord decision tree given morphological information from SMOR). This number seems much better but also has to be treated with care: this is not the realistic word error rate either, as CELEX does not contain any names, which would be contained in real text – and automatic g2p systems have been shown to exhibit higher word error rates for names.

4.4.2 Precision, Recall and F-Measure

When I evaluated alternative morphologies with respect to a gold standard morphology, I did not use word error rate but precision, recall and f-measure. Precision counts the number of correctly found morpheme boundaries divided by the number of all morpheme boundaries assigned by the algorithm (i.e. if the precision of a morpheme segmentation algorithm is 95%, 95% out of the boundaries it found were correct). Recall counts the number of morphological boundaries that were found by the algorithm divided by the target number of morpheme boundaries (i.e. a recall of 50% means that 50% of the morpheme boundaries were found.) f-measure is the harmonic mean of recall and precision.

$$precision = \frac{tp}{tp+fp}$$

$$recall = \frac{tp}{tp+fn}$$

$$f-measure = \frac{2tp}{2tp+fn+fp}$$

where *tp* stands for “true positives”, *fp* stands for “false positives” and *fn* stands for “false negatives”.

Chapter 5

Morphology

This chapter gives an overview of systems that segment words into *morphemes*. A morpheme is defined as the smallest meaning-bearing unit in language. The benefit from morphological segmentation for g2p conversion is motivated linguistically, and the performance of several versions of the rule-based system SMOR and a range of unsupervised data-based methods is evaluated with respect to a gold-standard morphological segmentation and their effect on the g2p conversion task. Strength and weaknesses of the alternative systems are pointed out and compared.

5.1 Motivation of Morphological Analysis for g2p

The method that would produce the best transcription results would be to lookup the pronunciation of every word in a dictionary. But language is highly productive, new words are created frequently and the problem is especially severe for a language like German, where compounding is extraordinarily productive. A lexicon can never cover all words from an arbitrary text. For the taz German newspaper corpus, which consists of articles from the last 19 years, only 3% of the different words occurring are covered by CELEX (many of the unknown words are proper names and words with stems that are not in the lexicon, but there is also a considerable amount of compounds and inflections of known words among them). Machine learning techniques for generalizing pronunciation of known words to unknown words and systems that use manually constructed rules have been designed to cope with the out-of-vocabulary (OOV) words. For both approaches, generalization quality has been shown to increase when information about the structure of an unknown word was taken into account.

German pronunciation heavily depends on the inner morphological structure of words. It has already been shown by several research groups that morphological preprocessing can improve German Text-to-Speech Synthesis (for details see Möbius [1998]; Pounder and Kommenda [1986]). Morphological preprocessing has shown to decrease the phoneme error rate by over 20% in the IBM TTS-system, which currently uses a decision tree for the g2p task. Furthermore, a much smaller tree is needed to achieve similar performance rates.

Taylor [2005] and Black *et al.* [1998] suggest that the use of a morphology would also be beneficial for English. This can probably also be generalized to e.g. Swedish or Dutch, whose morphological processes are similar to those of German.

5.2 Effect of Morphological Boundaries

Morphological boundaries improve performance of the letter-to-phoneme conversion system: morphological annotation from the ETI-morphology reduces phoneme error rate (PER) from 26.6% to 21.8% when using the AddWordTree for grapheme-to-phoneme conversion). But it is not only beneficial to segment words but also to have information about segment types (suffix, stem, prefix). More fine-grained information can further improve results, e.g. if the stress status of a prefix is coded in the morphological boundary used, or if derivational and inflectional suffixes are distinguished. As can be seen in table 5.9, the labelled version of the ETI-segmentation outperforms the unlabelled version.

Similar to the ETI morphology, SMOR outputs linguistically rich segmentation information from which boundary types can be deduced. But none of the unsupervised algorithms provide labelling (with the exceptions of the algorithm by Bernhard [2006] which distinguishes prefixes, stems and suffixes, and Morfessor 2.0, which is not yet available publicly).

Throughout this work, I used the same morphological annotation of letter strings, which result in the word *Familienangelegenheiten* to be annotated `1Familie2n5an4ge1leg2en3heit2en`. The labels for morphological boundaries are:

1. **stem**
   A stem is a content word without any inflectional affixes. Knowing where a stem ends and where another stem begins is crucial for the correct insertion of glottal stops, syllable boundaries, syllable-final devoicing of stops and fricatives (Auslautverhärtung) and for assigning word stress. My favourite example in German is *Granatapfel*: if we did not know that there was a morphological boundary between *Granat* and *Apfel*, we would have to pronounce it `/G R A . N A .1 T A . P F E# L/` instead of
   `/G R A .1 N A: T . ? A . P F E# L/`.

2. **inflectional suffix or infix (Fugenmorphem)**
   The inflectional suffix has a positive effect on the disambiguation of the letter *e*. Consider for example the pronunciations of *er* in `5Be1richt5er1statt3ung` vs. `1Richt3er2s` or `1Arbeit4er1schaffen` vs. `1Arbeit2er3schaft`.
   Consonant clusters are argued to influence the duration and quality of vowels (Pounder and Kommenda [1986]). The word *rast* is pronounced with a long `/A:/` because it is followed by just one stem consonant and can be morphologically decomposed into `ras+t` while *Rast* is pronounced with a short `/A/` because the vowel is followed by two consonants that belong to the same morpheme.

3. **derivational suffix**
   Similar to stems, derivational suffixes often influence syllabification, stress assignment and lead to final devoicing (e.g. for the correct pronunciation of *Höschen* `/H OE: S . CH2 E# N/` instead of `/H OE . S% E# N/` or `/H OE: Z . CH2 E# N/`).
   Some heavy syllables (which should be stressed according to the default rules) are never stressed when they are derivational suffixes, e.g. *-schaft*, *-heit*, *-keit*, *-lein*. And, similar to inflectional suffixes, derivational suffix markers can serve for disambiguating pronunciations of vowels.

4. **stressed prefix**
   There are three prefix tags, one for always unstressed prefixes, one for prefixes that are sometimes or always stressed and one for separable verb prefixes (only introduced in labelling morphological boundaries from the SMOR morphology in which this information is available). Detection of prefixes plays a crucial role in syllabification, glottal stop assignment (for all stems that begin with a vowel) and also for disambiguating pronunciation of e.g. *v* in
   `/V E: .1 R A N . D A:/` (Veranda) vs.
   `/F AE R .1 ? A N T . W O R# . T U N)/` (Ver+antwort+ung).

5. **unstressed prefix**
   see stressed prefixes

6. **separable prefix**
   Separable prefixes are always stressed. This is not only relevant for verbs but also for deverbal nouns, e.g. *Ǘbergang*, *Ǘbersicht*, *Ǘberfluß* vs. *Übersétzung*, *Überträgung*). Otherwise, the same arguments as for the other prefixes are valid.

5.3 Morphological Information from CELEX

To obtain a gold standard for morphological information, I extracted the manually annotated morphological information from the German CELEX database, and showed that perfect[1] morphological segmentation can lead to further improvement (see table 5.1). The morphological segmentation systems were not only compared with respect to their performance on the grapheme-to-phoneme conversion task, but also with respect to their similarity to CELEX manual morphological information. For results, see table 5.6.

5.4 Systems for Morphological Analysis

There are several dimensions of morphological analysis: finding the correct segmentation of a word, clustering words according to their stems, detecting morphological paradigms. Here, we are only interested in the first task: segmenting a word into the smallest meaning-bearing units, its *morphemes*.

There have been several approaches in NLP to build systems that accomplish this task. The most traditional systems are rule based systems. They rely on linguistically motivated rules and large affix lists and lexica. More recently, data-based approaches have also been taken to tackle morphological segmentation. We distinguish two types of data-based methods: *supervised* ones which take as an input morphologically annotated data and *unsupervised* methods, which work on raw text. This makes truly unsupervised methods particularly attractive, as virtually no work is needed to adapt them to other languages. However, state-of-the-art unsupervised algorithms have not yet reached the quality of rule-based systems.

[1]In fact, morphological information from CELEX is also far from perfect, see comments in chapter 3, but it's the best segmentation available, as there is no gold standard for morphological decomposition in German.

5.4.1 Rule Based Approaches

Rule-based systems are currently the most common approach to morphological decomposition and usually perform best in segmenting words. But there are several disadvantages in rule-based systems. Firstly, the acquisition of a substantial rule-based morphology is very labour-intense and can only be done by an expert linguist who knows the language concerned very well. Secondly, large knowledge bases are needed, such as comprehensive affix lists and a large lexicon, often annotated with linguistic features that are relevant for morphology. Experience from language processing shows that, due to the LNRE[2] property of language, there are always some words that are not in the lexicon, in particular foreign words and new words. So a rule-based system has to be updated permanently.

Another issue is the disambiguation of alternative analyses. In rule-based systems this can either be accomplished by assigning probabilities to analyses (which need to be acquired first from large data sets or be added to the morphology manually from linguistic insight), or by arranging rules in a certain order, which may prevent the desired analysis from being detected. In such a system, rules heavily depend on one another, making it very difficult to modify rules.

5.4.1.1 Bell Labs System

The Bell Labs System described in (Möbius [1998]) is a TTS-system for German that exploits the morphological structure of words and is implemented as a weighted finite state transducer (WFST). The transition weights of the transducer are assigned manually based on linguistic expert knowledge, and the productivity of affixes. Affix productivity has been shown to be proportional to the ratio of hapax legomena[3] of the words they occur with. The weights cause the analysis which includes a productive affix to be preferred over an analysis with an unproductive one.

Unknown words are segmented using a phonotactic[4] syllable model. This model tries to match known affixes within the word and assigns costs to any previously unaccounted orthographic substrings based on their phonotactics (whether the letter sequence can possibly be a stem, whether there is a letter sequence that can never occur within a morpheme and therefore gives high evidence for the insertion of a morpheme boundary). The order of stems and affixes is furthermore coordinated by specific rules (that require e.g. that a prefix cannot be directly followed by a suffix).

A difficult problem is furthermore the pronunciation of names, which make up a large amount of the incorrectly pronounced words. In Jannedy and Möbius [1997], the authors discuss the construction and use of a specialized finite state transducer for name analysis, which they show to perform much better on names than the generic system.

[2]Large Number of Rare Events.

[3]A hapax legomenon is a morpheme that only occurs with one single stem, for example *Schorn*stein, *Him*beere, *Schmetter*ling.

[4]Phonotactics is a branch of phonology that deals with restrictions in a language on the permissible combinations of phonemes.

5.4.1.2 ETI-Morphology

The ETI morphological analysis module is a rule-based system that relies on an extensive lexicon. The rules (when to peal off what affixes) are arranged in a specific order such that exactly one analysis is returned for each input word. Words that cannot be segmented because parts of them are not in the lexicon, are heuristically decomposed assuming a default “unknown” morpheme and pealing off as many affixes as possible. Mistakes can roughly be classified into two categories that are due to missing lexicon entries:

1. **unknown stems are decomposed too far**
   For example, if the word *Abderit* was not in the lexicon, it is segmented into `Ab+derit`, as *ab* is a common German prefix. This behaviour leads to relatively high recall but low precision on affixes (see tables 5.6 and 5.7 below).

2. **compounds with at least one unknown stem are not decomposed**
   A default “unknown” stem is not assumed in the case of compounds. Therefore, compounds are not decomposed when one of the stems is not in the lexicon (e.g. `1bardame` (desired analysis: `1bar1dame`), `5aus1sagesatz` (`5aus1sage1satz`), `1belegexemplar` (`1beleg1exemplar`)). This lowers recall on stem boundary detection.

The ETI-morphology was nevertheless the morphology which performed best on the grapheme-to-phoneme conversion task.

5.4.1.3 SMOR-Morphology with FSTs

SMOR, a computational morphology developed at the Institute for Natural Language Processing (IMS) at the University of Stuttgart, covers inflection, derivation and compounding. It is based on DMOR, DeKo and IMSLex and uses a finite state transducer (FST) to compute morphological analyses.

Finite state transducers have been widely used in rule-based systems, due to the fact that morphological rules and exception lexicons can be compiled into FSTs to work efficiently. Chapter 2 of Karttunen [2001] explains why a FST is adequate for the specific task of morphological analysis. Further systems that used FSTs are described in (Sproat [1996]; Beesley).

SMOR comprises a large lexicon containing affix lists and a stem list which, in addition to the canonical stem forms, also lists irregular derivational stems and compounding stems. To prevent over-generation, each entry is annotated with a set of features that are required to match in order for two morphemes to compose. These features ask e.g. for word class, stem type, origin, complexity, whether the verb can take a prefix, whether an *ß* is converted to *ss* (for a full list of features, cf. Schmid *et al.* [2004]).

The analysis for the word *Aktentaschen* looks like this:

```
Akt<NN>:<><kompos>:<><X>:<>ent<VPREF>:<><V>:<><X>:<>asche:<>n:<><V>:<><deriv>:<>
<X>:<><deriv>:<><V>:<><>:e<>:n<NN>:<><SUFF>:<><base>:<><X>:<><+NN>:<><Neut>:<>
<NDA>:<><Sg>:<>
Akte<>:n<NN>:<><kompos>:<><X>:<>T:tasche<NN>:<><base>:<><X>:<><+NN>:<><Fem>:<><>:n
```

```
<NGDA>:<><Pl>:<>
a:Akt<V>:<><deriv>:<><X>:<><deriv>:<><V>:<><NN>:<><SUFF>:<><kompos>:<><X>:<>ent
<VPREF>:<><V>:<><X>:<>asche:<>n:<><V>:<><deriv>:<><X>:<><deriv>:<><V>:<><>:e<>:n
<NN>:<><SUFF>:<><base>:<><X>:<><+NN>:<><Neut>:<><NDA>:<><Sg>:<>
a:Akt<V>:<><deriv>:<><X>:<><deriv>:<><V>:<><>:e<>:n<NN>:<><SUFF>:<><kompos>:<>
<X>:<>T:tasche<NN>:<><base>:<><X>:<><+NN>:<><Fem>:<><>:n<NGDA>:<><Pl>:<>
```

This form contains some information which is irrelevant to the actual task, for example agreement information. Therefore, all analyses that map on the same segmentation are only kept once. Furthermore, the representation is transferred to the format that is used by the ETI-morphology. For the current example, this would be:

```
1akt4ent1asch3en
1akten1tasche2n
1akt3en1tasche2n
```

The overall quality of analyses from SMOR is linguistically better than the ETI analyses (SMOR analyses have a much higher precision with respect to manually annotated data than ETI analyses, cf. table 5.6), but performance on the grapheme-to-phonem conversion task is worse for the following reasons:

1. **coverage**
   SMOR's coverage of the CELEX data is only 90%. OOV words include *Kirschsaft* (because the compositional stem *Kirsch-* is not included in the lexicon), *Abdampf*, *Adhäsion*, *Agglutination*, *entmagnetisieren*. Also, conjunctive II forms are included in CELEX in two versions: with and without an *e*. SMOR currently only analyses the version which contains an *e* (e.g. *entsprächst*, *entsprächt* vs. *entsprächest*, *entsprächet*).

2. **complex lexicon entries**
   The lexicon contains complex words which could be decomposed further (e.g. *Abbildung*, *Abschwung*, *Granatapfel*), and for which we do not get the information necessary to insert glottal stops (at the *a* in *apfel*), place syllable boundaries (between *t* and *a* in *Granatapfel*) or assign word stress correctly (to the first syllable of *Abbild*, instead of the second).

   SMOR uses a number of features to segment words and prevent over-generation. These features for example disallow the verb prefix *ab* to combine with a noun. For the word *Abbildungen*, there are thus two possible segmentations: `Ab+bild+ung+en` (`ab+bild` comes from the verb *abbilden*) and `Abbildung+en` (based on the lexicon entry *Abbildung*), but for the word *Abbild*, the only possible analysis is `Abbild`, as there is no regular process that would transform the prefixed verb *abbilden* into the noun *Abbild*. Additionally, SMOR has a filter that leads to under-segmentation in prefixes in order to preserve tractability in the transducer: every word can at most have one prefix, words as *unübersetzbar* are assumed to base on the complex stem *übersetz*.

3. **ambiguity**
   The finite state transducer is not weighted. Any alternative analyses can not be disambiguated by the transducer. It is therefore not clear, which out of the set

of analyses should be chosen. Most of the analyses generated by SMOR are linguistically possible, but many are extremely unlikely (such as `Abt+eil+ung+en` or `Akt+ent+aschen`, `Ab+fal+leim+er` or `Ab+end+kleid`). The average number of different segmentations per word is 2.4(!).

Possible solutions to these problems are the following:

1. **Introduction of a default stem to obtain wider coverage of the lexicon**
   The advantage of such an approach is that it would make use of the grammar's knowledge about possible affixes. On the other hand, the unknown words might begin or end in a prefix or suffix by chance, this is particularly true for very short (one letter) and rare or unproductive affixes. My suggestion is therefore to include a mechanism that takes into account affix length, probability and productivity as well as phonotactic and orthographic constraints (to prevent a segmentation like `A+ffix`). I did not implement the proposed constraints but only the default segmentation that peals of affixes.

2. **Segmentation of lexicon entries**
   For tasks in syntax or semantics it may be better not to segment lexicalized complex words such as *Abbild* or *Flugzeug*, but, as motivated above, we need to look at the lexicon from another perspective for this task. To integrate both types of granularities into the lexicon, I automatically inserted additional morphological boundaries into these complex words. The entry `Abbildung` was changed to `Ab<X>bild<X>ung`, where `<X>` is a morpheme boundary. Ideally, the newly inserted morpheme boundaries should also be encoded to state that *ab* is the prefix, *bild* the stem and *ung* a suffix. The automatic segmentation of lexicon entries was performed using the affix lists and stem lists from the lexicon itself. But the automatic segmentation introduces new ambiguities, and thus aggravate the disambiguation problem. In table 5.1, the approach is referred to as SMOR-newLex.

   Overall, introducing segment boundaries into the lexicon reduced precision from 92% to 88% (because segment boundaries were annotated automatically and not corrected manually, therefore they contain some errors) and improved recall from 75% to 80%. This results in an f-measure increase from 82.1% to 83.6% (significant at $p < 0.000001$ according to a two-tailed binomial test).

3. **Disambiguation**
   In a first, very simple approach, I used the system's inbuilt -d option. This option always selects the analysis with the smallest number of morphemes. In general, this leads to linguistically good results. But we face also the most severe problem of under-segmentation (i.e. very low recall) for words such as *Abbildung*, that can be segmented as `Ab+bild+ung` but also have own lexicon entries.

   To improve on recall, I then decided to always pick the analysis which splits the word into the largest number of segments, but this led to very low performance, too, this time due to low precision (e.g. `Akt+ent+asche+n` is preferred over `Akten+tasche+n`, and `Abt+ei+lunge+n` is preferred over `Abteilung+en`). A more sophisticated version of this heuristic (which actually returned the best overall results) was to always select the SMOR analysis with the largest number of segments *if* at least one stem segment from the analysis with the smallest number of segments was in the same position. This heuristic returns `Ab+bild+ung+en` instead of `Abbildung+en`, and prefers `Akten+tasche+n` over `Akt+ent+asch+en`. Unfortunately, it did not improve results on grapheme-to-phoneme conversion either (wrt. ETI), although it has a higher f-measure than the ETI-segmentation with respect to the CELEX manually annotated data.

Another way to disambiguate alternative analyses is to use *probabilities* for ranking the goodness of an analysis. In a finite state transducer, weights that represent transitional costs can be added to the arcs, to rank alternative paths with respect to one another. Weights can be added to a finite state transducer by manual annotation following human intuitions (this is the most laborious and difficult way, as it is hard to estimate the various interactions of weightings), based on transitional probabilities learnt from large amounts of annotated data, or based on productivity information (Möbius [1998]) for affix disambiguation (can be acquired even from raw text), or a combination of these. In a first approach, I derived the weighting for the finite state transducer from text through the acquisition of word and affix frequencies from the taz corpus. The underlying idea was that frequencies should help to correctly disambiguate cases where words are segmented into very infrequent morphemes, such as `Top+fes` (Engl: top (best) + Fes (fez = type of oriental hat)) instead of `Topf+es` (genitive of the word *Topf* = pot). The disambiguated analyses were then used to learn transitional probabilities for the SMOR finite state transducer. This did lead to some improvement, but frequency alone did not seem to be a sufficiently good disambiguator.

Much better results were in fact obtained when I used the morphological information contained in CELEX to disambiguate SMOR analyses. Disambiguation consisted of always choosing the analysis that was most similar to the CELEX analysis. However, due to the way arch probabilities are multiplied in the transducer, analyses with a low small number of segments are preferred over analyses with many segments. Therefore, under-segmentation is still a problem (cf. tables 5.6 and 5.1).

SMOR as a Resource

SMOR is a very rich resource, with information about stem type (native, classic, foreign), word category etc. Therefore, it would be interesting to at least use some of this information. The most direct usages are to extract stem and affix lists to build a low-resource naïve morphology (see section 5.4.1.4). SMOR annotation about separable prefixes can also be valuable for the pronunciation of verbs that are ambiguous with respect to their primary stress position, (i.e. úmfahren vs. umfáhren, ǘberlegen vs. überlégen,). Verbs that can be interpreted as having a separable prefix are marked with the "VPART"-tag. However, within our word list, this information is of limited value, as we have no means to disambiguate an ambiguous verb. But the information can help to determine the accent for verbs that only have one possible reading.

Another approach to exploiting SMOR would be to add pronunciations to each lexicon entry (e.g. add all the data available from CELEX). A new word could then be analysed by SMOR, and in a first step be given the concatenated pronunciation based on the separate pronunciations of each morpheme. To do justice to numerous processes like syllable-final consonant devoicing, stress shift etc., transformation based learning (TBL) could be used to learn transformation rules (from a training set) which could then be used on the concatenated string and would lead to the final pronunciation. The main drawback from this approach is that disambiguation and lacking coverage would still be an unsolved problem, and that pronunciations for all SMOR lexicon entries would have to be obtained.

Native vs. Foreign Words

In another experiment, foreign word were separated from native and classic words to learn two separate decision trees – one for native words and one for non-native words. To determine what are foreign words (like “Computer”, “Abandon”, “Bataille”, “Billard”), all words that were marked as “foreign” in SMOR were singled out. This process is a good approximation of what a foreign word is, but it is by no means perfect, as the annotation in SMOR marks only the kind of affixes a word can combine with – and a foreign word may be inflected like a German word even if its pronunciation does not follow German pronunciation rules. Compounds with a foreign bit in them may be marked as native, such as “computergesteuert”. Therefore, all words that contained other words with a “foreign”-mark were also removed from the set of native words.

The processing of words unknown to SMOR (and for which no consequently provenence tag is available), would work as follows: The new word would be processed by both the foreign and the native tree and the pronunciation from the tree that assigned it the higher probability would be chosen. This idea is based on the observation that analyses which turn out to be wrong have a considerably lower average probability than words that are transcribed correctly.

For each of the word sets (native and non-native words), two decision trees were learnt for grapheme-to-phoneme conversion were learnt. While results of the native-only tree were improved (PER 2.68%, WER 20.51%) (and even more so if all words that do not have an analysis in SMOR are taken out (PER 2.48%, WER 19.60%)), the pronunciations of the foreign words are very error-prone (PER 6.49%, WER 41.44%). The high error rate on foreign words can partly be explained by insufficient training set size and the heterogenity among the foreign words.

