Universität Karlsruhe (TH)
Research University · founded 1825

Fakultät für **Informatik**

# Statistical Methods for Automatic Diacritization of Arabic Text

Diplomarbeit am Lehrstuhl Prof. Waibel

Institut für Theoretische Informatik
Fakultät für Informatik
Universität Karlsruhe (TH)

Language Technologies Institute (LTI)
Carnegie Mellon University
Pittsburgh, USA

von

**Tim Schlippe**

Advisors:
Prof. Dr. Alex Waibel
Dr. Stephan Vogel

May 2008

interACT

lti

Carnegie Mellon

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, den 31. Mai 2008

[illegible]

Tim Schlippe

# Acknowledgements

I would like to thank cordially all the people who supported me with this thesis.

First of all, I express my gratitude to my advisor Dr. Stephan Vogel for his assistance and his inspirations.

I thank him and Prof. Waibel for making my two stays at Carnegie Mellon University possible.

Many thanks go to everyone at the machine translation group at Carnegie Mellon University and Universität Karlsruhe (TH) for interesting discussions and suggestions.

Furthermore, I would like to thank the people at AppTek for their collaboration.

Finally, I thank my family and my friends who supported me and had patience with me during the whole course of my studies.

# Abstract

In this thesis we present several statistical methods that automatically restore diacritics in Arabic text documents.

Modern Arabic texts are normally composed of scripts without diacritic marks. So the same word with different diacritizations may appear identical and have different semantics. This leads to an ambiguity when processing data for text-to-speech and speech-to-text applications. A reduction of the ambiguity with the help of diacritization in a text document may benefit further language processing tasks.

Basically, we regard the automatic diacritization as a statistical machine translation problem. With the help of this generative model, we achieve a word-based and a character-based translation from non-diacritized text to diacritized text. A method to integrate linguistic rules into our statistical approaches is suggested. We post-edit the output of a rule-based diacritizer by our statistical machine translation system.

To integrate global features like part-of-speech information, we also solve the diacritization problem as a labeling sequence problem. We tag each consonant with empty characters or diacritics which are inserted into the text after the consonants. The model is an instance of the Conditional Random Fields model [LMP01] which is a discriminative model.

# Zusammenfassung

In dieser Arbeit präsentieren wir verschiedene statistische Methoden, diakritische Zeichen maschinell in arabische Textdokumente einzufügen (Diacritization).

Moderne arabische Texte werden normalerweise mit Schriftzeichen ohne diakritische Zeichen dargestellt. Deshalb kann ein Wort in dieser Darstellung mehrere Bedeutungen besitzen. Dies führt zu einer Mehrdeutigkeit bei der Datenverarbeitung für Text-to-Speech und Speech-to-Text Anwendungen. Eine Reduzierung der Mehrdeutigkeit mit Hilfe maschinellen Einfügens der diakritischen Zeichen in Textdokumente würde eine weitere Sprachverarbeitung unterstützen.

Im Wesentlichen betrachten wir das Einfügen von diakritischen Zeichen als ein Problem in der statistischen maschinellen Sprachübersetzung. Mit Hilfe dieses generativen Models erreichen wir eine Übersetzung auf Wort- und Buchstaben-Ebene von Text ohne diakritische Zeichen in Text mit diakritischen Zeichen. Eine Methode wird vorgestellt, wie linguistische Regeln in unsere statistischen Ansätze integriert werden können. Die Ausgabe eines regelbasierten Systems wird dabei mit Hilfe unseres statistischen Systems nachbearbeitet.

Um globale Features wie Part-of-Speech Informationen zu integrieren, lösen wir das Einfügen von diakritischen Zeichen als ein Labeling Sequence Problem. Wir kennzeichnen jeden Konsonanten mit einem Auslassungszeichen oder einem diakritischen Zeichen und fügen die Zeichen jeweils nach den Konsonanten ein. Das Model ist eine Instanz des Conditional Random Fields Models [LMP01], einem diskriminativen Model.

# Contents

# List of Figures

# List of Tables

# 1. Introduction

Diacritization, also known as vocalization or vowelling, means the restoration of optional orthographical symbols for the vowelization. Especially in the Arabic language these symbols, called diacritics, do not appear in most written texts. Beside for people who are not able to understand text without diacritics such as learners of the Arabic language or sufferers of Dyslexia, an automatic diacritization of Arabic text is helpful for text-to-speech and speech-to-text applications as well as for translations from Arabic into another language and back.

Our focus is on statistical methods. We solve the diacritization problem as a statistical translation task. But once these concepts reach their limits, we integrate part-of-speech tags with the aid of conditional random fields and solve the diacritization problem as a labeling sequence task.

The statistical machine translation belongs to the generative models. The conditional random fields model is a discriminative model. A generative model is a full probability model of all variables. A discriminative model, however, provides a model exclusively of the target variables conditional on the observed variables. We use the generative model with the statistical machine translation systems to generate values of any variable in the model. To define dependencies between the non-diacritized input sequence, features like part-of-speech tags and the diacritized output, we decided to test a discriminative model as well. In contrast to the generative model, the discriminative model allows only sampling of the target variables conditional on the observed quantities. Thus we soften the independence conditions which we have by the SMT[1] system and gain well-defined dependencies by the conditional random fields.

Furthermore, we present a method to integrate rules into statistical approaches by post-editing the output of a rule-based diacritizer with our statistical translation system.

In order to have a starting point to implement and compare statistical features for the automatic diacritization, two SMT baseline systems are established with the

[1]SMT = statistical machine translation

components of the CMU[2] SMT system. One system operates on word level, the other one makes the decision which diacritic to choose on character level.

A hybrid system should combine the advantages of both the word- and the character-based approach. Two hybrid systems are proposed. The first one, consisting of a language model and a phrase table including characters in context with the next whole word, is easy to implement but does not lead to significantly better results. The second one takes more time to be established and to decode, but considerably enhances the results in comparison to the baseline systems.

As this second better hybrid system exclusively includes a character language model, we integrate an additional language model on word level, in order to gain a wider context in the language model.

By the means of the Moses package we improve the translation model of our SMT system by integrating additional lexical scores to the available conditional probabilities in the phrase table.

Since the word stems are conspicuously better restored than the word endings, we show experiments with modifications on the phrase table to reduce the errors at the word endings. Thereby we adapt the generation of specific diacritics at the word endings to the probability distribution of the development data by additional phrase table scores.

Furthermore, we observe improvements by post-editing the output of a rule-based diacritizer with statistical methods.

In the conditional random fields approach we carry out tests with feature functions that integrate part-of-speech tags and information about words in the context.

Our experiments are conducted on the LDC's Arabic Treebank of diacritized news stories and on data provided by AppTek[3], a company which develops machine translation systems for several language pairs.

## 1.1 Motivation

Modern Arabic texts are normally composed of scripts without diacritic marks. In a diacritic-less setting, however, different diacritic patterns may appear identical. This leads to a considerable ambiguity which is difficult to deal with when processing data for text-to-speech and speech-to-text applications such as speech recognition and speech synthesis [ZSS06][Zak06].

A reduction of the ambiguity with the help of diacritization in a text document might benefit a further processing. Due to the increasing word count, a restoration of all diacritics before the translation from Arabic into English by a statistical machine translation system not necessarily performs better. Yet, the diacritization of the passivization diacritic "damma" and the germination diacritic "shadda" may lead to a little bit better results. In [DGH07] an improvement from 0.4389 to 0.4416 BLEU scores [PRWZ02] in Arabic-English statistical machine translation by the diacritization of the passivization diacritic damma is reported.

[2]CMU = Carnegie Mellon University

[3]AppTek - Application Technology Inc., U.S. company, headquartered in McLean, Virginia

Since diacritics at the word endings mark the cases in Arabic, an output with diacritics after the translation from a language into Arabic is easier to understand even if the word order is wrong [Gha06].

Furthermore, a diacritization system would benefit non-native speakers, sufferers of Dyslexia, a specific learning disability which manifests primarily in having difficulties with written language [Elb04], as well as assist in restoring diacritics of Arabic children's and poetry books.

Beside the Arabic we find other languages such as Romanian, French [TC99] and Hebrew [Gal02] which contain diacritical characters whose lack may create ambiguity as well. When exporting from a diacritic-aware text editor into a non-diacritic-aware text format it may happen that diacritics are removed or substituted by other characters. After training our diacritization system with a parallel corpus of one of these languages it would be able to restore the diacritics in the appropriate language as we use language independent statistical approaches.

In the next section we describe the linguistic aspects of the Arabic language and the diacritics.

## 1.2 Arabic Language and the Use of Diacritics

Particularly the translation from Arabic into another language like English is a challenge due to syntactical alternatives like missing diacritics and glottal stop, idioms, proper names and nicknames, metonymy as well as multiple different meanings [Zak06].

The Arabic script is written, read and encoded from right to left. Many Arabic letters change their appearance depending on their position in a word. The Arabic alphabet is a set of 28 consonant letters, shown in Figure 1.1.

ي، و، هـ، ن ، م، ل، ك، ق، ف، غ، ع، ظ ط، ض، ص، ش، س، ز، ر، ذ، د، خ، ح، ج، ث ، ت، ب، أ

Figure 1.1: The Arabic Alphabet.

In our systems the Arabic letters are represented and processed in Buckwalter Transliteration, as described in Section 4.3.

Arabic diacritics are below and above each character within a word. These are marks for vowelization and usually absent, except in religious books like the Qur'an and textbooks for children and foreign learners. Only the consonant doubling "shadda" still appears in several modern Arabic scripts. By considering the context of the situation and the position of the words in a sentence, native speakers distinguish the right pronunciation and the correct words without diacritic marks in an automatic way. Their innate knowledge of grammar and lexicon of Arabic enable them to accurately vocalize any words in written texts based on the context.

A non-diacritized word without context may have many meanings since several diacritic marks are possible. A study shows that there is in fact an average of 11.6

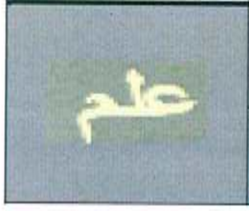
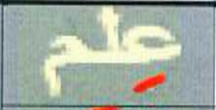
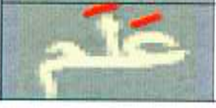
| without diacritics | with diacritics | meaning | pronunciation |
|---|---|---|---|
|  |  | science, learning | Eilm |
| |  | flag | Ealam |

Figure 1.2: Ambiguity in Arabic.

diacritizations for every non-diacritized word when analyzing a text of 23 000 script forms [DAS02].

For example the bare form "Elm" may have different lexical senses (figure 1.2): Diacritized as "Eilm" the meaning is "science" or "learning", diacritized as "Ealam" it stands for "flag". Furthermore, ambiguity may occur at the grammatical level where diacritics at the word endings are correlated with case and verbal information [MBK06].

Arabic diacritics can be split into short vowels pronounced as /a/ (fatha), /u/ (damma) and /i/ (kasra), double case endings pronounced as /an/ (fathatayn), /un/ (dammatayn) and /in/ (kasratayn), as well as syllabification marks. Double case endings are vowels used at the end of a word to distinguish cases. The syllabification mark "shadda" which is sometimes inserted in written text denotes the doubling of a consonant. The other syllabification mark "sukuun" indicates that a letter does not contain any vowels.

Figure 1.3 shows an overview of the diacritics we intend to restore.

| **Short vowels /a/, /u/, /i/** | | **Double case ending** | | **Syllabi-fication marks** | |
|---|---|---|---|---|---|
| Fatha | ◌َ | fathatayn | ◌ً | | |
| damma | ◌ُ | dammatayn | ◌ٌ | shadda | ◌ّ |
| kasra | ◌ِ | kasratayn | ◌ٍ | sukuun | ◌ْ |

Figure 1.3: Arabic Diacritics.

## 1.3 Objectives

The goal was to analyze statistical features for the automatic diacritization of Arabic text. We evaluate and compare these features with the aid of word error rates and diacritization error rates. These error rates are computed after the diacritization of a test set. We present all error rates in %.

In addition to preferably low error rates our methods are supposed to be convenient. For practical application, systems with a small disk space and a short running time are preferred.

The features that we analyze should restore the short vowels (fatha, damma, kasra), the germination mark "shadda", the doubled case endings (fathatayn, dammatayn, kasratayn) as well as the "sukun" that indicates a vowel absence.

In addition to the SMT system developed by the Carnegie Mellon University and the Universität Karlsruhe (TH) [VZH$^+$03] and any associated scripts, our experiments are conducted with tools that are well-established in the field of statistical machine translation such as the Suffix Array Language Model Toolkit, the SRI Language Model Toolkit, Moses, the Stanford Parser and CRF++.

## 1.4 Outline

So far we have disclosed our motivation, the particularities of the Arabic language in order to illustrate the issue and our objectives in Chapter 1.

In Chapter 2 we show how other researchers face the challenge of the restoration of Arabic diacritics.

We suggest two main approaches in this thesis. The ideas of our approaches are described in Chapter 3. First we regard the diacritization as a translation problem. We show statistical machine translation approaches and the integration of rules by post-editing the output of a rule-based system with the help of statistical methods. Secondly conditional random fields are utilized for the diacritization to integrate global features such as part-speech-tags.

In Chapter 4 we explain which data sources we make use of to establish, tune and test our systems. The data are represented by the Buckwalter Transliteration [Buc02] to process them morphologically by our systems. Besides, it is explained which evaluation measures are more reasonable than others for diacritization systems.

Chapter 5 gives an overview over the tools that support us to conduct the experiments. We apply the CMU SMT system for the translation approaches. Furthermore, tools such as the Suffix Array Language Model Toolkit, the SRI Language Model Toolkit, AppTek's rule-based Arabic Diacritizer, the Stanford Part-of-Speech Tagger and CRF++ are described. Moses is presented as well which provides a phrase table with the aid of the GIZA++ Alignment Toolkit [Mos06].

In Chapter 6 our implementations and experiments are delineated. The report starts with the statistical machine translation approach and the creation of the word level and the character level baseline systems. We enlarge the context of the character level system and continue with a system on both levels as well as the extension with an additional word language model. The integration of a phrase table with additional lexical scores besides the conditional translation probabilities leads to a better choice of the correct target phrase [Koe06]. Experiments on integrating the probability distribution of vowelized and real characters reduce the error rates at the word endings. We depict our post-editing of the output of AppTek's rule-based diacritication system. Finally we show how we create a diacritizer that is based on conditional random fields. We close Chapter 6 with the results, running times and the sizes of our systems as well as with the analysis of residual errors.

We conclude our work in Chapter 7 and point at further approaches based on our results which might lead to enhancements in the field of diacritization.

# 2. State of the Art

For the diacritization of Arabic text different methods such as rule-based, example-based, hierarchical, morphological and contextual-based methods as well as methods with Hidden Markov Models and weighted finite state machines have been applied.

Recent work shows that there are different approaches where the diacritization is treated as a machine translation problem.

In 1989 El-Sadany and Hashish [ESH89] and in 2003 El-Imam [EI03] proposed rule-based methods for the diacritic restoration. One drawback of these systems is their difficulty in maintaining the rules up-to-date and extending them to other Arabic dialects.

In 2004 Eman and Fisher suggested an example-based hierarchical top-down approach which works like this [EF04]: For a sentence in the test set the training data was searched for a matching sentence. If it was found, the whole sentence was used, if not, matching phrases were searched. In case of no matching phrases character n-grams were restored by diacritics.

In several approaches the diacritization is treated as a labeling sequence problem.

In 2004 Vergyri and Kirchhoff treated the diacritization as a tagging problem where each word was tagged as one of many possible forms provided by the Buckwalter's Morphological Analyser [VK04]. In order to learn the tag sequence the Expectation Maximization algorithm was applied. In that case morphological and contextual information with an acoustic signal was combined. Vergyri and Kirchhoff reported on a WER[1] of 27.3% and a DER[2] of 11.54%.

Nelken and Schieber restored diacritics by an algorithm which was based on weighted finite state machines in 2005 [NS05]. Like our systems their system was trained on LDC's Arabic Treebank Data as well. Words not occurring in the data were substituted by characters and larger morphological units. Altogether they gained a WER of 23.61% and a DER of 12.79%. Without regarding the final vowelization of

[1]WER = Word Error Rate

[2]DER = Diacritization Error Rate

the words which represents the case endings, a WER of 7.33% and a DER of 6.35% were achieved.

Zitouni, Sorensen and Sarikaya proposed a maximum entropy-based approach at IBM in 2006 [ZSS06]. Their system worked with lexical, segmented-based and part-of-speech tag features. The evaluation resulted in a WER of 18.0% and a DER of 5.5% with the final vowelization as well as a WER of 7.9% and a DER of 2.5% without the final vowelization.

In 2006 an approach was unveiled at the Saudi 18th National Computer Conference where the word sequence of undiacritized Arabic text was considered as an observation sequence from a Hidden Markov Model [EAMA06]. The hidden states were the possible diacritized expressions of the words. Finally the optimal sequence of diacritized words or states was obtained using the Viterbi Algorithm. For a test set of 995 words a DER of 4.1% was reported for their system trained on the Qur'an text. A further reduction to about 2.5% was reached by using a preprocessing stage and trigrams for a selected number of short words and the most frequent words.

A diacritization system which is based on the combination of a tagger and a lexeme language model was suggested by Habash and Rambow in 2007 [HR07]. Analogous to [VK04] their system used the Buckwalter Arabic Morphological Analyzer. The remaining WER is 14.9% and the DER 4.8%. Ignoring the last diacritic led to a WER of 5.5% and a DER of 2.2%.

