Institut für Logik, Komplexität und Deduktionssysteme
der Universität Karlsruhe
Lehrstuhl Prof. Dr. A. Waibel

Statistische Verfahren zur Bestimmung der Vokalisierung arabischer Texte

Diplomarbeit

von

Amélie Deltour

Bearbeitungszeitraum: 1. Oktober 2002 - 31. März 2003

Erstgutachter: Prof. Dr. Alexander Waibel
Betreuer: Dipl.-Inform. Jürgen Reichert

Hiermit erkläre ich, die vorliegende Arbeit selbständig erstellt und keine anderen als die angegebenen Quellen verwendet zu haben.

Karlsruhe, den 15. April 2003

Amélie Deltour

Abstrakt

Arabische Texte bestehen nur aus Konsonanten, die eventuell auch als lange Vokale dienen. Kurze Vokale und andere Hilfszeichen können durch sogenannte diakritische Zeichen angezeigt werden, welche aber im Normalfall in der Schrift weggelassen werden, da sie sich für Araber meistens aus dem Kontext ergeben. Wörter ohne diakritische Zeichen sind allerdings mehrdeutig. In der Regel sind nur ca. 30% der Wörter in einem Text eindeutig. In den meisten Fällen kann nur der Kontext helfen, die richtige Vokalisierung zu bestimmen.

Diese Arbeit untersucht verschiedene statistische Verfahren für die automatische Wiederherstellung dieser diakritischen Zeichen (insbesondere der Vokale). Unter anderem wurden Verfahren, die für die Wiederherstellung diakritischer Zeichen in europäischen Sprachen entwickelt wurden, an die arabische Sprache angepasst.

Die untersuchten Verfahren verwenden einen Korpus aus vokalisierten Texten. Für Wörter die im Korpus vorkommen, wurden wortbasierte Verfahren untersucht, die auf Kollokationen von Wörtern, auf Sprachmodellen und Hidden Markov Modellen basieren. Für nicht im Vokabular enthaltente Wörter wurden Verfahren auf Buchstabenebene untersucht, zuerst mit sogenannten „gemischten Kontexten“ und dann auch mit Sprachmodellen.

Die besten Ergebnisse wurden mit Sprachmodellen erreicht. Tests ergeben eine Wortfehlerrate von ca. 17% auf dem Korpus. Auf Zeichenebene beträgt die Fehlerrate 5%. Auf Texten aus anderen Bereichen, insbesondere Alltagstexten, die sich stark vom Trainingskorpus unterscheiden, liegt die Wortfehlerrate zwischen 20% und 50%.

Abstract

Arabic texts consist only of consonants, which can also be used for long vowels. Diacritics possibly stand for short vowels and other pronunciation marks, yet these diacritics are usually omitted in the written form since Arabs can most of the time restore them directly from the context. However, words without diacritical marks are ambiguous. Usually, only approx. 30% of the words in a text are unambiguous. In most of the cases only the context can help to determine the right vocalization.

This work investigates different statistical approaches for the automatic restoration of these diacritics (in particular the vowels). Among other things, methods developed for diacritic restoration in European languages were adapted to the Arabic language.

The investigated procedures use a corpus of vocalized texts. For words that occur in the corpus, methods at word level were investigated, which are based on collocations of words and on language models with Hidden Markov Models. For out-of-vocabulary words, letter-based methods were investigated, first with so-called „mixed contexts“ and then with language models too.

The best results were obtained with language models. Tests show that the word error rate on the corpus is approx. 17%. At the diacritic level the error rate amounts to 5%. On texts from other domains, in particular everyday life texts that strongly differ from the training corpus, the word error rate lies between 20% and 50%.

Danksagung

Hiermit möchte ich mich bei allen bedanken, die mir bei der Diplomarbeit geholfen haben. Zuerst geht mein Dank an Dipl.-Inform. Jürgen Reichert, der als mein Betreuer mich bei der Arbeit begleitet hat. Dann an Prof. Waibel, ohne den diese Arbeit an dem Institut nicht möglich gewesen wäre.