If the whole set of words is analysed using the tree trained on native words only, there are slightly more errors than when learning is performed on the mixed data (PER 2.89%, WER 21.77%). In conclusion, trying to process native and foreign words separately is difficult: the training set for foreign words is not big enough. To overcome the data sparsity problem, a model for words from each language would have to be built (i.e. one for German, one for French and one for Spanish, as these languages make up the largest proportion of foreign words in German.) These models should probably not be trained on original data sets from these languages, as typical German native speakers do not use the whole phonetic inventory of other languages but map many sounds onto sounds in their own language.

Combining two Morphologies

I was interested in seeing how the results from SMOR could be used to improve performance of the letter-to-phoneme conversion step, and therefore also tried to combine the output of SMOR and ETI in order to find an upper boundary for performance. This approach would of course be unattractive for a practical application, as it is a very resource-consuming solution to run both systems. For example, I always selected the SMOR analysis that was most similar to an analysis from the ETI-morphology. Although the increased f-measure showed that the morphological segmentation is more close to the target segmentation, it does not perform better on the g2p conversion task (see table 5.1).

version	WER	PER	f-meas.	Prec.	Recall
SMOR-d	24.99%	3.28%	71.8%	**95.4%**	57.6%
SMOR-heuristic	23.25%	**2.92%**	82.1%	90.2%	75.4%
SMOR+newLex	23.76%	3.00%	**83.6%**	87.1%	**80.4%**
SMOR-CELEX-WFST	24.16%	3.22%	76.4%	94.9%	63.9%
SMOR-ETI	22.33%	2.85%	81.7%	92.3%	73.3%

Table 5.1: Comparison of the effect of morphological decomposition from alternative versions of SMOR on the g2p task.

Evaluation on weighted word lists

The taz frequency information also allowed me to evaluate whether more mistakes were made with frequent words or whether errors were committed on infrequent words. A system that returns few errors on infrequent words but relatively many errors on frequent words is good for a design where all known words are simply looked up in a lexicon. A system that is designed for running on a device with very limited memory capacity would better be such that it makes very few errors on the frequent words, hoping that the rare words occur so seldom that performance is not impaired too much. For results from the frequency-weighted evaluation of morphological preprocessing for grapheme to phoneme conversion see table 5.2.

morphology	words corr.	words wrong	types corr.	types wrong	ratio
ETI	31,301,332	2,112,552	4,531	1,146	6.3%
SMOR-heur	32,108,944	1,304,940	4,445	1,232	3.9%

Table 5.2: Evaluating ETI and different versions of SMOR using weighted frequencies from taz.

As the ETI-morphology has shown to work best for the current task, it would also be interesting to augment ETI-lexicon with the entries from SMOR, thus covering a larger vocabulary, which would improve on precision.

5.4.1.4 Naïvely pealing off affixes

In the previous sections, I have argued that morphological annotation can improve conversion quality significantly. But a morphology with many rules is very time-consuming to build for new languages. What if we had a full affix list of this language? I wrote a simple script that peals off as many suffixes from a word as possible.

Alternative matches are disambiguated by pealing off the longest affixes first and introducing a restriction on minimal stem length. But unfortunately, this naïve morphology has a negative effect on letter-to-phoneme conversion quality.

An explication for this phenomenon is that the grapheme-to-phoneme conversion algorithms used here, the decision tree and the Hidden Markov model, capture the context and learn how an affix is typically pronounced. The morphological information is primarily needed for *disambiguating* letter strings as to whether they are an affix or part of the stem (like in *vertikal* where the *v* should be pronounced /V/ instead of /F/). The other big issue are morpheme boundaries in compounds, which the affix lists cannot resolve.

Information is "lost" by the simple disambiguation strategy of always trying to peal off the longest possible affix. The letter-to-phoneme conversion algorithm cannot relate the remaining (pseudo-)stem to the true stem any more. Another reason for why the annotation actually *harms* performance (instead of simple bringing no improvement), is that the algorithm actually loses some information as it sees less of the surrounding letter context when more characters are added to the string while keeping the context window size constant. This naïve affix list-based algorithm is used as a second baseline (first baseline: no morphological annotation) for comparing the performance of algorithms.

5.4.2 Data-based Approaches: Supervised

For an overview of past research on supervised approaches to morphology acquisition see chapter 2.1 in Schone and Jurafsky [2001]. Supervised approaches are less attractive than unsupervised approaches, as they require a lot of morphologically annotated data for learning. It would certainly be interesting to run a HMM similar to the syllable tagging model (see chapter 6) on the morphologically annotated data and observe what performance rates can be achieved when using supervised learning. Unfortunately, I did not have the time to try this in the scope of this thesis. Following the good results with the g2p-HMM (see chapter 8), I do not expect any significant improvements to be gained from introducing a separate module for morphological segmentation that uses the same method as the g2p algorithm.

5.4.3 Data-based Approaches: Unsupervised

Using unsupervised methods is attractive, because they can be applied by non-linguists and do not require any language-specific knowledge. The only input needed is a large corpus or list of words.

From 10th to 12th of April, I attended the 2nd Pascal[5] Workshop in Venice, Italy.

[5]European Network of Excellence, Pascal stands for Pattern Analysis, Statistical Modelling and

During the workshop, results from a recent challenge on language-independent automatic acquisition of morphemes were reported. For evaluation, algorithms had been applied to English, Finnish and Turkish corpora, and were also evaluated with respect to their usefulness for a speech recognition task. The approaches taken were very diverse and seemed interesting as a preprocessing step for the grapheme to phoneme conversion task. The three winners of the challenge (evaluation was done on different languages, so that there were different winners for the tasks) agreed to let me use their algorithms in order to test whether the their morphological segmentations can improve results for the letter-to-phoneme conversion task. Furthermore, I downloaded an older version (Morphessor 1.0) of the algorithm from the Morpho-Challenge organizers. The best-scoring versions (Morphessors 2.0 and 3.0) are not yet available for download (expected: June 2006).

All of these unsupervised methods assume morphology to be a concatenative process. The evaluation set-up was as follows: I tested each of the algorithms on English and German, and evaluated their benefits for g2p conversion with the AWT decision tree (and the best one also with the HMM model). The output of the unsupervised algorithms only indicates unlabelled morpheme boundaries.

Interestingly, none of the algorithms outperformed all others on all evaluation languages. This indicates firstly that different types of languages exhibit different types of morphological problems, and secondly that the unsupervised algorithms probably make some language-specific assumptions. Therefore, it is either necessary to choose the appropriate algorithm dependent on the type of language, or to combine different algorithms and exploit their strength for the different types of morphological phenomena, in order to obtain a system that can cope well with all of them.

Approaches to unsupervised acquisition of morphology were classified by Goldsmith [2001] into the following categories:

1. identification of morpheme boundaries based on conditional entropy
2. identification of bi-grams that have a high likelihood of being morpheme internal
3. discovery of rules for phonological relationships between pairs of related words
4. seeking an analysis that compresses the corpus optimally

Goldsmith [2001] discusses difficulties with first two approaches and concludes that morphology should not operate on a n-gram word environment, but rather on a higher level, because he finds statistical links between local sequencing of phonemes or letters to be too weak.

In the following four sections (using Goldsmith's categories for structuring the approaches), I will explain the most important approaches and those which I used to for morphological preprocessing in more detail.

5.4.3.1 Splitting based on segment predictability

The original idea for this approach was developed by Harris [1955]: word splitting should occur if the number of distinct letters (for words from a word list) after a given

Computational Reasoning.

sequence of letters rises above a certain threshold. This approach was implemented by Hafer and Weiss [1974], but they found it to give very noisy results if applied to a simple word list. Hafer and Weiss therefore experimented with the idea and developed 15 different algorithms that differed in their design and parameters. The psychological plausibility of this theory is supported by experiments discussed in Saffran *et al.* [1996] which suggest that humans also rely on transition probabilities to identify word boundaries. The idea underlying this approach is used (but not as strictly as in the first approaches) by Bernhard [2006]; Bordag [2006]; Keshava and Pitler [2006].

The approach presented in **Bordag [2006]** comprises two steps. A set of plausible affixes is obtained in the first step, and the second step uses these affixes to find morphemes for which less evidence can be found in the data. Such a two-step approach has also been used in another algorithm based on Harris' letter predictability approach (cf. Déjean [1998]).

Bordag's first step is based on Harris' letter successor variety (LSV). LSV is calculated among semantically similar words. The semantic similarity of words is thereby determined from context similarity and edit distance. According to this implementation, two words are similar if their contexts contain many identical words (latent semantic analysis (LSA) is not used), and if they look similar. The intuition here is that semantically similar words should have the same root or affix. This integration of semantic information alleviates the problem observed by Hafer and Weiss [1974] who had obtained very noisy results from using the whole lexicon. To reduce typical errors committed by the simple LSV-approach, the LSV value is combined with a measure for bi-gram weight (how probable it is in general to split two particular letters) and a substring frequency measure (to down-weight letter sequences that are graphemes (e.g. *sch*, *th*) but favour frequently occurring sequences are possible affixes).

What's interesting about this first step is the incorporation of semantic information and the use of edit distance to find similar words – using edit distance would also be generalizable to finding infixes or circumfixes, instead of always focussing on prefixes and suffixes (which are admittedly the most common phenomena in the Indo-European language group). Another nice property of this algorithm is that it is not limited to a certain number of morphemes but can split a word into an arbitrarily large number of morphemes.

But, as Bordag reported in Bordag [2005], precision for the segmentation obtained from this first step is high (about 80%), but recall is low. Therefore, a trie[6], made of the words for which there was enough evidence in the first step, is used to segment the rest of the words in a second step. The use of a forward and a backward tree is well suited for prefixation and suffixation, but not ideal for handling infixes. Several affixes are split off by iteratively applying the tries until no more splits are found. To prevent excessive over-segmentation, a threshold (either the affix or the stem must consist of minimally three letters) is introduced.

In MorphoChallenge 2006, a version of Bordag's algorithm which only performs the first step won the precision competitions for Finnish and Turkish. But recall was

[6]A trie (the word was derived from reTRIEval) is a special type of the more general tree. Tries are used to search simultaneously for strings that begin with the same substring. Each branch of a trie represents an additional letter, and every node corresponds to the string that can be obtained when appending the labels from all labels down to the node. The root node represents an empty string.

even much lower than for English and German, because the corpora were relatively small and because of the high number of word forms in these morphologically rich languages. This sufferance from data sparseness can also be observed for German (despite a corpus of 500 million words[7] and 9 million different word forms) where the algorithm (perhaps unsurprisingly) performs worse on infrequent words than on more frequent ones. For example on German, an f-measure of 0.704 (precision: 0.687, recall: 0.721) according to Bordag [2006] is reached on the 100,000 most common words of CELEX, but only 0.614 f-measure with 0.606 precision and 0.623 recall (values from personal communication) for the whole of German CELEX (approx. 250,000 words). If we assume that the word segmentation is only needed for very rarely occurring words (as we could store the pronunciation of frequently occurring words in a lexicon), this method seems to be not very well suited as a preprocessing step to letter-to-phoneme conversion, as it suffers more severely from data sparseness than some of the other algorithms considered.

The approach presented in **Bernhard [2006]** is also based on Harris' segment predictability, but uses transitional probabilities instead of counting the number of different letters following a given substring. Contrarily to the approaches by Bordag [2006]; Keshava and Pitler [2006]; Schone and Jurafsky [2001], it does not use trees to peal off prefixes or suffixes from the sides of the word but searches for similar stems. Similarly to the later Morfessor versions (refer to Creutz and Lagus [2004, 2005]), it uses three different categories: prefix, stem, and suffix as well as an additional category for linking elements (like the German Fugenmorpheme) and imposes constraints on how these can be combined. For all transitions between two letters in a word, the transitional probabilities to all substrings before the potential split and after the potential split are calculated. Minima in the transitional probability curve are accepted as morpheme boundaries if they deviate more than one standard deviation from the preceding and following maxima. The longest and least frequent segment is added to the stem list, if it occurs at least twice in the rest of the data and once in the beginning of a word, and if it is followed by at least two different letters. A problem is the data sparsity for stems in which consonant doubling, e-deletion, umlauting or ablauting occurs – these cannot be validated by this method. Once the affix and stem lists have been acquired, each word is segmented with respect to these lists. Any ambiguities are solved by using a cost function. Bernhard proposes two different cost functions:

$$cost_1(s_i) = -\log \frac{f(s_i)}{\sum_i f(f_i)} \quad cost_2(s_i) = -\log \frac{f(s_i)}{\max_i[f(s_i)]}$$

Following my experiments with the disambiguation of SMOR, it would be interesting not to use simple word segment frequencies but transitional probabilities (for example a HMM as in Creutz and Lagus [2004]), or affix productivity (as in Möbius [1998]).

The results of running three different data sets were compared for German CELEX: 1) unweighted CELEX, 2) weighted CELEX using weights from taz and disambiguating with cost function 1, 3) weighted CELEX using weights from taz and disambiguating with the second cost function.

[7]Projekt Deutscher Wortschatz: http://wortschatz.uni-leipzig.de/.

In MorphoChallenge 2006, Bernhard's algorithm was the winner[8] for Finnish and Turkish, and came second for English.

Morphology	Precision	Recall	F-Measure	PER	WER
unweighted CELEX	0.380	0.566	0.452	4.46%	31.89%
weighted CELEX, method 1	0.649	0.621	0.635	3.88%	28.26%
weighted CELEX, method 2	0.566	0.687	0.621	5.04%	34.98%

Table 5.3: Different data sets for Bernhard's algorithm.

The approach presented in **Keshava and Pitler [2006]** combines the ideas of using words that appear as substrings of other words and of using transitional probabilities.

The core element of the algorithm are a forward tree and a backward tree, which are used to calculate the conditional probabilities of a letter given the preceding letter sequence. The forward tree is used to determine suffixes while the backward tree is used to find prefixes. The trees encode the probability of a letter given the letters that precede it in the word (for the forward tree), and the probability of each letter given all letters that follow it (for the backward tree) respectively.

Then, two lists of potential prefixes and suffixes are constructed using three conditions:

1. The letter sequence before the potential suffix should also be a word in the corpus.

2. The transitional probability of the last two stem letters should be close to 1.

3. The transitional probability from the last letter of the stem to the first letter of the hypothesized suffix should be < 1.

If all three conditions are met, the potential suffix is rewarded (it gets some credit), otherwise it is punished (its score is deduced). Prefixes and suffixes that have a positive score in the end are included into the final prefix and suffix lists, which is then pruned in order to eliminate affixes that are concatenations of two or more suffixes (such as *-ungen* in German).

In the word segmentation step, morpheme boundaries are inserted using the prefix and suffix lists: For each word, all possible suffixes are pealed off. In the case of ambiguities, probabilities from the trees are used. Then the same process is repeated for prefixes. Compounding is solved partially because the prefix and suffix lists may also contain stems, especially those which often occur in a certain position in the word.

The algorithm achieved by far the best results for English (80% f-score), but had low recall for Finnish and Turkish, which is due to the invalid assumption that a proper word is obtained after pealing off a suffix. Another problem are infixes, as the set of potential affixes only comprises prefixes and suffixes that occur in the beginning or end of the integral words from word list. The algorithm has the same problems as Bernhard's algorithm concerning stem variations due to consonant doubling, umlauting and ablauting.

[8]Among participants; the organizer's best Morfessor system was slightly better.

I tried the algorithm on my German data with different data sets: 1) German taz (only words that occur more than once), 2) German CELEX unweighted word list, and 3) German CELEX frequency weighted word list. In approaches 2) and 3), no suffixes were validated, whereas the prefix lists looked very good for all three approaches. The algorithm's failure to find suffixes in German can be explained as follows: The first condition, saying that a word whose affix was stripped off should still be a word in the corpus is not met. German verb stems do not occur on their own (except from certain imperative forms), so a word like *abholst* that is stripped of its suffix is *abhol* which cannot be found in the lexicon, although words like *abholen*, *abholt*, *abhole* or *abholung* are part of the corpus. The problem for German nouns is similar. While many foreign suffixes such as *-ation* or *-ismus* (engl. *-ation*, *-ism*) are found for English, due to word pairs like *catholic – catholicism*, *commercial – commercialism*, *cynic – cynicism*, *accept – acceptation*, *adapt – adaptation*, such pairs are very rare for the German equivalents: *katholisch – Katholizismus – Katholik*, *adaptieren – Adaptation*, *akzeptieren – Akteptanz*.

These suffixes are punished far too often and thus even lead the list for lowest scores! Therefore, the first condition would have be changed for languages with agglutinative morphology. The introduction of a stem list in addition to the prefix and suffix lists would probably also be beneficial; such a list would also help to alleviate the problem of over-segmentation in short words that are decomposed into a prefix and a suffix but no stem or a very short stem (e.g. `über+kl+ei+de+n`, `über+f+uhr`, `über+b+last`, `über+b+leib+e`, `über+brachtet`, `über+b+ring+e`).

The version that was trained on taz (running the corpus for this corpus took more than 10 hours on a 2.8 GHz machine) achieved much more promising results. Here, a long list of suffixes could be identified, many of which are real suffixes. But, supporting my hypothesis about the problem resulting from criterion 1, some very common suffixes, such as *-ung* and prefixes such as *ver-* did not obtain sufficiently high scores.

Another common source of mistakes is that suffixes can be stripped off in any order. Here, a HMM or similar model as used in the second version of Morfessor (see below) could help to overcome this problem.

Remarkably, the RePortS algorithm does not suffer from problems observed with the MDL-approach Creutz and Lagus [2006] or with Bordag [2006], namely the detection of graphemes such as *sch* in German or *th* in English. These are punished too often when the remaining word is not part of the lexicon.

Morphology	Precision	Recall	F-Measure
unweighted CELEX	0.832	0.250	0.385
weighted CELEX	0.597	0.388	0.470
unweighted taz	0.711	0.507	0.592

Table 5.4: Different Data Sets for RePortS.

I now want to compare this approach to my naïve approach of segmenting words by using linguistically verified prefix and suffix lists.
Intuitive advantages of the unsupervised approach:

1. prefix and suffix lists contain stems, therefore compounding can be resolved to

a certain extent.

2. if segmentation is ambiguous, probabilities are used.
3. if the transitional probability equals 1, the affixes is not pealed off (e.g. *vertikal* will not be segmented)

Intuitive disadvantages:

1. some affixes may not be found
2. the algorithm may find incorrect affixes

5.4.3.2 Probability of letter sequences to occur morpheme-internally

The second type of approaches is based on the heuristic that certain letter sequences do not occur morpheme-internally but are only seen at morphological boundaries. In German, for example, the letter sequences *tw* or *hh* never occur morpheme-internally but only in morphologically complex words such as *Nachtwache* or *nachholen*, whereas the strings *ch* or *qu* only occur morph-internally. For a broader discussion of and references to algorithms that use this approach see Goldsmith [2001].

The approach is not very promising, in particular when considering that this kind of information is captured by the statistical models for g2p conversion any way, as their decision on how to transcribe a letter is dependent on the neighbour letters.

5.4.3.3 Word Comparison (similarity and dissimilarity of parts of words)

Measuring the similarity of words is an interesting approach in that the basic idea also allows to capture non-concatenative processes such as umlauting, ablauting, letter insertions or deletions or similar regular letter-variance patterns.
In Neuvel and Fulop [2002], the authors describe an implementation based on Whole Word Morphology, which does not recur to the notion of a morpheme but learn transformation patterns for whole words. Two words are assumed to be morphologically related if all differences between them are also found in at least one other pair of words from the same lexicon. The input of the algorithm is a tagged lexicon. Evaluation was not done by analysing words (the approach does not allow to do this as it does not have the notion of a morpheme) but by generating word inflections and measuring the correctness of these word forms; the authors report precision levels of 70–80%. The approach is interesting because it is well-suited for detecting morphological relatedness, which is an important concept that most of the other unsupervised algorithms lack and which causes them to perform badly on words that exhibit stem variation.

The following two approaches (by Jacquemin [1997]; Schone and Jurafsky [2001]) make use of contextual information (like Bordag, see above). They work on full text instead of word lists and extract co-occurrences of words, thus exploiting *syntactic* and *semantic* information.

In Jacquemin [1997], similarities and dissimilarities are identified as word formation strategies. Similarities are thereby determined using tries (cf. second step in Bordag [2006]). Jacquemin's approach is, like many of the others discussed here, specific to word formation by prefixing and suffixing. Words are deemed to be related if they have the same first few letters as other words and satisfy length conditions for the remaining stem. The simple application of this intuition leads to very noisy results. Therefore, Jacquemin only accepted as affixes differences between words from same context (again, cf. Bordag [2006]).

As in (Neuvel and Fulop [2002]), differences that are only seen for one hypothesized word pair are rejected. Jacquemin furthermore improved performance by clustering groups of words with the same ending, which was later adopted by Goldsmith who called these sets of strings that occur with the same stem *signatures*. In another approach, (Jacquemin [1997]) exploits similar words in small contexts to introduce semantic relatedness. But as the contexts considered are very strict and the data set seems rather small, recall is inhibitively low.

The approach by Schone and Jurafsky [2000, 2001] is quite similar to Jacquemin's approach in that it also uses two tries to find frequently occurring differences at the beginning and end of words, which are then regarded as potential affixes. Schone and Jurafsky also cluster words into conjugation groups, but extend the usage of semantic and syntactic relatedness to filter affix candidates, such as the potential prefix *t-*, which comes up because there are many words in English that only differ in this letter (like *train* and *rain* or *table* and *able*). They exploit the fact that semantic vectors obtained from latent semantic analysis for *train* and *rain* are not very strongly correlated, while those of e.g. *rain* and *raining* are. Unsurprisingly, this method suffers from the same problems as Bordag's approach: many stems occur very rarely, so that not enough contextual information can be gathered for doing LSA. Therefore, a reliable affix lists and a stem inventory is determined in a first step, while the affix sets are then applied to more rarely occurring words in a second step. Jurafsky and Schone's approach tackles the detection of suffixes, prefixes and circumfixes, but it is not clear how well composition would be covered. The approach is very resource-intense as the calculation of LSA takes up a lot of time and space. Unfortunately, it is not possible to directly estimate the segmentation quality of this algorithm as it was not evaluated for segmentation but only with respect to its detection of conflation sets.

5.4.3.4 Minimal Description Length (MDL)

The use of minimal description length is motivated by the intuition that a good morphological decomposition of words must be from a model which compresses the data well and which is simple at the same time. To compress the data means that the number of letters in a lexicon and affix list is smaller than the number of letters in a word list (because for all words on *-ing* (working, eating, sleeping), the ending *ing* is just stored once, and for all inflections of the same word such as *working*, *works* and *worked*, the root *work* is only stored once, too). If we would only optimize this criterion, the optimal compression would be obtained by storing each letter on its own. But then the model required to describe the words would be very complicated, having to explain the relationship between each of the letters in a word. Therefore, minimal description length also optimizes for simplicity of the model (the most simple model would be a look-up word list).

However, the theory of minimal description length does not tell us how to generate the model or how to determine what is a stem and what is an affix. It is used to *compare* models and select the best model out of a set of alternative models.

A shortcoming of the approach is that it does not take into account frequencies and might therefore detect high frequency letter sequences even if they are not morphemes (such as *th*, *str* in English or *sch* in German).