Analogous to the mentioned previous approaches we use LDC's Arabic Treebank data, part-of-speech tags and rule-based methods. To process the Arabic text morphologically it is represented in Buckwalter Transliteration, too. In comparison to the previous approaches our experiments are conducted on word level, on character level and on both levels. We work with statistical machine translation techniques. Furthermore, we regard the diacritization as a labeling sequence problem by tagging each consonant with empty characters or diacritics which are inserted into the text after the consonants. The model is an instance of the Conditional Random Fields model [LMP01].

# 3. Automatic Diacritization

First we regard the diacritization problem as a simplified phrase-based translation task. Since the CMU SMT system can perform a phrase-based translation we use it as a tool for our experiments.

In cooperation with AppTek we merge a rule-based approach with statistical methods by post-editing the output of AppTek's rule-based diacritizer.

To integrate global features for the diacritization we determine our diacritized output sequence by the help of conditional random fields. In that approach we are able to create dependencies between a non-diacritized input sequence, global features such as part-of-speech tags and the output sequence by feature functions.

In the following sections we illuminate our intentions to realize the diacritization as a translation process, describe how we earn benefits from the output of a rule-based system and which global features we select for the conditional random fields approach.

## 3.1 Diacritization as Translation

The challenge we face is the restoration from non-vowelized strings like characters, words and phrases to vowelized characters, words or phrases. These assignments may be regarded as a simplified translation process. In this simplified translation process the non-vowelized string may be regarded as the source language and the vowelized string as the target language. We say "simplified translation" since in contrast to the translation between different languages neither a word reordering nor a change in the number of words is necessary. It is an n-to-n mapping from non-vowelized words to vowelized words and from non-vowelized characters to vowelized characters respectively.

There are different approaches to machine translation [KK03] [ABM+93] [Hut03]. On the one hand there are translation systems based on grammar like interlingua- and transfer-based, on the other hand there are direct approaches such as example-based and statistical. These approaches are illustrated with the Vauquois triangle shown in Figure 3.1.

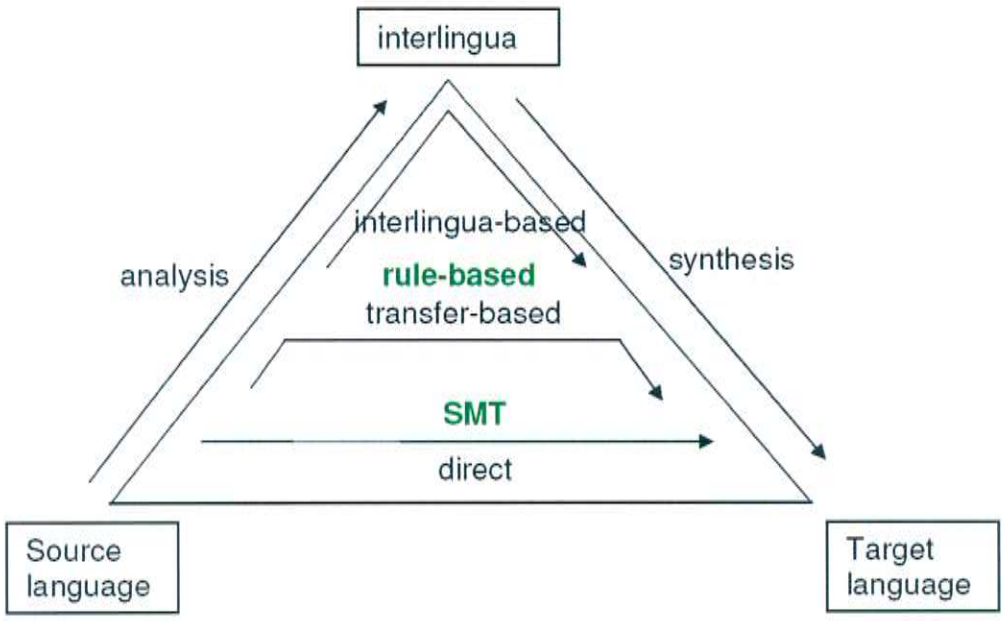


Figure 3.1: Vauquois Triangle.

For the translation from non-diacritized text to diacritized text we use the CMU machine translation system as a translation tool which is a statistical machine translation system [VZH+03].

We create a phrase table with non-diacritized entries on the source side and diacritized entries on the target side and build a language model with diacritized text.

With the appropriate representation of the non-diacritized and diacritized text we are able to operate on character level, on word level as well as on both levels by the help of the CMU SMT system.

By post-editing the output of the rule-based machine translation system of the company AppTek, we do not remain on the direct strategy that our SMT approach contains. A rule-based strategy includes transfer-based and interlingual machine translation paradigms.

In the next sections we illuminate the translation process and go on with the language and translation model that are basic components of our CMU SMT system.

### 3.1.1 Statistical Machine Translation

In this section we give an introduction into the statistical machine translation as we derive benefits from it by applying the CMU SMT system for our translation approach to face the challenge of the diacritization.

The basic idea of statistical machine translation is to automatically analyse existing human sentences or phrases, with an eye towards building translation rules [Kni99]. Subsequently these rules are used to translate new texts automatically.

The automatic translation is called decoding. When decoding a sentence in the source language, the system is seeking the most likely translation in the target language.

Using Bayes Rule, the probability for the most likely translation is computed in the following way:

$$\arg\max_{e} P(e|f) = \arg\max_{e} P(e) * P(f|e)$$

The language model $P(e)$ is the probability that the phrase $e$ would occur in the target language. The translation model $P(f|e)$ is the probability that, if $e$ occurs, it could be translated into $f$. The sentence $e$ is searched that maximizes the product of these terms.

The a priori probability $P(e)$ is computed from a monolingual corpus and the conditional probability $P(f|e)$ from a bilingual one.

Particular data structures as well as heuristic search algorithms are used during the decoding phase. In the memory the phrase table is represented as a prefix tree with nodes emitting to the multiple translations and the text to translate as a lattice. A run over the prefix tree and the lattice is executed. When expanding a sentence, one word is expanded at a time. Once a final state has been reached in the tree, a new lattice edge will be created for each translation. The first or the n-best path is searched through the lattice. Since every lattice has one or multiple scores of e.g. language model, translation model or word count, which could be implemented as costs or bonuses, the lattices are assigned different quality ratings.

Figure 3.2 illustrates the basic components of a statistical machine translation system. The steps from preprocessing the source language text, the decoding with the influence of the translation model and the language model up to the target language text are demonstrated.

The analysis of the existing training sentences and the construction of the statistical models happen during the training phase. The system can be trained by input text in source language and the exact translation in target language. Then words and phrases are automatically aligned within pairs in that parallel corpus. The probabilities are determined by training statistical models for the parallel corpus. Statistical models allow many alternatives called hypotheses to be created. Afterwards a score is given to each hypothesis. Finally the best hypothesis among all is selected.

The speed of an SMT system can be traded with the quality [Och05]. The more alternatives the system applies to the statistical models, the more accurate the result, but the longer it takes for the system to compute.

The statistical approach can be applied on any language pair that has been given enough training data to translate a text by using the statistical models deriving from it. Thus new systems can be realized on low costs. No linguistic knowledge is necessary for building a statistical machine translation system as the system simply has to be fed with a parallel corpus.

In addition to the language model and the translation model, models like the word count or the distortion model are normally applied for the statistical translation. In our case however, the alignment from words or characters without diacritics to words or characters with diacritics is a monotonic mapping. Therefore models that concern the distortion of words or phrases do not have to be established and the Viterbi algorithm used for aligning phrases need not be applied.

In the next sections we dwell on the language model and the translation model.

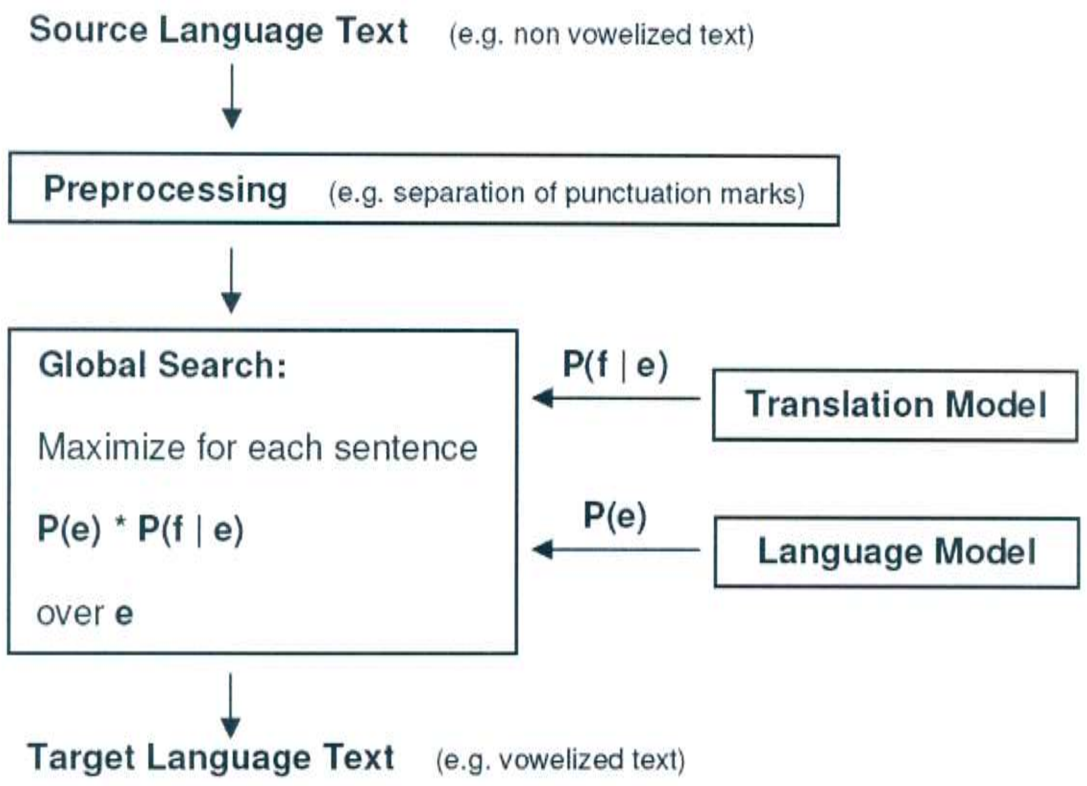

Source Language Text (e.g. non vowelized text)
Preprocessing (e.g. separation of punctuation marks)
Global Search:
Maximize for each sentence
P(e) * P(f | e)
over e
P(f | e)
Translation Model
P(e)
Language Model
Target Language Text (e.g. vowelized text)

Figure 3.2: Components of an SMT system.

### 3.1.2 Translation Model

On the basis of the parallel corpus, probabilities for the translation model can be computed. Each probability $P(f|e)$ of a source language string given a target language string is assigned. This chance is called conditional probability. Roughly speaking, the translation model provides the "fulfilness" [Kni99].

The translation model is realized by a phrase table. This phrase table contains phrases of the source language on the source side, translated phrases in the target language on the target side as well as scores that influence the choice of the phrase pair in the decoding. The scores may derive from the conditional phrase translation probabilites, may be lexical weights or numbers to boost or penalize the choice of the phrase pairs.

### 3.1.3 Language Model

In order to guarantee a certain level of grammatical and syntactical quality, with the a priori probability $P(e)$, a probability is assigned to each target language sentence. Thereby a higher priority is given to expressions which occur more often in the target language and which consequently are more common and specific. Roughly speaking, the language model provides the "fluency" [Kni99].

With the training data we have a huge database containing continuous sequences of words that ever appeared in the target language. Such subsequences of n words are called *n-grams*. Depending on the number of occurrences, this leads to a certain probability for each n-gram in the database. It is important to not assign a probability 0 to an n-gram, as it could be perfect, even if it has never appeared in the database before. For example the much higher frequency of the 3-gram "the Grand Canyon" in contrast to the rare occurrence of "the Great Canyon" is a good indicator that "the Great Canyon" is rather likely to be an incorrect expression. Since the number of calculations and the storage requirement increases rapidly with higher n-grams an appropriate $n$ should be selected depending on the system requirements and the execution time.

Several kinds of word alignment models are used in different systems for the implementation of the language model [BPPM93][VNT96][OTN99][Wu97]. In the first model, called IBM1, the calculation as to how likely the continuous sequences of words are, depends only on lexical probabilities. In addition IBM2, IMB3, IBM4, IBM5, HMM and Bilingual Bracketing are models which consider further information like position and fertilities, for example. Yet, we do not need to work with a model that depends on more than the lexical probabilities in view of the monotonic mapping from non-diacritized words or characters to diacritized words or characters.

Toolkits like the Suffix Array Language Model Toolkit and the SRI Language Model Toolkit provide additional features like for instance the collection of n-gram statistics on the fly or rescoring an n-best list of sentences with additional language model scores.

In the next three sections we demonstrate our trains of thoughts on how to perform the diacritization on word level, characters level and on both levels.

### 3.1.4 Diacritization on Word Level

Due to being represented as words, we are allowed to use the original words as input for our diacritizer.

In contrast to the character representation that is used to process the characters by the CMU SMT system we have no blanks within a word in the representation utilized for the word level. The CMU SMT system treats everything between a blank as a single word or token. For example 5-gram entries in the phrase table include five words in word representation and only five characters in character representation. An enlargement of phrase table and language model on character level up to the same context as on word level would involve considerably more calculations and thereby time complexity.

| | | | |
|---|---|---|---|
| **Extract from the original test set without diacritics** | mwskw | Jf | b |
| **Extract from the original reference with diacritics** | muwsokuw | Jaf | b |

Figure 3.3: Word Level Representation.

Therefore the advantage of dealing with entire words is that we have a wider context than when dealing with characters, both within the phrase table and the language model.

Figure 3.3 illustrates the representation of the Arabic text on word level. The characters are ASCII characters since we decided to process the text in Buckwalter Transliteration. In Section 4.3 we dwell on the Buckwalter Transliteration.

Due to the limited training data there are words in the test set that do not occur in the phrase table and in the language model. Therefore diacritizing on word level results in the drawback that the translation system can recognize unknown words only by inserting the non-vowelized source word into the output sentence. Thus an unknown word leads to a word error.

Due to this drawback we diacritize words on character level as described in the next section.

### 3.1.5 Diacritization on Character Level

As proposed for the restoration of diacritics in [Mih02], the idea is to develop a system that works on character level.

Operating on character level in contrast to the system that deals with words has the advantage that there is no word which is translated by inserting the non-vowelized source word since each consonant character is assigned to the same character with a vowel.

| | |
|---|---|
| **Extract from the original test set without diacritics** | m w s k w space J f space |
| **Extract from the original reference with diacritics** | mu w so ku w space Ja f space |

Figure 3.4: Character Level Representation.

The representation of the Arabic text on character level is shown in Figure 3.4. Each consonant or consonant with appropriate diacritic is represented in Buckwalter Transliteration and separated by a blank. The word endings are marked by the word "space" to allow a conversion to the word level representation.

A drawback is that the context is not as wide as in our word-based system since everything between blank is treated as a single word by our translation systems.

Thus we create a system that diacritizes words on word level as well as characters on character level.

### 3.1.6 Diacritization on both Levels

By creating a diacritizer that functions on both levels we plan to combine the benefits of both the diacritization on word level and on character level. We call the diacritization systems on both levels *hybrid systems*.

The goal is primarily to obtain a character-based system with wider context. Furthermore, a system that allows to diacritize known words on word level and switches to character level for unknown words is considered as a solution to combine the advantages of both approaches.

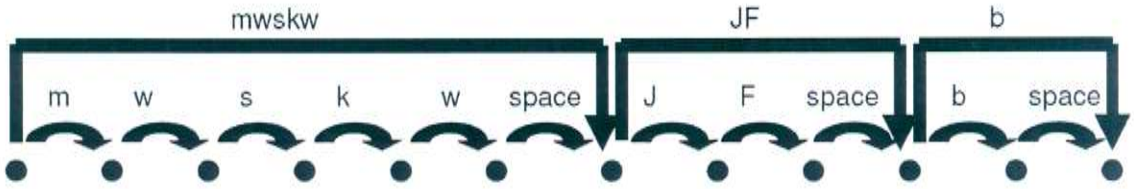


Figure 3.5: Lattice with Edges from Char to Char and from Word to Word

The CMU SMT system provides the opportunity to input the test data as a lattice. The lattice representation allows defining edges from word to word as well as from character to character. So the decoder is able to switch from word level to character level and back depending on the current word - whether it is known or unknown.

An example for the lattice that enables the hybrid approach is illustrated in Figure 3.5. The edges are located so that a path to word level or a path to character level is selectable at the beginning of each word. If the word "mwskw" can be found in the phrase table it is translated as a word. If not, each character is assigned to an appropriate character with a diacritic.

### 3.1.7 Diacritization: Rule-based Translation with Statistical Phrase-based Post-Editing

The approaches we have described yet do not make use of real linguistic rules for the restoration of the diacritics. So we intend to observe the impact of a rule-based approach on a statistical approach.

In 2007 the National Research Council Canada developed a machine translation system referring to an automatic post-editing strategy [SUIK07]. Appropriate training material provided, it was possible to train an SMT system to do automatical corrections of systematic errors made by rule-based systems. Initially the scientists translated the input text into the target language using a rule-based machine translation system. Then they automatically post-edited the output with the help of a statistical phrase-based system. The statistical system considered the output of the rule-based system as the source language, and reference human translations as the target language.

| Source | Target |
|---|---|
| AppTek's output (vow) | vowelized (human) |

Figure 3.6: Phrase Table of the Post-Editing Approach.