Zu danken sind noch allen, die sich Zeit genommen haben, meine Fragen zu klären: Leila Chouchane, Raja Gouigem und Dr. Gharieb M. Gharieb im Bereich Arabisch, Alicia Tribble bei der Literatursuche, Prof. Dr. Klaus Lagally und Kenneth R. Beesley bei der Arbeit mit ArabTeX, John Hudson über Unicode und Andreas Stolcke über das SRILM Toolkit. Prof. Dr. Klaus Lagally und Andreas Stolcke möchte ich besonders für die Werkzeuge, die sie entwickelt und kostenlos zur Verfügung gestellt haben, danken.

Zuletzt sind allen zu danken, die bei der Ausarbeitung auf Deutsch geholfen haben.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

1.1 Aufgabenstellung und Motivation

Arabisch ist in der heutigen Zeit eine wichtige Sprache, da es von über 180 Millionen Menschen[1] gesprochen wird. Diese stammen vor allem aus den arabischen Staaten. Deshalb interessiert man sich mehr und mehr dafür, die im Bereich des *Natural Language Processing* (NLP) entwickelten Verfahren auf das Arabische anzuwenden, so zum Beispiel in der Sprachübersetzung.

Die arabische Sprache stellt aber mehrere Herausforderungen an das NLP. Eine von diesen ist, dass die sogenannten diakritischen Zeichen und im Besonderen die Vokale, in arabischen Texten meistens nicht geschrieben werden. Es wäre aufwändig diese Zeichen zu schreiben und Araber brauchen sie nicht, um den Sinn eines Textes zu verstehen. Nur in Ausnahmefällen, beispielsweise in Texten für Kinder, für Arabisch-Lernende und in religiösen Schriften werden diese Zeichen geschrieben. In Alltagstexten, wie z.B. Zeitungsartikeln, sind sie aber nie vorhanden.

Wegen dieses Fehlens der diakritischen Zeichen sind die Wörter in einem arabischen Text mehrdeutig. Nach [Beesley, 1996] gibt es durchschnittlich fünf mögliche Vokalisierungen für ein Wort. Das ist unter anderem für die morphologische Analyse des Arabischen problematisch, denn die fehlenden diakritischen Zeichen erhöhen die Zahl der möglichen Analysen für ein Wort.

Durch den Kontext, die Kenntnis des arabischen Lexikons und der arabischen Grammatik kann ein Araber die richtige Form eines Wortes erkennen. Für Leute, deren Muttersprache nicht Arabisch ist, und für computerbasierte Verfahren ist dies aber nicht so einfach.

Nehmen wir zum Beispiel den deutschen Satz:

Arbsch st n schwrg Sprch.

Wer kann diesen Satz lesen? Für einen Deutschen ist es nicht schwierig. Aber wie sieht es für einen Franzosen (ohne Deutschkenntnisse) aus? Oder gar für einen

[1] Nach `http://www.al-bab.com/arab/language/lang.htm`

Rechner? Woher soll man wissen, dass es sich um das Wort „Sprache“ handelt und nicht zum Beispiel um das Wort „Spruch“?

Eine automatische Vokalisierung[2] arabischer Texte durch den Rechner wäre für viele Anwendungen hilfreich, z.B. für Arabisch-Lernende oder für die Sprachsynthese des Arabischen. Die automatische Vokalisierung kann auch als erster Schritt betrachtet werden, um die richtige morphologische Analyse eines Wortes zu finden.

1.2 Vorgehensweise

Ziel dieser Diplomarbeit ist die Untersuchung möglicher statistischer Verfahren zur automatischen Vokalisierung arabischer Texte.

Das Prinzip statistischer Verfahren ist, wertvolle Informationen aus Statistiken über eine große Menge von Daten (den Korpus) zu extrahieren. Solche Verfahren haben sich schon für die maschinelle Übersetzung oder die Spracherkennung als sehr gut erwiesen. In [Goldsmith, 2001] wurde gezeigt, dass man mit statistischen Verfahren linguistische Informationen, nämlich Informationen über die Morphologie, automatisch aus einem Korpus gewinnen kann.

Der Vorteil von statistischen Verfahren ist also, dass keine große Wissensbasis über die Sprache vorhanden sein muss. Man braucht auch keine aufwändigen Regeln von Hand zu schreiben, wie es beispielsweise bei der wissensbasierten morphologischen Analyse von Arabisch in [Kiraz, 1995] der Fall ist.

Da für statistische Verfahren eine große Menge von Texten nötig ist, wurden zuerst diese Daten gesammelt. Als vokalisierte arabische Texte wurden die Bibel und der Koran benutzt. Sie wurden um einen kleinen Wortschatz gebräuchlicher Wörter ergänzt.