A very influential paper that used minimal description length to learn morphological segmentation for English and other European languages was (**Goldsmith [2001]**). He proposed several heuristics for developing a probabilistic morphological grammar which are then evaluated using MDL. The input of the algorithm is raw data and the algorithm tries to split words into stems and affixes. Goldsmith's implementation of the approach in the program Linguistica is freely available for download[9].

An important drawback is that longer prefixes such as *zusammen-* or stems in compounds are not found by the current version of Linguistica, as the threshold for maximal length of a prefix is fixed to 5. German infixes (Fugenmorpheme) cannot be found either.

The main steps in Goldsmith's algorithm are:

1. **discovery of candidate suffixes using weighted mutual information**
   Two heuristics are used to identify candidate suffixes. The first one is to split all words into two parts (disregarding whether they can be split or not) and optimizing for description length. The method turns out to be strongly biased towards single letters. The problem is fixed by strongly dispreferring one-letter segments, but the heuristic still tends to prefer graphemes that consist of several letters. (In German, these are e.g. *sch*, *ch*.)

   Goldsmith's second heuristic is to try to peal off the word suffixes of different length (e.g. 1 *g*, 2 *ng* to 6 *orking*) and use weighted mutual information to rank them. The best ones are kept (they contain the most frequently seen sequences of letters at the end of a word) and used as (high-quality) suffix-candidates to parse all words. The MDL criterion is then used to identify the optimally compressing suffixes.

2. **detection of regular signatures**
   Each stem is assigned its signature, i.e. all endings it was seen with. For example in German, the stem *repar-* has the signature `ation.iert`. All signatures that appear just once are deleted, as well as those that appear only with one stem, thus improving the quality of the endings and removing pseudo-stems.

3. **use maximum description length to correct errors generated by heuristics**
   After these two steps there are still a number of typical problems, which Goldsmith points out and tries to solve using, again, the MDL criterion:

[9]http://humanities.uchicago.edu/faculty/goldsmith/Linguistica2000

- suffixes that contain several suffixes (e.g. *-ungen*)
  All suffixes that can be generated by composing two other suffixes are removed and additional words are generated (e.g. *Reinigung* from original *Reinigungen*).
- spurious signatures
  e.g. `atzes.ätze` due to the words *Aussagesatz* and *Aussagesätze*
- relatedness between stems is not detected
  This problem (due to umlauting, ablauting, consonant doubling, e-deletion etc.) is tackled by introducing a procedure for identifying related stems, which tries to delete one letter, insert a letter or replace a letter and thereby tries to find consistent patterns (e.g. “replace *a* by *ä*” in German).

However, there are of course some mistakes which remain: for example there are some words for which evidence is not large enough: for example, *ver-* is a very frequent prefix but should not be pealed off a word as *vertikal*. This is a difficult problem: if a prefix is not assumed for previously unseen stems (here *tikal*), then recall will be low, as there are many words that are only seen in their prefixed variant, or are extremely sparse due to stem changes such as ablauts, e.g. *Verkehr*, *versäumen*, *versunken*.

Goldsmith reports very high precision (85.9%) and recall (90.4%), but it is difficult to compare these values to my results or the results from the Morpho-Challenge workshop, as Goldsmith’s evaluation is based on manual inspection and did not use the same corpus.

Another MDL model that is designed to cope with an arbitrary number of morphemes in a word is the Morfessor System **Creutz and Lagus [2006]**, that can also be downloaded freely[10]. The Morfessor system is from the organizers of the Pascal Morpho-Challenge. It system has shown to perform well on highly inflecting and compounding languages such as Finnish and Turkish and its second version (not available for download yet) returned very good results for English, too.

The Morfessor algorithm uses a greedy search strategy for segmenting words in order to obtain a more concise representation of the lexicon. Every word is split in all positions and candidate boundaries with the highest probability (i.e. those whose parts also occur in other words) are kept. This process is repeated until convergence. The results are dependent on the order in which words are selected. The order of the words is therefore randomized in each step, to limit the effects of artefacts. Another decision that needs to be taken when running the algorithm is, whether it should be run on a full corpus, i.e. take into account word tokens, or whether it should be run on a word list in which each word occurs only once, i.e. on word types. Creutz [2003] observed that learning from word types (i.e. using a word list) leads to significantly higher recall but slightly lower precision. Also, the lexicon size influences precision and recall quality. A larger corpus leads to higher precision but lower recall. It is not clear what lexicon or word list size to choose in order to obtain optimal results. F-measure was found to

[10]http://www.cis.hut.fi/projects/morpho/

increase for English with increasing corpus length, but for Finnish, best results were obtained with a smaller lexicon (see Creutz and Lagus [2005]).

For this evaluation, I used the Morfessor 1.0 software package. The more recent versions "Categories-ML" and "Categories MAP", which were also presented at Morpho-Challenge, perform better, and additionally give information about boundary types and even return a hierarchical analysis of the morphemes. These new versions would therefore probably be more interesting than the currently available version. The new algorithms will probably be made available during summer 2006.

The main problems with the results from Morfessor 1.0 are

1. under-segmentation of frequent words
   Can be explained by the way the algorithm works: A frequent morpheme is best stored as a single entity, whereas rare words are more efficiently split up into parts, using pointers to these various parts to compose the word.

2. over-segmentation of rare words (e.g. `ab+m+e+ierung`, `s+ie+st+en`, `s+en+s+e`)

3. morphological violations (e.g. `t+ränkung`, `t+röstung`, `s+en+s+ation`, `s+e+e+gang`).
   In these cases, the frequent suffixes *-s*, *-e*, *-t*, *-en*, should only be used as suffixes and not be allowed in any place in the word.

The majority of these problems are tackled in the second version, called Morfessor Categories-ML (see Creutz and Lagus [2004]), which has not been made available yet. At Morpho-Challenge, the authors reported that they introduces categories "prefix", "suffix", "stem", and "noise". In a first step, the baseline algorithms is run as usual. Then, a HMM is used to estimate the probability of a morpheme of belonging to these classes and the context-sensitivity of the HMM makes morphotactic violations less likely to occur. In several iterations, words are split further and re-concatenated, while minimizing description length.

These modifications increased f-measure values for Finnish and Turkish considerably (by 10 to 20 points), and also helped for English. A positive effect can therefore also be expected for German.

A further extension called Categories-MAP Creutz and Lagus [2005] outputs a hierarchy of morphemes. Under-segmentation is prevented by enforcing the splitting of a string into morphemes where possible. The problem of over-segmentation is alleviated by not allowing for "noise"-morphs. F-measure is for all languages similar to that of the third version, but in English, precision is much higher and recall lower than for the Categories-ML approach.

The Categories-ML and Categories-MAP algorithms have a language-dependent parameter, whereas the baseline version, Morphessor1.0, is knowledge-free and language-independent. The parameter determines the likeliness of seeing a prefix / stem / suffix in a certain place and is tuned to the target language. The language-specific parameter can be estimated using a small development corpus.

When inspecting the data, I found recall to be very low. Many affixes (prefixes as well as suffixes) were not recognized, because they were too frequent or because they did not occur often enough in a particular form but with a different or no affix. Here, German doubling of consonants, deletion of *e*'s, ablauting and umlauting aggravate the problem of data sparseness (as seen for most algorithms).

morphology	Precision	Recall	F-Measure	PER	WER
taz + CELEX	0.651	0.251	0.363	4.17%	29.9%
weighted CELEX	0.639	0.205	0.310	4.10%	29.6%
unweighted CELEX	0.709	0.418	0.526	4.17%	30.0%

Table 5.5: Different Data Sets for Morfessor1.0.

5.5 Homographs

A difficult problem within text-to-speech synthesis is the contextually appropriate pronunciation of homographs. Homographs are words that are spelled identically but pronounced differently. English has a very serious problem with homographs, for example there are many verb–noun homographs such as “At present, we present him a present.” These can only be disambiguated in context; in word lists the difference cannot be learnt (unless the relevant information, e.g. POS tag, is provided). German has similar phenomena: e.g. *modern* (modern / to rot), *rast/Rast* (speed / break), *Heroin* (heroine or female hero), *Knie* (knee, pronounced `/K N I:/` in its singular form and `/K N I: E#/` in plural). Another source of homographs that with different stress patterns can arise from compounds such as *Lebensmittelpunkt*, which can mean *center of life* or *food badge*. The most common case however are ambiguities in stress assignment for verbs that have the same form with a separable and non-separable prefix: e.g., *durchströmen* (flow through), *umfahren* (drive around / knock over), *umspannen* (embrace / transform), *unterstellen* (to take shelter / assume).

To quantify the problem of homographs in German: CELEX contains 1,238 homographs that are not homophones (that is about 0.5% of all words). There are two problems for machine learning and evaluation: A machine learning algorithm can only return a single best answer if it is not given any context, so for each of these pairs, it will get one of the words wrong, even if it has learnt German pronunciation perfectly. At the same time, the chance of getting at least one of the different versions right is higher.

In previous research, homographs have often been taken out of the corpus altogether or have been disambiguated manually. The problem of homographs in TTS are discussed in more detail in Yarowsky [1996].

5.6 Results

In this section, I will give an overview of the algorithms' performances on evaluation with respect to the CELEX manually annotated data and on the conversion task. A first important point is that higher f-measure with respect to CELEX correlates with a lower grapheme-to-phoneme conversion error rate only in very rough terms: The rule-based methods achieved consistently higher f-score than the data-based methods (between 71% and 84% f-measure, see table 5.6) and their transcription scores were also all better than the no-morphology baseline, while segmentations from unsupervised morphologies obtained f-scores of only 53%–64%. Disappointingly, all segmentations from unsupervised algorithms made the conversion results worse than they are without morphological annotation.

But smaller improvements in f-measure did not always entail an improvement in phoneme error rate. For example, ETI segmentation reaches an f-score of 79.5%, but a lower PER than the f-measure-wise best version of SMOR, which had an f-score of over 83%.

The interpretation of the data is difficult: It is not clear whether precision or recall is more important for performing well on the conversion task. Therefore, I analysed the data further to see what kind of morphological boundaries each of the algorithms got right or wrong, and also looked into the types of mistakes made in g2p conversion.

morphology	Precision	Recall	F-Measure	PER AWT
CELEX-gold standard				2.64%
ETI	0.754	**0.841**	0.795	**2.78**%
SMOR-ETI	0.923	0.733	0.817	2.85%
SMOR-heuristic	0.902	0.754	0.821	2.92%
SMOR-newLex	0.871	0.804	**0.836**	3.00%
SMOR-CELEX-WFST	0.949	0.639	0.764	3.22%
SMOR-large segments	**0.954**	0.576	0.718	3.28%
`no morphology`				3.63%
best RePortS (taz+CELEX)	0.711	0.507	0.592	3.83%
best Bernhard (w. CELEX)	0.649	0.621	0.635	3.88%
naïvely cut affixes	0.553	0.683	0.611	4.00%
best Morfessor (unw. CELEX)	0.709	0.418	0.526	4.10%
Bordag	0.665	0.619	0.641	4.38%

Table 5.6: Evaluating rule-based and data-based systems for morphological segmentation with respect to German CELEX (Cen [1995]) manual morphological annotation.

When looking at the types of morphological segments found by the algorithm, the high performance of the ETI morphology on stems and prefixes but comparably low recall on suffixes is striking. This finding also indicates a special problem in assessing precision and recall for suffixes: the correct place for a suffix boundary is often much less well-defined than for prefix and stem boundaries. It is not always evident how far a word should be decomposed – for example, when consonant doubling occurs, should the doubled consonant be part of the stem or part or the affix?

Most interesting is perhaps the question why the Bordag algorithm performs so badly on the conversion task (PER= 4.38%, which is worst of all morphologies tested), although it achieves the highest f-score among unsupervised methods. When looking at the boundary type recall figures, the Bordag algorithm performs best on suffix boundaries, while both the Bernhard algorithm and the RePortS algorithm outperform it on the detection of stems. This indicates that identifying stems correctly has a larger impact on g2p conversion than the identification of suffixes.

morphology	stem	prefix	suffix
ETI	0.937	0.954	0.790
SMOR-d	0.527	0.658	0.595
SMOR-heur.2+newLex	0.848	0.896	0.778
best RePortS (taz+CELEX)	0.711	0.507	0.592
best Bernhard	0.638	0.593	0.616
cut-affs	0.599	0.651	0.724
Bordag	0.575	0.522	0.647
best Morfessor (unw. CELEX)	0.415	0.367	0.425

Table 5.7: Recall for types of morphological segments.

To reduce errors, it is crucial to know what kinds of errors are made in order to identify their causes and effectively improve performance of the system. Table 5.8 provides an overview of the types of errors in function of the morphological preprocessing used. For a more detailed discussion of conversion problems please refer to chapter 8.Tpyes

of errors:

1. **syllabification or stress errors**
   `/.1 ? A P . F A: R# . T/` instead of `/.1 ? A P . F A: R# T/`
   (spurious syllable boundary in *Abfahrt*)
   `/.1 ? A P . N O R# . M I: . T AE: . T E# N/` instead of
   `/. ? A P . N O R# . M I: .1 T AE: . T E# N/`
   (wrong word stress in *Abnormitäten*)

2. **incorrect vowel quality**
   `/.1 L AE S . ? A: R# T/` instead of `/.1 L E: S . ? A: R# T/`
   (wrong quality of *e* in *Lesart*)

3. **incorrect vowel length**
   `/.1 K Y: S . CH2 E# N/` instead of `/.1 K Y S . CH2 E# N/`
   (incorrect long `/Y:/` sound in Küßchen)

4. **consonant errors**
   `/.1 ? A CH1 . T S I CH2 . S% T E# L/` instead of
   `/.1 ? A CH1 . T S I CH2 . S T E# L/`
   (*st* pronounced as `/S% T/` instead of `/S T/` in *achzigstel*)
   `/. P A T . T S E# .1 R EI/` instead of
   `/. P A T . T S E# .1 R EI/`
   (*t* pronounced twice in *Patzerei*)

Finally, different kinds of errors may be propagated differently with different algorithms. Table 5.9 lists the phoneme error rates and word error rates with the decision tree and the HMM.

Morphology	WER	vowel length	vowel quality	syllab. or stress	consonants
Bordag	4.38%	397 (9%)	1194 (28%)	2637 (62%)	37 (1%)
best Morfessor	4.1%	524 (9%)	1799 (31%)	3551 (60%)	72 (1%)
affix only baseline	4.00%	541 (9%)	1647 (29%)	3569 (61%)	68 (1%)
best Bernhard	3.88%	459 (8%)	1503 (26%)	3691 (65%)	36 (1%)
best RePortS	3.83%	369 (7%)	1554 (28%)	3616 (65%)	56 (1%)
no morphology	3.63%	455 (9%)	1487 (28%)	3312 (63%)	41 (1%)
WFST-SMOR	3.22%	381 (8%)	1156 (24%)	3189 (67%)	34 (1%)
ETI	2.78%	393 (9%)	1031 (26%)	2632 (64%)	42 (1%)
CELEX	2.64%	291 (7%)	867 (21%)	2880 (71%)	20 (0.5%)

Table 5.8: An overview of the conversion error classes for each morphological system.

Morphology	AWT-PER	AWT-WER	g2p-HMM with constr.
Bordag	4.38%	31.35%	
Morfessor	4.1%	30.00%	
affix only baseline	4.00%	29.32%	
Bernhard	3.88%	28.26%	
RePortS	3.83%	28.03%	15.1%
no morphology	3.63%	26.59%	13.7%
WFST-SMOR	3.22%	24.16%	
ETI unlabeled	2.94%	21.95%	
ETI	2.78%	21.13%	13.6%
CELEX	2.64%	21.64%	13.2%

Table 5.9: Using a Morphology as a Preprocessing Step for AWT and the HMM.

5.7 Discussion

My experiments showed that good morphological annotation can significantly improve grapheme-to-phoneme conversion with the AWT algorithm. Annotating the graphemic representation of words with morphological boundaries eliminates some of the typical mistakes that a system which does not have access to such information frequently commits (no glottal stop, incorrect syllabification, errors on vowels that cannot be disambiguated without knowledge about morphological structure). These results therefore seem to confirm the finding of Richard Sproat (Sproat [1996]), saying that optimal results cannot be expected from a gp2 system for languages such as German, English or Dutch, if it does not have access to morphological information.

However, if morphological information is not good enough (i.e. if it does not have an f-score $> 70\%$ for German), the wrong or missing morphological information makes the algorithm perform worse. None of the unsupervised algorithms reached sufficiently good performance to lead to an improvement on the g2p task. However, I think that using the Morfessor 2.0 and 3.0 version, which will soon become available, might be worth trying, as they achieved large performance gains on Turkish, Finnish and English. The approach by Keshava and Pitler is also very promising, in particular when the good performance of their algorithm on English is considered. Substituting the inappropriate assumption that a word whose affixes have been pealed off is a again a word, by a notion of stems can possibly lead to a performance boost for highly inflecting languages.

The experiments on the SMOR morphology showed that SMOR lacks some information that is crucial to g2p conversion. Evaluation with respect to a gold-standard morphology (extracted from the CELEX database) showed that precision is much better for SMOR than for the ETI morphology. I think that SMOR can outperform ETI if 1) morphological boundaries are manually inserted into the lexicon, 2) weights are learnt for the transducer, and 3) if a default strategy for coping with OOV words is implemented.

Chapter 6

Syllabification

In this chapter, I first motivate the implementation of a separate syllabification component, explain how syllabification works in German and give an overview of previous research on the problem. I then present the statistical syllabification model used and very briefly address the smoothing issue. Results from experiments with alternative input representations (letters, morphologically annotated letters, phonemes) are presented and remaining errors analysed. In the final section, performance within the TTS pipeline is discussed.

6.1 Motivation for a Syllabification Component

There were two main reasons for designing a separate syllabification component and for not letting the information be learnt in just one step by the letter-to-phoneme conversion module. Firstly, about 60% of the phoneme conversion errors are syllabification or stress errors, the majority being stress errors. The decision tree cannot capture enough context and does not apply any restrictions to prevent the violation of phonological contstraints. Therefore, there is a virtue in building a component that can take into account the relevant context and apply constraints. While I want to keep the model as language-independent as possible, a general constraint saying that every syllable must have a nucleus is to my knowledge valid for all languages. The set of possible nuclei may differ from one language to another[1], but the construction of a list of possible syllable nuclei for a language is still a reasonalby easy and staightforward task.

Syllable boundaries are needed as a basis for performing stress assignment. And because vowel length and quality sometimes depend on word stress in German, arranging the modules in the order: 1. mophological analysis, 2. syllabification (syllable boundaries depend on morphological boundaries), 3. word stress assignment and 4. letter to phoneme conversion, is well-motivated.

The second reason for introducing a separate syllabification step is that Marchand and Damper [2005] have shown perfect syllabification to improve g2p conversion for English, while automatically induced syllabification using the same method as for grapheme-to-phoneme conversion (Pronunciation by Analogy), did not yield better

[1]Typical syllable nuclei in Indo-Germanic languages are vowels and liquides.

results. So why did I hope for an improvement? Firstly, the introduction of constraints introduces linguistic knowledge to the system that is otherwise not available. Secondly, syllabification results reported in Schmid *et al.* [2005] attained near-to-perfect syllabification quality (over 99%). Syllabification is furthermore a process that is necessary for grapheme-to-phoneme conversion in any language, whereas morphological analysis is particularly important for German and some other languages with rich morphology.

6.2 Syllabification in German

A syllable in German has the general form C*VC*[2]. The consonant cluster at the beginning of a syllable is called the *onset*, it is followed by an obligatory *nucleus*, which usually consists of a vowel. The consonant cluster following the nucleus is called the syllable's *coda*.

When a string CCVCCCVC is seen, the segmentation into syllables may thus be ambiguous between CCV-CCCVC, CCVC-CCVC, CCVCC-CVC and CCVCCC-VC. Fortunately, this high ambiguity is alleviated by restrictions from phonotactics (the reader is referred to Kiraz and Möbius [1998] for a detailed analysis of German phonotactics). The constraints define e.g. the maximal length of a consonant cluster (3 consonant phonemes in the onset and 5 in the coda for German) and what types of consonants they can consist of (for example, if a three-consonant German onset begins with `/S% P/`, the third consonant must be either `/R/` or `/L/`). This observation is captured by the *sonority hierarchy*, which states that consonants in the beginning of the onset and end of the coda are less sonore than consonants in positions closer to the nucleus. Despite these restrictions, ambiguities arise frequently, in particular in morphologically complex words that involve compounding or derivations. The default is that the onset attracts as many consonants as phonotactically possible (this is also known as the *maximum-onset principle*). In German, morphological structure often overrides this default, such as in the compound *Parkende* (Engl. = end of parc), which has a morphological boundary between its two components *Park* and *Ende* and should therefore be syllabified as `Park-en-de`, in contrast to the present participle *parkende* which is syllabified as `par-ken-de`.

6.3 Previous Work on Syllabification

Previous approaches have tried a range of different approaches to the problem: Rule-based systems mainly relied on the maximum onset principle or the sonority hierarchy. An example is the finite state transducer by Kiraz and Möbius [1998]. Data-based approaches adapted Pronunciation by Analogy (PbA) to syllables (cf. Marchand and Damper [2005]), used forward-feeding connectionist networks with back-propagation (cf. Daelemans and van den Bosch [1992]), or a PCFG (cf. Müller [2001]). In recent work, Schmid *et al.* [2005] applied a Hidden Markov Model (HMM) for placing syllable boundaries and obtained excellent results. I therefore used their algorithm and modified it later, introducing another smoothing method (see below).

[2]C* = zero or more consonants, V = syllable nucleus

6.4 Syllabification as a Tagging Problem – Using a HMM

The basic idea in Schmid *et al.* [2005] is to look at the syllabification task from a tagging perspective. Each letter is to be annotated with one of two tags: `B` (syllable boundary follows) or `N` (no syllable boundary). The observed states are the phonemes or letters (depending on whether the algorithm is run on graphemic or phonemic input), and the hidden states are a pair $\langle l; s\rangle$ of the observed phoneme or letter l and the tag s (see figure 6.1).

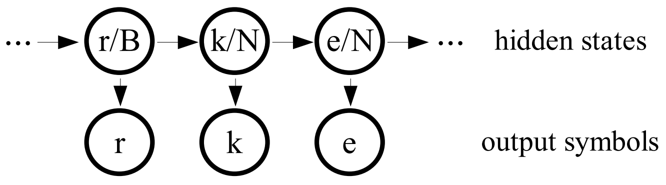

Figure 6.1: HMM for syllable tagging with states shown for a sequence from the word "parkende".

The syllable tagger uses the Viterbi algorithm (see chapter 2 or (Manning and Schütze [1999])) to efficiently compute the most probable sequence $\hat{s}_1^n$ of syllable boundaries $\hat{s}_1, \hat{s}_2, \hat{s}_3, ..., \hat{s}_n$, (that satisfies the constraints specified below) for a given letter sequence l_1^n. The Markov assumption states that a given letter-boundary pair only depends on the preceding k states (i.e. letter/phoneme–syllable-boundary pairs).

$$\hat{s}_1^n = \arg\max_{s_1^n} \prod_{i=1}^{n+1} P(\langle s; l\rangle_i \,|\, \langle s; l\rangle_{i-k}^{i-1})$$

Dummy letters '#' are appended to both ends of the word to indicate the word boundary and ensure that all conditional probabilities are well-defined.