The company AppTek owns a rule-based system for their diacritization tasks. They provide us with the output of their rule-based diacritizer that already contained some correct diacritized words and the analogical human reference translation. AppTek's rule-based Arabic diacritizer is introduced in Section 5.5.

Our intention is to establish a statistical phrase-based system that translates the output of the rule-based system into correct vowelized text as a post-editing step. The benefit is that the rule-based system excludes a large number of possible forms.

The idea is to create the phrase table as shown in Figure 3.6. The source side contains AppTek's output, the target side the correct vowelized text. The language model is created by the correct vowelized text.

We also intend to integrate unvowelized text into the source side of the phrase table to test the influence of this additional information.

## 3.2 Conditional Random Fields to Diacritization

Our approaches using the CMU SMT system have the advantage that operating on word and on character level can be realized without large effort.

A shortcoming is, however, that the integration of grammatical information that may help particularly for the restoration of the diacritics at the word endings is more difficult.

With the Conditional Random Fields, however, it is easy to influence the generation of the diacritics by dependencies on more global features rather than on the context as in our previous approaches.

Conditional random fields are a probabilistic framework for labeling and segmenting sequential data [Wal04]. A CRF[1] is a form of undirected graphical model that defines a single log-linear distribution over label sequences given a particular observation sequence.

As shown in the formular of CRFs, conditional probabilities are computed for the decision which diacritic to choose deriving on dependencies between features in the training data expressed by feature functions.

$$Y^* = \arg\max_{Y} \log P(Y|X) = \sum \Theta_i f_i(X, Y)$$

For the observation sequence $X$, in our case the sequence of consonants, the sequence of the most likely vowels $Y$ is computed by the highest sum of the defined feature functions $f_i$ that are weighted by feature weights $\Theta_i$.

### 3.2.1 Features

Global features that may influence the generation of the diacritics are part-of-speech tags. Part-of-speech tags provide us an identification of the words based on their definition and relationship to other words such as adjective, perfect verb, relative pronoun etc. Particularly the diacritics at the word endings that mark the case distinctions and verbal information may be better restored by this information.

Specific words in the sentence e.g. the first or the last word of a sentence or a word in a certain distance to the current word may have positive influence on the

[1]CRF = conditional random field

correct restoration. So features with word or position dependent information may be interesting, too.

In the CRF model we use features of word level and character level to influence the generation of the diacritics. On word level, we apply features that capture the lexical part-of-speech tags up to 2 neighboring words. On character level, we make use of features which are the combination of up to 6 previous and subsequent characters.

In the next chapter we illustrate the data that we use to establish our systems, the representation of the Arabic characters in Buckwalter Transliteration and the evaluation metrics.

# 4. Data

We work with two data sources. On the one hand we make use of the diacritized LDC's Arabic Treebank data for the translation approaches and the conditional random fields approach. On the other hand we received data by the company AppTek which help us conduct the post-editing experiments. Furthermore, we adduce reasons for the representation of the Arabic text in Buckwalter Transliteration in this chapter and describe the evaluation metrics we use for our data.

## 4.1 LDC's Arabic Treebank

The data to train, tune and to test the translation- and the conditional random fields-based diacritizers are extracted from the LDC's Arabic Treebank of diacritized An Nahar News stories. Arabic Treebank data are produced by the Linguistic Data Consortium (LDC) to support language research and development of language technology for modern standard Arabic. As many researchers work with LDC's data and the results of diacritization systems are heavily dependent on the used data, we are able to compare our approaches with other approaches [MBK06][NS05][ZSS06].

Our sentences are in modified Buckwalter Transliteration, pictured in Section 4.3, and do not include any punctuation marks. Since the corpus contains complete vowelization including case endings, the transliterated diacritics "BK", "Bu", "BN", "Ba", "Bi", "BK", "F", "a", "K", "u", "o", "i", "B" and "N" had to be deleted in order to create the non-vowelized part of the parallel corpus.

For all systems except the post-editing system a training set of 613 k words within 23 k sentences has been used from this data source. A development set to improve the established systems and a test set to test them have been established each with 32 k words in 1 190 sentences.

For the systems operating on character level, characters are separated from each other by whitespaces in the test set and on the non-vowelized part of the training corpus. Also, within the reference and the vowelized section of the training corpus, each character and its appropriate following diacritic are separated from those of the other characters. To distinguish between words in the character representation the word "space" is inserted between each word which is depicted through characters or vowelized characters.

## 4.2 AppTek Data

We were given data by AppTek. Applications Technology, Inc. (AppTek) is a company in the fields of natural language processing and computational-linguistics technology. It develops industry level machine translation systems for several language pairs.

AppTek provides multilingual information-retrieval solutions enabled with machine translation; web-based dynamic translation of online content; and integration with automatic speech recognition and text-to-speech software, enabling speech-to-speech machine translation. The suite of linguistic tools includes proper-noun recognition, morphological analysis, and large annotated lexicons, all with API and SDK integration tools.

First they provided us with a test set that contained 69 sentences with punctuation marks in UTF8 format. After converting the text into Buckwalter Transliteration and doing some modifications in our translation systems such as the separation of the punctuation marks, the text was ready to be diacritized. Unfortunately, we did not receive the human reference translation for this test set to evaluate our translation. Yet, the test set supported us to initiate the preprocessing steps to deal with punctuation marks.

AppTek develops an Arabic rule-based diacritizer. They sent us the output of their latest diacritizer. After removing some dublicates 116 k words in 40 k sentences remained. We split the data and built a training set of 104 k words within 36 k sentences, a development set to tune the system and a test set with each 6 k words within 2 k sentences.

In comparison to the LDC's Arabic Treebank data the sentences provided by AppTek are more similar to each other and shorter. Due to the similarity, word stems or even whole words which are included in the test set are contained in the training data. So the error rates with AppTek's data are lower than those with the LDC's Arabic Treebank data.

## 4.3 Buckwalter Transliteration

Since the CMU SMT system deals with ASCII characters the Arabic script is transliterated into Buckwalter Transliteration which was developed at Xerox by Tim Buckwalter with the goal to transmit Arabic text into environments where the display of genuine Arabic characters proves either impossible or inconvenient [Buc02]. Roman character equivalents are chosen to be reasonably mnemonic.

From Unicode to Buckwalter Transliteration and back it is a one-to-one mapping without gain or loss of ambiguity. Thus, modern Arabic orthographical symbols are transliterated by using only 7-bit ASCII characters.

In contrast to Arabic characters the Buckwalter Transliteration is editable and displayable on ASCII terminals and printers. The Arabic orthography is strictly represented, differently to the common romanization like e.g. Qalam that adds morphological information not expressed in Arabic script. Whereas unmodified Arabic letters are straightforward to read, the transliteration of letters with diacritics takes some time to get used to.

Figure 4.1 describes the function of each diacritic, their pronunciation as well as their adequate character in Buckwalter Transliteration. As can be seen from the table seven diacritics that we restore are displayed exactly in Buckwalter Transliteration, except the "shadda" for processing reasons.

The CMU SMT System deals with text files containing ASCII characters. An overview of all Arabic characters in Buckwalter Transliteration is given by the Buckwalter Transliteration Table in the Appendix A.

## 4.4 Evaluation Metrics

There are currently several ways of evaluating a machine translation system, one of which is evaluating with regard to the adequacy and fluency of the translated text. Whereas, on the one hand, the meaning is to be conveyed both accurately and clearly, it is equally important for the final output to be grammatically correct and fluent.

So far there have also been human evaluations such as TC-Star SLT Evaluation (English-Spanish) and Darpa TIDES MT Evaluation (Chinese-English). Due to inefficient costs and the lack of foreign speakers that are needed for human evaluations, metrics have been defined to automatically evaluate the output of SMT systems. Two of these automatic systems to be explained here are BLEU[1] and NIST.

BLEU [PRWZ02] works by assigning enhanced values to outputs which hold a rather large number of n-grams matching with given references. Another useful feature of BLEU is that it helps in guaranteeing for some balance between the length of the translation and the reference sentence. BLEU scores range from 0 to 1.

NIST, however, works differently: It compares the output with given reference, calculating the information gain for matching n-grams. Sentences whose length differs significantly from the reference considered will be penalized.

Other conventional evaluation metrics are the mWER[2] and the mPER[3].

When translating words without diacritics into such with diacritics, the translated sentence is as long as the reference sentence. Therefore, we need evaluation metrics which do not regard sentence length, e.g. Word Error Rate or Character Error Rate.

In our case the Word Error Rate returns the percentage of incorrectly diacritized white-space delimited words. The Character Error Rate is also called Letter Error Rate and Diacritization Error Rate as it illustrates the percentage of diacritics incorrectly restored. In this thesis we use the term "Diacritization Error Rate".

The word endings including the last diacritic indicate whether or not case distinction or verbal information is translated properly. As the character error rate is noticeably higher at the end of words and as we tested methods to improve especially the final vowelization, we evaluate the diacritized text with final vowelization as well as the one without final vowelization.

[1]BLEU = BiLingual Evaluation Unterstudy
[2]mWER = Multi reference Word Error Rate
[3]mPER = Multi reference position independent WER

| Name | | Buckwalter Transliteration | Pronunciation |
|---|---|---|---|
| **Short vowels /a/, /u/, /i/** | | | |
| Fatha | ◌َ | a | /a/ |
| damma | ◌ُ | u | /u/ |
| kasra | ◌ِ | i | /i/ |
| **Double case ending** | | | |
| fathatayn | ◌ً | F | /an/ |
| dammatayn | ◌ٌ | N | /un/ |
| kasratayn | ◌ٍ | K | /in/ |
| **Syllabification marks** | | | |
| shadda | ◌ّ | B (normally ~) | consonant doubling vowel |
| sukuun | ◌ْ | o | vowel absence |

Figure 4.1: Buckwalter Transliteration.

By these distinctions we are able to analyze how the diacritization systems perform in restoring the diacritics at the word endings and at the stems. Besides, it is evident whether the errors disperse to many words or are rather located in a few words.

These interpretations support us to appropriate measures in order to enhance our systems. For example measures to reduce the errors at the word endings may influence the generation of the final diacritics by boosting and penalizing some endings, or integrating rules or grammatical information. Errors located in a few words might arise from many unknown words. In this case we may add new training data.

To compute word and diacritization error rates we use the evaluation tool *sclite*[4] which is originally used for scoring the output of speech recognition systems, and adapted it according to our purposes. sclite uses dynamic programming to find alignments between reference and hypothesis word strings. The evaluation tool was modified to gain the word error rate as well as the diacritization error rate with and without the final vowelizations.

To tune our translation systems we also make use of BLEU as the metric for the Minimum Error Rate Training, characterized in Section 5.1.2.

In addition to sclite different tools are applied in our experiments. They are outlined in the next chapter.

[4]sclite = Score-Lite

# 5. Tools

In this chapter the tools which we apply to conduct our experiments are characterized. In addition to the CMU SMT System which is helpful for the diacritization as a translation process, we utilize the Suffix Array Language Model Toolkit, SRI Language Model Toolkit, the Moses package with GIZA++, AppTek's rule-based Arabic diacritizer, the Stanford Part-of-Speech Tagger and CRF++.

## 5.1 The CMU SMT System

As the CMU SMT system which is also used at Universität Karlsruhe (TH) provides phrase to phrase translations extracted from a bilingual corpus, we employ it and any associated scripts for our translation purposes within the frame of diacritization [VZH+03].

The basics of the statistical machine translation which we described in Chapter 1.3 e.g. the decoder, different models or the phrase extraction are implemented in the system. The system allows to be tuned by weighting the different statistical models with scaling factors.

### 5.1.1 Tuning the SMT System

In the CMU SMT system different models $Q_i$ namely the language, translation, distortion model, word and phrase count are used. Since these models can be trained with different amounts of data, they each represent different levels of reliability.

Thus different weights $c_i$, also called scaling factors, are given to these different models. When decoding, the weighted scores of these models are summed up for each sentence. It is important to find the optimal scaling factors $c_1 \dots c_n$ so that $Q = c_1Q_1 + c_2Q_2 + \dots + c_nQ_n$ leads to the highest score for the chosen evaluation metric.

### 5.1.2 Minimum Error Rate Training

For each scaling factor of a model the sum $Q = c_kQ_k + Q_{rest}$ is given. To compute the optimal scaling factors for the different models the Minimum Error Rate (MER) training [Och03] can be used in the CMU SMT system. By having a reference text the Minimum Error Rate Training makes use of automatic evaluation metrics like NIST or BLEU, mentioned in section 4.4, and adapts the scaling factors depending on the evaluation results. The final evaluation criterion is taken directly into the computation as part of the training procedure.

In order to determine the optimal scaling factors for the translation of the test set we applied the Minimum Error Rate Training to our development set. We use BLEU as evaluation metric for the MER training.

For the Minimum Error Rate Training automatic tuning algorithms like Maximum Entropy, Minimum Bayes, Simplex and genetic algorithms are used.

Next, the Suffix Array Language Model and the SRI Language Model are presented. Our baseline diacritization systems are realized by the Suffix Array Language Model which is customary at the SMT group of the Carnegie Mellon University. Due to the fact that the SRI Language Model fares better in our hybrid experiments with integrating an additional language model on word level, we work with that language model as well.

## 5.2 Suffix Array Language Model

The Suffix Array Language Model (SALM) Toolkit has been developed since 2002. It is widely used by the SMT group of the Carnegie Mellon University and the Universität Karlsruhe (TH) in the research of data-driven machine translation [Zha07].

The C++ package provides functions to locate and estimate statistics of the n-grams within a large corpus, e.g. estimating type/token frequency or locating n-gram occurrences. The Suffix Array Language Model may feature an arbitrarily long history for a large training corpus. With a huge corpus the total number of n-grams becomes very large. In this way storing all n-grams in the memory becomes infeasible. The SALM indexes the corpus according to its suffix array order and provides efficient algorithms to collect n-gram probabilities on the fly. Since the n-gram frequencies are counted online there is no need to store the n-gram statistics offline on disk which could be very large for a huge corpus.

The advantage of the suffix array is its space efficiency due to only one pointer being saved for one word instead of the whole word. With the complexity of a binary search a word can be located in the training data. Moreover, restricted by the sentence boundaries only, n-grams up to any $n$ can be found.

By the help of the Suffix Array Language Model the a priori probabilities for our baseline systems on character and word level are computed.

Like in the CMU SMT system, each token between a whitespace in a text is treated as a word by the functions of the SALM Toolkit.

## 5.3 SRI Language Model

The SRI Language Model is a toolkit for building and applying statistical language models, primarily for use in speech recognition, statistical tagging and segmentation [Sto02].

On the one hand the SRI Language Model Toolkit is able to create a model from any training data; on the other hand it renders it possible to determine the probability of a test corpus - conventionally as the test set perplexity.

This toolkit also offers programs and utility scripts for counting n-grams or n-best rescoring. Furthermore, algorithms for smoothing, like Katz Backoff, Good Turing or Witten Bell, are implemented as well.

After working with the character level Suffix Array Language Model in our system which is able to create diacritized text on both word and character level, we expanded the system by including scores on world level computed by the SRI Language Model Toolkit.

Each token between a whitespace is handled as a word by the scripts of the SRILM Toolkit like in the SALM Toolkit.

## 5.4 Moses and GIZA++

By the means of the Moses package and the GIZA++ Alignment Toolkit we improve our system by integrating additional lexical scores to the available phrase translation probabilites.

Moses is a statistical machine translation system which allows to automatically train translation models for any pair of language by a parallel corpus [Mos06]. Among the exponential number of choices an efficient search algorithm quickly finds the highest probability translation. The phrase-based system can translate short text chunks. The words within the phrase table and the training data may have a factored representation with for example lemma, surface forms, part-of-speech, word classes or morphology. Thus linguistic and other information at many stages of the translation process are enabled. Moses is a drop-in replacement for Pharaoh, a beam search decoder for phrase-based statistical machine translation models [Koe06].

GIZA++ is a freely available implementation of statistical translation models. During the training process of Moses the data are prepared, GIZA++ runs, the words are aligned, a lexical translation table is built, the phrases are extracted and scored, and a lexical reordering model and generation models are built.

Since we utilize the CMU SMT system as the translation system for our experiments, we do not apply Moses as a translation system. For the creation of a phrase table that includes phrase translation probabilities and lexical weights, we benefit from the Moses scripts and GIZA++. The generated phrase table is adapted and integrated in our system.

By default a phrase table containing at most 7-gram entrances with conditional phrase translation probabilities as well as lexical weights in both directions is created by GIZA++.

## 5.5 AppTek's rule-based Arabic Diacritizer

AppTek's Arabic diacritizer is a software component that diacritizes Arabic words and sentences.

It makes use of Arabic morphology rules, prefixes and suffixes applicability, and idioms to generate iterations of possible diacritized word forms. A series of disqualifying steps eliminate the iterations to the appropriate ones.

The diacritizer processes the data in UTF8 format. To deal with the data we convert the output of the diacritizer and the human reference translation into Buckwalter Transliteration.

At the moment the diacritizer is still on development and is outperformed by our systems as shown in Section 6.1.7. Yet, the system is helpful for us since many possibilities of diacritized words are excluded. We apply a statistical-based post-processing step to the output of AppTek's diacritizer.

There are efforts to improve the rule-based diacritizer by integrating statistical methods at AppTek.

## 5.6 Stanford Part-of-Speech Tagger

The Stanford Part-of-Speech Tagger reads text in some language and assigns parts of speech to each word and other token, such as noun, verb, adjective, etc. [TM00]

The software is a Java implementation of log-linear part-of-speech taggers. The tagger uses a subset of the POS tags employed in the Penn English Treebank.

For the Arabic language the Stanford Arabic Tagger has been developed. We use that tool to generate part-of-speech (POS) tags to the non-vowelized words that influence the restoration of diacritics in our conditional random fields systems, as described in section 6.2.