Als Referenz wurde ein *Baseline*-Verfahren entwickelt, welches Texte mit ca. 30% Fehlerrate vokalisieren kann. Anschließend wurden verschiedene Verfahren untersucht, um diese Fehlerrate zu reduzieren. Unter anderem wurde versucht, Verfahren, die für die Wiederherstellung der Akzente in europäischen Sprachen entwickelt wurden, an Arabisch anzupassen. Für Wörter, die im Korpus vorkommen, wurden wortbasierte Verfahren verwendet, während unbekannte Wörter nach buchstabenbasierten Verfahren vokalisiert wurden. Auf Wortebene wurden zwei Verfahren untersucht: eines mit einem Kollokationsmodell und eines mit einem Sprachmodell. Auf Buchstabenebene wurden zwei weitere Verfahren entwickelt.

[2]Mit Vokalisierung wird hier allgemein die Wiederherstellung der diakritischen Zeichen gemeint

1.3 Inhaltsübersicht

Zuerst werden im Kapitel 2 die Grundlagen der arabischen Sprache vorgestellt, die der Leser benötigt, um die folgende Arbeit zu verstehen.

Im Kapitel 3 werden Arbeiten anderer Autoren erläutert, die sich schon mit dem Thema oder damit verbundenen Fragestellungen beschäftigt haben. Einige davon haben als Basis dieser Arbeit gedient. Sie ermöglichen auch einen Vergleich mit den hier gefundenen Ergebnissen.

Kapitel 4 beschreibt den Korpus. Die Quellen der Daten und ihre Besonderheiten werden vorgestellt sowie die jeweils nötige Vorverarbeitung. Ferner wird auf Problemstellungen eingegangen, die sich bei der Datensammlung ergeben haben.

Im Kapitel 5 werden die verschiedenen Verfahren erklärt, die untersucht wurden. Theoretische Grundlagen und deren Einsatz werden präsentiert, ebenso Experimente und deren Ergebnisse.

Kapitel 6 stellt die abschließenden Ergebnisse des entwickelten Systems vor und vergleicht diese mit anderen Arbeiten. Diese Resultate erlauben es, eine Bewertung des Systems vorzunehmen.

Schließlich fasst Kapitel 7 die Arbeit zusammen und ermöglicht einen Ausblick auf offengebliebene Fragen und mögliche zukünftige Erweiterungen.

Kapitel 2

Grundlagen der arabischen Sprache

Für den Leser, der mit dem Arabischen nicht vertraut ist, enthält dieses Kapitel eine kurze Einleitung um die folgende Arbeit besser zu verstehen. Eine ausführlichere Einführung in die Arabische Sprache kann man in [Brockelmann, 1965] oder [Gharieb, 1996] finden. Hier werden lediglich die Grundlagen angegeben, die für das Verstehen der Arbeit nötig sind, im Besonderen die arabische Rechtschreibung.

2.1 Die arabische Sprache

Arabisch ist eine semitische Sprache, wie auch Hebräisch. Beide Sprachen haben viele Ähnlichkeiten: sie werden von rechts nach links geschrieben und haben eine Konsonantenschrift (nur Konsonanten werden geschrieben, siehe 2.2).

In den arabischen Staaten werden im Alltag Mundarten gesprochen. Es gibt eine Vielzahl solcher Mundarten, die so unterschiedlich sind, dass Sprecher aus verschiedenen Regionen sich nicht verstehen können. Für das öffentliche und religiöse Leben wird aber das sogenannte Hocharabisch benutzt. Hocharabisch ist die Sprache, die in allen arabischen Staaten in der Schule gelehrt wird. Sie ermöglicht also die Kommunikation zwischen Sprechern mit verschiedenen Mundarten. In Zeitungen wird ebenfalls Hocharabisch verwendet.

Wenn in dieser Arbeit der Ausdruck „Arabisch“ benutzt wird, wird immer Hocharabisch gemeint.

2.2 Die arabische Schrift

Wie schon erwähnt, hat Arabisch eine Konsonantenschrift. Das heißt, die Buchstaben sind fast ausschließlich Konsonanten. Die kurzen Vokale und andere Hilfszeichen für das Lesen werden mit sogenannten diakritischen Zeichen über und unter den Buchstaben gekennzeichnet.