Consider the example in figure 6.2: two hidden states are generated for each observed state, one for each of the possible tags[3]. Then the probabilities for each possible state sequence are calculated and the one with the highest probability is chosen. Thereby, the algorithm always has a limited size context window (k=1 in the example figure).

Two crucial aspects of the algorithm, smoothing and the introduction of constraints, will be discussed in the next two sections.

6.4.1 Constraints

There is one main constraint for syllabification: each syllable must contain exactly one nucleus. When running the decision tree or simple HMM where this constraint is not in-built, phonologically impossible syllables are generated, e.g. `/. T/` at the end of a word, probably because the sequences `/. T E#/`, `/. T E# S T/`, `/. T E# T/`, `/. T A#/` etc. were seen frequently. In rare words, syllables with more than two nuclei

[3]It would be mathematically more correct to also generate a state for all other letters of the alphabet and the two tags, but as such a state can never be seen in training, its probability unsmoothed probability would be zero any way, so such states can safely be pruned.

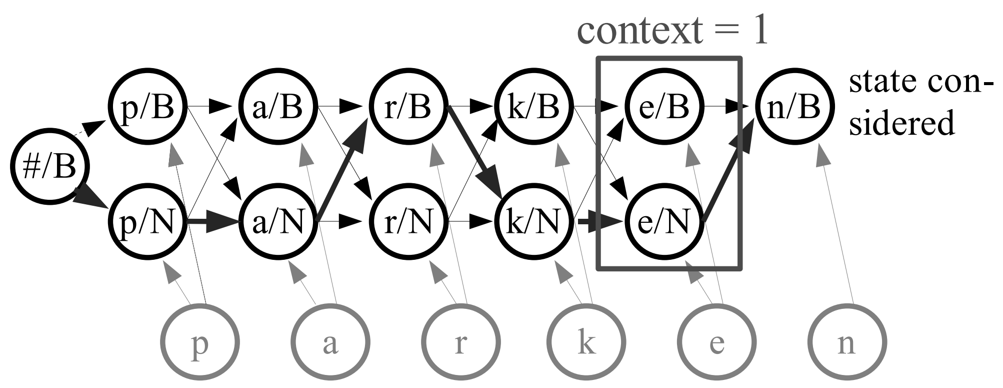

Figure 6.2: HMM for syllable tagging with states shown for a sequence from the word "parkende", the target sequence is indicated with bold arrows.

are an important source of errors. Example errors from the Hidden Markov Model for syllable boundary detection include: `schi-llers-c-h`, `spathal-ti-gen`.

For the exact constraint conditions, refer to Schmid *et al.* [2005], who first introduced them to the HMM. I was interested in quantifying the benefit from these constraints in function of the context window, to estimate whether it is worth introducing the constraint (see table 6.1). Mathematically, the probabilities in the HMM do not sum to one any more in the current implementation, as all placements of syllable boundaries that do not conform to the constraints are pruned off immediately.

WER	k=3	k=4	k=5
constraint	3.70%	3.10%	2.96%
no constraint	5.68%	3.48%	3.66%

Table 6.1: Syllabification on letters with and without the syllable constraints.

6.4.2 Smoothing

Smoothing is needed to avoid zero probabilities for sequences that were not seen in the training data. Some of the probability mass is taken off the probability values and redistributed to unseen events.

To approximate a probability of an unseen event well, a popular strategy is to back-off to a smaller context. A smaller context is of course relatively more probable. Therefore, the probability of the smaller context is combined with a backoff-factor that "punishes" the unavailability of the larger context.

An important aspect is the choice of what context to back-off to. The variant used in these experiments does not simply back-off to the next smaller n-gram but to phoneme-types, i.e. consonants or non-consonants. This is intuitively sensible in that consonants behave similarly (that is, they are either in the onset or the coda of a syllable, while vowels are in the syllable nucleus). At this point, some (easy to acquire) language-specific information is added to the algorithm.

As an alternative to backing off only if an n-gram has not been seen in training, linear interpolation can be performed. In linear interpolation, probabilities from all smaller n-grams combined. The relative contribution of a lower-order n-gram to the overall probability is determined by weights which sum to one. The choice of the smoothing strategy is important because it determines the ranking of alternative state sequences to one another and can thus strongly influence results. For an extensive and comprehensible study of different smoothing strategies see Chen [2003].

In the initial implementation of the syllabification HMM, a very simple smoothing method, which I will refer to as "Schmid-smoothing" was used. When I adapted the model to stress assignment (see next chapter), I found some problems that were due to Schmid-smoothing and substituted it by Modified Kneser-Ney smoothing (proposed in Chen [2003]). For an elaborated explication of reasons, see section 7.8. Kneser-Ney smoothing improves performance by 10%–20% for the syllabification task (see table 6.2 for the results).

From an analysis of the different types of mistakes made by the two algorithms, Kneser-Ney smoothing seems to capture morphological phenomena (rules such as "do not syllabify `1Wurf2e` as `1Wurf-2e` but `1Wur-f2e`" because `-2e` can only stand on its own if it follows a vowel or mute *h*). For a more detailed comparison of performance, also see section 6.5.

6.5 Experiments and Results

A range of experiments was conducted to estimate the gain to be drawn from alternative conditions:

- **different smoothing techniques**
  Schmid smoothing vs. Modified Kneser-Ney smoothing
- **type of syllable boundaries**
  orthographic vs. projected from target syllables
- **richness of input**
  letters vs. letters annotated with morphological boundaries, vs. phonemes
- **alternative test sets**
  random 90-10 split vs. test set design where stems from test and training sets are disjunct. (See chapter 4 for a discussion on training and test set design.)

The choice of the smoothing technique has a great impact on syllabification performance, and can improve results by 10% to 20% (see results in table 6.2). Typical errors that are made using Schmid smoothing but not with Kneser-Ney smoothing are:

```
ö r t - e r - t e n
g r ö - b l i c h - s t e s
5 u n - 1 s a n f t - 2 e - r e m
1 e i n - 1 t o p - f 1 e - s s 2 e n
/T S U: . K U N F T S T . R AE CH2 . T I . G A#/
/? A . P F L AU . T E# T/
```

These examples show that morphemes and general principles are captured better with Kneser-Ney smoothing.

WER	Schmid	Kneser-Ney
nomorph, orth	2.60%	2.13%
nomorph, proj	3.43%	3.10%
Celex, proj	2.17%	1.91%
ETI, proj	2.95%	2.63%
Phonemes	1.84%	1.53%
nomorph (90/10)	0.73%	0.72%
Celex (90/10)	0.53%	0.53%
Phonemes (90/10)	0.18%	0.18%

Table 6.2: Results for Syllabification HMM Schmid backoff vs. Kneser-Ney smoothing, context window length k=4.

There are two possible choices for what syllables to learn: orthographic syllables or syllables from the target phonemization. These can differ considerably, as the following examples illustrate:

- **approximants**
  Vowel sequences that are not graphemes (*ei*, *au*, *eu*) are often realized as approximants[4] if the first vowel is a front vowel. In that case, the two syllables constitute only one syllable together with the following vowel.
  `ü-ber-re-gi-o-na-le` (orth.) vs. `ü-ber-re-gio-na-le` (proj. phon.)
  `ja-gu-ar` (orth.) vs. `ja-guar` (proj. phon.)

- **double consonants**
  If there is no morpheme boundary in-between a sequence of two identical consonants, they have the effect of shortening the preceding vowel and are not pronounced as two consonants. Therefore, their projection onto the letter representation of the word is difficult.
  `ü-ber-ren-nest` (orth.) vs. `ü-ber-re-nnest` (proj. phon)

- **graphemes made of several letters**
  When projecting a syllable boundary from phonemes, there are some cases, e.g. words on *-ungen*, where the orthograhic syllable boundary is between *n* and *g*, but where the sequence *ng* is transcribed to a single phoneme `/N)/`. To project the syllable boundary back, one has to decide where to put it.
  `set-zet` (orth.) vs. `ü-ber-se-tzet` (proj. phon)
  `zu-hän-gend` (orth.) vs. `zu-hä-ngend` (proj. phon)

- **contestable phonemization**
  Finally, there are also some cases, where syllabification annotation in CELEX is at least debatable. `ü-ber-ge-ord-net` (orth.) vs. `ü-ber-ge-or-dnet` (proj. phon)
  `zweif-le-risch` (orth.) vs. `zwei-fle-risch` (proj. phon)

[4]German approximants are [w] and [j], which are transcribed as /U/ and /J/ in the phoneme set used.

The difference from projected syllabification to orthographic word separation may be learnt by a machine learning algorithm if the deviation is regular. “Real” problems only arise when the data is inconsistent, like the phonemization of CELEX. The inconsistencies presumably also lead to the considerably lower performance of the algorithm on projected syllable boundaries (see table 6.3).

WER	k=3	k=4	k=5
noMorph proj.	3.70%	3.10%	2.96%
noMorph orth.	2.56%	2.13%	2.34%

Table 6.3: Results for syllabification HMM for letters with different morphological annotations and phonemes.

The results from using information about morphological boundaries for syllabification correspond to my expectations: the algorithm performs significantly better on the morphologically annotated words than on raw letters (see table 6.4). I also observed that the algorithm performs always better on input in all lower-case than on input with upper-case letters, because case distinction leads to data sparsity and is not well-motivated. The cases in which syllabification based on ETI annotation is wrong are generally due to incorrect morphological annotation. Some of the syllabification errors on the data annotated with CELEX morphological information can also be explained by wrong or missing morphological annotation (e.g. `1fas-te-la-bend2s`), and errors due to the fact of working on letters instead of graphemes or phonemes (discussion see below).

WER	k=3	k=4	k=5
noMorph proj.	3.70%	3.10%	2.96%
ETI proj.	3.58%	2.63%	2.88%
Celex proj.	2.67%	1.91 %	2.08%
Phonemes	1.55%	1.53 %	1.57%

Table 6.4: Results for syllabification HMM for letters with different morphological annotations and phonemes.

As the results obtained by me did not correspond the good results reported in Schmid *et al.* [2005], I changed the training and test set design to replicate the original experiment. Instead of the training and test set design where stems are disjunct, I now used the randomly generated training and test sets and evaluated them using 10-fold cross evaluation. Word error rates (WER) were much lower when using the random test set than with the test set design where inflections or derivations from the same word did not occur both in the test and training set (table see 6.5). One interesting point is that the optimal window length was determined to be k=4 for the disjunct-stems-design, whereas a context window of k=5 performed better on the randomly split data. Although the results do not reach significance level, this might be a sign for over-training due to the high similarity of training and test data in the randomly split data: a window size of four is better for generalizing than a window length of five, which seems to generalize less well to new data.

WER	k=3	k=4	k=5
noMorph proj.	3.70%	3.10%	2.96%
noMorph proj. 90/10	0.72%	0.72%	0.53%
Celex proj., disj. stems	2.67%	1.91%	2.08%
Celex proj., 90/10	0.54%	0.53%	0.45%
Phonemes disj. stems	1.55%	1.53%	1.57%
Phonemes 90/10	0.43%	0.14%	0.11%

Table 6.5: Results for syllabification HMM for different training-test set designs.

6.6 Error Analysis

An interesting error that occurs relatively frequently, in particular with small context windows is `wich-est` instead of `wi-chest` or `Deich-e` instead of `Dei-che`. While I first wondered about the lacking ability for generalization (syllable boundaries before schwa can only occur when the previous syllable does not have a coda), it became then clear that this is an artefact due to working on letters instead of graphemes: the sequence `h-e` is highly frequent in words with a mute *h*. So in fact, the algorithm has seen in training that `he` is actually bad syllabification. Results could therefore be improved by substituting graphemes like *ch* or *sch* by different symbols. But of course in some cases, disambiguation might be needed to do this - not every *sch* is actually pronounced `/S%/`. In some cases, e.g. in *Röschen* (Engl: little rose), *sch* has to be transcribed `/S . CH2/`. Experiments showed that the context window may be shrunk if graphemes are substituted by other characters.

Further remaining errors in the phoneme transcription can be explained by inconsistencies in the training data, e.g. `/F AE R# . D I N) . E# S T/` has the boundary after `/N)/` instead of before `/N)/`, because the two only occurrences of `/D I N) E# S T/` in the training data were annotated as `/? A P . D I N) . E# S T/` and `/F O: . R AU S . B E# . D I N) . E# S T/`; a more frequently occurring inconsistency is e.g. for `zer-stö-re-risch-ste`, 1,450 words on *-ischste* in the training data have the boundary after *schs*, while only 35 insert it after *sch*. Some of the instances in the evaluation set are also wrong or at least questionable, such as `/A K . N E# N/`.

Another important source of errors both on graphemes and phonemes is the ambiguity between *st* and *schst* in the end of words. For example, we want `/G R OE: S . T E#/` but `/S% OE: N . S T E/`, because the *s* belongs to the stem in *größte*, while it is a pure suffix in *schönste*. Consider also `lau-ste` (Engl: tepidest) vs. `laus-te` (Engl: deloused). The algorithm would not have to simply count what is the most probable sequence (*-ste* or *s-te*) but also take into account whether there is a stem ending in *s*.

Finally, there are also mistakes due to missing morphological data or wrong phonemization, such as `/? EI N . T O . P F AE . S E# N/` (Eintopfessen). Syllables beginning in a vowel are very rare, therefore the analysis follows the default phonotactic rules of German when it inserts the syllable boundary before `/P F/`. Furthermore, there should be a glottal stop before `/AE/`, which would have rendered the syllabification unambiguous. Many other analyses that seem extremely undesired at the first view, can be explained by the training data: `ko-tzlang-wei-lig` (Engl: puke-boring) may be explained by the missing morphological boundary, but even then, *tzl* is not a good syllable onset. The problem is that `otz-l` has never been seen in the training data, while `o-tzl` has been seen once – in `bro-tzle` which may also be considered as a syllabification error.

6.7 Performance on g2p task and Discussion

The central question in syllabification is whether automatically acquired syllabification does improve grapheme to phoneme conversion, or whether sole use of such a component is to provide required information for automatic stress assignment. Marchand and Damper [2005] found that grapheme-to-phoneme conversion performance was significantly improved if words were annotated with perfect syllable boundaries, but observed no performance gains when syllable boundaries were acquired from an automatic process. I replicated their experiments with my data and modules and found automatic syllabification to lead to a small but significant improvement (25.6% instead of 26.6% WER; p=0.015 on a two-tailed t-test). The effectiveness of the automatic syllabification module can be explained by two factors: Firstly, I used a method for syllabification that worked differently from the g2p conversion algorithm and therefore had access to a different kind of information, whereas Marchand and Damper used the same method for both g2p conversion and syllable boundary assignment. Secondly, the integration of phonological constraints adds additional linguistic information to the module that is not available to the decision tree.

When these two factors are not valid, i.e. when g2p is performed with the HMM and if the syllable structure constraint is implemented into this HMM, separate automatic syllabification does not lead to an improvement.

6.7.1 Preprocessing vs. Postprocessing

Syllabification works better on phonemes than on letters, because phonemes are less ambiguous. Consider for example the word *Drecksturm* (Engl: either “dirt tower” or “storm of dirt”) or the well-known example of *Staubecken* which can either mean “reservoir” or “dust corners”. The correct phonemic transcription (even without syllable boundaries) resolves these ambiguities: the sequences `/S T/` and `/B ?/` in the `Drecks+turm` and `Staub+ecken` interpretation are not allowed as an onset in standard German. On the other hand, the g2p conversion process needs the information that there is a morphemic and therefore syllabic boundary here in order to generate the correct phonemes. The question is whether this information has to be provided explicitly by annotating a morphemic or syllabic boundary, or whether the g2p component can achieve the correct disambiguation using statistical evidence (having seen that *Dreck* is quite probable to be appended an *s* if it is followed by further letters (i.e. if it is a compound), or that *turm* has been seen more frequently than *sturm*.

Evidence from experiments shows that higher overall performance is achieved if syllabification and stress assignment are done in a postprocessing step rather than in preprocessing (see table 8.2).

6.7.2 Orthographic vs. Projected Syllable Boundaries

In table 6.3 we saw that learning orthographic syllable boundaries is a less difficult problem than learning syllable boundaries that were projected from the phoneme output. The reason is probably that orthographic syllabification annotation is of higher quality and less inconsistent. When integrated into the pipeline system, the g2p component has to compensate for the difference between orthographic and projected syllable boundaries, which leads to correspondingly worse performance at this later step. The word error rate is 18.6% if orthographic syllables are learnt as opposed to 16.6% when syllabification is learnt on the projected syllables.

Chapter 7

Word Stress

This chapter addresses German word stress assignment. I firstly motivate, why a special treatment for stress assignment is needed and explain how German word stress works. Problems with low quality training data are then addressed and alternative designs for a stress module are discussed.

I implemented three different approaches for stress assignment: a rule-based system, a probabilistic baseline system and an adaptation of the HMM also used for stress assignment. The advantages and disadvantages of the models are analysed. Significant data-sparsity problems led to the discovery of problems with the smoothing method used and its substitution by another method, which subsequently also led to better results for the syllabification and g2p-HMMs. In the last section of this chapter, I summarize my findings and draw conclusions.

7.1 Why a separate module for Stress Assignment?

Using the ETI morphological analysis and the AddWordTree for grapheme-to-phoneme conversion, 21% of the words in the list are transcribed incorrectly (at least one pronunciation or stress error). Within these errors, there are particularly many word stress errors. Stress errors account for two thirds of the mistakes (i.e. 14.5% of the words are stressed incorrectly when using the ETI-morphology in preprocessing, 16.8% with CELEX morphological annotation, and even 17.3% are assigned incorrect stress without morphological preprocessing). There are three roughly equally big categories within the stress errors.

1. words with more than one main stress ($\approx$ 5.3%)
   Most words in this category are very long and complex (often compounds or prefixed words).

2. words with no stress ($\approx$ 4%)
   This category typically contains some rare words, and words with few affixes.

3. stress on the wrong syllable ($\approx$ 5.2%)
   There are relatively many non-native words in this category, but not exclusively. Word stress is a hard problem in German. So far, no rules have been found which can perfectly predict word stress.

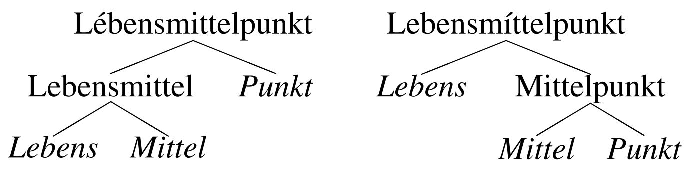

Figure 7.1: Two possible analyses for the homograph “Lebensmittelpunkt”.

7.2 German Word Stress

Many linguists have now agreed on the hypothesis that syllable weight and syllable position within the word are the most relevant parameters in German word stress. (Although the influence of syllable weight is not uncontroversial among experts. For an overview of the discussion in the field, see Jessen [1998].)

7.2.1 Primary Word Stress

There are several levels on which word stress is determined:

1. compounds
   German compounds are usually stressed on the left element of the compound, with very few exceptions (e.g. *Jahrzéhnt*, *Lebewóhl*), which can be looked up from a special list. In compounds with three elements, stress position is ambiguous and arguably depends on the structure of the compound as well as on its lexicalization status. Firstly, consider the minimal pair *Lebensmíttelpunkt* (Engl: center of life) and *Lébensmittelpunkt* (Engl: food badge) (alternative analyses are shown in Figure 1). When the first two components belong together more closely, the first compound element is stressed. Otherwise, stress may be on the second element, but for left-branching structure, the definition of rules is more complex: Stress has been proposed to fall on the middle element if the last two words are freely composed (as in [Bundes+[kriminál+amt]]) but on the first element if the last two elements constitute a lexicalized unit (as in [Háupt+[bahn+hof]]).

   In CELEX however, all compounds are stressed on the first syllable, with the very few exceptions of *Rotkreuzmannschaft*, *Nordsüdgefälle* and *Ostwestroute*. This is on the one hand due to wrong stress (or at least questionable) assignments (*Állhéilmittel*, *Últrakurzwellenempfänger*, *Ángestelltenversicherungsgesetz*), on the other hand, there are relatively few real (non-lexicalized) compounds in CELEX.

2. affixes
   The next level for determining word stress are affixes.

   (a) prefixes

   - There are some prefixes which can never be stressed, such as *ver-*, *er-*, *ent-*

- and some which are always stressed, such as *ein-*, *auf-*
- there is also a group of ambiguous prefixes whose stressing cannot be explained, e.g. *un-*
- and finally there are those prefixes whose stressing depends on whether they are separable or non-separable verb prefixes, e.g. *um-*, *über-*, *unter-*, *voll-*

When there are several prefixes that belong to the group of usually stressed affixes, overrules the other. Therefore, it would be good to extract a ranked (according to frequency and stressing probability) list for stressing probabilities from the training data.

(b) suffixes

- inflectional suffixes
  Inflectional suffixes are never stressed but can cause stress shift in words with certain endings (*Musík* but *Músiker* or *Áutor* but *Autóren*). Suffixes that leave the stress pattern of the stem untouched are classified as class-II suffixes, whereas suffixes that are stressed themselves or that have an impact on the stem they modify, are classified as class-I suffixes (for a full list of inflectional and derivational suffixes, see Jessen [1998]).
- derivational suffixes
  Some derivational suffixes are always stressed, some are never stressed, very few ambiguous with respect to their stress status. As for inflectional suffixes, some seem to influence the stress position of their word: *Amérika* vs. *amerikánisch*, *Amerikáner*.

3. stems

German word stress has been studied for a long time, first publications on syllable weight and stress go back to the Sixties. A recent extensive study which claims to be able to explain 95% of German word stress for non-prefixed and non-compounded words (see Wagner [2003, 2001]). Wagner reviewed and refined a set of rules introduced in Jessen [1998]. These rules are based on the account of syllable weight and syllable position after splitting off "unstressable" suffixes, while keeping some other affixes which have the power to attract stress. Wagner's rules (based on Jessen):

(a) stress the last syllable if it is heavy

(b) otherwise stress the second last syllable if it is not a schwa-syllable

(c) otherwise go from right to left and stress the next non-schwa syllable

The second last ("penultima") syllable is stressed most often. A baseline assignment that always stresses the second last syllable in a stem achieves a success rate of 63%.

There remain some exceptions that cannot be solved without recurrence to etymological information. For example, *Arbeit* was originally a compound, but is not perceived as a compound any more in contemporary German. Exceptions also include borrowings from other languages in particular from French.

In some rare cases (e.g. *Télefon* vs. *Telefónanruf*, *Telefónanschluss*), stress is shifted within the same stem due to compounding.

Since syllable form and weight has an influence on stress assignment in German, it seems sensible to try to learn syllable classes, i.e. group syllables according to their weight (possibly in function of the position). One option is to use linguistic information to do this. But generalization to other languages in which syllable structure also plays a role for stress assignment would be easier if syllable weight could be derived automatically. The notion of syllable weight could for example be approximated from a syllable's probability of being stressed. This would also enable us to capture syllables with a special stress status due the fact that they represent a certain morpheme.

Information about the morphological structure, syllabification and phonemes (syllable weight is only defined for phonemes, because letters are ambiguous) is necessary to apply the rules.

7.2.2 Secondary Word Stress and Stress Clash

The discussion so far focussed on primary word stress in German. Complex words, which tend to be quite long in German, often also have secondary stress, especially in compounds or prefixed words.