The Stanford Part-of-Speech Tagger exclusively adds tags to unvowelized Arabic text. Therefore in text with diacritics the diacritics have to be removed to be applied by the Stanford Part-of-Speech Tagger. As the Stanford Part-of-Speech Tagger processes data in UTF8 format, we convert the text from Buckwalter Transliteration into UFT8 format. After the generation of the POS tags the data are converted back into Buckwalter Transliteration.

In addition to Arabic data, Chinese, German and English tagger data are provided by the Stanford Natural Language Processing Group.

## 5.7 CRF++

CRF++ is a simple, customizable, and open source implementation of Conditional Random Fields (CRFs) for segmenting and labeling sequential data. The tool is designed for generic purpose and is applied to a variety of natural language processing (NLP) tasks, such as named entity recognition, information extraction and text chunking.

The training file is in a particular format consisting of multiple columns where information for the input sequence, additional information such as POS tags and words as well as the output sequence may be defined. The test file has the same format without the last column that represents the output sequence.

Moreover, for the training some feature templates have to be specified which describe which features are used in training and testing. Dependent on the relative position from the current focusing token dependencies can be defined.

An advantage of CRF++ in comparison to the statistical machine translation approaches is that the execution is considerably faster after training. The running time can be regarded in Chapter 6.3.

In the following chapter we present our diacritization systems, the challenges and their results.

# 6. Experiments and Results

After having delineated the challenge of diacritization, the approaches of other researchers in this field, our ideas, methods and tools we describe our experiments in this chapter. First the diacritization is treated as a statistical machine translation task. Starting from two baseline systems we suggest several methods to improve different components with the goal of gaining lower error rates. The combination of AppTek's rule-based diacritizer and our statistical-based approach is shown and afterwards the integration of global features by the help of conditional random fields. We present our results in this chapter and analyze the systems which we established.

## 6.1 Diacritization as Translation

In the following sections we describe the implementations of our translation approaches. We start with essential preprocessing steps, go on with the baseline systems on character and word level, the hybrid approaches, the approach to improve the phrase table with attempts to enhance the diacritization at the word endings and end with translating the output of a rule-based diacritizer.

For all statistical machine translation tasks the systems are tuned by a Minimum Error Rate Training with 8 iterations on the development data. The presented error rates of our systems are the results on an unseen test set.

### 6.1.1 Preprocessing and Postprocessing

It is important to provide for the consistency between the training and test texts. In a preprocessing step different input modifications may be applied on the raw texts. For example, morphological preprocessing like stemming and part-of-speech tagging or the separation of punctuation marks may noticeably enhance the quality of the decoding. Furthermore, in a postprocessing step the output text can be modified according to adapt a certain format for evaluation or further processing steps.

In order to be able to process AppTek's data, we separated any punctuation marks and removed special characters which the CMU SMT decoder is not able to process in a preprocessing step. Moreover, we converted texts in UTF8 format into our modified Buckwalter Transliteration representation for processing purposes.

For the constitution of the training and test data for our systems operating on character level, we separated the consonants on source side and each consonant along with its adequate diacritic on target side. To create part-of-speech tags for training and test data in our CRF systems, we use the Stanford Part-of-Speech Tagger.

Our postprocessing steps include an n-best rescoring and reordering as well as the integration of additional language model scores.

### 6.1.2 The Baseline Systems

In order to have a starting point to implement statistical methods for the automatic diacritization, both a word-based and a character-based baseline SMT system with the components of the CMU SMT System were established and evaluated.

For the two baseline systems test and training data are converted into Buckwalter Transliteration and normalized in a preprocessing step. Depending on the language model and the translation model, the hypotheses are determined. A delineation of the components which the baseline systems include is given in Figure 6.1.

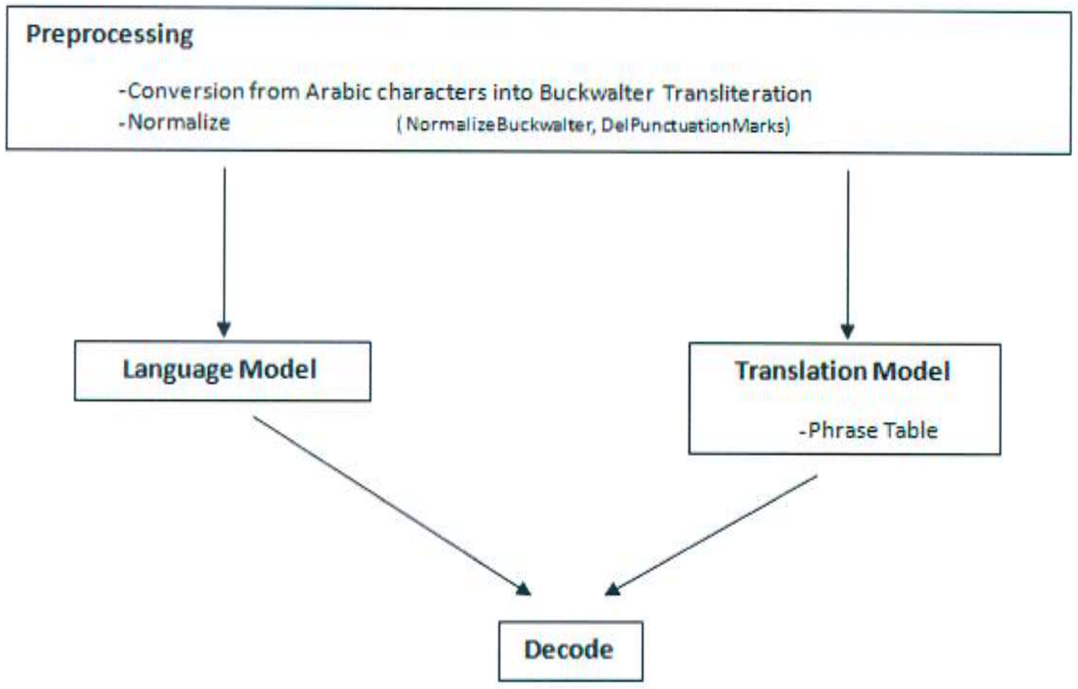

Preprocessing
-Conversion from Arabic characters into Buckwalter Transliteration
-Normalize
(NormalizeBuckwalter, DelPunctuationMarks)
Language Model
Translation Model
-Phrase Table
Decode

Figure 6.1: Components of the Baseline Systems

The language model is a 6-gram Suffix Array Language Model. It belongs to the category IBM1, which means that the calculation as to how likely the continuous sequences of words are, depends on lexical probabilities only. The input to build the language model is the corpus name and the corpus size in millions of words. So one file stores the vocabulary built for the corpus and three files to index the corpus are generated.

The translation model consists of the phrase table which is organized in the following way: *source phrase # target phrase # probability*. The probability entries are conditional phrase translation probabilities.

The system allows dealing with unknown words (which are not within the phrase table) by inserting the source word into the output sentence. In this way an unknown word does not become lost, with the reader being able to look it up after the decoding.

| | | WORD-BASED | CHAR-BASED |
|---|---|---|---|
| Language Model | | words, max 10-gram | chars, max 10-gram |
| Phrase Table | | words, max 5-gram | chars, max 5-gram |
| Training Data | | 613 k words | 613 k words |
| Test Data | | 23 k words | 23 k words |
| final vow | WER | **22.8** | **21.8** |
| | DER | **7.4** | **4.8** |
| no final vow | WER | **9.9** | **7.4** |
| | DER | **4.3** | **1.8** |

Table 6.1: Results of the Baseline Systems.

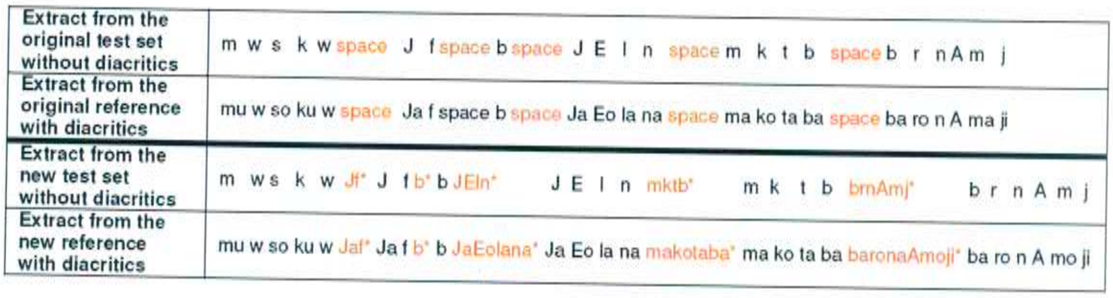
For the decoding optimizations and evaluations parameters can be specified. Among others, there are parameters for activating the Minimum Error Rate Training, specifying the number of iterations for the Minimum Error Rate Training, the adopted evaluation metric and the number of references for evaluation.

#### 6.1.2.1 The System on Word Level

First a system which includes a phrase table and a language model consisting of words was built.

The language model is a 10-gram Suffix Array Language Model belonging to the category IBM1. The phrase table contains phrases with 1 word up to 5 words.

Table 6.1 shows that the word-based SMT system accounted for a WER of 22.8% and a DER of 7.4%. Without concerning the word endings the WER was only 9.9% and the DER only 4.3%.

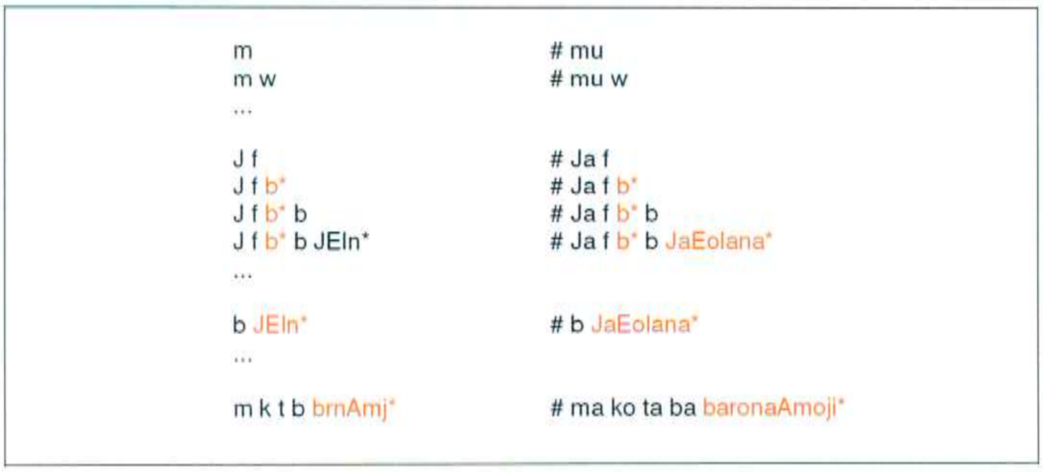


| | |
|---|---|
| **Extract from the original test set without diacritics** | m w s k w space J f space b space J E I n space m k t b space b r n A m j |
| **Extract from the original reference with diacritics** | mu w so ku w space Ja f space b space Ja Eo la na space ma ko ta ba space ba ro n A ma ji |
| **Extract from the new test set without diacritics** | m w s k w Jf' J f b' b JEln' J E I n mktb' m k t b brnAmj' b r n A m j |
| **Extract from the new reference with diacritics** | mu w so ku w Jaf' Ja f b' b JaEolana' Ja Eo la na makotaba' ma ko ta ba baronaAmoji' ba ro n A mo ji |

Figure 6.2: Representation of Characters in Context with the Next Whole Word.

#### 6.1.2.2 The System on Character Level

As proposed for the restoration of diacritics in [Mih02], the next step was to learn from characters, as the word-based system did not know a large amount of words. We developed



| | |
|---|---|
| m | # mu |
| m w | # mu w |
| ... | |
| J f | # Ja f |
| J f b* | # Ja f b* |
| J f b* b | # Ja f b* b |
| J f b* b JEln* | # Ja f b* b JaEolana* |
| ... | |
| b JEln* | # b JaEolana* |
| ... | |
| m k t b brnAmj* | # ma ko ta ba baronaAmoji* |

Figure 6.3: Phrase Table of Chars in Context with the Next Whole Word.

a system with a phrase table where non-vowelized characters are arranged on the source side and the vowelized ones on the target side.

The character-based system has a 10-gram Suffix Array Language Model of the category IBM1, with the phrase table containing at most 5-gram character entries.

With a WER of 21.8% and a DER of 4.8% the results of the evaluation system are slightly better than those of the word-based system. Also, with 7.4% and 1.8% the WER and DER are lower without concerning the word endings. All results are presented in Table 6.1.

### 6.1.3 Characters in Context with the Next Whole Word

As we concluded above, the results of the character-based system prove better than those of the word-based system. The advantage of dealing with words, however, is that more context is provided within the phrase table and the language model.

Thus the aim was to expand the context of the characters. Therefore a system, which translates characters in context with the next word, was established. We achieved this by a preprocessing step: After each word in character representation the next word in word representation was inserted into training and test data. In order to extract later words in character representation from the hypothesis each real word was marked by a "*". So in contrast to the previous character-based system the word "space" is not necessary, since "word*" is a delimiter between words in character representation. Figure 6.2 and Figure 6.3 show extracts from the modified test set reference and phrase table. The modifications are color-coded.

As the phrase table contains up to 5-gram entries, at most four characters stand in context with the next whole word. The 10-gram Suffix Array Language Model has redundant information since the same word in character representation follows after a word in word representation.

After the decoding the character part was extracted from the hypothesis. As shown in Table 6.2, the extraction of the character part comes along with an improvement of 0.5% in the WER in contrast to the character-based system. The context of the following word makes sure that more word endings are translated correctly. However, the recognition of the word stems is worse.

| | | CHAR-BASED | CHARS IN CONTEXT WITH THE NEXT WHOLE WORD |
|---|---|---|---|
| Language Model | | chars, max 10-gram | max 10-gram |
| Phrase Table | | chars, max 5-gram | max 5-gram |
| Training Data | | 613 k words | 613 k words |
| Test Data | | 23 k words | 23 k words |
| final | WER | **21.8** | **21.5** |
| vow | DER | **4.8** | **7.1** |
| no final | WER | **7.4** | **10.5** |
| vow | DER | **1.8** | **4.4** |

Table 6.2: Results of Chars in Context with the Next Whole Word.

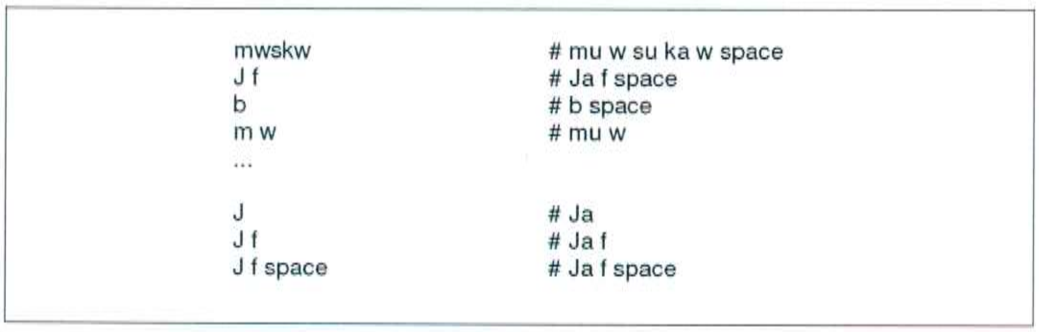
Due to the high error rates concerning the word stems we turned our attention to the following auspicious hybrid approach.

### 6.1.4 The Systems on both Levels

As depicted in Section 3.1.6 the CMU SMT system provides the opportunity to input the test data as a lattice. We utilize the lattice representation to define edges from word to word as well as from character to character. So we archieve a switch from word level to character level and back by the decoder.

#### 6.1.4.1 The System without Language Model on Word Level

The idea was to establish a hybrid SMT system which translates a word as soon as it is found in the phrase table. Otherwise each character of the unknown word is translated.



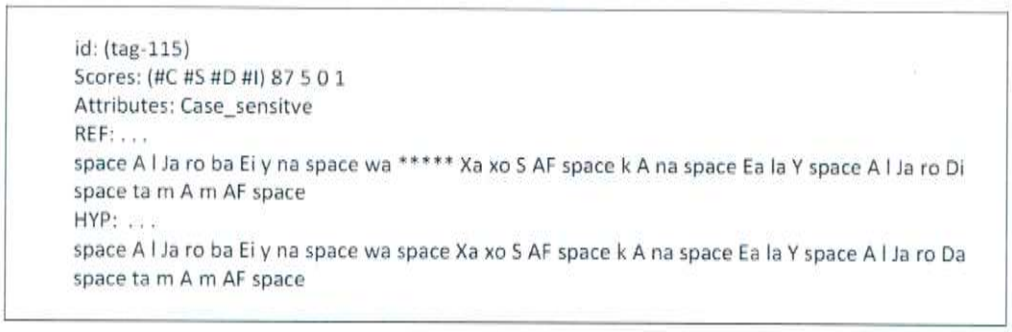
| | |
|---|---|
| mwskw | # mu w su ka w space |
| J f | # Ja f space |
| b | # b space |
| m w | # mu w |
| ... | |
| J | # Ja |
| J f | # Ja f |
| J f space | # Ja f space |

Figure 6.4: Extract from Phrase Table with Word Part and Character Part Entries

Each sentence to translate is represented as a lattice with edges from character to character and from word to word. For each translation a new lattice edge is created, the model scores through each of them being calculated.