Die Buchstaben Im Arabischen gibt es 28 Konsonanten und einen Kehlkopfverschlusslaut (*hamza*).

Desweiteren gibt es noch einige Buchstaben, die abgeleitete Formen von den anderen sind (z.B. die Femininendung *tāʾ marbūṭa*).

Für eine Übersicht der arabischen Buchstaben, siehe Tabelle A.1.

Die Vokale Es gibt im Arabischen lange und kurze Vokale.

Für die langen Vokale verwendet man Buchstaben:

- ا (*ʾalif*) für das lange *ā*;
- ي (*yāʾ*) für das lange *ī*;
- و (*wāw*) für das lange *ū*.

Wenn diese Buchstaben als lange Vokale verwendet werden, befindet sich vor ihnen der entsprechende kurze Vokal: z.B. بَا *bā*, بِي *bī*, بُو *bū*.

Die kurzen Vokale werden durch kleine Zeichen (meistens Striche) über und unter der Schriftlinie angezeigt: z.B. بَ *ba*, بِ *bi*, بُ *bu*.

Die diakritischen Zeichen Das Arabische verwendet orthographische Hilfszeichen für die Wiedergabe von einigen Aussprachen:

- *fatḥa*, *kasra* und *ḍamma* für die kurzen Vokale;
- *fatḥa tawīla*, *kasra tawīla*, *ḍamma tawīla* für die Nunation, d.h. einen Vokal, der mit einem folgenden *nūn* (*n*) ausgesprochen wird. Das Zeichen ist eine Verdopplung vom entsprechenden Vokalzeichen. Diese Zeichen sind lediglich als Endungen zu finden;
- *sukūn* für die Vokallosigkeit (wenn ein Konsonant ohne Vokal auszusprechen ist);
- *šadda* (auch *tašdīd* genannt) für die Verdopplung eines Konsonanten;
- *waṣla* für die Verbindung (um einen *ʾalif* anzuzeigen, der als Verbindungs-*ʾalif* dient und nicht auszusprechen ist).

Die Tabelle 2.1 gibt eine Zusammenfassung des Aussehens und der Aussprache dieser Zeichen.

Bei jedem Konsonanten in einem arabischen Wort ist eventuell ein Verdopplungszeichen und auf jeden Fall ein Vokalzeichen (oder Vokallosigkeitszeichen) zu finden, außer bei Buchstaben, die als lange Vokale dienen. Es gibt auch besondere Fälle, in denen ein Buchstabe kein Vokalzeichen haben darf (z.B. beim Verbindungs-*ʾalif* wenn er nicht am Anfang des Satzes steht).

Name	Zeichen	Aussprache
fatḥa	ـَ	*a*
kasra	ـِ	*i*
ḍamma	ـُ	*u*
fatḥa tawīla	ـً	*an*
kasra tawīla	ـٍ	*in*
ḍamma tawīla	ـٌ	*un*
sukūn	ـْ	Vokallosigkeit
šadda	ـّ	Verdopplung
waṣla	ٱ	Verbindung

Tabelle 2.1: Diakritische Zeichen im Arabischen

Außer *šadda* (siehe 2.4 und 2.5) werden diese diakritischen Zeichen in alltäglichen Texten (Zeitungen, Romane, Briefe, ...) weggelassen. Sie werden lediglich in religiösen Texten (z.B. im Koran), in alten Gedichten, in Kinderbüchern oder bei Bedarf in schwierigen Texten (wenn ein Araber selbst den Sinn nicht klar finden kann) geschrieben.

Hier ist ein Beispiel eines Satzes, zuerst mit und dann ohne diakritische Zeichen geschrieben (Arabisch wird von rechts nach links geschrieben und die Transkription von links nach rechts):

يَتَحَدَّثُ ٱلْوَزِيرُ مَعَ ٱلسَّفِيرِ.

yataḥaddaṯu 'l-wazīru maʿa 's-safīri.

يتحدّث الوزير مع السفير.

'Der Minister diskutiert mit dem Botschafter.'

2.3 Vokalisierung des Arabischen

In dieser Arbeit werden wir Texte mit diakritischen Zeichen als „vokalisierte Texte" und Texte ohne diakritische Zeichen als „nicht vokalisierte Texte" bezeichnen.

Ein Araber selbst kann nicht eindeutig definieren, was eine vokalisierte Form und was eine nicht vokalisierte Form ist. Es gibt nämlich verschiedene Ebenen der Vokalisierung.