Another problem arises from the fact that there is some evidence that German is a stress-shifting language. This means that two adjacent syllables must not both be stressed (e.g. *Fách* + *Hóchschule* = *Fáchhochschúle*). This phenomenon can even be observed to occur across word boundaries: den *Róck* + *ánziehen* = *den Róck anzíehen*. Wagner and Fischenbeck [2002] conducted experiments on this phenomenon but reported that stress shifting is in fact a very irregular pattern in German, which makes it very difficult to automatically assign stress correctly.

In this work, only primary word stress is considered. Stress clash across words boundaries does not occur in the actual problem setting, because we only work with word lists. Any stress assignment that would be sensitive to its context cannot be evaluated with the data available to me any way.

7.3 Working with Low Quality Training / Test Data

As mentioned above, the word stress assignments which were learnt from CELEX using the AddWordTree were subject to many errors. There are two reasons for this. The first one was discussed above, and is caused by the fact that the learning mechanisms do not have access to all of the information they would need to solve the problem. But a second problem must also not be underestimated. The CELEX stress data is not very reliable, in many cases it is wrong and (perhaps even worse) inconsistent. For example, the suffix *-tion* should always be stressed in German unless it is followed by another affix that attracts stress more strongly, like *-al* or unless it is part of a compound. But in CELEX, only half of the words on *-tion* are stressed on this syllable. When I manually inspected the first 40 cases, there were only two correct ones among them.

Such inconsistencies have a very negative effect on the data based algorithms, which consequently learn wrong or no generalizations. Furthermore, the data is only partially valuable for evaluation – any system will get punished for not committing the mistakes which are in the "gold-standard" data. Therefore, it is difficult to evaluate the rule-based system using this data. For the data-based algorithms, the case is perhaps less dramatic because (at least if errors are consistent) we can assess how well the algorithm learns from what it is given.

7.4 Approaches considered

- Rule-based Stress Assignment
  Caseiro *et al.* [2002] obtained best results on a g2p conversion task for Portuguese when they introduced a separate preprocessing step for determining word stress. However, the stress patterns were not learnt automatically, but instead a set of manually constructed rules was compiled into a finite state transducer. An important argument for using a rule-based approach is the low quality of word stress in our resources.

  It would be possible to compile Wagner's rules into such an FST. But one problem is that Wagner's description of weight is based on phonemes and not on letters. It is therefore necessary to determine the weight status of the syllables in phonemes and then, using the letter-phoneme alignment, project these categories onto the letters and generalize the rules found in order to assign weights to letter-syllable.

  However, this mapping will cause ambiguity. Therefore, it has to be tested whether a stress assignment step should better be done in preprocessing or post-processing.

- Separate Decision Tree
  Why would one want another decision tree, although the Addwordtree has already proven to return unsatisfactory results? What can be hoped to be gained from another, separate decision tree? The decision tree could be better adapted to the task. It could consider more context, and could ask different questions.

  The most important negative aspect is that again, it cannot be guaranteed that exactly one stress is produced for each word. Furthermore, Dege [2003] already tried to do stress assignment in a separate decision tree and obtained even higher word error rates than for the integrated solution.

- Pronunciation by Analogy
  Similarly, PbA can neither guarantee that there is exactly one stress in a new unseen word. The weighting algorithm which chooses the best path through the lattice, could however be modified to allow for exactly one stress, but such a method would not be able to model stress attraction. Marchand and Damper attempted to do letter-to-stress with PbA but found that PbA is not well suited for this task: they obtained only 54% of words correctly stressed for English.

- HMM
  In analogy to the syllabification problem, stress assignment can also be regarded as a tagging problem. The necessary constraints can be defined similarly (in syllable boundary assignment, a syllable must have exactly one nucleus; in stress assignment each word must have exactly one main stress.) Issues in designing the HMM are what entities to choose (single letters or syllables), how big a context is needed and what information to include as features (stress status, native-foreign info, morphological information, position in the word). Furthermore, it is necessary to define a smoothing strategy. I have eventually chosen this approach – see below for more detailed design description.

7.5 Multi-level Process

Because of the different levels that need to be considered to assign German word stress theoretically (compounding, affixation, stress within stems), using a multi-level-architecture based on the morphological structure of a word seems to do justice to the problem. On the first level, which only applies to compounds, the part of the compound which should be stressed needs to be decided on. In a next step, words with affixes which are always stressed can be filtered out (as the affix stress overrules the stressing from stems or suffixes), and only in a final step does an algorithm apply that would determine the stress position on the word or compound element.

The advantage of such an architecture is that it reflects the nature of the problem well. Also, this design prevents some basic errors that typically occur in the decision tree or the HMM (see chapter 8), which do not have access to this kind of level-dependent rules. Drawbacks are that this approach only works with good morphological annotation. Otherwise it is impossible to apply the different steps. Furthermore, this architecture is specific to German and related languages such as Dutch, but can not be transferred to languages with different stress assignment processes. To build something equivalent for another language, a linguist would be required to determine the other language's stress determining factors. Finally, the problem with the introduction of several steps is that they will probably only work better, if their input is flawless. In real life, the danger of propagating errors is high.

7.6 Rule-based Stress Assignment

Because the training data available to me is very unreliable and a data-based algorithm can only ever be as good as its training data, I implemented Wagner's rules to see how well they scale to large amounts of naturally occurring text. Although the rules seemed sufficiently simple on the first view, they proved labour-intensive to implement.

Wagner's rules only apply to word stems, or word stems with class-I suffixes (i.e. suffixes that are stressed themselves or lead to stress shift when they are attached to a word). Therefore, the multi-level-process had to be applied. In my implementation, compounds are first segmented into their elements and only the first element is processed further, (I here exploited the fact that CELEX almost exclusively only contains compounds that are stressed on the first element). Then, the rest of the word is segmented into its prefixes and suffixes, and stress is assigned appropriately. Only those word stems that did not concatenate with stressed affixes are processed with Wagner's rules (for the set of rules, see above or refer to Wagner [2001]).

A first problem with the rule based system is that morphological information is crucially necessary. This model simply does not work if no morphological information is available. In the case of un-recognized compounds or prefixes, the rule system assign a wrong stress in almost 100% of the cases: the rules try to assign stress to the second last or word-final syllable (processing the word from right to left), while some prefixed words and most compounds should be stressed on the first element of the word.

Syllable weight rules are defined on phonemes, because the graphemic presentation is much more ambiguous. At the same time, the rules concerning affixes (to distinguish class-I and class-II affixes) are defined on the orthographic letter-strings. Therefore, I implemented a the rule-based approach on a combined representation of phonemes and letters. Such a component can therefore not be used in preprocessing but only for postprocessing after the g2p conversion step.

In the first version, which did not take into account morphological information, the correctness rate was as low as 52.41% on a corpus of approx. 239,700 German words from CELEX (the training corpus). When I include morphological information, correct assigment of stress rises to 84.11%, which is much better but still does not even reach the success rate of the one-step integrated Addword Tree or HMM. There is some room for improvement though, as can be seen from the below list of error categories:

- **ambiguous prefixes**
  20% of the errors are on prefixes that are ambiguous with respect to their stress status. In future work, information e.g. from SMOR should be used to disambiguate these cases (time does not allow me to implement this in the scope of this thesis).

- **words not covered by the rules**
  This category make up approximately another 25% of the errors. The words it inclused are often foreign words, mainly from Latin and French, but also some native German words: *Arbeit*, *Ablativ*, *Abenteuer*, *Abakus*, *Allee*

- **missing morphological information**
  Morphological annotation, even the one from CELEX, is not perfect and accounts for about another 25% of the mistakes. There are some words that lack compound annotation, such as *Aberglaube*, *Aberwitz*, *Achillesferse* and words where prefixes are not marked, e.g. *Abgeschmacktheit*, *Abwesenheit*

- **words on *-ien***
  There are three types of words that end on *-ien* and that are stressed differently. The first group includes plurals of words that end in long stressed `/I:/`, like *Harmonien*, *Fantasie*. The plural of these words should also be stressed on `/I:/`. In contrast, the plurals *Sanatorien*, *Materialien* looks the same and is also transcribed `/I: . E# N/`, but should not be stressed on `/I:/` but the previous syllable. If we had access to the singular forms of these words[1], we could decide

[1]Singular of *Sanatorien* is *Sanatorium*. The singular of *Materialien* is *Material*.

the matter more easily, but there are also words like *Linie* or *Akazie* which look like the words from the *Harmonie*-class but whose plural forms are pronounced similar to those from the second class.

- **words that are wrong in CELEX**
  This concerns mostly words on *-tion*, *-ell*, *-ieren*, but includes also words like *abtáuschen*, *abtranspórtieren* and of course all words that are annotated with more than one or no stress. Altogether, these cases make up about 15% of the mistakes (this number was estimated from looking manually through 200 errors).

- **composita that are not stressed on the first syllable**
  As explained above, the stressing of compounds with three elements depends on the compound's structure and lexicalization status. In the version of the rule-based system which was implemented here, I do not have access to such information. However, it could be retrieved from large text sources by estimating the probability of compound elements of occurring together and constituting a lexicalized unit. For this purpose, e.g. for the word `Haupt+bahn+hof`, one could count the occurrence frequencies of *Hauptbahn* and *Bahnhof*, and also count with how many different other words they are compounded. We would expect to find *Bahnhof* much more frequently than *Hauptbahn* and to see loads of words that compound with *Bahnhof*, e.g. *Nordbahnhof*, *Güterbahnhof* etc., but few words like *Hauptbahnstrecke*.

I conclude from this experience that the rule-based method is not appropriate for the task at hand – it cannot cope with data that is not morphologically annotated and degrades quickly if the morphological annotation is noisy. Furthermore, a large exception list would be needed to handle all those cases which cannot be covered by the rule system. Moreover, such a rule-based system does not generalize well to other languages.

7.7 Baseline Approaches to Learning Stress for Stems

The first possibility to assign stress to stems is to simply do a lexicon lookup. The lookup method is the best quality method for known stems. The only source of mistakes made would be homographs. But, as no patterns are learnt, it is not possible to assign stress to unseen stems.

Secondly, one could store for each syllable its probability of being stressed. For a new word, stress could then always be assigned to the syllable with highest stressing probability. Doing this, we disregard the importance of syllable position, and as can be expected, one performs less well on the training set. If we only store the syllable form and not the syllable position, only 85% of the training set are predicted correctly. These probabilities can of course be combined with a syllable position model, but in general, stress probability can only be assigned to words that are made of syllables that were already seen in training, new syllables need to be assigned some default value.

The third approach is to learn syllable classes, i.e. to group syllables according to their probability of being stressed and to generalize the patterns that are common to these syllables. We then again expect to lose some contextual information, i.e. perform worse on the training set. But we can now cope with any unseen word, which may even be composed of syllables not contained in the training set. As syllables are also subject to the LNRE[2] problem, it seems sensible to try such an approach. The goal is to learn the classification rules for different syllable weights automatically without using any linguistic knowledge. If we succeed in doing this, the approach can promisingly be applied to other languages as well.

In the rest of this subsection, two knowledge-lean (syllable weights are not known in advance) and quickly-running approaches are described. The motivation for implementing these approaches are twofold. On the one hand, I wanted to provide a baseline algorithm to compare against. The second goal is to learn syllable classes that are meaningful for the stress assignment task.

Assumptions in this approach are that syllable structure and syllable position are relevant for stress assignment. As discussed above, the weight of a syllable (which contributes to its probability of being stressed) depends on the vowel it contains, and on the structure of its onset and coda. Ideally, these should be derived from the phonemes, and not the letters, as one grapheme is much more ambiguous than a phoneme (*e* can be the heavy nucleus /`E:`/, the light nucleus /`AE`/ or the super-light nucleus schwa /`E#`/).

The stressing probability of an entity is the frequency of seeing this entity in a stressed syllable divided by its overall frequency. The stressing probability of a syllable is thereby composed of the stressing probabilitis of its onset, nucleus and coda. Composing the syllable stress probability from these units allows to generalize to unseen syllables. As a result, all syllables are assigned gradual values.

To estimate these probabilities well, it is necessary to learn from "clean" words, that is from syllables that are not stressed for being a certain affix, but only from syllables that are part of stressed stems. Furthermore, only words from the subset of CELEX and BOMP with the same transcription were used to ensure good quality.

Once the stress probabilities have been calculated for all syllables in the training set, new words from the test set were evaluated. If the heaviest syllable is chosen for stressing without taking into account syllable position, 72% of words are stressed correctly. The position-only baseline for stress assignment yields 63% of stems correct. Introducing a simple heuristics for combining the information from syllable position and syllable weight, a precision of over 82% on the test set can easily be reached.

This approach does not scale well to a situation where morphological analysis is not available, because syllable stress probabilities would be biased by e.g. heavy affixes that are always unstressed, such as *-keit* or *-schaft*.

[2]large number of rare events: The number of extremely rare events is extremely large in many NLP problems. Seeing a previously unseen word or syllable occurs very often, although the probability of a particular instance of such a rare event is very low. Khmaladze [1987] found that the law of large numbers does not hold for such lnre distributions.

7.8 HMM for Stress Assignment

The second serious approach to word stress assignment was the implementation of a Hidden Markov Model (HMM), similar to the HMM used for syllabification. The HMM differs from the HMM for syllabification and letter-to-phoneme conversion, (see chapter 8), in that its hidden states are syllables-stress pairs instead of letter–syllable-boundary pairs.

7.8.1 Definition

The hidden states are pairs of syllables and stress markers, the observed states are syllables. The Hidden Markov Model returns the most likely syllable-stress sequence $\hat{str}_1^n = \hat{str}_1, \hat{str}_2, \hat{str}_3, ..., \hat{str}_n$, (that satisfies the constraint that each word contain exactly one stress) for a given syllable sequence syl_1^n. Thereby, the Markov assumption is applied which states that a given syllable-stress pair only depends on the preceding k states (i.e. syllable–stress pairs).

$$\hat{str}_1^n = \arg\max_{str_1^n} \prod_{i=1}^{n+1} P(\langle str; syl\rangle_i \,|\, \langle str; syl\rangle_{i-k}^{i-1})$$

To each word, dummy syllables '#' are appended at both ends to indicate the word boundary and ensure that all conditional probabilities are well-defined.

Syllables were chosen as a basic entity to allow the HMM to acquire a notion of syllable weight, with the idea that the model can back-off from unseen syllables using the results from the syllable-class learning experiment for learning stress probabilities of syllables. Using syllables as basic units has advantages and disadvantages. The advantage is that the model can look at a larger context. In addition, every unit that may be tagged with the stress-tag is meaningful in that it contains a syllable nucleus (stress can only be realized on syllable nuclei). A disadvantage is that it leads to a strong data sparsity issue: The two following words are inflectional variants of another but look very different to the algorithm: 1) `1un-ter-1pri-vi-le-#gier-2t2es-tes`[3] 2) `#1un-ter-1pri-vi-le-gier2t2st` (`gier-2t2es-tes` and `gier2t2st` are not related for the algorithm. If they were, they should be assigned the same stress status.). The algorithm does not learn to generalize from the training data that it should stress all endings on *-ier* (unless the word is prefixed / part of a compound etc.) irrespective of the preceding stem (that determines the syllable onset) and of the inflectional suffixes.

The HMM does not have any explicit attribute for the syllable position but contains this information implicitly through its context window. The advantage of the HMM is that it can be generalized to other languages easily.

I ran the HMM with several different context window sizes from $k = 1$ to $k = 3$. Choosing $k = 1$ makes the HMM most attractive: performance cannot be significantly improved by choosing a larger window size. Furthermore, a context length of 1 in training is the only constellation in which the training file takes less space than the full training data (5.9 MB for Phonemes). The file sizes and performance in function

[3] '#' is used to mark stress in letter-strings while I use the '.1' notation to mark stress in phonemic representations

of the window length choice is shown in table 7.1, note that the sizes refer to the uncompressed text files.

context window	training file size	Word Error Rate
k=1	2.3 MB	8.8%
k=2	6.9 MB	8.7%
k=3	11.9 MB	9.4%

Table 7.1: Training file sizes and Word Error Rate in function of the context window for the HMM on Phoneme Syllables.

7.8.2 Constraints

One important problem with the HMM and decision tree for integrating grapheme-to-phoneme conversion, syllabification and stress assignment in one single step was that many words were assigned no stress or more than one word stress. The idea in modularizing the task is to introduce constraints to counter these obvious mistakes. Therefore, a constraint that only accepts words with exactly one stress was integrated into the model. As can be seen from the word error rates below shown in table 7.2, this constraint improves results significantly, and reduces word error rate by over 60%.

To integrate the stress constraint into the model, every sequence is stored with a flag indicating whether it has already been assigned a stress or not. Any state that would introduce a second stress is pruned directly for efficiency reasons. This pruning causes the probabilities in the model not to sum to 1 any more, but only eliminates analyses that we do not want any way – the ranking with respect to the other analyses does not change.

Without the constraint, 14.5% of the words are not stressed at all and 13.7% of words have more than one stress, the remaining 3.6% of errors are due to wrong stress position.

context window	WER with constraint	WER without constraint
k=1	10.3%	31.9%
k=2	10.3%	30.2%
k=3	11.1%	29.9%

Table 7.2: Effect of integrating a constraint to the stress HMM.

7.8.3 Training data

The algorithm was furthermore run on several different data sets. First, of course, on the full training data, not using any knowledge about how compounds are stressed or which affixes are stressed. In a second run, I used linguistic information about German through introducing the restriction that all compounds must be stressed on the first compound element. But, surprisingly, this did not lead to an improvement, which is probably due to incorrect morphological annotation (some prefixes were annotated as stems). I did not go further into correcting morphological annotation because compounding is much less of an issue for the Hidden Markov Model than for the rule-based approach: Many compound elements seem to occur in a typical position in the compound. For example, *Mann* occurs often as the last element of a compound, which is perhaps partially an artefact of the compounding stems in German – if it was to occur as a first element, it would be either have the form *Manns* or *Männer*. Therefore, the algorithm learns to rather stress the syllable *Manns* or *Mä* followed by *nner* than the syllable *Mann*.

To estimate the proportion of errors due to inconsistent stress annotation, the data was run on a smaller set of high-quality data, which was obtained from retaining only those words in the training set that had the same stress annotation in BOMP (for more information on BOMP, see chapter 3) and CELEX. The results are difficult to interpret, as there are two factors that influence performance: the amount of training data and the quality of training data. While quality was increased when using the CELEX-BOMP intersection set, size was reduced to roughly one quarter (61,000 words instead of 239,000 words), which lead to data sparsity problems. From the results in table 7.3, we can see that the negative effects of the smaller training set weighted heavier than an improvement in quality. An important lesson to be learnt from this experiment is that performance drops dramatically with a decrease of the amount of training data, at least in models where data sparsity is an important issue.

training set	size	test on	
		full CELEX	CELEX-BOMP
CELEX training	239,000 wds	8.8%	-
CELEX-BOMP overlap	91,000 wds	17.8%	-
CELEX-BOMP same-stress	61,000 wds	19.3%	13.6%

Table 7.3: Training file sizes and Word Error Rate in function of the context window for the HMM on Phoneme Syllables.

Like the syllabification HMM, this algorithm was also run on different types of input data: phoneme-syllables, letter-syllables and morphologically annotated letter-syllables. The model worked best on phoneme-syllables, reaching a word error rate of only 8,7% for k=2. At first, I was surprized to find that morphological annotation from CELEX did not improve performance over using no morphological annotation, although overall conversion quality was best with CELEX annotation (remember results in table 5.9). A possible explication for this observation is that while the number of conversion errors on phonemes decreases, the number of conversion errors in stressing increased. This negative effect of morphological annotation is on the one hand probably due to increased data sparsity: syllables that are seen rarely are seen even more rarely with one particular morphological annotation, in particular if the morphological annotation is not perfectly consistent.

The results in table 7.4 show performance of the algorithm on the supposedly perfect data, i.e. projected syllable boundaries for the letter syllables, CELEX morphological annotation and CELEX phonemization. For an account of how well the module works when integrated in the TTS-architecture, and how it degrades with noisy input, see section 7.8.6 below.

training set	p=1	p=2	reverse (p=1)	reverse (p=2)
Phonemes	8.8%	8.7%	8.6%	9.1%
Letters	9.9%	9.5%	9.7%	9.9%
Letters + CELEX-morph	10.3%	10.3%	9.9%	9.9%

Table 7.4: Word Error Rates in function of the input data and processing direction.

In another experiment, I let the HMM run backwards through the words, because the linguistically motivated rule-based system also assigns stress to syllables beginning from the end of the word. So I hoped that the Hidden Markov Model might also profit from this order. But the results are difficult to interpret - while an improvement was obtained for the k=1 context window, this success was not consistent for the higher-order HMMs. Also, the difference is small and only weakly statistically significant ($p \approx 0.1$ using a two-tailed binomial t-test).

7.8.4 Smoothing

Smoothing is always necessary in statistical models to prevent the model from assigning zero probabilities to unseen events. For the stress-assignment HMM, smoothing is even a more important issue than for the letter-based syllabification algorithm: While it is very improbable to see a new letter–stress-tag pair in the input word, encountering a previously unseen syllable–stress-tag pair is quite a frequent event even with a training set as large as the one used here (239,700 words). Even more often, we will not have seen a certain syllable sequence. In that case, we try to estimate its probability from a smaller n-gram, just as done in the Hidden Markov Model for syllabification. To alleviate the data sparsity problem, I substituted all upper case letters by the corresponding lower case letters.

7.8.4.1 Back-off steps

I tried out several strategies to back off to smaller contexts and evaluated their influence on the results.

- **backing off directly to the next smaller n-gram**
  The simplest strategy to estimate the probability of an n-gram based on the n-1-gram gave me the best results.

  A word error rate of 8.6% on phoneme syllables for k=1 was reached, which is significantly better than the WERs from the rule-based system (15.8% WER) or the integrated decision tree step (14.5% WER with ETI-morphology, 16.8% with CELEX morphological annotation and 17.3% WER without morphological annotation).

  In another experiment, all syllables that occurred just once were mapped onto a default “unknown” syllable. Syllable stress probability of new syllables was then estimated from this value. But (probably because of more serious data sparsity problems from flattening out all syllables that occurred once), this approach increased the word error rate by almost 20% (relative increase).

- **backing off to the syllable structure**
  Instead of estimating an unseen syllable’s probability from the next smaller context alone, I first backed off to the syllable structure of the outermost syllable in the context window. The structures I backed off to were the ones used in the rule-based algorithm that are relevant for syllable weight in German. Unfortunately, results did not differ significantly from results that did not include this step.

- **backing off to the syllable’s stress status**
  In a last approach, I tried another stepwise backoff-approach. If a syllable sequence had not been observed, it was generalized to an unknown syllable with the same stress status. E.g. for estimating the probability of stressing the syllable *der* in *Kalender* (_’ stands for “no stress”, ‘#’ stands for “stress”, and ‘-’ for a syllable boundary):

  _ Ka-#len-_ der
  _ *any*-#len-_ der
  #len-_ der
  #*any*-_ der
  _ der
  _*any*

  Although this worked a bit better than backing off to the syllable structure, it was clearly outperformed by the simplest backoff step.

Further possibilites, which I found intuitively less promising, and therefore did not spend time on, are 1) to backoff to the syllables morphemic type (part of a stem, part of a suffix etc.) but this is less well-defined as syllable boundaries do not always coincide with morpheme boundaries, 2) to back off to the stressing probability as acquired from the naive approach outlined in the previous section and 3) to back off to the syllables position (e.g. counted from the back and the beginning in order to abstract away from particular word lengths).