The phrase table is built of a word part and a character part. The word part contains up to 5-gram word entries, the character part at most 5-gram character entries. The source side of the word part is assigned into vowelized characters and the word "space" to mark the whitespace between any words. Whereas a whitespace is assigned into "space", characters within the source side of the character part are allocated into vowelized ones. The 10-gram Suffix Array Language Model is established with vowelized characters since the result sentences are made up of these.

In Arabic text which lack diacritics words consisting of only one character may appear. The characters "b", "d", "E", "f", "F", "g", "h", "H", "I", "j", "J", "k", "l", "m", "M", "n", "q", "r", "s", "S", "t", "T", "v", "w", "X", "y" and "z" represented in Buckwalter Transliteration are also words in the Arabic language. It also may happen that real characters occur in Arabic text with words.



```
id: (tag-115)
Scores: (#C #S #D #I) 87 5 0 1
Attributes: Case_sensitve
REF: . . .
space A l Ja ro ba Ei y na space wa ***** Xa xo S AF space k A na space Ea la Y space A l Ja ro Di
space ta m A m AF space
HYP: . . .
space A l Ja ro ba Ei y na space wa space Xa xo S AF space k A na space Ea la Y space A l Ja ro Da
space ta m A m AF space
```

Figure 6.5: Evaluation without Distinguishing between One Char Words and Chars.

Thus we can find characters in both parts of the phrase table, within the character part as real characters and within the word part as words, as presented in Figure 6.6. If we do not distinguish between a word consisting of only one character and a real character, the system may possibly misinterpret and translate a character within a word as an individual word, inserting the word "space" into the target side. Or vice versa, meaning the system mistakenly confuses a word consisting of only one character for a character. In this case the word "space" would not be inserted. Figure 6.5 alleges an example where the word "space" is inserted mistakenly into the hypothesis after the character "wa" since it is not distinguished between one character words and characters.

Translating a test set without marking words consisting of one character leads to insertions and deletions which increase the error rates, as demonstrated in Table 6.4. We realized the distinction between words consisting of one character by inserting a "*" in the source side of the phrase table and in the lattice, displayed in Figure 6.7.

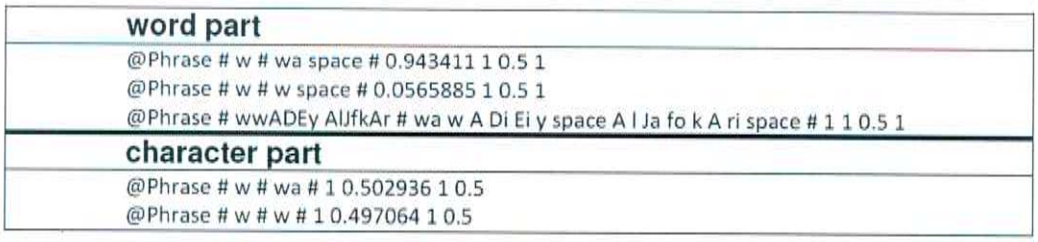
The results of our system on both levels with a character language model are presented in Table 6.3. We were able to reduce the WER to 20.1% and the DER to 4.3%. Even at the word stems we declined the WER to 6.6% and the DER to 1.6%. In the framework of this approach the system was also evaluated with a narrower context within the phrase table. Thereby we demonstrate that a wide context within the phrase table is important to receive better results.

| | | HYBRID SYSTEM | | | | |
|---|---|---|---|---|---|---|
| Training Data<br>Test Data | | 613 k words<br>23 k words | | | | |
| Language Model<br>Phrase Table | | chars, max. 10-gram<br>chars, max. 5-gram | | | | |
| | | words, max. 1-gram | words, max. 2-gram | words, max. 3-gram | words, max. 4-gram | words, max. 5-gram |
| final vow | WER | **21.5** | **20.4** | **20.2** | **20.2** | **20.1** |
| | DER | **4.5** | **4.3** | **4.3** | **4.3** | **4.3** |
| no final vow | WER | **6.7** | **6.6** | **6.6** | **6.6** | **6.6** |
| | DER | **1.6** | **1.6** | **1.6** | **1.6** | **1.6** |

Table 6.3: Results of the Hybrid Systems.

| | | HYBRID SYSTEM | |
|---|---|---|---|
| | | **no marked one char words** | **marked one char words** |
| final vow | WER | **20.4** | **20.1** |
| | DER | **4.4** | **4.3** |
| no final vow | WER | **7.5** | **6.6** |
| | DER | **1.7** | **1.6** |

Table 6.4: Hybrid Systems - Marked and No Marked One Character Words.



| word part |
|---|
| @Phrase # w # wa space # 0.943411 1 0.5 1 |
| @Phrase # w # w space # 0.0565885 1 0.5 1 |
| @Phrase # wwADEy AlJfkAr # wa w A Di Ei y space A l Ja fo k A ri space # 1 1 0.5 1 |
| **character part** |
| @Phrase # w # wa # 1 0.502936 1 0.5 |
| @Phrase # w # w # 1 0.497064 1 0.5 |

Figure 6.6: Extract from Phrase Table without Distinguishing between One Character Words and Characters.

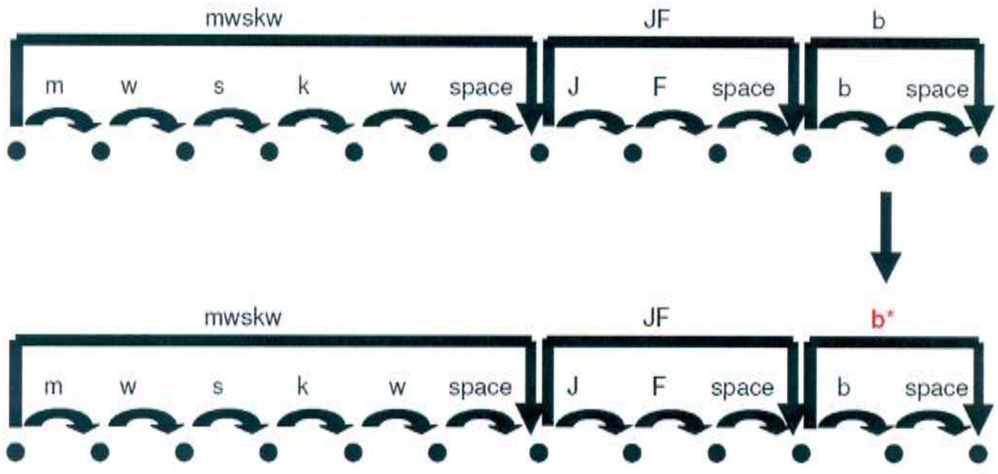


Figure 6.7: Modification of the Lattice to Mark One Character Words

Although we work with a 10-gram character language model, the context in a word language model would be wider. Thus we proceeded to expand the diacritization system through a word language model.

#### 6.1.4.2 The System with Language Model on Word Level

The hybrid system goes to word level for known words and to character level for unknown words. Since due to the phrase table the target phrases are in character representation the language model scores were computed on character level so far. In order to enlarge the context our intention was to integrate a language model on word level.

This is realized as follows: 1000-best lists of each sentence are converted from character representation into word representation. Then for each sentence in the 1000-best lists a word language model score is assigned. With the help of the Minimum Error Rate Training, the new optimal scaling factors for the models are calculated. After a rescoring the sentence from the 1000-best list with the highest score is chosen as hypothesis.

To calculate the logarithmic language model probability for each sentence we experimented with the Suffix Array Language Model Toolkit [Zha07] and the SRI Language Model Toolkit [Sto02]. The results show that in contrast to the translation model a large context in the language model on word level does not perform better. As shown in Table 6.5

| | | HYBRID SYSTEM | HYBRID SYSTEMS WITH WORD LEVEL SALM | | |
|---|---|---|---|---|---|
| Language Model | | chars, max 10-gram | chars, max 10-gram<br>words, max 3-gram | chars, max 10-gram<br>words, max 4-gram | chars, max 10-gram<br>words, max 6-gram |
| final vow | WER | 21.8 | 20.0 | 20.0 | 20.1 |
| | DER | 4.8 | 4.3 | 4.3 | 4.3 |
| no final vow | WER | 7.4 | 6.9 | 6.9 | 6.9 |
| | DER | 1.8 | 1.7 | 1.7 | 1.7 |

Table 6.5: Results of the Hybrid Systems with Word Level SA Language.

| | | HYBRID SYSTEM | HYBRID SYSTEMS WITH WORD LEVEL SRILM | | |
|---|---|---|---|---|---|
| Language Model | | chars, max 10-gram | chars, max 10-gram<br>words, max 3-gram | chars, max 10-gram<br>words, max 4-gram | chars, max 10-gram<br>words, max 6-gram |
| final vow | WER | 21.8 | 19.9 | 19.9 | 20.0 |
| | DER | 4.8 | 4.3 | 4.3 | 4.3 |
| no final vow | WER | 7.4 | 6.8 | 6.8 | 6.9 |
| | DER | 1.8 | 1.6 | 1.7 | 1.7 |

Table 6.6: Results of the Hybrid Systems with Word Level SRI Language.

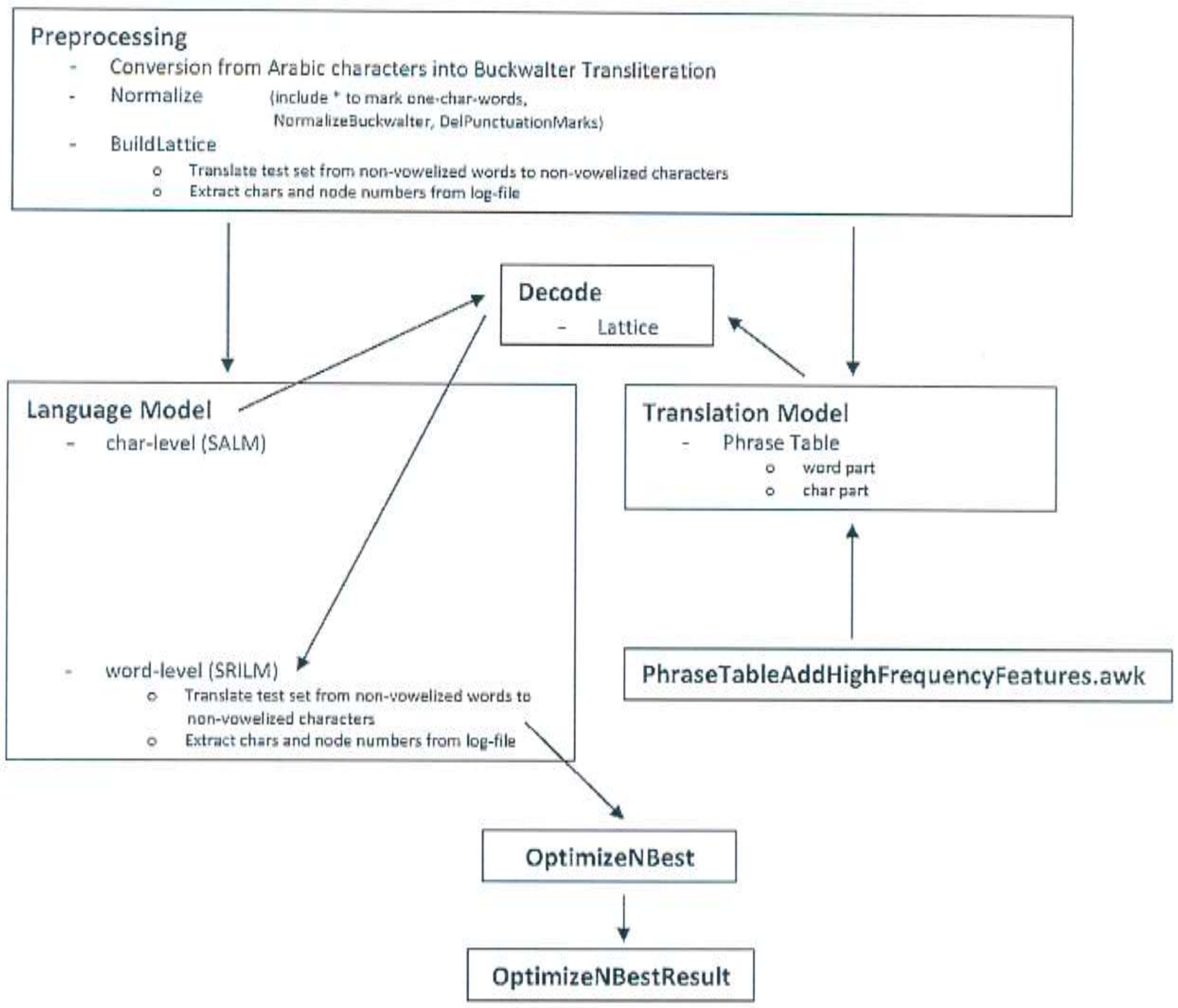

Preprocessing
- Conversion from Arabic characters into Buckwalter Transliteration
- Normalize (include * to mark one-char-words, NormalizeBuckwalter, DelPunctuationMarks)
- BuildLattice
o Translate test set from non-vowelized words to non-vowelized characters
o Extract chars and node numbers from log-file
Decode
- Lattice
Language Model
- char-level (SALM)
- word-level (SRILM)
o Translate test set from non-vowelized words to non-vowelized characters
o Extract chars and node numbers from log-file
Translation Model
- Phrase Table
o word part
o char part
PhraseTableAddHighFrequencyFeatures.awk
OptimizeNBest
OptimizeNBestResult

Figure 6.8: Components of the Hybrid System with Language Model on Word Level

and Table 6.6, the 3-gram word language models scored better than any 4- or 6-gram language model. The SRI Language Model came off a little bit better than the Suffix Array Language Model. The best hybrid system with language model on word level has a WER of 19.9% and a DER of 4.3%. The WER and the DER without the final diacritics are 6.8% and 1.6%.

Even if the hybrid approach with the language model on word level results in lower error rates the system is not as convenient as the systems we established before. The phrase table has significantly more entries as it contains both characters and words. The test set in text format has to be converted into the lattice format containing edges for the characters and the words. Furthermore, computing the language model scores for each sentence in each 1000-best list, the rescoring and the reordering extends the decoding time. In Section 6.3.2 an overview of the duration of the systems is given. The complexity of the system becomes apparent by the characterization of the components allegorized in Figure 6.8.

Whether this system is applicable depends on the desire. - Is a system designated that provides accuracy or is the attention turned on speed?

| | | **BASELINE SYSTEM** | **BASELINE SYSTEM with 7-gram phrase table** | **ADDITIONAL LEXICAL SCORE beside phrase translation prob in phrase table** |
|---|---|---|---|---|
| | | **char-based** | **char-based** | **char-based** |
| Phrase Table | | chars, max 5-gram | chars, max 7-gram | chars, max 7-gram (by Moses/GIZA++) |
| final vow | WER | **21.8** | **21.6** | **21.5** |
| | DER | **4.8** | **4.8** | **4.7** |
| no final vow | WER | **7.4** | **7.5** | **7.4** |
| | DER | **1.8** | **1.9** | **1.8** |

Table 6.7: Results with additional Lexical Weight.

After having enhanced the language model we take a look at the translation model in the next section.

## 6.1.5 Additional Lexical Scores for the Translation Model

Having improved the language model our goal was to enhance the translation model. So far the phrase table contained exclusively the phrase translation probabilities. Thus a sentence was scored by summing up the chances that a certain phrase is translated into another certain phrase.

By integrating a lexical score for each phrase the word translation probabilities are also integrated as can be seen in the formular of the lexical weighting shown in Table 6.9. As the diacritization is a monotone alignment the factor in front of the sum is 1 in our case.

| | |
|---|---|
| Formular of the Phrase Translation Probability [Pang 05] | $p(\tilde{c} \mid \tilde{e}) = \prod_i \sum_j p(c_i \mid e_j)$ |
| Formular of Lexical Weighting [Koehn06] | $\text{lex}(\bar{f}\mid\bar{e}, a) = \prod_{i=1}^{n} \frac{1}{\lvert\{j\mid(i,j)\in a\}\rvert} \sum_{\forall(i,j)\in a} w(f_i\mid e_j)$ |

Figure 6.9: Phrase Translation Probability and Lexical Weighting

We conducted our experiments with additional lexical scores within the phrase table by expanding the baseline system on character level.

To establish a phrase table containing both the phrase translation probability $\varphi(vow/non\_vow)$ and the lexical weighting $lex(vow/non\_vow)$ we made use of the Moses

| Distribution of the Word Endings in the Hypothesis of the Hybrid System with word LM | | Human Reference Translation | | Training Data | |
|---|---|---|---|---|---|
| pi | 10.477 | pi | 8.508 | pi | 8.828 |
| y | 6.876 | y | 6.890 | y | 7.122 |
| A | 6.477 | A | 6.432 | A | 6.252 |
| n | 4.906 | n | 4.956 | n | 4.716 |
| Y | 4.459 | Y | 4.436 | Y | 4.398 |
| na | 3.285 | na | 3.184 | na | 3.244 |
| ti | 2.590 | AF | 2.394 | AF | 2.415 |
| AF | 2.349 | ti | 2.251 | ti | 2.233 |
| ri | 2.201 | pK | 2.054 | li | 1.894 |
| li | 2.173 | t | 2.048 | t | 1.889 |
| ... | | ... | | ... | |

Table 6.8: Distribution of the Final Consonants and Diacritics.

Package with GIZA++ [Mos06]. By default a phrase table containing at most 7-gram entrances is created by GIZA++.

In order to have a comparable phrase table to see the improvement resulting by the additional lexical score we also utilized a 7-gram phrase table in our baseline system. Figure 6.7 demonstrates that enlarging the maximal context within the phrase table from 5 to 7 characters lowers the word error rate by 2% while the lexical weights reduce it by 1%.