2.3.1 Vokalisiertes Arabisch

Meistens werden bei vokalisierten Formen nicht alle Zeichen geschrieben, sondern nur die wichtigsten. In Wörterbüchern, in denen die Wörter vokalisiert sind, gibt es beispielsweise kein *fatḥa* (*a*) Zeichen vor dem langen Vokal *ʾalif* (*ā*), weil dieses Zeichen als implizit betrachtet wird. Dieses überflüssige Zeichen wird fast nie geschrieben, ebenso das *waṣla* Zeichen. Es kann auch sein, dass Konsonanten ohne Vokalzeichen geschrieben werden, anstatt den *sukūn* zu tragen.

Im Koran und in der Bibel werden normalerweise alle Zeichen geschrieben. Diese Texte sind die einzigen, bei denen die Vokalisierung so vollständig ist. Bevor eine Version des Koran oder der Bibel veröffentlicht werden darf, muss der Text von verschiedenen Personen sorgfältig überprüft werden, um sicher zu sein, dass er keine Fehler enthält.

2.3.2 Nicht vokalisiertes Arabisch

Auch für nicht vokalisierte Texte, die im Alltag zu finden sind, gibt es mehrere Möglichkeiten: sie können einige Zeichen tragen oder nicht. In Lehrbüchern wird zum Beispiel das Zeichen bei der Endung ـًا auch in der nicht vokalisierten Form geschrieben.

In vielen Texten wird die Vokalisierung folgendermaßen behandelt:

- Das *šadda* Zeichen, das die Assimilation (siehe 2.4) anzeigt, wird nicht geschrieben. Im Beispiel von S. 7 ist السفير die nicht vokalisierte Form für اَلسَّفِيرِ.

- Das *šadda*, das aus morphologischen Gründen auf einem Konsonant steht (siehe 2.5), wird geschrieben. Im Beispiel von S. 7 ist يتحدّث die nicht vokalisierte Form für يَتَحَدَّثُ.

- Vokalzeichen werden nicht geschrieben, außer wenn mehrere Vokalisierungen möglich sind, die zu einem unterschiedlichen Sinn führen. Um Eindeutigkeit zu gewährleisten, kann in diesem Fall ein Vokalzeichen geschrieben werden.

Es ist häufig auch der Fall, dass nur die Verdopplungszeichen (*šadda*), die wirklich benötigt werden, geschrieben werden. Verdopplungszeichen, welche einem arabischen Leser klar sind, werden weggelassen.

2.4 Assimilation

Als Assimilation wird die Ausspracheänderung auf Grund der direkten Folge zweier Konsonanten bezeichnet.

Die häufigste Assimilation im Arabischen ist die Assimilation des Artikels.

Der Artikel (ال *al*) wird nämlich mit dem Substantiv in einem Wort zusammengeschrieben. Beginnt der Substantiv mit einem bestimmten Konsonant, wird der *lām* (ل *l*) des Artikels assimiliert, d.h. er wird nicht ausgesprochen (und trägt auch kein *sukūn* wie er eigentlich sollte) und der folgende Konsonant wird mit *šadda* verdoppelt.

Im Beispiel von Seite 7 ist die Assimilation bei اَلسَّفِيرِ *as-safīri* zu sehen.

Die assimilierenden Konsonanten werden „Sonnenbuchstaben" genannt, weil das Wort Sonne ein gutes Beispiel der Assimilation ist: اَلشَّمْس *aš-šams*. Die nicht assimilierenden Konsonanten werden „Mondbuchstaben" genannt, weil es beim Wort Mond (اَلْقَمَر *al-qamar*) keine Assimilation gibt.

2.5 Morphologie

Eine wichtige Besonderheit des Arabischen ist der Ableitungsmechanismus. Die meisten Wörter werden aus einer Verb-Wurzel (meistens triliterale Wurzel) abgeleitet. Diese Ableitungen bestehen aus verschiedenen Vokalschablonen, Infixen, Präfixen, Suffixen, Verdopplungen, ...[1]

Wenn die diakritischen Zeichen nicht geschrieben werden, kann ein Wort also mehrere morphologische Analysen und somit Bedeutungen haben, je nachdem, wie es vokalisiert wird.