7.8.4.2 Smoothing Algorithm

The second issue in smoothing is what method to use for discounting probability mass from seen events and redistribute it to unseen events. In a first approach, I used the smoothing strategy that was implemented in the syllable-HMM (which this model for stress assignment is based on Schmid *et al.* [2005]). It used a non-standard procedure which I will refer to as “Schmid-Smoothing” throughout this work. The smoothed probabilities are

$$P(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i}) + \Theta P(w_i|w_{i-n+2}^{i-1})}{c(w_{i-n+1}^{i-1}) + \Theta}$$

prob(# 5ver) = 0.0323	**prob(# %5ver) = 2.5044e-05**
prob(%1web2st) = 9.1138e-07	prob(1web2st) = 9.1138e-07
backoff(5ver) = 0.0001	**backoff(%5ver) = 0.1428**
prob(5ver %1web2st) = 1.0041e-10	prob(%5ver 1web2st) = 1.3019e-07
prob(%1web2st #) = 0.6091	prob(1web2st #) = 0.6091

Table 7.5: Backoff Probabilities for the word *verwebst*.

where Θ is a predefined smoothing parameter. In this work, I set $\Theta = 1$. The backoff-factor can then be derived as:

$$\begin{aligned}
\alpha(w_{i-n+1}^{i-1}) &= \frac{1-\sum_{w_i} p(w_i|w_{i-n+1}^{i-1})}{1-\sum_{w_i} p(w_i|w_{i-n+2}^{i-1})} \\
&= \frac{1-\sum_{w_i} \frac{c(w_{i-n+1}^{i})+\Theta p(w_i|w_{i-n+2}^{i-1})}{c(w_{i-n+1}^{i-1})}}{1-\sum_{w_i} p(w_i|w_{i-n+2}^{i-1})} \\
&= \frac{c(w_{i-n+1}^{i})+\Theta-\sum_{w_i} c(w_{i-n+1} i-\Theta\sum_{w_i} p(w_i|w_{i-n+2}^{i-1}))}{(c(w_{i-n+1}^{i})+\Theta)(1-\sum_{w_i} p(w_i|w_{i-n+2}^{i-1}))} \\
&= \frac{\Theta(1-\sum_{w_i} p(w_i|w_{i-n+2}^{i-1}))}{(c(w_{i-n+1}^{i})+\Theta)(1-\sum_{w_i} p(w_i|w_{i-n+2}^{i-1}))} \\
&= \frac{\Theta}{c(w_{i-n+1}^{i-1})+\Theta}
\end{aligned}$$

Using this smoothing method, the error rate is inacceptably high. When I analysed the results, I identified some patterns of mistakes that "should not occur" i.e. that are not explicable based on the data or due to exceptions. My observation was that with Schmid-smoothing, the stress was very often assigned to unstressable prefixes or suffixes. Particularly often-occurring examples were in words prefixed by *ver-*, *be-*, *er-*, which for some reason were stressed by the algorithm. On the other hand, the normally stressed prefixes *ab-*, *auf-*, *aus-* etc. were not stressed!

The assumption in Schmid-backoff is that if we have a context that we saw very often, we should also have seen it with the actually considered state, otherwise our current state is probably not very grammatical or otherwise bad and should therefore be punished for having to backoff. On the other hand, it is probable to not have seen a particular state after a state that occurs very infrequently any way. In table 7.5, the effect of Schmid-smoothing on the word `5ver-1web2st` is shown (only the analysis that was assigned the highest probability and the desired analysis are shown). The syllable `1web2st` was seen twice in the training data, once stressed and once unstressed (from the two possible stress positions in the word *überwebst*).

The prefix `5ver`, has been seen in the training data 6,502 times without stress and only 6 times with stress (these were actually annotation errors: `5vér1wasch2en`). In the test list, there are 336 words with the prefix `5ver`. Out of these, 83 were wrongly stressed on the *ver*-prefix. A typical example is `5ver-1web2st`: As we expect, a much higher probability is assigned to unstressed `5ver` initially. But then, the sequence `5ver-1web2st` was not in the training data. To estimate its probability, the next smaller context, which consists only of `1web2st`, is considered. In the case shown here, the probabilities for stressed and unstressed `1web2st` are the same (I chose this example for simplicity, the effect occurs in cases when the algorithm has to back-off). But then the algorithm combines the estimate of the smaller n-gram with the Schmid-backoff-value for seeing `5ver`. This backoff probability is high for stressed `5ver` as it has not been seen often, but very low for unstressed `5ver`.

The assumption in Schmid's smoothing is sensible for e.g. POS-tagging, where we want to punish ungrammatical sequences. A very common tag such as `ART` for an article should have been seen in training with all POS-tags that may follow it, whereas a very rare tag is expected to not have been seen in all possible contexts.

The assumption used for POS-tagging does however not hold for stress assignment on syllables. We expect to see many different stems (and also a lot of stems that were not previously seen in that context) with morphemes that are productive, such as *ver-*. That is, we want to have a higher probability for contexts that occur with a lot of different syllables. I then looked for some approach that would take this idea of productivity into account. A comprehensive summary and comparison of commonly used smoothing strategies can be found in Chen [2003].

Kneser-Ney Smoothing is a variant of Absolute Discounting. It is an interpolated model. That means that all smaller n-grams are always taken into account even if the actually considered sequence has been seen. The formula for Interpolated Kneser-Ney Smoothing has the following form:

$$p_{KN}(w_i|w_{i-n+1}^{i-1}) = \frac{\max\{c(w_{i-n+1}^{i}) - D, 0\}}{\sum_{w_i} c(w_{i-n+1}^{i})} + \gamma(w_{i-n+1}^{i-1}) * p_{KN}(w_i|w_{i-n+2}^{i-1})$$

where the backoff factor is

$$\gamma(w_{i-n+1}^{i-1}) = \frac{D}{\sum_{w_i} c(w_{i-n+1}^{i})} N_{1+}(w_{i-n+1}^{i-1}\bullet)$$

D is the discounting value, $c(w_i)$ is the frequency of w_i, and $N_{1+}(w_{i-n+1}^{i-1}\bullet)$ is a function that returns the number of contexts from w_{i-n+1} to w_i followed by any syllable, that have a count of one or more:

$$N_{1+}(w_{i-n+1}^{i-1}\bullet) = |\{w_i : c(w_{i-n+1}^{i}) 0\}|$$

Kneser-Ney Smoothing differs from Absolute Discounting in that the probability of the lower order n-grams are estimated as

$$p_{KN}(w_i|w_{i-n+2}^{i-1}) = \frac{N_{1+}(\bullet w_{i-n+2}^{i})}{N_{1+}(\bullet w_{i-n+2}^{i}\bullet)}$$

where

$$N_{1+}(\bullet w_{i-n+2}^{i}) = |\{w_{i-n+1} : c(w_{i-n+1}^{i}) > 0\}|$$

and

$$N_{1+}(\bullet w_{i-n+2}^{i}) = |\{(w_{i-n+1}, w_i) : c(w_{i-n+1}^{i}) > 0\}| = \sum_{w_i} N_{1+}(\bullet w_{i-n+2}^{i})$$

that is, all lower-order models are calculated as:

$$p_{KN}(w_i|w_{i-n+1}^{i-1}) = \frac{\max\{N_{1+}(w_{i-n+1}^{i}) - D, 0\}}{\sum_{w_i} N_{1+}(w_{i-n+1}^{i})} + \gamma(w_{i-n+1}^{i-1}) * p_{KN}(w_i|w_{i-n+2}^{i-1})$$

In this implementation, I used Modified Kneser-Ney Smoothing which was proposed by Chen and Goodman, who conducted a large comparative analysis of alternative smoothing algorithms and found modified Kneser-Ney Smoothing to work best, especially for small non-zero counts. The good performance on this task can thus be explained by the important data sparsity problem encountered. Modified Kneser-Ney Smoothing differs from Kneser-Ney Smoothing only in that it uses more specific parameters, that is N_{1+} is substituted by three different terms: N_1, N_2 and N_{3+} and matching specialized discount values D_1, D_2 and D_{3+}. The model is given as:

$$p_{KN}(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i}) - D(c(w_{i-n+1}^{i}))}{\sum_{w_i} c(w_{i-n+1}^{i})} + \gamma(w_{i-n+1}^{i-1}) p_{KN}(w_i|w_{i-n+2}^{i-1})$$

where

$$D(c) = \begin{cases} 0 & if \quad c = 0 \\ D_1 & if \quad c = 1 \\ D_2 & if \quad c = 2 \\ D_{3+} & if \quad c \geq 3 \end{cases}$$

and

$$\gamma(w_{i-n+1}^{i-1}) = \frac{D_1 N_1(w_{i-n+1}^{i-1}\bullet) + D_2 N_2(w_{i-n+1}^{i-1}\bullet) + D_{3+} N_{3+}(w_{i-n+1}^{i-1}\bullet)}{\sum_{w_i} c(w_{i-n+1}^{i})}$$

Optimal values for D_1, D_2, D_{3+} were determined as:

$$Y = \frac{n_1}{n_1 + 2n_2}$$

$$D_1 = 1 - 2Y\frac{n_2}{n_1}$$

$$D_2 = 2 - 3Y\frac{n_3}{n_2}$$

$$D_{3+} = 3 - 4Y\frac{n_4}{n_3}$$

and n_x is the number of entities that were seen exactly x times in training.

An assumption in the back-off and interpolation models is that all w_i are part of a closed vocabulary W. It is therefore not specified how to proceed with previously unseen entities. In the current application however, there are a lot of such unseen syllable-stress pairs. These unseen entities can either occur as part of the context. In that case, we simply go down to the largest m-gram that does not contain any such syllables, and estimate the probability from that m-gram model.

If the syllable-stress pair which we want to estimate has not been seen, we still need to assign it some probability. The simplest solution is to assign such an entity a uniform probability, but this is not optimal, we can do better by estimating a syllable-stress pair's probability by basing on the probability of seeing a stressed syllable. As long as both the stressed and the unstressed version of the syllable were seen in training, it is fine to estimate the probability of the syllable and its stress status from the probability of stressing a syllable that was seen only once in training.

But, as the probability of the stress status is relatively high, we run into problems when the syllable was seen stressed but not unstressed or the other way around, because then we estimate $P(\langle syll, stress\rangle)$ for the one case and $P(stress)$ for the other case. This leads to severe errors of preferring syllables that were not seen in training with a particular stress status (and from which we should therefore possibly conclude that they were not seen *because* it is very bad to stress them (e.g. a schwa syllable)). Therefore, the probability of a syllable that has been seen but not with a particular stress status is estimated as $P(\langle syll, stress\rangle) = P(syll) * \hat{P}(stress|syll)$. This version showed to prevent errors of e.g. stressing schwa-syllables and led to an improvement in performance.

The overall performance of Kneser-Ney Smoothing is consistently and significantly better than Schmid Smoothing. Errors like stressing unstressed prefixes (*ver-*, *be-* etc.) do not occur.

But there are still some mistakes we would like to prevent. For example, we definitely would like our algorithm not do make mistakes like `5Ge-1rinn-#3sel2s` instead of `5Ge-#1rinn-3sel2s`, because suffixes containing an *e* are in principle never stressed, because these *e*s are realized as schwas. But, as can be seen from the current example, the algorithm does not learn such rules. While it correctly determined that `5Ge` should not be stressed, `1rinn` and `3sel2s` were both unstressed in all occurrences in the training data, and both had very few counts. To solve this problem, the algorithm would need to estimate the probability of `1rinn` being stressed via its being the first syllable of a stem, and `3sel2s` as a suffix containing (very probably) a schwa.

context window	k=1		k=2		k=3	
training set	schmid	knes.	schmid	knes.	schmid	knes.
Phonemes	12.6%	8.8%	17.3%	8.7%	20.2%	9.4%
Letters	14.2%	9.9%	19.7%	9.5%	22.6%	10.4%
Lett. + morph	13.2%	10.3%	18.6%	10.3%	21.5%	11.1%

Table 7.6: Performance for different input data, for different smoothing methods and context sizes.

The improvements obtained through using modified Kneser-Ney Smoothing instead of Schmid Smoothing (refer to table 7.6) are all strongly statistically significant ($p < 0.0001$, using a two-tailed binomial test).

7.8.5 Summary of Experiment Results

In addition to generalizing the data quite well by reaching a word error rate of below 10%, the algorithm also remembers the training data well. If run on the data it was trained on, it stresses over 99% of the words correctly. This means that no exception lists is required stress assignment to known words.

A good parameter setting for the HMM seems to be a small context window of 1. Data sparseness (and therefore the appropriate choice of a smoothing algorithm) is an important issue that arises from defining the HMM on syllables rather than letters. I showed that using Kneser-Ney-Smoothing instead of Schmid-Smoothing can reduce the word error rate for word stress assignment by almost 50%. I also showed that in designs with important data sparsity problems, using as much data as possible for training is important, even if the quality is not perfect.

The algorithm performs best on phonemes, which are the least ambiguous representation of words. This is relevant because the letter *e* can be either realized as an unstressable schwa (`/E#/`), or `/AE/` or `/E:/`, which can both port stress.

Morphological information from CELEX does not improve results and the initially proposed process of exploiting morphological annotation for using several steps for stress assignment, did not lead to any better results either, although morphology has an important influence on stressing, as argued above.

7.8.6 Performance in Pipeline System

When the module has to handle imperfect input (i.e. input containing wrong syllable boundaries, or incorrect phonemizations), we can expect it to also suffer from the errors. The below table shows the performance of the module when working on automatically generated syllable boundaries or phonemes. Overall performance is best when stress assignment and syllabification are performed in one step with grapheme to phoneme conversion.

The advantage of the modular design is that a component can be exchanged easily. For example, if some system for stress assignment became available that modelled the problem better, it could substitute the old module, only adapting the input and output formats. Best results were obtained when the syllabification and stressing steps were performed after g2p conversion (see result table 8.2).

	perfect input	automatic input
no morph	9.9%	12.7% (3.1% syll. errors)
CELEX	10.3%	11.7% (1.9% syll. errors)
Phonemes	8.8%	14.3% (6.2% syll./phon. errors)

Table 7.7: Word Error Rates of stress assignment HMM on imperfect syllabification and imperfect phonemes input respectively.

Words annotated with automatically generated syllabification and stress information can also be used as an input to the decision tree for the g2p conversion task. When the letters without morphological information are annotated with automatically generated orthographic syllable boundaries and stress as returned from the HMM stress assignment component, word error rate is reduced from 26.6% to 21.15%, and phoneme error rate is reduced from 3.63% to 3.32%. Both reductions are highly statistically significant.

7.9 Discussion

The original goal of significantly reducing the conversion error rate by running separate syllabification and stress components and using the information for the input to the decision tree was reached, and WER was reduced from 26.6% to 21.1%.

In combination with the g2p-HMM that does not incorporate any constraints, the syllabification and stress modules also lead to an error reduction from 22%WER to 16.6%WER if used as preprocessing components and even down to 15.2%WER if used in postprocessing. However, best results were achieved when the constraints used in the syllabification and stress components were directly incorporated into the g2p-HMM, when a word error rate of only 13.7%WER was reached (refer to chapter 8).

Despite this success in improving overall performance by quite a large margin, my opinion is that none of the approaches outlined above fully fits the stress assignment problem:

- The performance of the rule-based approach is not good enough because of a high number of exceptions to the rules and because it heavily depends on a morphology that correctly identifies morpheme boundaries between compounds. When no statistical information is available to the system, even semantic information is needed to correctly stress compounds with more than two elements.

- The baseline approach did not actually lead to much benefit other than general information on what kind of performance rates can be achieved using statistical information for the stressing probability of syllables with a certain structure. The results were not used to alleviate the data sparsity problems in the HMM component because this kind of stepwise backoff turned out to not bring enough benefits. But I think that there may still be room for improvement, if more time is invested – estimating probabilities more exactly *should* lead to performance gains.

- Finally, although the quality of stress assignment by the HMM is much better than the one from the other models tried, it is not satisfying with an error rate close to 10%. The main problem is that the model does not have the power to model the processes that underlie stress assignment: it does not model the structure of compounds, and does not “understand” the difference between affixes that are stressed because of their semantics, and a syllable in a stem that is stressed because of its weight, position or the origin of the word it is part of.

Chapter 8

Grapheme-to-Phoneme Conversion

The first part of this chapter motivates the use of automatic grapheme-to-phoneme conversion for TTS systems and provides an overview of previous research in the field. I then briefly address the problem of graphemic parsing, which is also known as letter to phoneme alignment. The second part describes the adaptation of the HMM used for syllabification and stress assignment to the grapheme-to-phoneme conversion problem. For smoothing, I obtained best results with a variant of Modified Kneser-Ney Smoothing. The differences between this version and Chen and Goodman's smoothing method are discussed. The chapter furthermore addresses the issues of pruning the HMM and the integration of the constraints that were applied for syllabification and stress assignment in the previous modules to the g2p component. Finally, I briefly report results for English and French and compare them to state-of-the-art methods.

8.1 Motivation for Automatic Grapheme-to-Phoneme Conversion

The term Grapheme-to-phoneme (g2p) conversion is commonly used for describing the process of automatic conversion of letters into their phonemic transcription. The simplest solution to the problem is to do a lexical lookup in large databases where each word is listed together with its phonemic representation. But many words cannot be found in the lexicon. The phonemic representation of OOV words is obtained from transcribing known parts of the words (this can be substrings or single letters) and composing these parts. This compositional strategy tries to mirror the ability of human native speakers of a language to read and pronounce unknown words following the systematics of their language.

Although German orthography is certainly much closer to the pronunciation of German words than English orthography is to the pronunciations of English words, g2p is still a hard problem in German. The German alphabet contains 30 letters, while the phoneme set of German contains 48 phonemes, so there is necessarily some ambiguity. Furthermore, there are some notoriously ambiguous letters such as *e* that can be transcribed as `/AE:/`, `/AE/`, `/E:/` or `/E /` in function of its context.

For a long time, a majority opinion in the speech research community has been that that grapheme-to-phoneme conversion is a solved problem and that linguistically motivated rule-based methods outperform machine learning approaches. However, Damper *et al.* [1999] have argued that neither of these presumptions is true: rule-based methods actually showed to work very poorly on large texts, and even the best machine learning approaches only obtain below 70% of words correct on the English NetTalk corpus, indicating that this is not a solved problem at all.

8.2 Previous Approaches to g2p

A wide range of techniques have been applied to grapheme-to-phoneme conversion. An overview and comparative evaluation of four different approaches: a rule-based method, a nearest neighbour algorithm, a neural network by Sejnowski and Rosenberg [1987] and Pronunciation by Analogy (PbA), is provided in Damper *et al.* [1999].

The oldest approach to grapheme-to-phoneme conversion is to use **phonlogical transcription rules**, e.g. "pronounce the sequence *ab* as `/A: B/`, unless it is a prefix or followed by a syllable boundary, in which case it has to be pronounced `/A P/`". Rule-based systems for g2p conversion are still in use today. Möbius [1998] describes a finite state transducer for the Bell Labs system. They report to use a total of 259 rules in addition to a separate component that deals with affix pronunciations. As Damper remarks, most of the rule-based methods are part of commercial systems and can therefore not be tested by independent researchers to measure their performance and compare them to other systems. The rule-based method tested in the evaluation by Damper used the 372 phonological transcription rules from Elovitz *et al.* [1976]. It performed worst among the tested systems - only 25.7% of the words were transcribed correctly, evaluating on a subset of Webster's dictionary which contained 16,280 words. Damper et al. also report to have evaluated an up-to-date commercial rule-set, for which they obtained similarly low success rates. An obvious disadvantage of rule-based systems is also that they have to be developed by a specialist and are very labour-intensive to create (because of the large number of rules and because of possible interactions between them, so that they have to be arranged carefully).

To overcome this rule acquisition bottleneck, Bouma [2000] proposed an approach for bootstrapping the generation of rules from a very basic subset of rules. The author uses a lazy learning method called **transformation based learning (TBL)**: a set of rule patterns is used to generate new rules that minimize the errors made by the baseline system (i.e. the rule-based system) and bring the largest overall improvement on the data. These rules are then added to the rule set until no more improvements can be found. Applying this method to a rule-based system for Dutch, Bouma achieved a reduction in word error rate from 39.6% to 7.4%. This seems to be a good method for languages where a rule-based system has already been developed and where only small data sets with letters and their phonemic transcriptions are available. Unfortunately, it is difficult to compare results across languages.

The first data-based approach to get much attention was the NetSpeak system, a **neural network** by Sejnowski and Rosenberg [1987], whose performance was 46.0% on unseen test data in Damper *et al.*'s evaluation. It is from this project that the manually annotated and aligned NetTalk corpus originates.

Bosch and Daelemans [1993] compared a connectionist model with an **instance based learning (IBL)** method, a table lookup method (that used heuristics to also provide transcriptions for unseen words) and the rule-based system “Morpa-cum-Morphon”. They found IBL and the table to outperform the neural network. The table performed best and also returned significantly better results than the rule-based system.

In Daelemans and van den Bosch [1997] the table-approach was developed further and stored as a search trie, called **IG-tree** (IG stands for Information Gain, see section 2). In this approach, a separate tree is built for each letter in the alphabet. Thereby, the exact amount of context is added which is needed to completely disambiguate all words from the training data. The branches of the tree represent the context, the leaves contain the phonemes. For new input words, each of the word’s letters is looked up in the trie and a trie branch is followed down to the leaves as far as possible. If the context of a letter has been seen in training and a leaf can thus be reached, the corresponding phoneme is chosen. If no leaf is reached (e.g. because a particular context did not occur in the training data and is ambiguous), majority vote among the leaves is used. The advantage of this approach is that the training data is stored efficiently and the method will transcribe all words seen in training correctly (except homographs). But a problem is that the choice of a transcription is not dependent on the transcription of the preceding letters. Therefore, phonologically impossible sequences that are not compatible with one another can occur.

The IG-tree differs from decision trees (e.g. ID3 or C4.5 (Quinlan [1993]) in that it does not use pruning and in that the order of questions is not computed for the generation of each branch but deterministically dependent on the letter sequence in the word. While the IG-tree only asks for specific letters in specific context positions, many implementations of decision trees use a wider range of questions.

Decision trees were one of the first data-based approaches to be applied to the problem (refer to Lucassen and Mercer [1984]) and are widely used for grapheme-to-phoneme conversion (e.g. Andersen and Dalsgaard [1995]; Kienappel and Kneser [2001]; Black *et al.* [1998]; Suontausta and Häkkinen [2000]). A variant of Lucassen and Mercer’s decision tree, the AddWordTree (AWT), is used in the current IBM TTS system. The AddWordTree is a binary entropy-based decision tree with a context window of 5 letters to each side and 5 phonemes back.

Decision trees allow for a compact representation of the data and are relatively easy and quick to use. When using decision trees, the g2p problem is viewed as a categorization problem. Each letter is assigned to its phoneme-category. For assigning this category, the decision tree asks questions about the letter under consideration and its context (for an example decision tree, see figure 8.2). Good compactness of the model is achieved by firstly using the information gain criterion to decide at each branch of the tree (see section 2), which question to use and secondly defining a effective set of questions.