The Moses Package with GIZA++ also computes the phrase translation probability $\varphi(non\_vow/vow)$ as well as the lexical weighting $lex(non\_vow/vow)$. Since translating a diacritized consonant into a sole consonant always leads to the same consonant these two probabilities are 1. Thus they have no influence on the decision which phrase should be taken and were dropped to save computing time.

When creating the post-editing system we worked with these four scores as some phrases on the target side do not have the same source phrase due to some incorrect translations by AppTek's diacritizer.

In the next sections we analyze the generation of the word endings and try to influence the generation by modifications within the phrase table.

## 6.1.6 The Word Endings

### 6.1.6.1 Analysis of the Generation

Considering our results so far it is evident that the error rates at the word endings are significantly higher than at the word stems. Therefore the destination was to analyze the word endings.

We computed the probability distribution of the final consonant and diacritics as well as the real final characters for all words in the hypothesis of our system operating on word and character level with the additional word language model. In order to compare our

| Distribution of the Word Endings in the | | | | | |
|---|---|---|---|---|---|
| Hypothesis of the Hybrid System with word LM | | Human Reference Translation | | Training Data | |
| i | 35.961 | i | 30.402 | i | 30.496 |
| a | 15.117 | a | 16.925 | a | 16.868 |
| u | 7.958 | u | 10.333 | u | 10.320 |
| y | 4.906 | y | 6.890 | y | 7.122 |
| A | 4.459 | A | 6.432 | A | 6.252 |
| K | 3.285 | K | 5.520 | K | 5.249 |
| n | 2.590 | n | 4.956 | n | 4.716 |
| Y | 2.349 | Y | 4.436 | Y | 4.398 |
| F | 2.201 | F | 3.519 | F | 3.527 |
| t | 2.173 | t | 2.048 | t | 1.889 |
| ... | | ... | | ... | |

Table 6.9: Distribution of the Final Real Characters.

probability distribution we also calculated the probability distribution of the reference and the training data.

The 10 most occurring word endings are listed in Table 6.8 and Table 6.9.

While the probabilities of occurrence for most word endings are close together, it is noticeable that the word ending "pi" (ta marbouta with kasra) is generated almost 2% more than it should be. Regarding the real final characters shows that the word ending "i" (kasra) occurs even more than 5.5% more in our hypothesis than in the reference or in the training data.

Thus the idea was to influence the generation of the word endings by our decoder. Our first attempt was to penalize the generation of the word ending "pi". Then we planned to continue with penalizing the ending "i". If that was successful our goal would be to go on with influencing further endings. As we describe in the next section, we realized these experiments with the help of an additional score in the phrase table.

```
...
@Phrase # A b D p space # A bi Da pu space # 0.5 1.0
@Phrase # A b D p space # A bi Da pK space # 0.25 1.0
@Phrase # A b D p space # A bi Da pi space # 0.25 0.5
@Phrase # A b d space A # A bi di space A # 0.5 1.0
...
```

Figure 6.10: Phrase Table with Additional Score to Penalize "pi" at the Word Ending

#### 6.1.6.2 Weighting of the Word Endings

In this approach the intention was to boost word endings that appear infrequent and to penalize word endings that occur very frequent in the hypothesis in comparison to the

| | | **BASELINE SYSTEM** | **Experiments on the char-based system with WEIGHTING OF THE WORD ENDINGS** | | |
|---|---|---|---|---|---|
| Phrase Table Start Scores | | char-based | char-based "pi space" : 0.5 others : 1.0 | char-based "pK space", "pu space", "pa space", "pF space", "pN space", "p space" : 1.0 others : 0.5 | char-based "i space": 0.5 others : 1.0 |
| final vow | WER | **21.8** | **21.8** | 21.7 | 21.8 |
| | DER | **4.8** | **4.8** | **4.8** | **4.8** |
| no final vow | WER | **7.4** | **7.4** | **7.4** | **7.4** |
| | DER | **1.8** | **1.8** | **1.9** | **1.9** |

Table 6.10: Results of Weighting the Word Endings.

reference translation and the training data. The experiments are carried out with the baseline system in order to have a short duration for the decoding. The idea is that a token in the phrase table can be weighted differently from the other tokens by assigning an additional score to it. In our case the word "token" stands for character or word. By the means of the Minimum Error Rate Training [Och03] the optimal scaling factors for the new score as well as for the already existing scores can be assigned depending on the reference translation.

Our decoder converts all probabilities into the negative logarithm. Instead of multiplying some probabilities together the logarithm versions are added. Thereby, underflowing the floating point scheme is avoided [Kni99]. Since the logarithmical representation leads to negative numbers the decoder omits the minus sign to process positive numbers. So the numbers are treated as costs. If a token has been assigned a different score than the other tokens and occurs more frequent in the reference translation than the other ones, the scaling factor for that score increases the costs for that token in the Minimum Error Rate Training. If it appears more infrequent the computed scaling factor decreases the costs. An extract from the phrase table of the character level baseline system with an additional score to penalize "pi" at the word endings is demonstrated in Table 6.10.

If more tokens are supposed to weight differently from others more columns can be inserted into the phrase table and the generation of these tokens is adapted to their occurrences by the help of the Minimum Error Rate Training.

For the start score of a token which is supposed to be weighted it is important to be different from the other scores in the same column within the phrase table. How far the distance between the scores is and which score is higher does not play a role for determining the start score since the scaling factors are computed so that the optimal distance between the scores and the rank are achieved by the Minimum Error Rate Training. Attempts

| | | **RULE-BASED SYSTEM** | **WORD-BASED SYSTEM** with additional lexical score in phrase table | | **POST-EDITING** | **POST-EDITING** (rule-based output and non-vow data in source side of phrase table) |
|---|---|---|---|---|---|---|
| Training Data (words) | | | 613 k Treebank | 104 k AppTek | 104 k AppTek | 104 k AppTek |
| Test Data (words) | | 6 k AppTek (non-vow) | 6 k AppTek (non-vow) | 6 k AppTek (non-vow) | 6 k AppTek (vow output) | 6 k AppTek (vow output) |
| final vow | WER | **53.2** | **70.2** | **15.6** | **13.8** | **13.8** |
| | DER | **18.0** | **27.4** | **5.5** | **4.9** | **4.9** |
| no final vow | WER | **28.4** | **57.8** | **10.3** | **9.3** | **9.5** |
| | DER | **10.3** | **20.0** | **3.5** | **3.2** | **3.3** |

Table 6.11: Results of the Post-Editing Systems.

with several start scores lead only to insignificant fluctuations between the results after a sufficient number of iterations due to the approximation in the training. Yet, attention should be paid to the range of the scaling factors since it is possible that the optimal scaling factors are located outside the defined range.

In order to determine the optimal scaling factors the Minimum Error Rate Training with 8 iterations for each experiment was executed. First we intended to penalize the word ending "pi" which is produced almost 2% too frequently by the hybrid system with the word language model. Since influencing the creation of the word ending "pi" was not successful the next idea was to boost the other diacritics affiliated to the consonant "p". Thus "pi" was penalized along with the other endings. This approach decreased the word error rate somewhat from 22.0% to 21.9%. Moreover, we aimed to penalize the ending "i" that appeared more than 5.5% too frequent. This system did not lead to better results either. The error rates of experiments with the weighting of the word endings on character level are illustrated in Table 6.10.

A reason for the declines is overfitting. Overfitting means fitting the training data too much, so the model starts to degrade performance on test data [NR05].

Experiments on weighting too frequent or too infrequent words in the hypothesis were also conducted with the baseline system on word level. Unfortunately these attempts did not lead to better results.

### 6.1.7 The Post-Editing Systems

We converted the output of AppTek's diacritizer into our modified Buckwalter Transliteration. Then we created the phrase table with AppTek's output on the source side and the human reference translation on the target size by the help of GIZA++. Therefore the phrase table scores were the conditional phrase translation probabilities in both directions as well as the lexical weights in both directions. The language model was built by the human reference translation.

The results of our post-editing systems are outlined in Figure 6.11. To see the improvement of our post-editing system we evaluated the results of AppTek's diacritizer and translated the non-vowelized human reference text by our baseline system on character level with the phrase table created by GIZA++. Whereas the rule-based diacritizer came off with a WER of 53.2% for the part that we selected as test set, our baseline system trained by AppTek's data achieved a WER of 15.6%. Post-editing the output of the rule-based diacritizer with a phrase table containing the output of the rule-based system on the source side as well as adding the same non-vowelized training data into the phrase table led to a WER of 13.8%.

We also translated the test set by the baseline system on character level trained with Treebank data instead of AppTek's data to show how dependent our systems are on the training data. We report a WER of 70.2%. It is important to train the statistical diacritization systems with the appropriate data to obtain good results.

## 6.2 The Conditional Random Fields System

To overcome the limits of the statistical machine translation concepts we determined to explore the integration of grammatical information into a statistical approach for the diacritization with the help of conditional random fields.

In the subsequent approach the intention was to improve the restoration of diacritics by making the decision how to diacritize dependent on grammatical information. We intended to establish dependencies between words and part-of-speech tags and so influence the word endings.

### 6.2.1 Part-of-Speech Tagging

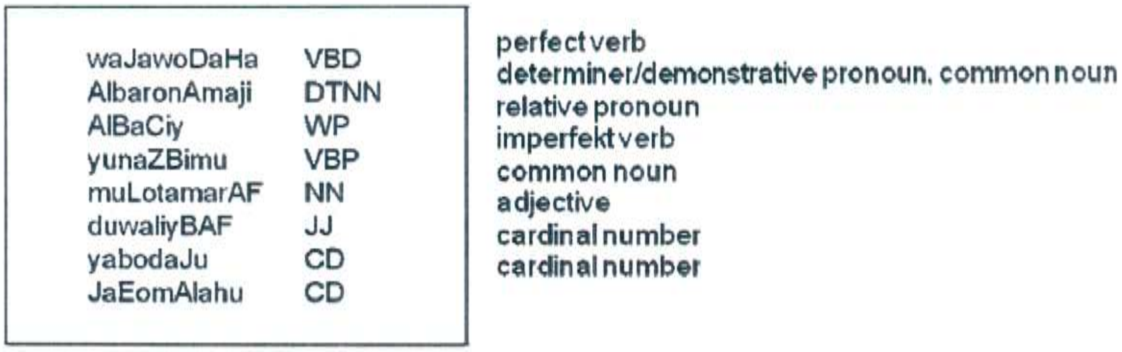
The parts of speech are assigned to each word by the Stanford Part-of-Speech Tagger [TM00], introduced in Section 5.6. The tagger uses a subset of the POS tags employed in the Penn English Treebank. Figure 6.11 shows an example for the POS Tags created by the Stanford Arabic Tagger.

The Stanford Part-of-Speech Tagger exclusively adds tags to unvowelized Arabic text. To establish the training data we define dependencies between the POS tags, the non-vowelized and the vowelized Arabic text with the aid of conditional random fields.

### 6.2.2 Conditional Random Field Approach

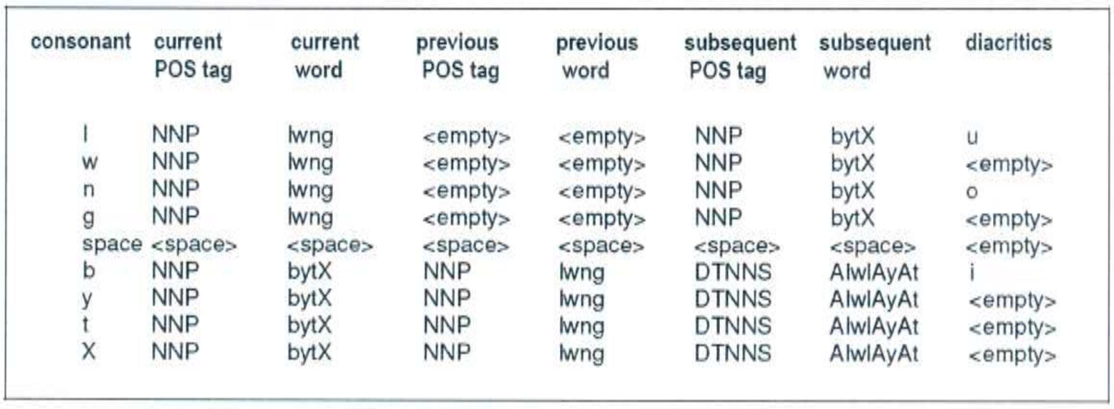
CRF++ assigned the diacritics to the consonants on character level [LMP01]. Beside the non-vowelized input the output sequence was supposed to depend on current part-of-speech information as well as on previous and subsequent word and POS information.

While the assignment of the output sequence to our test set takes only a few seconds, the training to establish the dependencies can be very resource-intensive depending on the



| | | |
|---|---|---|
| waJawoDaHa | VBD | perfect verb |
| AlbaronAmaji | DTNN | determiner/demonstrative pronoun, common noun |
| AlBaCiy | WP | relative pronoun |
| yunaZBimu | VBP | imperfekt verb |
| muLotamarAF | NN | common noun |
| duwaliyBAF | JJ | adjective |
| yabodaJu | CD | cardinal number |
| JaEomAlahu | CD | cardinal number |

Figure 6.11: Example for POS Tags in Arabic.



| consonant | current POS tag | current word | previous POS tag | previous word | subsequent POS tag | subsequent word | diacritics |
|---|---|---|---|---|---|---|---|
| l | NNP | lwng | <empty> | <empty> | NNP | bytX | u |
| w | NNP | lwng | <empty> | <empty> | NNP | bytX | <empty> |
| n | NNP | lwng | <empty> | <empty> | NNP | bytX | o |
| g | NNP | lwng | <empty> | <empty> | NNP | bytX | <empty> |
| space | <space> | <space> | <space> | <space> | <space> | <space> | <empty> |
| b | NNP | bytX | NNP | lwng | DTNNS | AlwlAyAt | i |
| y | NNP | bytX | NNP | lwng | DTNNS | AlwlAyAt | <empty> |
| t | NNP | bytX | NNP | lwng | DTNNS | AlwlAyAt | <empty> |
| X | NNP | bytX | NNP | lwng | DTNNS | AlwlAyAt | <empty> |

Figure 6.12: Training Data for CRF++.

number of defined features. For example the training with part-of-speech tags and dependencies on the previous and subsequent 4 characters took about 20 hours on a machine with 4 processors and 16 GB memory.

The format of our file to train CRF++ is allegorized in Figure 6.12. In the first column the observation sequence with the consonants is listed. In the following columns POS tags of the current, subsequent and previous word are itemized, current, subsequent and previous words themselves and finally the correct diacritic belonging to the current consonant. If there is no entry in a column the word "<empty>" is inserted.

In a template file we described the dependencies by regarding the training data as a coordinate system and indicating the relative positions of a dependent entry to the current consonant. For example the line *U03:%x[0,0]/%x[1,0]* in the template file for the training data in Figure 6.12 defines dependencies between each character and the current POS tags in the same line. Thus in the training a conditional probability is calculated for the diacritic "u" in case of the consonant "l" and the POS tag "NNP" adjacent on the right. *U03* is a unique identifier for a line in the template file. In Figure 6.13 we set an example how the dependencies can be defined.

Since the CRF training led to memory constraints we had to use only the most effective features. Furthermore, we reduced the amount of words in the training data to 75% to be able to define further dependencies for the context.

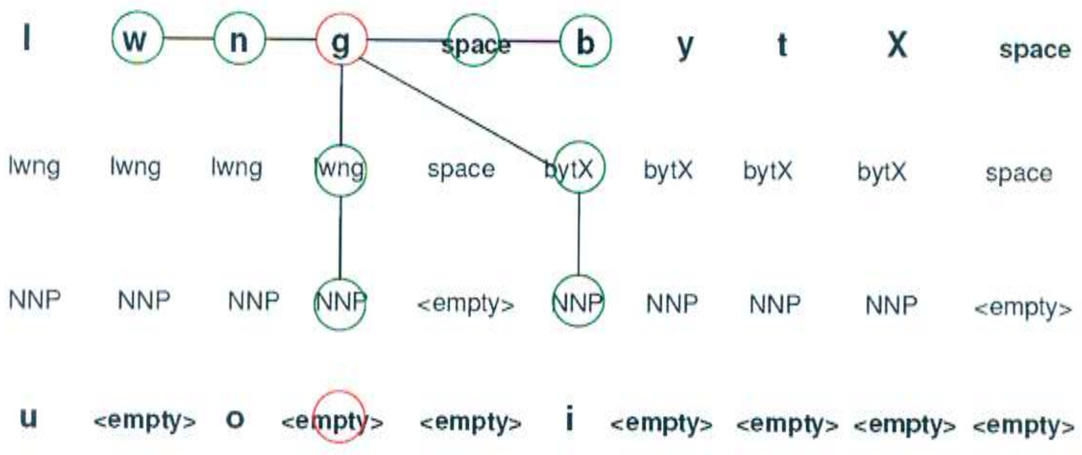

l w n g space b y t X space
lwng lwng lwng wng space bytX bytX bytX bytX space
NNP NNP NNP NNP <empty> NNP NNP NNP NNP <empty>
u <empty> o <empty> <empty> i <empty> <empty> <empty> <empty>

Figure 6.13: Dependencies with CRFs.

Figure 6.12 demonstrates the best results of our CRF systems. With a character context of 2 neighbors on the left and right (char context = 4) we archieved a WER of 22.8%. We reduced the amount of training data and obtained space in the memory to enlarge the character context up to 6 neighbors on the left and right (char context = 12). By the system with the character context of 12 which has a WER of 21.9% and a DER of 4.7% we could archieve a slightly lower diacritization error rate than in our character-based baseline translation system.