Nehmen wir zum Beispiel das Wort حسب, aus der triliteralen Wurzel *ḥsb*, die den Sinn von 'glauben, denken' hat. Es kann folgende Ableitungen und Bedeutungen haben (nach [Karboul, 1999]):

- حَسِبَ *ḥasiba* → 'er dachte'
- حَسَبَ *ḥasaba* → 'er zählte'
- حَسْبَ *ḥasba* → 'entsprechend'
- حَسَب *ḥasab* → 'Hoheit'

Zum Glück unterscheiden sich nicht alle Ableitungen nur durch diakritische Zeichen. Beispielsweise gibt es aus derselben Wurzel das Wort حِسَاب *ḥisāb* ('Rechnung'), bei dem die Ableitung aus einer Verlängerung des Vokals besteht.

[1] Eine Beschreibung dieser Mechanismen kann z.B. in [Darwish, 2002] gefunden werden, mögliche formale Analysen in [Kiraz, 1995] oder [Beesley, 1998].

2.6 Kasus und Indeterminierung

Der Kasus (d.h. der Fall: Nominativ, Akkusativ oder Genitiv) wird durch Endvokale bei Substantiven und Adjektiven angezeigt.

Für den Nominativ steht der kurze Vokal *u*, für den Akkusativ der Vokal *a* und für den Genitiv oder das Präpositionalobjekt der Vokal *i*.

Beispiel:

اَلْبَاصُ *al-bāṣu* → 'der Bus'

يَرْكَبُ اَلْبَاصَ *yarkabu 'l-bāṣa* → 'er nimmt den Bus'

فِي اَلْبَاصِ *fī 'l-bāṣi* → 'im Bus'

Die Indeterminierung wird durch die Nunation oder *tanwīn* angezeigt (siehe S. 6).

Beispiel:

اَلْبَاصُ *al-bāṣu* → 'der Bus'

بَاصٌ *bāṣun* → 'ein Bus'

2.7 Mehrdeutigkeit

Da die kurzen Vokale wie gesagt Ableitungen, Kasus und Determinierung anzeigen, sind Wörter ohne diakritische Zeichen mehrdeutig. Aus dem Sinn des Satzes soll die korrekte abgeleitete Form gefunden werden und aus der Syntax der richtige Kasus sowie die richtige Determinierung für Substantive und Adjektive.

Die Mehrdeutigkeit kommt nicht nur von den erwähnten Fällen. Es gibt auch Wörter, die auf der morphologischen Ebene nichts miteinander zu tun haben, aber dieselbe nicht vokalisierte Form haben, z.B. مَنْ *man* (wer) und مِنْ *min* (von).

2.8 Kodierung

Es gibt verschiedene Kodierungen, die für Arabisch entwickelt wurden. Unter den bekanntesten sind:

- Die Windows-Kodierung für Arabisch (cp-1256). Diese 8-Bit-Kodierung wurde ursprünglich von Windows-Anwendungen benutzt, um arabische Zeichen zu kodieren.
- Der *Unicode Character Set* (siehe [The Unicode Consortium et al., 2000]) unterstützt auch die arabische Schrift.

Das Ziel des Unicode-Standards ist, alte Kodierungssysteme, die länderspezifisch und miteinander nicht kompatibel sind, zu vereinheitlichen. Alle Zeichen der wichtigsten Sprachen können mit dieser einzigen Kodierung kodiert werden. Zu jedem Zeichen wird eine Zahl (*code point*) assoziiert. Eine solche Zahl ist eindeutig, was in alten Kodierungssystemen nicht der Fall war. Die Zahl `E1` ist zum Beispiel in der *codepage* cp-1256 der arabische Buchstabe *lām*, während es in der *codepage* cp-1255 (die Windows-Kodierung für Hebräisch) der Buchstabe *beth* ist. Mit Unicode haben diese Zeichen verschiedene *code points*: `0644` für *lām*, `05D1` für *beth*. So kann man in einem Dokument mehrere Schriften gemeinsam verwenden (z.B. lateinische, arabische, griechische oder asiatische Schriften).