The tree’s compactness, performance and ability to generalize also largely depend on the questions it is allowed to ask. A very simple tree would only be able to ask for letter identities. The entropy criterion is in such a simple tree only used to choose the letter position to ask about. Only asking for exact identities leads to very large trees. For example, *ch* is pronounced /CH2/ after consonants. If the decision tree can only ask for single letters, it would have to ask the same question for every single consonant: is there a ’r’ at position -1, is there a ’n’ at position -1,... Furthermore, the decision tree can not generalize to letter sequences it did not see in training. Kienappel and Kneser

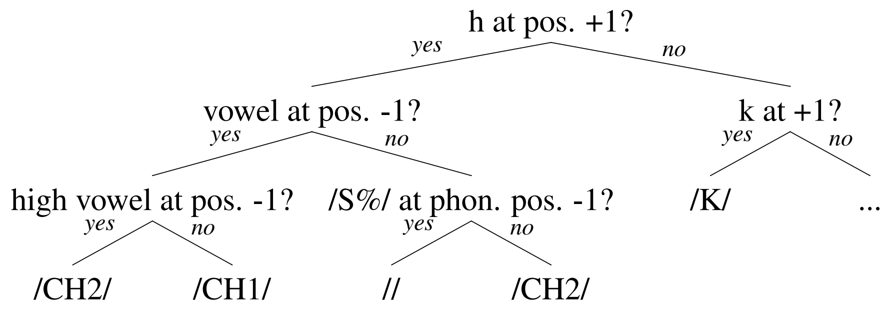

Figure 8.1: Example binary decision tree branch for the letter *c* in position 0.

[2001] experimented with different questions and found that for German and English, it is beneficial to ask questions about:

- letters within a certain context window
- questions about preceding phonemes
- grouping questions (sets of letters, e.g. consonants, front vowels, back vowels)
- "double" questions (i.e. whether two adjacent letters are identical)

The grouping questions can either be defined manually (as done in the AWT) or automatically (cf. Kienappel and Kneser [2001]). Kienappel and Kneser reported that including the more complex questions about groups of letters or phonemes and the question about double consonants lead to a relative improvement of 8% with respect to the decision tree that can only ask for single letters and phonemes.

Advantages of the decision tree are that it is quick to run and compact, because it only asks the questions that are relevant according to the information gain criterion and therefore ideally only stores the relevant parts of the context. However, as Bellegarda [2003] pointed out, pruning and choosing the relevant questions by looking at what patterns can be found in the data has a price: the effective set of questions ends up best covering those phenomena which are most represented in the training data. Rarely seen contexts tend to be overlooked, regardless of their degree of relevance to a given situation. New words that comprise a number of irregular patterns that were rarely seen in the training set will suffer from this limitation. This observation may explain why the joint HMM model which I am using (see below) returns better results even when a smaller context window is used.

The basic idea in analogy-based methods is to compose the pronunciation of an unseen words from large substrings of words seen in training. The approach was first implemented by Dedina and Nusbaum [1996], and was most recently applied by Damper *et al.* [1999] in their **Pronunciation by Analogy** system. An attractive aspect of the analogy-based methods is that they do not use a fixed context window size but can match arbitrarily long sequences. During the training phase, a large lattice of the words seen from training data is built up. Any new words can the be analysed by searching for a matching path through this lattice. In most cases, there are many such paths. A range of different strategies for heuristically choosing the best path have been proposed through the years. Recently, Damper *et al.* [2002] used a combination of five selection strategies to determine the best combination of sub-sequences and achieved a significant improvement over using just one strategy. The original system did not tempt to also assign syllable boundaries or stress. In Marchand and Damper [2005], the authors enrich the graphemic representation of the words with syllabification information and achieved a significant improvement from 65% WER to 71% WER on the NetTalk corpus. However, this improvement was only attained with "perfect" syllabification. Syllable boundaries from an automatic system (using essentially the same mechanism, which they called Syllabification by Analogy (SbA)) did not lead to an performance improvement as errors were propagated and the syllabification module did not have access to any knowledge or data that the simple system would not have had access to.

Another type of approaches for which very good results were reported are **joint n-gram** models. In joint n-gram models, letters and phonemes are combined to one single state, a "graphone", similar to the combined syllable-stress states and letter-syllable-boundary states used in the previous chapters. Such an approach was taken by Bisani and Ney [2002] and Galescu and Allen [2001]. Both use an integrated letter-phoneme alignment procedure (all of the other data-based approaches are dependent on a preprocessing step that determines the alignment between letters and phonemes, which is not a trivial task (see section 8.3). Galescu and Allen [2001] pre-derive letter-phone chunks that consist of at most zero or one letters and zero or one phones (i.e. they introduce null-letters and null-phonemes for the alignment). Bisani and Ney [2002] on the other hand experimented with different maximal letter and phoneme sequences for the joint states in function of the context window size. Best results (PER=0.52%) were found for a context window size of 3 and using $\langle(l:|1..2|):(g:|1..2|)\rangle$[1] as length constraints for the joint states. On the other hand, a design with $\langle(l:|1..6|):(g:|1..6|)\rangle$-chunks yielded an error rate of 2.3%. When using a smaller context window size, they found larger chunks to work better than smaller chunks: for k=1, a $\langle(l:|1..6|):(g:|1..6|)\rangle$-chunk achieved a phoneme error rate of 2.79%, while the $\langle(l:|1..2|):(g:|1..2|)\rangle$ lead to 14.08% PER. A plausible explication for these results is that large chunks lead to heavy data-sparsity problems, and that small chunks cannot capture enough of the contextual information if the context window size is also small.

Chen [2003] systematically compares conditional n-gram models (decision trees) and joint n-gram models (similar to the ones introduced by Bisani and Ney [2002] and Galescu and Allen [2001]). Chen iteratively optimizes the letter-phoneme alignment during training for the g2p conversion task, uses small chunks ($\langle(l:|0..1|):(g:|0..1|)\rangle$ pairs only) and applies a smoothing method based on Gaussian priors instead of ML estimates. This smoothing method allows him to profitably run the algorithm on 7-grams and larger, while results already degrade for 5-grams when other smoothing methods (Kneser-Ney (used by myself, see below), absolute discounting (used by Bisani and Ney) or Witten-Bell Smoothing (used by Galescu and Allen)) are applied. Chen [2003] reports evaluation results superior to Marchand's PbA approach (evaluating on the NetTalk data), and outperforms both Bisani and Ney [2002] and Galescu and Allen [2001].

[1]This notation stands for a graphone made of one or two letters and one or two graphemes.

An approach using joint n-grams was also implemented by Caseiro *et al.* [2002] for Portuguese. But he did not obtain results superior to the rule-base system used.

Models that do not use joint letter-phoneme states and therefore cannot make the decision of the actual phoneme dependent on the preceding letters but only on the actual letter and the preceding phonemes, did not achieve very good results. Examples of such approaches using **Hidden Markov Models** are Rentzepopoulos and Kokkinakis [1991](who applied the HMM to the related task of phoneme-to-grapheme conversion), Taylor [2005] and Minker [1996].

8.3 Alignment

In the current IBM system, data alignment is the most laborious step in data-based grapheme-to-phoneme conversion. The alignment process is done semi-automatically where the algorithm is given a set of letters, phonemes and mappings from single letters to phonemes or phoneme-sequences. Then, all mappings are calculated, and the ambiguous cases are disambiguated manually, a process that easily takes one or two weeks of full-time work for every new data set. (An example for a sequence whose alignment has to be disambiguated is *tts* to `/T S/` (e.g. in the word *Betts* (Engl: genitive of *bed*)). Each of the *t*s can either map onto `/T/` or the empty phoneme-sequence `//`.)

8.3.1 Alignment with Morphological Boundaries

For the German data, morphological boundaries initially mapped onto syllable boundaries. This meant that every time some morphological annotation was changed (e.g. due to using another morphological preprocessing algorithm), hundreds of disambiguation rules had to be adapted manually, before the effect of the change on g2p conversion performance could be evaluated. This made the process very ineffective and constituted a real bottle-neck. (Coming back to the above example, let us assume that the morphological process annotated a morphological boundary `tt2s`. This has now again be disambiguated to `/T S/`.)

Although a mapping from morphological boundaries onto syllable boundaries may be intuitive, it is not necessary for the decision tree to learn that a morphological boundary should map onto a syllable boundary. As it does ask questions about the context, it may as well learn that whenever a morphological boundary was seen at position -1, it should use the phone-sequence version that contains a syllable boundary (e.g. `/. ? A/` instead of `/A/`).

When the alignment algorithm is run on new morphologically annotated data, it generates all ambiguous sequences that contain the morphological boundary. The decisions on how to align the word does not change through the morphological annotation. If we chose to make the first *t* map onto `/T/` and the second onto silence for *tts*-example, we want to make the same decision for the morphologically annotated sequence `tt3s`. I implemented a simple script that automatically annotates these cases. For each newly generated ambiguity, it looks up the disambiguation of the same string without the morphological annotation, copies the alignment to the new string and inserts a null-mapping for the morphological boundary.

And indeed, using the automatic script for making all syllable boundaries map onto an empty string did not only perform as good as the old version but even a bit better – probably because manual disambiguation always also introduces inconsistencies that do not occur when the automatic process is applied. New versions for morphological annotation can now be tested without manual effort.

This script does however not solve the problem of manually aligning data for corpora in other languages.

8.3.2 Fully Automatic Alignment

The most widely used approach to do letter to phoneme alignment automatically is to use the Expectation Maximization (EM) Algorithm. The EM-algorithm iteratively maximizes the likelihood of the letter-phoneme alignment until convergence. For a description of the alignment problem and of how the EM algorithm works, refer to Damper *et al.* [2004].

Some of the English and French data bases I evaluated the g2p-HMM on were aligned using an automatic alignment algorithm and still did return quite good results (see table 8.5 below).

I can however not comment on the exact performance differences, because currently, no corpus is available to me for which I have access to both manual and automatic alignment. But I hope to supplement this information later: Yannick Marchand very kindly sent me a small excerpt of automatically aligned CELEX for German, and promised to send the full aligned CELEX in the beginning of June (which is unfortunately after the deadline of this thesis, so that I cannot report results on comparison between the manually aligned versions and state-of-the-art automatic alignment in this thesis).

8.4 A HMM for g2p conversion

The strength of an HMM model defined in analogy to the stress and syllabification models is that it captures the context well, that is, it has a good memory for the examples it saw (in function of its context window width) and does have the power to generalize to new data by looking at sub-sequences of the data. Furthermore, using the Viterbi algorithm, it guaranties to find the optimal state sequence (i.e. the most probable one according to the estimation methods used) if no pruning is used. Therefore, it seems only sensible to try and adapt the model to grapheme-to-phoneme conversion, which can also be regarded as a tagging task: tagging letters with phonemes.

After I implemented this model, I actually found that a very similar approach had already been taken by Chen [2003] and Bisani and Ney [2002] and Galescu and Allen [2001] (see section 8.2).

8.4.1 Definition

The observed states in the model are the letters, the hidden states are letter-phoneme pairs. This is why I prefer to call this method “letter-to-phoneme conversion” instead of “grapheme-to-phoneme conversion” – I work on letters and not on graphemes. The phoneme sequence can therefore contain zero to n phonemes, where n is typically ≤ 3. A typical hidden state (for the model that does g2p, syllabification and stress assignment in one step) looks like this: `a-.1_?_ A:`. (The underscores mark the boundaries between two phonemes.)

But where do these states come from? The letter and phoneme sequences from the training data need to be aligned in a separate step first, as I did not implement an Expectation-Maximization step to align letter to phoneme sequences. The training input format looks as follows:

```
...
a-.1_?_A: a- s-S g-._G e-EI i- e-._E\# r-R\# s-S
a-.1_?_A: b-._B a-A k-._K u-U s-S
a-._?_A b-._B a-AN n- d-.1_D o-ON n-
...
```

The HMM returns the most likely phoneme sequence $\hat{p}_1^n = \hat{p}_1, \hat{p}_2, \hat{p}_3, ..., \hat{p}_n$, for a given letter sequence l_1^n. Thereby, the markov assumption is applied which states that a given letter - phoneme-sequence pair $\langle p;l\rangle$ only depends on the preceding k states (which are also letter - phoneme-sequence pairs).

$$\hat{p}_1^n = \arg\max_{p_1^n} \prod_{i=1}^{n+1} P(\langle p;l\rangle_i \,|\, \langle p;l\rangle_{i-k}^{i-1})$$

To each word, dummy states '#' are appended at both ends to indicate the word boundary and ensure that all conditional probabilities are well-defined. When testing HMM with several different context window sizes from k=3 to k=5, I found that k=4 actually worked best.

8.4.2 Smoothing and Backoff

Several smoothing techniques, which I also tried on the other Hidden Markov Models were implemented. For a detailed discussion of how these techniques work and what effects they have on the performance of the algorithm, see section 7.8.4.2.

- **Schmid Smoothing**
  The HMM outperforms the decision tree when using Schmid backoff, but Kneser-Ney Smoothing proved again to work better.

- **Interpolated Modified Kneser-Ney Smoothing**
  I tried two different versions: in the first one the unigram probability was estimated using a uniform distribution: $P(\langle l,p\rangle) = \frac{1}{|\langle l,p\rangle : c(\langle l,p\rangle)>0|}$. The second version was to estimate the unigram probability from its distribution seen in the training data: $P(\langle l,p\rangle) = \frac{c(\langle l,p\rangle)}{\sum_{\langle l,p\rangle} c(\langle l,p\rangle)}$. The difference in performance is marginal (22.0% vs. 21.99%) and not significant. I therefore use the uniform distribution, because of its simplicity. In Chen [2003], the authors also use a uniform distribution.

- **Variant of Interpolated Modified Kneser-Ney Smoothing**
  Best results were obtained with a variant of Interpolated Modified Kneser-Ney smoothing. In this variant, only the next smaller n-gram model was taken into account for events that had been seen.

A variant of Modified Kneser-Ney Smoothing:
$p_{vKN}(w_i|w_{i-n+1}^{i-1}) =$

$$\begin{cases} \frac{\max\{c(w_{i-n+1}^{i})-D,0\}}{\sum_{w_i} c(w_{i-n+1}^{i})} + \frac{D}{\sum_{w_i} c(w_{i-n+1}^{i})} N_{1+}(w_{i-n+1}^{i-1}\bullet) \frac{N_{1+}(\bullet w_{i-n+2}^{i})}{N_{1+}(\bullet w_{i-n+2}^{i}\bullet)} & if \quad c(w_{i-n+1}^{i}) > 0 \\ \frac{D}{\sum_{w_i} c(w_{i-n+1}^{i})} N_{1+}(w_{i-n+1}^{i-1}\bullet) p_{vKN}(w_{i-1}|w_{i-n+2}^{i-1}) & if \quad c(w_{i-n+1}^{i}) = 0 \end{cases}$$

This version differs from true Modified Kneser-Ney Smoothing in two ways: firstly, it does just one interpolation step for $c(w_{i-n+1} > 0)$. The term $\frac{N_{1+}(\bullet w_{i-n+2}^{i})}{N_{1+}(\bullet w_{i-n+2}^{i}\bullet)}$ is itself smoothed with lower-order n-grams in the original version, but not in my variant. Secondly, the probability when backing off to lower-order n-grams (i.e. when $c(w_{i-n+1}^{i}) = 0$) is estimated from frequencies instead of counts (as done in Absolute Discounting). I therefore also implemented Absolute Discounting, to see whether it would work better than my version of Modified Kneser-Ney Smoothing, but that was not the case. The difference between using my variant of Modified Kneser-Ney Smoothing or original Modified Kneser-Ney Smoothing makes up 1%-2% in WER (which corresponds to up to 15% relative improvement).

Table 8.1 shows the word error rates for using Schmid smoothing and my variant of Modified Kneser-Ney Smoothing respectively. This version outperforms Schmid smoothing in particular on the phoneme-error rates when stress is not counted, and on input that is not morphologically annotated. Significant differences ($p < 0.00001$) with respect to Schmid smoothing are marked with an asterisk.

	stress counted		phonemes only	
	Schmid	Kneser-Ney	Schmid	Kneser-Ney
nomorph	23.4%	21.5%*	9.2%	7.3%*
CELEX	25.0%	25.2%	7.3%	6.2%*
ETI	27.0%	26.5%	8.3%	7.2%*

Table 8.1: Results for g2p with HMM - Schmid Backoff vs. my variant of Modified Kneser-Ney Smoothing.

One of the problems encountered in the model for stress assignment, namely how to cope with previously unseen states, does not occur in this implementation of the letter-to-phoneme conversion HMM. The model looks up all mappings from letters to phonemes that were seen in the training data and only tries to tag a letter with phonemes that it was already aligned with in the training data. This may in principle prevent the algorithm from aligning a letter with the phoneme-sequence that would be required to correctly transcribe the word, but if it was not ever seen in the training data, the algorithm should assign it a very low probability any way. This restriction of only using already seen letter–phoneme-sequence pairs allows me to have a closed vocabulary and therefore a real probability distribution (which is not given otherwise, at it is not known how much of the probability mass should be given to unseen entities).

8.4.3 Pruning

The 30 letters of the German alphabet combine with the 48 single phonemes, the syllable tag and the stress sign to 363 different pairs of letters and phoneme-sequences. This means that the average number of tags for an observed letter is more than 12. This leads to a huge lattice (the full lattice for a 10 letter word would contain 10^{12} sequences). Even when using the Viterbi algorithm, the processing times for calculating all relevant sequences and comparing them is about 8 minutes per word on a 1.50GHz machine, which is of course totally unacceptable. In fact, we need something that runs at least in real time.

This processing time problem is well-known from typical HMM applications such as speech recognition. It is very common to use pruning in these cases. The idea in pruning is that sequences with low probabilities are not calculated to the end, as it is very improbable that they will come out to be the finally best ones. That is we try to cut off those sequences which we assume not to contain the optimal sequence. A widely known pruning strategy is beam search pruning (proposed by Lowerre and Reddy [1980]). In beam search pruning, only those sequences are considered, which have a probability "similar" to the current best sequence. A threshold is needed to define what "similar" means. Using this pruning strategy, processing time depends on the number of sequences that are similarly probable as the best sequence. Therefore, search time is not necessarily proportional to the length of the word but may depend on the actual search space and data configuration. Alternatively it is also possible to very simply always consider a certain number of sequences that are followed, independent of how similar they are to the actually best sequence. I tried various thresholds: 5, 10, 15, 25, 35. While the very low thresholds actually do prune of some desired sequences, the sizes above 15 returned reasonable results that did not significantly differ from searching the whole space. Choosing a threshold of 35 still allows to analyse about 80 words per second on a 1.5GHz machine, while a more than 120 word per minute can be processed with a threshold of 15 and almost 400 with a threshold of 5 considered sequences. For the HMM without thresholds on raw letters, the performance degrades from 22.0% to 22.9% when using a threshold of 5 instead of 35 (see figure 8.2).

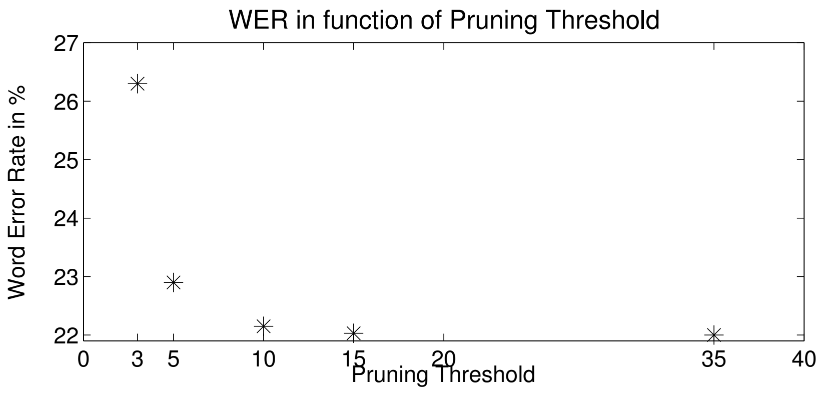

Figure 8.2: The Effect of Pruning.

Furthermore, the current implementation is in Perl, so that I am confident that a reimplementation in C would lead to better time performance.

8.4.4 Multiple Solutions

The Viterbi algorithm guaranties to return an overall best state sequence (at least if it is not pruned), by maximizing the overall probability $P(\langle p,l\rangle_i \,|\, \langle p,l\rangle_{i-n+1}^{i-1})$. In some cases, we would like to know the two best pronunciations of a word, e.g. for homographs. The Viterbi algorithm has to be modified in order to do this, as it always only remembers the single best sequence leading to a certain state. The two globally best sequences might share the last node – in which case the second best analysis would be lost. Therefore, it is necessary to modify the Viterbi algorithm so that it always remembers the two best states. A formal derivation and proof can be found in Rentzepopoulos and Kokkinakis [1991]. The authors considered the mirror problem: they wanted the algorithm to return two alternative spellings for homophones on their phoneme-to-grapheme conversion task. Unfortunately, I did not have time to implement this modification to the Viterbi algorithm and therefore cannot comment on the quality of the results.

8.4.5 Integration of Constraints

Integrating constraints into the syllabification and stress assignment HMMs leads to hughe performance gains. Therefore, it is an obvious idea to also integrate these constraints into the grapheme-to-phoneme conversion model. The constraints are just the same, as in the other models:

- allow exactly one main stress per word
  Once a stressed syllable has been generated, no more phoneme-sequences that also have a stress are allowed. The globally best sequence is only chosen among those sequences that have one stress.

- allow exactly one nucleus per syllable
  No syllable boundary can be generated, if no vowel has been seen since the last syllable boundary. Also, all sequences that generate a second vowel without inserting a syllable boundary in-between are eliminated. The final sequence is only accepted, if the last syllable also contains a vowel.

Including the stress and syllabification constraints leads to a highly significant improvement: the word error rate for not morphologically annotated words is almost reduced by 50% (from 26.6% WER to 14.6% WER) with respect to the decision tree results for the same input.

The integration of the constraints into the g2p component, to which the syllabification and stress module owe a large part of their tagging success, is also superior to a pipeline-architecture. When the constraints are used, the integrated approach of converting graphemes to phonemes, setting syllable boundaries and assigning word stress in one single module achieves a lower word error rate than tackling the three problems in separate steps.

For a model that does not contain the constraints, the separate syllabification and stressing steps do contribute positively (see results in table 8.2 below).

	Preprocessing	Postprocessing	one-step + constr.	one-step no constr.
no morph	syllables only: 96.9% w corr	phonemes only: 94.0% w corr	phonemes only: 92.2% w corr	phonemes only: 93.6% w corr
	syll + stress: 86.7% w corr	phon + syll: 92.8% w corr	phon + syll: 91.5% w corr	phon + syll: 92.4% w corr
	syll+str+phon: 83.4% w corr	phon+syll+str: 84.8% w corr	phon+syll+str: **86.3%** w corr	phon+syll+str: 78.5% w corr
CELEX	syllables only: 98.1% w corr	phonemes only: 95.0% w corr	phonemes only: 93.5% w corr	phonemes only: 94.7% w corr
	syll + stress: 88.2% w corr	phon + syll: 94.9% w corr	phon + syll: 92.8% w corr	phon + syll: 93.4% w corr
	syll+str+phon: 83.9% w corr	phon+syll+str: 85.6% w corr	phon+syll+str: **86.7%** w corr	phon+syll+str: 74.7% w corr

Table 8.2: Comparing the effects of modularization, and preprocessing and post-processing of syllabification and stress.