## 6.3 Results

After having described our experiments we analyze the methods which we have proposed. This section starts with a comparison of the diacritization systems which we established. We proceed with the convenience of the techniques. The execution time and the disk space for the systems are checked against each other. Finally we look at the residual errors in the diacritization and reflect how we may attain improvements.

### 6.3.1 Comparison

We proposed several statistical methods for the diacritization of Arabic text. Table 6.13 gives an overview of the results of the systems that represent the different diacritization methods. The translation systems and the conditional random fields system are trained, tuned and translated by LDC's Arabic Treebank data. The post-editing system and the system on word level which is comparable to the post-editing system are trained, tuned and translated with the data provided by the company AppTek.

Two translation baseline systems were established, a system on word level and one on character level. We improved the system on character level by enlaring the phrase table and integrating additional lexical weights beside the phrase translation probabilities. By using a 7-gram phrase table instead of a 5-gram phrase table we report an improvement of 0.2%. The lexical scores reduce the WER of 1% further.

We inserted whole words into the phrase table and the language model of the character-based system. This modification allowed us to lower the word error rate at the word ending by 1.2%. However, the other numbers increased.

| | | CRF SYSTEMS | | | | | |
|---|---|---|---|---|---|---|---|
| Char Context | | 4 | 4 | 6 | 8 | 10 | 12 |
| Training Data | | 613 k 100% | 460 k 75% | 460 k 75% | 460 k 75% | 460 k 75% | 460 k 75% |
| Test Data | | 23 k | 23 k | 23 k | 23 k | 23 k | 23 k |
| final vow | WER | **22.8** | **24.1** | **22.6** | **22.2** | **22.0** | **21.9** |
| | DER | **5.1** | **5.4** | **4.9** | **4.8** | **4.7** | **4.7** |
| no final vow | WER | **9.4** | **10.0** | **8.5** | **8.3** | **8.3** | **8.4** |
| | DER | **22.2** | **2.4** | **2.0** | **1.9** | **1.9** | **2.0** |

Table 6.12: Results of CRF Systems.

The creation of a hybrid system that operates on both, the character and the word level, led to an enhancement of 1.7% in WER as well as in the error rates at the word stems compared to the character-level baseline system. We expanded the system which obtained a character language model so far with a word language model. With that system we achieved a WER of 19.9%, the lowest WER of our systems, which is 1.9% lower than in the character baseline system.

With the CRF system where we integrated global features we gained a WER of 21.9%. Exclusively in DER we report a reduction of 0.1% in comparison to the baseline system on character level.

Post-editing the output of AppTek's rule-based diacritizer improved the system that is trained, tuned and translated by the same data by 1.8%.

In the next section we are concerned with the convenience of the features.

### 6.3.2 Convenience

An application field of the diacritization of Arabic text may be the improvement of the machine translation from Arabic into another language and back. For example, a foreign tourist can negotiate his way in an Arabic country with the help of a small device which runs a machine translation task [FWS+01]. Soldiers in Arabic-language war zones, where it is too dangerous for human translators, are enabled to communicate with the local population with the aid of machine translation devices [RMKM06]. Usually such small hand-held or wearable devices have limited resources since the size of the devices is supposed to be small, the weight to be light and the power consumption to be low. The diacritization as a task among speech recogition, machine translation and speech synthesis tasks should not require too much time and resources.

So features that lower the error rates only slightly but blow up the running time and the disk space are less convenient. We took the time for the translation of our Treebank test set with 2 k sentences and the 2 k test sentences by AppTek. Furthermore, the size of the systems is analyzed.

| | | BASELINE SYSTEM ON WORD LEVEL | BASELINE SYSTEM ON CHAR LEVEL | 7-GRAM PHRASE TABLE | 7-GRAM PHRASE TABLE, LEXICAL SCORE | CHARS IN CONTEXT WITH THE NEXT WHOLE WORD | SYSTEM ON BOTH LEVELS, 5-GRAM PHRASE TABLE | SYSTEM ON BOTH LEVELS, WORD LM | CRF, CHAR CONTEXT 12 | CHAR-BASED SYSTEM (APPTEK) DATA) | POST-EDITING (APPTEK DATA) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| final vow | WER | 22.8 | 21.8 | 21.6 | 21.5 | 21.5 | 20.1 | 19.9 | 21.9 | 15.6 | 13.8 |
| | DER | 7.4 | 4.8 | 4.8 | 4.7 | 7.1 | 4.3 | 4.3 | 4.7 | 5.5 | 4.9 |
| no final vow | WER | 9.9 | 7.4 | 7.5 | 7.4 | 10.5 | 6.6 | 6.8 | 8.4 | 10.3 | 9.3 |
| | DER | 4.3 | 1.8 | 1.9 | 1.8 | 4.4 | 1.6 | 1.6 | 2.0 | 3.5 | 3.2 |

Table 6.13: Results of the Diacritization Systems.

| | BASELINE SYSTEM ON WORD LEVEL | BASELINE SYSTEM ON CHAR LEVEL | 7-GRAM PHRASE TABLE | 7-GRAM PHRASE TABLE, LEXICAL SCORE |
|---|---|---|---|---|
| Running Time | 00:02:12 h | 00:51:29 h | 01:28:59 h | 01:57:27 h |

| | CHARS IN CONTEXT WITH THE NEXT WHOLE WORD | SYSTEM ON BOTH LEVELS, 5-GRAM PHRASE TABLE | SYSTEM ON BOTH LEVELS, WORD LM | CRF, CHAR CONTEXT 12 |
|---|---|---|---|---|
| Running Time | 1:04:23 h | 1:36:37 h | > 1:36:37 h | 2:45:57 h |

| | CHAR-BASED SYSTEM (APPTEK) DATA) | POST-EDITING (APPTEK DATA) |
|---|---|---|
| Running Time | 00:05:55 h | 00:03:58 h |

Table 6.14: Running Times (in hh:mm:ss).

#### 6.3.2.1 Running Time

In Table 6.14 an overview of the duration of our machine translation systems is given. Figure 6.14 contrasts the running times of the systems trained and tested with Treebank data with their word error rates. The time to diacritize 2 k sentences was stopped on an Intel(R) Xeon(TM) Dual Core CPU, with each 3.60 GHz. The machine has 6 GB memory. The parallel processing was not used.

With a running time of 00:02:12 h the fastest of our diacritizers is the baseline system on word level. Since the Arabic text on character level is divided into noticeably more tokens than the text consisting of words, much more time is required by the character-based systems to decode. Thus with a running time of 00:51:29 h the baseline system on character level takes 23 times longer than the system on word level.

With the additional phrases of the length of 6 and 7 tokens in the phrase table of the character-based system, the running time expands by 00:37:30 h. The incorporation of the lexical scores lengthens the time by 00:28:28 h.

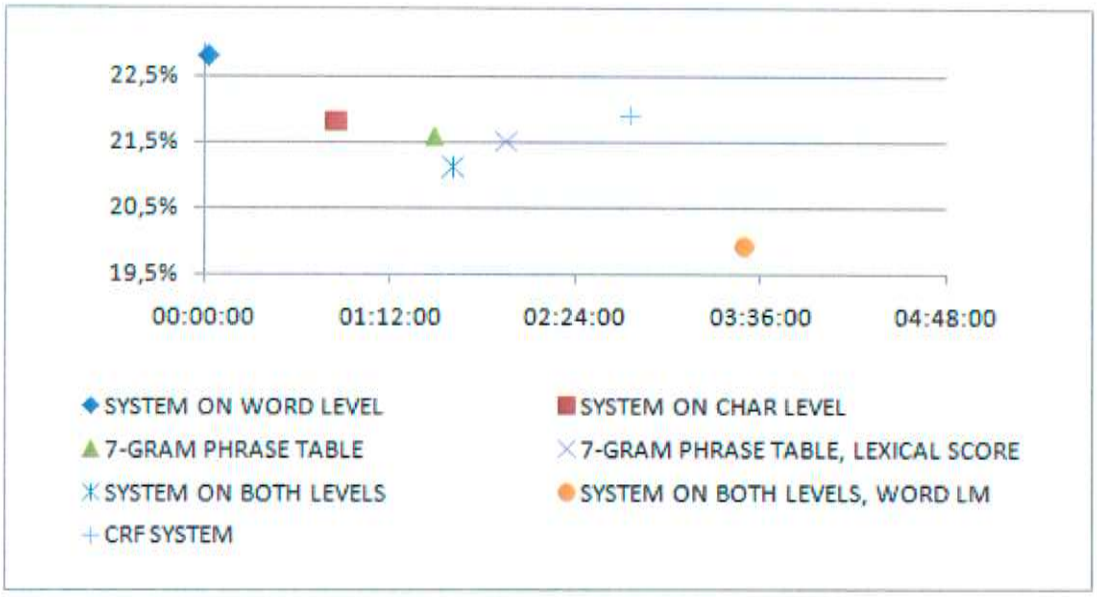


Figure 6.14: Running Time - WER.

The system which deals with characters in context with the next whole word obtains a running time of 1:04:23 h. Operating on word and character level takes 1:36:37 h. Approximately 4 minutes of that is due to the creation of the lattice. The computation of the word language model scores, the rescoring and the reordering extends the time by at least 2 hours depending on the n-gram size.

The diacritization by our CRF systems consists of two steps: First the allocation of the part-of-speech tags. For that a duration of about 2.75 hours is needed. The second step is the real labeling sequence process that takes only about 20 seconds.

The post-editing system deals with characters. AppTek's sentences are significant shorter than the Treebank sentences. The running time of our post-editing takes 00:03:58 h, somewhat shorter than the character-based translation system trained and tuned with the human reference and the vowelized training data. The post-editing system is faster since less entries are located in the phrase table which the system needs to browse through. Many possible forms are excluded by the rule-based diacritization. Unfortunatelly, information about the running time of AppTek's rule-based diacritizer is not available.

In addition to the running times of our systems we detected their disk spaces, as described in the next section.

#### 6.3.2.2 Size

To small devices which need disk space for the files required for speech recognition, machine translation and speech synthesis, a small diacritization system is advantageous. In Table 6.15 we show the disk spaces of the diacritization systems which we propose. The size information is rounded to MB. In Figure 6.15 we contrast the disk spaces of the systems established with Treebank data with their word error rates.

The systems which solve the diacritization as a statistical translation task require disk space for the phrase table, the language model, the decoder as well as corresponding scripts and parameter files. In the hybrid system with the word language model additional files for the lattice, the n-best list and the language model scores are saved on the disk.

| | BASELINE SYSTEM ON WORD LEVEL | BASELINE SYSTEM ON CHAR LEVEL | 7-GRAM PHRASE TABLE | 7-GRAM PHRASE TABLE, LEXICAL SCORE |
|---|---|---|---|---|
| Disk Space | 141 MB | 86 MB | 230 MB | 261 MB |

| | CHARS IN CONTEXT WITH THE NEXT WHOLE WORD | SYSTEM ON BOTH LEVELS, 5-GRAM PHRASE TABLE | SYSTEM ON BOTH LEVELS, WORD LM | CRF, CHAR CONTEXT 12 |
|---|---|---|---|---|
| Disk Space | 220 MB | 310 MB | 473 MB | 792 MB |

| | CHAR-BASED SYSTEM (APPTEK) DATA) | POST-EDITING (APPTEK DATA) |
|---|---|---|
| Disk Space | 52 MB | 52 MB |

Table 6.15: Disk Space.

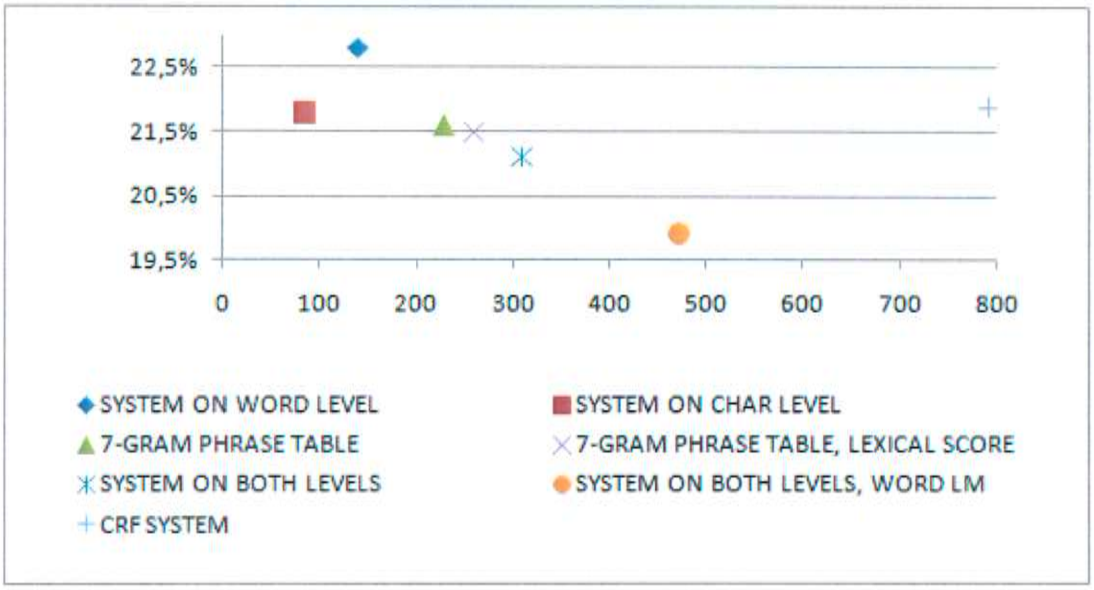


Figure 6.15: Disk Space - WER.

The CRF system lacks disk space for the model file, the part-of-speech tagger, the decoding file as well as the test file that includes the global features.

Our baseline system on word level has a size of 141 MB, while we save the baseline system on character level with only 86 MB. The reason is that the phrase table on word level contains much more characters, and therefore it is almost three times bigger than the phrase table on character level. The volume of the suffix array word language model files is 7.6 MB, for the character language model files it is 33.6 MB.

A phrase table on character level that includes 6 and 7 character phrases as well increases the volume of the system by 144 MB. Lexical scores in addition to the phrase translation probabilities in the phrase table blow it up by 31 MB.

The system which deals with characters in context with the next whole word holds a language model with 35.3 MB and a phrase table with 184 MB. In total, we calculate a size of 220 MB.

The system on both levels needs a disk space of 310 MB. The hybrid system with language model on word level saves the 1000-best lists with 154 MB and the word language model scores for each sentence with 8 MB. The sizes of the 1000-best lists and of the word language model scores depend on the amount and the length of the test sentences.

The biggest diacritization system is our CRF system which assigns diacritics depending on current, previous and subsequent words and part-of-speech tags as well as on a character context of up to 6 neighbors on the left and right. The model file has a size of 784 MB and the test file size with the global features is 6 MB. The decoding file has a size of 5 KB. The Stanford Part-of-Speech Tagger requires 1.5 MB.

Our post-editing system which is trained and tuned by AppTek's data has a volume of 52 MB. The comparable word-based system obtains the same size. As for the running time, we have no information about the disk space required byAppTek's rule-based diacritizer.

In the next section we observe the residual errors of our systems with the lowest error rates.

### 6.3.3 Analysis of Residual Errors

In order to reveal the weak points of our diacritizers we analyze the hypothesis of our system which scores the smallest error rates and is trained, tuned and translated by Treebank data as well as the hypothesis of our post-editing system established by AppTek's data.

The results of the systems show that more errors occur at the word endings than at the stems. If the errors at the word endings were eliminated, our diacritizers would outperform the state of the art systems, as described in Chapter 2. Therefore, we focus on the word endings. A reduction of the residual errors at the stems may be accomplished by training the system with more data which would enhance the statistical models.

Our best system which makes use of the Treebank data is the hybrid system with the language model on word level. Table 6.16 and Table 6.17 itemize the 10 most occurring word endings of the hypothesis and the reference in our modified Buckwalter Transliteration.

In our approaches to weight the word endings in Section 6.1.6.1 we were not able to improve the generation of the word endings "pi" (ta marbouta with kasra) and "i" (kasra) which is a huge part of our residual errors by statistical methods.

When comparing Table 6.16 and Table 6.17 to Table 6.18 and Table 6.19 it becomes clear that the creation of the word endings can be controlled much better by a set of rules. For example, Table 6.18 and Table 6.19 show that the word ending "pi" is issued 2.323% too frequently and "i" 6.271% too frequently by the character-based system trained and tuned by AppTek's data. With the help of the rules of AppTek's diacritizer the generation of "pi" is only 0.02% and the restoration of "i" only 0.781% too high. Moreover, there are further word endings whose generation is significantly closer to the probability distribution of the human reference translation with the help of rules such as "A" (alef) and "na" (nuun with fatha).

It is evident that rules, which for example define the generation of the final damma ("u") which indicates the determined nominative case and the kasra ("i") which defines the determined genitive case, support the parts where our statistical methods are not able to achieve much due to not enough training data or probabilities which are very close to each other.