Ein Zeichen (*code point*) wird in einer Unicode-kodierten Datei mit einem oder mehreren *code units* repräsentiert. Die Repräsentation von *code points* mit einer Folge von *code units* wird von sogenannten *Unicode Transformation Formats* (UTF) definiert. Zum Beispiel wird mit der UTF-16 Kodierungsform ein *code point* immer mit zwei Bytes repräsentiert, während bei UTF-8 ein *code point* durch 1 bis 4 Bytes dargestellt wird. Für europäische Sprachen ist es vorteilhaft, UTF-8 zu verwenden, da die meisten Zeichen in diesen Sprachen nur mit einem Byte repräsentiert werden. Für arabische Buchstaben werden zwei Bytes benötigt.

2.9 Transliteration

Es kann aus verschiedenen Gründen interessant sein, arabische Texte zu romanisieren, d.h. mittels romanischem Alphabet zu schreiben. So kann man Speicherplatz sparen (7-bit ASCII anstelle UTF-8 z.B.), den Text einfacher auf dem Bildschirm eines nicht arabischen Systems anschauen, per Email schicken, als Eingabe für ein Programm benutzen, das nur ASCII Zeichen unterstützt oder lesen, wenn man mit der arabischen Schrift nicht sehr vertraut ist.

Wie im Artikel *Romanization, Transcription and Transliteration* von Kenneth R. Beesley[2] erklärt wird, soll man bei diesen Umschriften zwischen Transkription und Transliteration unterscheiden (wir verwenden hier dieselben Bezeichnungen wie im Artikel).

- Eine Transkription (*transcription*) ist eine Darstellung des arabischen Textes, die eher phonetisch ist. Das heißt, sie ist für nicht arabischsprechende Leute nötig, um einen Text aussprechen zu können, aber entspricht nicht der genauen Rechtschreibung des Textes. Ein Beispiel einer Transkription ist die Transkription, die in dieser Arbeit verwendet wird[3], um neben den

[2]Siehe `http://www.xrce.xerox.com/competencies/content-analysis/arabic/info/romanization.html`

[3]In dieser Arbeit wurde die ZDMG Transkription verwendet, siehe [Lagally, 1999].

arabischen Beispielen eine romanisierte Form anzugeben, welche für den Leser verständlicher ist und auch eine Idee der Aussprache wiedergibt.

- Eine Transliteration entspricht dagegen der Rechtschreibung des Originaltextes. In einer Transliteration wird jeder arabische Buchstabe mit einem romanischen Buchstaben assoziiert. Diese 1-zu-1 Assoziierung garantiert, dass die Transliteration wieder rückgängig gemacht werden kann, ohne den Text zu ändern oder Information zu verlieren (solange der Originaltext ausschließlich übliche Arabische Zeichen enthält).

Da in dieser Arbeit die Orthographie sehr wichtig war, wurde eine Transliteration verwendet, um den Text zu romanisieren.

Die gewählte Transliteration ist die Buckwalter-Transliteration, die von Tim Buckwalter entwickelt wurde.[4] Diese verwendet lediglich 7-Bit ASCII-Zeichen. Diese Transliteration hat mehrere Vorteile. Sie entspricht der Phonetik ziemlich gut und ist somit relativ einfach zu lesen. Zudem ist die Umwandlung zwischen Unicode und Buckwalter-Zeichen einfach. Jeder Unicode *code point*, welcher für einen arabischen Buchstaben steht, wird mit dem entsprechenden ASCII Zeichen ersetzt.

Die Tabelle A.1 im Anhang A zeigt die Assoziation zwischen den arabischen Buchstaben, dem Unicode *code point* und den Buckwalter-Buchstaben.[5]

Nehmen wir wieder das Beispiel von Seite 7.
Der Satz in vokalisierter arabischer Schrift ist:

يَتَحَدَّثُ ٱلْوَزِيرُ مَعَ ٱلسَّفِيرِ.

Die Transkription (ZDMG) lautet:

yataḥaddaṯu 'l-wazīru maʿa 's-safīri.

Die Transliteration (Buckwalter) lautet:

`yataHad~avu Alwaziyru maEa Als~afiyri.`

[4] Siehe `http://www.xrce.xerox.com/competencies/content-analysis/arabic/info/buckwalter-about.html`

[5] Die Tabelle ist unter `http://www.xrce.xerox.com/competencies/content-analysis/arabic/info/translit-chart.html` zu finden.

Kapitel 3

Stand der Forschung

Im Bereich der Computerlinguistik wurde bisher viel im Bereich rechnerunterstützter, morphologischer Analyse der arabischen Sprache gearbeitet. Diese Methoden wurden insbesondere zur Verwendung in *Information Retrieval*-Systemen entwickelt. In diesem Kontext haben sich einige Forscher für das Problem der Mehrdeutigkeit des Arabischen (oder Hebräischen, da beide Sprachen ähnlich sind) interessiert.