8.4.6 Results

The German CELEX corpus includes information about stress and syllabification. Therefore, we want to also learn the stress assignment and syllabification from this data. I compared several possibilities for doing so. The experiment conditions were

1. WER
   word error rate using the variant of Modified Kneser-Ney-smoothed HMM without any constraints

2. WER-sc
   WER condition augmented by the constraint for stress assignment (exactly one main stress per word)

3. WER-ssc
   WER-sc condition augmented by the constraint for syllables: each syllable must contain exactly one vowel
   Due to suboptimal transcription of the suffix *-tion* to `/T I: O: N/` in CELEX, this constraint makes the suffix get split up into `/T I: . O: N/` or `/T I: .1 O: N/`.

4. WER-pss
   Perfect syllable boundaries and stress positions are projected onto the letters, so that the algorithm will not make any mistakes on these, and in addition has a richer input, which should allow it to more successfully decide on syllable-final consonant devoicing, glottal stops etc.

5. WER-S1
   the same condition as WER, but errors on stress assignment are not counted

6. WER-S2
   the model does not try to learn syllable boundaries or stress patterns, i.e. it does not see them in training either.

	WER	WER-sc	WER-ssc	WER-pss	WER-S1	WER-S2
nomorph HMM	21.5%	14.9%	13.7%	5.7%	7.5%	7.3%
nomorph AWT	26.6%			7.05%	15.0%	
ETI HMM	26.5%	14.9%	13.6%		7.5%	7.2%
ETI AWT	21.1%			11.3%		
CELEX HMM	25.2%	14.7%	13.2%	5.6%	6.5%	6.2%
CELEX AWT	21.7%			7.0%		

Table 8.3: Results for g2p with HMM and Schmid-Smoothing vs. AWT, alternative morphological annotations. p is set to 4 in the HMM, pruning threshold t=15.

In a last experiment setting, I wondered about the optimal place of the stress mark in a setting where states are defined on graphones as units - are the syllable boundaries really a good place to mark stress, or wouldn't it be better to mark stress with the syllable nuclei? The syllable onset is the least important part of a syllable to decide stress, the nucleus and coda play a more important role to determine syllable weight. Each stress marker was mapped onto the following vowel. But disappointingly, this change did not lead to a performance improvement.

Table 8.4 displays performances for different complexities of the model. The word error rates shown were achieved with a HMM that incorporates the stress and syllabification constraints, and was corrected for the incorrect or at least questionable transcriptions of `/T I: O: N/` for the suffix *-tion*, which should better be `/T J O: N/`. Without this automatic correction, word error rates are about 0.5–1 percentage points higher. The size of the training set is about 13 MB.

	k=3	k=4	k=5
nomorph HMM	14.4% (2.2 MB)	13.7% (5.1 MB)	16.1% (9.3 MB)
ETI HMM	15.1% (1.8 MB)	13.6% (3.9 MB)	16.3% (7.1 MB)
CELEX-Morph HMM	14.6% (1.8 MB)	13.2% (4.1 MB)	14.9% (7.5 MB)

Table 8.4: The effect of different context window sizes for the HMM with constraints, the size of the training data files (uncompressed text files) are shown in brackets.

An important result that can be read from this table is furthermore, that the HMM captures morphological information much better than the decision tree because it captures the structure more adequately, which seems to be significant when it comes to translating the effects of morphological boundaries automatically. That is, the HMM has a better memory for exact sequences and recognizes a morpheme better. To achieve any improvement at all, it is necessary to have a near to perfect morphology - even the ETI morphology, which lead to a significant improvement of performance with the decision tree, only leads to a minor improvement that does not reach statistical significance.

One typical type of errors that occurs quite frequently is inaccurate vowel length before doubled consonants, although there exists a strict rule in German, that a vowel which is followed by a doubled consonant (in the same morpheme) is always short. Both the decision tree and the HMM do not capture this information because they lack concept for abstracting to the level "two identical consonants". The model does not have the power to generalize from seen sequences of short vowels followed by two identical consonants (or *ck*) that any vowel followed by any two identical consonants should be short – to achieve this it would have to see in the training data any sequence of vowels followed by double consonants, as it does not have a concept of that short vowels belong to a class of sounds. To capture the overall rule for double consonants, these would have to be stored as graphemes, and also would need to be grouped together in a class.

8.5 Performance on other languages

The HMM model is very straightforward to apply to other languages. An aligned corpus with words and their pronunciations is needed, but no further work or adaptation is required.

Performances of the algorithm on the English NetTalk corpus and Teacher's Word Book were evaluated using 10-fold cross-validation because of the small lexicon size. All other corpora were evaluated on a simple testset that was generated from taking from the full set every tenth word. The training corpus contained the remaining 90% of the data. In their 2005 paper, Marchand and Damper reported that a significant performance gain could be achieved when the graphemic input was annotated with "perfect" syllable boundaries. I tried the same for the HMM and obtained a corresponding improvement rate. The NetTalk set contains manually generated alignment information for the whole set of 20,008 words. On the Pascal Pronalsyn webpage[2], Damper also provided the Teacher's Word Book data set containing 16,280 words. The beep corpus and the French data were aligned automatically with the EM-algorithm presented in Damper *et al.* [2004].

corpus	HMM-Schm	HMM-KN	PbA	Chen	AWT
E - Nettalk (part. set)			65.5%	67.9%	
E - Nettalk	59.5%	64.6%		65.4%	
E - Nettalk (+syll)	65%	70.6%	71.7%		
E - Teacher's WB		71.5%	71.8%		
E - beep	84.5%	85.7%	86.7%		
E - CELEX		76.3%			68.3%
French - Brulex	84.9%	88.4%			

Table 8.5: Results for g2p with HMM compared to latest PbA results.

[2]http://www.pascal-network.org/Challenges/PRONALSYL/Datasets/

8.6 Discussion

The HMM very significantly outperforms the decision tree, in particular when phonotactic constraints concerning the syllable structure and stress are incorporated into the module. For input data that is not morphologically annotated, the Hidden Markov Model reduces the word error rate by almost 50% (from 26.6% to 13.7%).

When morphological information from the best-performing automatic morphology (ETI) is used to annotate the input, only very slight improvements can be reached with the Hidden Markov Model, as it captures the exact context very well any way, but still leading to an improvement of 36% with respect to the AddWordTree (from 21.5% to 13.6%). These results indicate that a morphological component is not necessary even for a language that is very sensitive to the effects of morphological boundaries, such as German, if a Hidden Markov Model is used instead of a decision tree.

I strongly recommend the reimplementation (for efficiency reasons, as the current demonstration module implemented in the scope of this work is in Perl) of the Hidden Markov Model (or a similar model that has comparable effects, such as PbA) for use in the IBM TTS system.

Chapter 9

Summary and Future Work

9.1 Conclusion

The present thesis analysed grapheme-to-phoneme conversion for a German TTS system. The problems of morphological preprocessing, syllabification, stress assignment and letter-to-phoneme conversion were considered. An error reduction by 50% with respect to the AddWord Tree module were reached by using a Hidden Markov Model that integrates phonological constraints. The HMM was also shown to generalize to other languages and compares well to other state-of-the art approaches. I strongly recommend to substitute the AddWordTree by a joint n-gram model in the IBM TTS system.

9.1.1 Morphology for German TTS

Morphological information is important for German grapheme-to-phoneme conversion because it determines the pronunciation of certain letters, has a strong influence on syllabification and is crucial for stress assignment in morphologically complex words. It is however not necessary to represent morphological boundaries explicitly in all architectures. If a model which captures word structure well and has a good memory for letter sequences is used in grapheme-to-phoneme conversion, the relevant information can to a large proportion be captured implicitly. A morphological preprocessing component then only leads to very slight improvements that are perhaps not worth the effort of integrating a separate morphological component.

Good morphological analysers are hard to acquire. Best results on the letter-to-phoneme conversion task were obtained with the rule-base ETI morphology. Using the computational morphology SMOR did not lead to an improvement because the lexicon IMSLex contains complex lexicon entries, so that many morphological boundaries are not found. To alleviate this problem, reliable morphological boundaries need to be inserted into IMSLex. The disambiguation problem can be solved by using heuristics, or training a weighted FST on disambiguated morphological analyses. Finally, a default morpheme should be introduced to cope with out-of-vocabulary words.

State-of-the-art unsupervised approaches, although in principle attractive because they require no development cost and only raw text as a resource, do not yet reach sufficient quality to improve grapheme-to-phoneme conversion.

9.1.2 Syllabification and Stress

The integration of phonological constraints leads to very significant performance gains for German. The constraints used (requiring exactly one nucleus per syllable and one primary stress per word) are very general and, to my knowledge, language-independent.

Stress is a hard problem in German, as it depends on morphology as well as on syllable weight and position. The data used for data-based learning of word stress is not optimal, as the source used contains many stress errors. A rule-based approach did not perform well on the data set, as it requires perfect morphological information, which is typically not available. Furthermore, the rule-based approach cannot cope with foreign words and even among German words there is quite a large proportion of words which is not covered by rules.

The data-based module for word stress assignment lead to a significant improvement in performance. But results are still far from optimal, which can be understood when considering the fact that the stress module does not have access to the information it needs. In fact, the HMM models the problem rather poorly, as it does not generalize over syllables and has no representation for the higher level stress assignment processes.

The choice of an appropriate smoothing method matters a lot, in particular if we face hard data sparsity problems. A variant of Modified Kneser-Ney Smoothing was found to perform best.

9.1.3 Modularity

Best results were obtained from the Hidden Markov Model component that does syllabification, stress assignment and grapheme-to-phoneme conversion all in one step and integrates phonological constraints for syllabification and word stress. We can conclude from this finding that an integrated approach is superior to a pipeline architecture for strongly inter-dependent tasks.

Modularity is an advantage if the individual components are more specialized to their task than the component that does several tasks in one step, and if the exact scope of knowledge needed is used. The syllabification and stress modules lead to performance improvements when used in a pipeline with the AddWordTree. In a modular system, one component can furthermore be substituted by another one – for example, if a better way of doing stress assignment in German was found, this could be integrated much more easily. If the modular architecture is chosen, best results are obtained when g2p conversion is done before syllabification and stress assignment (refer to table 9.1).

9.2 Suggestions for Future Work

In future work, I suggest to use the EM-algorithm for letter to phoneme alignment. Alignment is needed as a preprocessing step for both the decision tree and the HMM. Automating this step would allow to more easily adapt the letter-to-phoneme conversion module to other languages.

All of the algorithms evaluated in this thesis were implemented in Perl – for serious use, these should be reimplemented in e.g. C, to improve time efficiency.

	modular		one-step	
	Preproc.	Postproc.	constr.	no constr.
no morph	83.4%	84.8%	**86.3%**	78.5%
AWT no morph	78.9%			**73.4%**
ETI morph			86.4%	
AWT ETI morph				**78.2%**
CELEX morph	83.9%	85.6%	**86.7%**	74.7%
AWT CELEX morph	84.3%	84.1%		**78.4%**

Table 9.1: Comparing the effects of preprocessing and postprocessing vs. an integral architecture.

Although morphological information did not lead to any improvements in g2p conversion, I think that the approach taken by Keshava and Pitler is very promising. Substituting the incorrect assumption that a word whose affix has been pealed off is again a word by an account of what a word stem is can be expected to lead to significant performance gains.

Finally, it would be good to find a better way for coding morphological information. Currently, low quality morphological information has a negative effect on grapheme-to-phoneme conversion quality, because it is directly encoded into the letter string and therefore causes the algorithm to have access to a smaller letter context within its context window.

Bibliography

O. Andersen and P. Dalsgaard. Multi-lingual testing of a self-learning approach to phonemic transcription of orthography. In *EUROSPEECH95*, pages 1117–1120, 1995.

Kenneth R. Beesley. Finite-state morphological analysis and generation of Arabic at Xerox research: Status and plans in 2001.

J.R. Bellegarda. Unsupervised, language-independent grapheme-to-phoneme conversion by latent analogy. In *Spoken Language Group, Apple Comput. Inc.*, pages 244–247, 2003.

Delphine Bernhard. Unsupervised morphological segmentation based on segment predictability and word segments alignment. In *Proceedings of 2nd Pascal Challenges Workshop*, pages 19–24, Venice, Italy, 2006.

M. Bisani and H. Ney. Investigations on joint multigram models for grapheme-to-phoneme conversion. In *Proceedings of the 7th International Conference on Spoken Language Processing*, pages 105–108, 2002.

Alan W Black, Kevin Lenzo, and Vincent Pagel. Issues in building general letter to sound rules. In *Third ESCA on Speech Synthesis*, 1998.

Stefan Bordag. Unsupervised knowledge-free morpheme boundary detection. In *Proceedings of RANLP 05*, 2005.

Stefan Bordag. Two-step approach to unsupervised morpheme segmentation. In *Proceedings of 2nd Pascal Challenges Workshop*, pages 25–29, Venice, Italy, 2006.

A. van den Bosch and W. Daelemans. Data oriented methods for grapheme-to-phoneme conversion. In *Proceedings of European Chapter of ACL*, pages 45–53, Utrecht, 1993.

G. Bouma. A finite-state and data-oriented method for grapheme to phoneme conversion. In *Proceedings of the first conference of the North-American Chapter of the Association for Computational Linguistics*, pages 303–310, Seattle, 2000.

K.-U. Carstensen, C. Ebert, C. Endriss, S. Jekat, R. Klabunde, and H. Langer. *Computerlinguistik und Sprachtechnologie*. Spektrum Akademischer Verlag, 2001.

D. Caseiro, I. Trancoso, L. Oliveira, and C. Viana. Grapheme-to-phone using finite-state transducers. In *Proceedings of the IEEE 2002 Workshop on Speech Synthesis*, Santa Monica, California USA, 2002.

Center for Lexical Information. Max-Planck-Institut for Psycholinguistics, Nijmegen. *CELEX German Linguistic User Guide*, 1995.

Stanley F. Chen. Conditional and joint models for grapheme-to-phoneme conversion. In *Eurospeech*, 2003.

Mathias Creutz and Krista Lagus. Induction of a simple morphology for highly-inflecting languages. In *Proceedings of the 7th Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON)*, pages 43–51, 2004.

Mathias Creutz and Krista Lagus. Unsupervised morpheme segmentation and morphology induction from text corpora using morfessor 1.0. In *Publications in Computer and Information Science, Report A81, Helsinki University of Technology*, 2005.

Mathias Creutz and Krista Lagus. Unsupervised models for morpheme segmentation and morphology learning. In *ACM Transaction on Speech and Language Processing*, 2006.

Mathias Creutz. Unsupervised segmentation of words using prior distributions of morph length and frequency. In *Proceedings of ACL-03, the 41st Annual Meeting of the Association of Computational Linguistics*, pages 280–287, 2003.

W. Daelemans and A. van den Bosch. Generalization performance of backpropagation learning on a syllabification task. In *Proceedings of the 3rd Twente Workshop on Language Technology. Enschede*, pages 27–37, 1992.

Walter M. P. Daelemans and Antal P. J. van den Bosch. Language-independent data-oriented grapheme-to-phoneme conversion. In Jan P. H. van Santen, Richard W. Sproat, Joseph P. Olive, and Julia Hirschberg, editors, *Progress in Speech Synthesis*, pages 77–89. Springer, New York, 1997.

Robert I. Damper, Yannick Marchand, M.J.Adamson, and K. Gustafson. Evaluating the pronunciation component of text-to-speech systems for English: A performance comparison of different approaches. In *Computer Speech and Language*, volume 13, pages 155–176, 1999.

Robert I. Damper, Craig Z. Stanbridge, and Yannick Marchand. A pronunciation-by-analogy module for the festival text-to-speech synthesiser. In *Proceedings of 4th ISCA Workshop on Speech Synthesis*, pages 97–102, Pitlochry, Scotland, 2002.

Robert I. Damper, Yannick Marchand, John-David Marseters, and Alex Bazin. Aligning letters and phonemes for speech synthesis. In *Fifth ISCA ITRW on Speech Synthesis*, 2004.

M. Dedina and H. Nusbaum. Pronounce: a program for pronunciation by analogy. In *Computer Speech and Language 5*, pages 55–64, 1996.

Christopher Dege. *Integration prosodischer Information in die Phonetisierung mit Entscheidungsbäumen*. Diplomarbeit, Universität des Saarlandes, 2003.

H. Déjean. Morphemes as necessary concepts for structures: Discovery from untagged corpora. In *Workshop on paradigms and Grounding in Natural Language Learning*, pages 295–299, Adelaide, Australia, 1998.

H. Elovitz, R. Johnson, A. McHugh, and J. Shore. Letter-to-sound rules for automatic translation of English text to phonetics. In *Acoustics, Speech, and Signal Processing*, 1976.

Alexander Fraser and Daniel Marcu. Measuring word alignment quality for statistical machine translation. In *Technical Report ISI-TR-616, ISI/University of Southern California*, 2006.

Lucian Galescu and James Allen. Bi-directional conversion between graphemes and phonemes using a joint n-gram model. In *Proceedings of the 4th ISCA Tutorial and Research Workshop on Speech Synthesis*, 2001.

John Goldsmith. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198, 2001.

Francois Guérillon. French ETI phonemes with ADDWORD. In *IBM CONFIDENTIAL*, June 2002.

Margaret A. Hafer and Stephen F. Weiss. Word segmentation by letter successor varieties. In *Information Storage and Retrieval 10*, pages 371–385, 1974.

Zellig Harris. From phoneme to morpheme. In *Language 31*, pages 190–222, 1955.

Christian Jacquemin. Guessing morphology from terms and corpora. In *Research and Development in Information Retrieval*, pages 156–165, 1997.

Stefanie Jannedy and Bernd Möbius. Name pronunciation in German text-to-speech synthesis. In *5th Conference on Applied Natural Language Processing, ACL*, pages 49–56, Washington, DC, 1997.

Michael Jessen. *Word Prosodic Systems in the Languages of Europe*. Mouton de Gruyter: Berlin, 1998.

Lauri Karttunen. Applications of finite-state transducers in natural language processing. *Lecture Notes in Computer Science*, 2088:34–46, 2001.

Samarth Keshava and Emily Pitler. A simpler, intuitive approach to morpheme induction. In *Proceedings of 2nd Pascal Challenges Workshop*, pages 31–35, Venice, Italy, 2006.

Estate Khmaladze. The statistical analysis of large number of rare events. In *Technical Report MS-R8804, Dept. of Mathematical Statistics*, 1987.

Anne K. Kienappel and Reinhard Kneser. Designing very compact decision trees for grapheme-to-phoneme transcription. In *Eurospeech*, Scandinavia, 2001.

George A. Kiraz and Bernd Möbius. Multilingual syllabification using weighted finite-state transducers. In *Proceedings of the Third International Workshop on Speech Synthesis*, pages 71–76, Jenolan Caves, Australia, 1998.

B. T. Lowerre and R. Reddy. The harpy speech understanding system. In *Trends in Speech Recognition, Prentice- Hall, Englewood Cliffs, NJ*, pages 340–360, 1980.

J. Lucassen and R. Mercer. An information theoretic approach to the automatic determination of phonemic baseforms. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP 9*, pages 304– 307, 1984.

Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

Yannick Marchand and Robert I. Damper. Can syllabification improve pronunciation by analogy of English? *Natural Language Engineering*, 2005.

Wolfgang Minker. Grapheme-to-phoneme conversion - an approach based on hidden markov models. 1996.

Bernd Möbius. Word and syllable models for German text-to-speech synthesis. In *Proceedings of the Third International Workshop on Speech Synthesis*, pages 59–64, Jenolan Caves, Australia, 1998.

Bernd Möbius. *German and Multilingual Speech Synthesis*. phonetic AIMS, Arbeitspapiere des Instituts für Maschinelle Spachverarbeitung, 2001.

Karin Müller. Automatic detection of syllable boundaries combining the advantages of treebank and bracketed corpora training. In *Proceedings of ACL*, pages 402–409, 2001.

Sylvain Neuvel and Sean Fulop. Unsupervised learning of morphology without morphemes. In *Proceedings of the Workshop on Morphological and Phonological Learning, ACL Publications*, 2002.

Vincent Pagel, Alan W Black, and Kevin Lenzo. Letter to sound rules for accented lexicon compression. In *ICSLP98, Sydney, Australia*, 1998.

Amanda Pounder and Markus Kommenda. Morphological analysis for a German text-to-speech system. In *COLING 1986*, pages 263–268, 1986.

J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

P.A. Rentzepopoulos and G.K. Kokkinakis. Phoneme to grapheme conversion using hmm. In *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, pages 797–800, 1991.

Jenny R. Saffran, Elissa L. Newport, and Richard N. Aslin. Word segmentation: The role of distributional cues. *Journal of Memory and Language*, 35:606–621, 1996.

Helmut Schmid, Arne Fitschen, and Ulrich Heid. SMOR: A German computational morphology covering derivation, composition and inflection. In *Proceedings of LREC*, 2004.

Helmut Schmid, Bernd Möbius, and Julia Weidenkaff. Tagging syllable boundaries with hidden Markov models. IMS, unpublished, 2005.

Patrick Schone and Daniel Jurafsky. Knowledge-free induction of morphology using latent semantic analysis. In *Proceedings of the Computational Natural Language Learning Conference*, pages 67–72, Lisbon, 2000.

Patrick Schone and Daniel Jurafsky. Knowledge-free induction of inflectional morphologies. In *Proceedings of the North American chapter of the Association for Computational Linguistics (NAACL-2001)*, 2001.

Uwe Schöning. *Algorithmik*. Spektrum Akademischer Verlag, 2001.

T. J. Sejnowski and C. R. Rosenberg. Parallel networks that learn to pronouce English text. In *Complex Systems 1*, pages 145–168, 1987.

J.A. Sonquist and J.N. Morgan. The detection of interaction effects. In *Survey Research Center, Institute for Social Research, University of Michigan, Ann Arbor*, 1964.

R. Sproat. Multilingual text analysis for text-to-speech synthesis. In *Proc. ICSLP '96*, volume 3, pages 1365–1368, Philadelphia, PA, 1996.

Janne Suontausta and Juha Häkkinen. Decision tree based text-to-phoneme mapping for speech recognition. In *ICSLP-2000, vol.2*, pages 831–834, 2000.

Paul Taylor. Hidden Markov models for grapheme to phoneme conversion. In *INTERSPEECH*, pages 1973–1976, Lisbon, Portugal, 2005.

Isabel Trancoso and Diamantino Antònio Caseiro. Spoken language processing using weighted finite state transducers. *SWIM'2004 - Special Workshop in Maui: Lectures by Masters in Speech Processing*, January 2004.

I. Trancoso, D. Caseiro, C. Viana, F. Silva, and I. Mascarenhas. Pronunciation modeling using finite state transducers. In *Proc. ICPhS'2003 - 15th International Congress of Phonetic Sciences*, Barcelona, Spain, 2003.

A. van den Bosch, T. Weijters, and W. Daelemans. Modularity in inductive-learned word pronunciation systems. In *Proceedings NeMLaP3/CoNNL98*, page 185194, Sydney, 1998.

Petra Wagner. Systematische Überprüfung deutscher Wortbetonungsregeln. In *Elektronische Sprachsignalverarbeitung aus Studientexte zur Sprachkommunikation*, number 22, pages 329–338, 2001.

Petra Wagner. Improving automatic prediction of German lexical stress. In *Proceedings of the 15th ICPhS*, pages 2069–2072, Barcelona, Spain, 2003.

David Yarowsky. *Progress in Speech Synthesis*. New York: Springer, 1996.