We conclude the thesis in the next chapter and point at future work based on our results.

| Distribution of the Word Endings in the Hypothesis | | Distribution of the Word Endings in the Reference | |
|---|---|---|---|
| pi | 10.477 | pi | 8.508 |
| y | 6.876 | y | 6.890 |
| A | 6.477 | A | 6.432 |
| n | 4.906 | n | 4.956 |
| Y | 4.459 | Y | 4.436 |
| na | 3.285 | na | 3.184 |
| ti | 2.590 | AF | 2.394 |
| AF | 2.349 | ti | 2.251 |
| ri | 2.201 | pK | 2.054 |
| li | 2.173 | t | 2.048 |

Table 6.16: Distribution of the Final Consonants and Diacritics (Treebank).

| Distribution of the Word Endings in the Hypothesis | | Distribution of the Word Endings in the Reference | |
|---|---|---|---|
| i | 35.961 | i | 30.402 |
| a | 15.117 | a | 16.925 |
| u | 7.958 | u | 10.333 |
| y | 4.906 | y | 6.890 |
| A | 4.459 | A | 6.432 |
| K | 3.285 | K | 5.520 |
| n | 2.590 | n | 4.956 |
| Y | 2.349 | Y | 4.436 |
| F | 2.201 | F | 3.519 |
| t | 2.173 | t | 2.048 |

Table 6.17: Distribution of the Final Real Characters (Treebank).

| Distribution of the Word Endings in the | | | | | |
|---|---|---|---|---|---|
| Hypothesis of the Char-Based system | | Hypothesis of the Post-Editing system | | Human Reference Translation | |
| A | 12.993 | A | 13.117 | A | 13.199 |
| n | 7.299 | na | 6.497 | na | 5.469 |
| pi | 6.065 | ni | 5.715 | ni | 5.160 |
| y | 5.037 | y | 4.400 | y | 4.502 |
| na | 4.770 | pi | 3.762 | pi | 3.742 |
| ni | 4.544 | no | 3.166 | ti | 3.475 |
| t | 4.071 | Y | 3.166 | Y | 3.166 |
| AF | 3.187 | mo | 2.632 | no | 3.104 |
| Y | 3.166 | tu | 2.303 | mo | 2.467 |
| ri | 3.104 | nBa | 2.282 | n | 2.241 |

Table 6.18: Distribution of the Final Consonants and Diacritics (AppTek).

| Distribution of the Word Endings in the | | | | | |
|---|---|---|---|---|---|
| Hypothesis of the Char-Based system | | Hypothesis of the Post-Editing system | | Human Reference Translation | |
| i | 29.852 | a | 25.658 | i | 23.581 |
| a | 15.275 | i | 22.800 | a | 22.985 |
| A | 12.993 | A | 13.117 | A | 13.199 |
| u | 8.306 | u | 11.719 | u | 10.465 |
| n | 7.299 | o | 7.936 | o | 6.887 |
| y | 5.037 | y | 4.400 | y | 4.502 |
| t | 4.071 | Y | 3.166 | Y | 3.166 |
| F | 3.906 | F | 2.488 | F | 2.426 |
| Y | 3.166 | K | 2.179 | n | 2.241 |
| m | 2.611 | N | 1.439 | K | 2.138 |

Table 6.19: Distribution of the Final Real Characters (AppTek).

# 7. Conclusion and Future Work

In this thesis several statistical methods to diacritize Arabic text were presented. Basically, the automatic diacritization was regarded as a translation task. To integrate global features like part-of-speech information, we also solved the diacritization problem as a labeling sequence task with the help of conditional random fields.

The CMU SMT system was used as a translation tool to conduct experiments with statistical features which improve the language model and the translation model for the diacritization. To determine the optimal scaling factors for the models, we applied the Minimum Error Rate Training. We introduced further tools which support the creation of statistical diacritizers such as the Suffix Array Language Model Toolkit, SRI Language Model Toolkit, the Moses package with GIZA++, AppTek's rule-based Arabic diacritizer, the Stanford Part-of-Speech Tagger and CRF++.

The restoration of the diacritics was performed on word and character level. We realized a hybrid system by representing the test set as a lattice with edges from characters to characters and from words to words as well as a phrase table with word and character entries.

Proposals for the improvement of the language model were made by the additional word language model scores to the hybrid system. Thereby, we detected that the integration of word language model scores provided by the SRI Language Model enhances the system more than those of the Suffix Array Language Model.

With additional lexical weights and the enlargement of the phrases in the phrase table, we demonstrated a method to meliorate the translation model. In contrast to the enlargement of the context in the phrase table, the enlargement of the n-grams in the language model did not lead to lower error rates.

A system, which translates characters in context with the next word, achieved a lower word error rate than the system on character level.

We made proposals for the improvement of the diacritization at the word endings by weighting the phrase table entries with additional scores.

Furthermore, we revealed a method to integrate rules into our statistical approaches by post-editing the output of a rule-based diacritizer with our statistical translation system.

By using conditional random fields, we softened the independence conditions which we have with the SMT system and gained well defined dependencies to part-of-speech tags.

characters and words in the observation sequence and the target sequence. The experiments showed that the machine translation approaches outperform the labeling sequence approach in word and character error rates. So far we could achieve only a slightly lower diacritization error rate than in our baseline translation approaches by the CRF system. We intend to pursue the CRF research to establish dependencies on more than 6 neighbors on the left and right on character level.

Most benefits were gained by operating on both word and character level. Additionally, large n-grams in the phrase table supported the translation approach.

In post-editing the output of the rule-based system we learned that large n-grams in phrase table, phrase table probabilities and lexical scores in both directions lead to magnificent results.

We figured the running times and the disk spaces of the systems to show the convenience of the methods we are presenting. Thereby we showed that some features reduce the error rates only slightly but blow up the running time and the disk space.

With a word error rate of 19.9% in our system on word and character level we get close to state-of-the-art systems such as [ZSS06](18.0%) or [HR07](14.9%).

Further steps may be to test the combination of all features that we demonstrated and to compare these results with the results of other state-of-the-art systems. A combination system, however, would be exclusively applicable if the intention is to have a preferably accurate system since due to the size and running time it may not be convenient.

Moreover, the systems may be enhanced at a grammatical level as many errors at the word endings remain. Based on the results of our post-editing system we recommend to integrate a set of rules into a pure statistical approach.

The integration of the diacritization features in Arabic-English or English-Arabic translation systems is interesting for us. A next step may be the integration of a diacritizer into the ISL Lecture Translation System of the Carnegie Mellon University and the Universität Karlsruhe (TH) [Hel07].

Also, the intention is to improve AppTek's rule-based diacritizer by the integration of our statistical diacritization features.

# A. Buckwalter Transliteration

| | | | | | |
|---|---|---|---|---|---|
| ء | ' | ذ | * | ل | l |
| آ | \| | ر | r | م | m |
| أ | > | ز | z | ن | n |
| ؤ | & | س | s | ه | h |
| إ | < | ش | $ | و | w |
| ئ | } | ص | S | ى | Y |
| ا | A | ض | D | ي | y |
| ب | b | ط | T | ـً | F |
| ة | p | ظ | Z | ـٌ | N |
| ت | t | ع | E | ـٍ | K |
| ث | v | غ | g | ـَ | a |
| ج | j | ـ | _ | ـُ | u |
| ح | H | ف | f | ـِ | i |
| خ | x | ق | q | ـّ | ~ |
| د | d | ك | k | ـْ | o |

Figure A.1: Buckwalter Transliteration Table (according to QAMUS LLC).

# Bibliography

[ABM+93] ARNOLD, D.J. ; BALKAN, Lorna ; MEIJER, Siety ; HUMPHREYS, R.Lee ; SADLER, Louisa: *Machine Translation: an Introductory Guide*. London : Blackwells-NCC, 1993

[BPPM93] BROWN, Peter F. ; PIETRA, Vincent J. D. ; PIETRA, Stephen A. D. ; MERCER, Robert L.: The mathematics of statistical machine translation: parameter estimation. In: *Comput. Linguist.* 19 (1993), Nr. 2, S. 263–311. – ISSN 0891–2017

[Buc02] BUCKWALTER, Tim ; LINGUISTIC DATA CONSORTIUM (Hrsg.): *Buckwalter Arabic Morphological Analyzer Version 1.0*. Philadelphia: Linguistic Data Consortium, 2002. http://www.qamus.org

[DAS02] DEBILI, Fathi ; ACHOUR, Hadhemi ; SOUISSI, Emna: De l'etiquetage grammatical á la voyellation automatique de l'arabe / Correspondances de l'Institut de Recherche sur le Maghreb Contemporain 17. 2002. – Forschungsbericht

[DGH07] DIAB, Mona ; GHONEIM, Mahmoud ; HABASH, Nizar: Arabic Diacritization in the Context of Statistical Machine Translation. Copenhagen, Denmark, 10-14 September 2007 (Proceedings), S. 143–149

[EAMA06] ELSHAFEI, Moustafa ; AL-MUHTASEB, Husni ; ALGHAMDI, Mansour: Statistical Methods for Automatic Diacritization of Arabic Text. In: *The Saudi 18th National Computer Conference*. Riyadh, 2006 (18), S. 301–306

[EF04] EMAM, Ossama ; FISCHER, Volker: Hierarchical approach for the statistical vowelization of arabic text / IBM patent filed. DE9-2004-0006. 2004. – Forschungsbericht

[EI03] EL-IMAM, Yousif A.: Phonetization of Arabic: rules and algorithms. P.O. Box 27272, Sharjah, United Arab Emirates, 2003

[Elb04] ELBEHERI, Gad ; SONS, Chichester: Wiley (Hrsg.): *Dyslexia in Egypt*. International Book of Dyslexia: A cross langauge comparison and practice guide (Second Edition). In Smythe, Everatt & Salter (Eds), 2004

[ESH89] EL-SADANY, T. A. ; HASHISH, M. A.: An Arabic morphological system. In: *IBM Syst. J.* 28 (1989), Nr. 4, S. 600–612. – ISSN 0018–8670

[EVW04] ECK, Matthias ; VOGEL, Stephan ; WAIBEL, Alex: Improving statistical machine translation in the medical domain using the unified medical language system. In: *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*. Morristown, NJ, USA : Association for Computational Linguistics, 2004, S. 792

[FKPW06] FÜGEN, Christian ; KOLSS, Muntsin ; PAULIK, Matthias ; WAIBEL, Alex: Open Domain Speech Recognition & Translation: Lectures and Speeches. In: *In Proceedings of the TC-STAR Workshop on Speech-to-Speech Translation*. Barcelona, Spain, June 2006

[FWS$^+$01] FÜGEN, C. ; WESTPHAL, M. ; SCHNEIDER, M. ; SCHULTZ, T. ; WAIBEL, A: LingWear: a mobile tourist information system. In: *Proceedings of the First international Conference on Human Language Technology Research*. Morristown, NJ : Association for Computational Linguistics, March 18 - 21 2001

[Gal02] GAL, Ya'akov: An HMM approach to vowel restoration in Arabic and Hebrew. In: *Proceedings of the ACL-02 workshop on Computational approaches to semitic languages*. Morristown, NJ, USA : Association for Computational Linguistics, 2002, S. 1–7

[Gha06] GHARIEB, Dr. Gharieb M. ; SELBSTVERLAG (Hrsg.): *Arabisch für alle*. 2006 (ISBN: 3-00-017880-5)

[Hel07] HELAOUI, Rim: *Building an English-to-Arabic Statistical Machine Translations System for the ISLs Lecture Translator*. Universität Karlsruhe (TH) Institut für Theoretische Informatik Carnegie Mellon University (CMU) Language Technology Institute (LTI), June 2007

[HR07] HABASH, Nizar ; RAMBOW, Owen: Arabic Diacritization through Full Morphological Tagging. In: *Proceedings of NAACL/HLT (Companion Volume, Short Papers)*, 2007

[Hut03] *Kapitel* The Oxford Handbook of Computational Linguistics, Chapter 27. In: HUTCHINS, John: *Machine translation: general overview*. Oxford: University Press, 2003, S. 501–511

[KK03] KNIGHT, Kevin ; KOEHN, Philipp: What's new in statistical machine translation. In: *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*. Morristown, NJ, USA : Association for Computational Linguistics, 2003, S. 5–5

[Kni99] KNIGHT, Kevin: A Statistical MT Tutorial Workbook. In: *prepared in connection with the JHU summer workshop*, 1999

[Koe06] KOEHN, Philipp ; UNIVERSITY OF EDINBURGH (Hrsg.): *Pharao Training Manual Version 1.3*. Scotland: University of Edinburgh, 2006

[LMP01] LAFFERTY, John D. ; MCCALLUM, Andrew ; PEREIRA, Fernando C. N.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2001. – ISBN 1–55860–778–1, S. 282–289

[MBK06] MAAMOURI, Mohamed ; BIES, Ann ; KULICK, Seth: Diacritization: A Challenge to Arabic Treebank Annotation and Parsing. University of Pennsylvania, USA, 2006

[Mih02] MIHALCEA, Rada: Diacritics Restoration: Learning from Letters versus Learning from Words. In: *CICLing*, 2002, S. 339–348

[Mos06] *Moses - A Factored Phrase-Based Beam-Search Decoder for MT.* : *Moses - A Factored Phrase-Based Beam-Search Decoder for MT.* JHU Summer Workshop 2006, 2006. http://www.statmt.org/moses/

[NR05] NABHAN, Ahmed R. ; RAFEA, Ahmed: Tuning Statistical Machine Translation Parameters Using Perplexity. In: *Information Reuse and Integration, Conf, 2005. IRI -2005 IEEE International Conference* Dept. of Math., Cairo Univ., Egypt, 2005

[NS05] NELKEN, Rani ; SHIEBER, Stuart M.: Arabic Diacritization Using Weighted Finite-State Transducers. In: *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*. Ann Arbor, Michigan : Association for Computational Linguistics, June 2005, S. 79–86

[Och03] OCH, Franz J.: Minimum error rate training in statistical machine translation. In: *ACL '03: Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*. Morristown, NJ, USA : Association for Computational Linguistics, 2003, S. 160–167

[Och05] OCH, Franz J.: Statistical Machine Translation: Foundations and Recent Advances. In: *Tutorial at MT Summit 2005* Google, Inc., 2005

[OTN99] OCH, F. J. ; TILLMANN, Christoph ; NEY, Hermann: Improved alignment models for statistical machine translation. In: *Proc. of the Joint Conf. of Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999, S. 20–28

[PRWZ02] PAPINENI, Kishore ; ROUKOS, Salim ; WARD, Todd ; ZHU, Wei-Jing: BLEU: a Method for Automatic Evaluation of Machine Translation. In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*,. Philadelphia, July 2002, S. 311–318

[PYZ$^+$05] PANG, Wei ; YANG, Zhendong ; ZHENBIAO, Chen ; WEI, Wei ; XU, Bo ; ZONG, Chengqing: The CASIA Phrase-Based Machine Translation System. In: *Institute of Automation Chinese Academy of Sciences* (2005)

[RMKM06] RIESA, Jason ; MOHIT, Behrang ; KNIGHT, Kevin ; MARCU, Daniel: Building an English-Iraqi Arabic Machine Translation System for Spoken Utterances with Limited Resources. In: *Proc. Interspeech*, 2006

[Sto02] STOLCKE, Andreas: SRILM - The SRI Language Modeling Toolkit / Speech Technology and Research Laboratory. Version: September 2002. http://www.speech.sri.com/projects/srilm. SRI International, Menlo Park, CA, U.S.A., September 2002. – Forschungsbericht

[SUIK07] SIMARD, M. ; UEFFING, N. ; ISABELLE, P. ; KUHN, R.: Rule-based Translation With Statistical Phrase-based Post-editing. In: *ACL 2007 Second Workshop on Statistical Machine Translation*. Prague, Czech Republic, June 23 2007, S. 203–206

[TC99] TUFIS, Dan ; CHITU, Adrian: Automatic Diacritics Insertion in Romanian Texts. In: *Proceedings of COMPLEX'99 International Conference on Computational Lexicography, Pecs*. Bucharest : Romanian Academy (RACAI), 2ICI, 1999

[TM00] Toutanova, Kristina ; Manning, Christopher D.: Stanford Arabic Parser - Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger. In: *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000).* Hong Kong, 2000

[VK04] Vergyri, Dimitra ; Kirchhoff, Katrin: Automatic Diacritization of Arabic for Acoustic Modeling in Speech Recognition. In: Farghaly, Ali (Hrsg.) ; Megerdoomian, Karine (Hrsg.): *COLING 2004 Computational Approaches to Arabic Script-based Languages.* Geneva, Switzerland : COLING, August 28th 2004, S. 66–73

[VNT96] Vogel, Stephan ; Ney, Hermann ; Tillmann, Christoph: HMM-based word alignment in statistical translation. In: *Proceedings of the 16th conference on Computational linguistics.* Morristown, NJ, USA : Association for Computational Linguistics, 1996, S. 836–841

[VZH$^+$03] Vogel, S. ; Zhang, Y. ; Huang, F. ; Tribble, A. ; Venugopal, A. ; Zhao, B. ; Waibel, A.: The CMU Statistical Machine Translation System. In: *Proceedings of MT-Summit IX, LA*, 2003

[Wal04] Wallach, Hanna M.: Conditional Random Fields: An Introduction / Department of Computer and Information Science, University of Pennsylvania. 2004. – MS-CIS-04-21

[Wu97] Wu, Dekai: Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. In: *Comput. Linguist.* 23 (1997), Nr. 3, S. 377–403. – ISSN 0891–2017

[Zak06] Zakhary, Dalal ; Linguistic Data Consortium (Hrsg.): *GALE Evaluation Reference Translation Alternation Guidelines - GALE Translation QC, Arabic-English.* Version 1.0 11/10/2006. Linguistic Data Consortium, 2006

[Zha07] Zhang, Ying: SALM: Suffix Array and its Applications in Empirical Language Processing. Version: 2007. http://projectile.is.cs.cmu.edu/research/public/tools/salm/salm.htm. 2007. – Forschungsbericht

[ZSS06] Zitouni, Imed ; Sorensen, Jeffrey S. ; Sarikaya, Ruhi: Maximum entropy based restoration of Arabic diacritics. In: *ACL '06: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL.* Morristown, NJ, USA : Association for Computational Linguistics, 2006, 577–584