Die Wiederherstellung von diakritischen Zeichen wurde bisher fast nur für europäischen Sprachen erforscht.

Nur in einem Artikel der Literatur kann man eine Arbeit über die automatische Vokalisierung des Arabischen und Hebräischen mit statistischen Verfahren finden.

3.1 Morphologische Analyse und Vokalisierung der arabischen Sprache

Es wurden mehrere Systeme entwickelt, die eine morphologische Analyse von arabischen Wörter ermöglichen, im Besonderen nach der Theorie der *two-level morphology* von Koskenniemi (siehe [Koskenniemi, 1984]).

Viele Systeme für die morphologische Analyse des Arabischen, im Besonderen die ersten Systeme, nehmen an, dass die Wörter vokalisiert sind. Das ist zwar einfacher, aber in der Praxis nicht so gut anwendbar, da die Wörter meistens nicht oder kaum vokalisiert sind! Ein System, welches in [Beesley, 1996] vorgestellt wird, kann als Eingabe arabische Wörter nehmen, die beliebig vokalisiert sein können. Es wird aber erwähnt, dass nicht vokalisierte Wörter zu mehreren Möglichkeiten führen, und das System zwischen diesen Möglichkeiten nicht unterscheiden kann. Wegen dieser Mehrdeutigkeit kann man Arbeiten über die sogenannte *morphological disambiguation* des Arabischen finden. Diese Disambiguierung ist insofern mit der Vokalisierung verbunden, als jede mögliche morphologische Analyse in der Regel einer Vokalisierung entspricht.

In [Levinger et al., 1995] werden mehrdeutige Analysen durch die Berechnung von sogenannten *morpho-lexical probabilities* gelöst (allerdings handelt es sich in dieser Arbeit um Hebräisch).

Diese Wahrscheinlichkeiten werden aus einem Korpus gelernt, indem ähnliche Wortformen betrachtet werden. Für jede mögliche Analyse eines mehrdeutigen Wortes wird eine Menge von ähnlichen Wörtern (*similar words set*) durch Regeln erzeugt. Die Wahrscheinlichkeit der Wörter dieser Menge im Korpus gibt einen Näherungswert für die Wahrscheinlichkeit der Analyse.

Diese Methode betrachtet nicht den Kontext des Wortes.

Mit demselben Ziel wird in [Itai and Segal, 2000] ein System beschrieben, das statistische Methoden mit Regel-basierten syntaktischen Analysen in drei Schritten kombiniert (auch für die hebräische Sprache).

In einem ersten Schritt („*word phase*") werden mit einem Basis-Analysator die möglichen Analysen für jedes Wort erzeugt. Durch eine Methode ähnlich der Methode von [Levinger et al., 1995] wird jeder Analyse eine Punktzahl zugewiesen, welche von der Wahrscheinlichkeit der Analyse im Korpus abhängt, aber unabhängig vom Wortkontext ist.

Im zweiten Schritt („*pair phase*") werden Wortpaare betrachtet. Mit einem Trainingskorpus werden *transformation rules* gelernt, um die Punktzahl der Analysen dem Kontext entsprechend zu ändern.

Im dritten Schritt („*sentence phase*") wird ein syntaktischer Analysator verwendet, um die Informationen der zwei früheren Schritte mit syntaktischen Informationen zu kombinieren. Damit wird dann die beste Analyse für den ganzen Satz gesucht.

In [Ramsay and Mansur, 2001] werden mittels Morphologie die diakritischen Zeichen von arabischen Wörtern zur Verwendung in *text-to-speech* Systemen ermittelt.

Das System benutzt hier eine kategoriebasierte Grammatik, um die Struktur der arabischen Wörter zu beschreiben. Mit den Regeln dieser Grammatik ist es möglich, die morphologische Analyse und dadurch die diakritischen Zeichen der Wörter zu finden.

Das System kann aber nicht zwischen mehrdeutigen Formen, die derselben Kategorie angehören, unterscheiden. Nur durch eine Kopplung mit einer syntaktischen und semantischen Analyse könnte diese Mehrdeutigkeit gelöst werden.

In [Debili and Achour, 1998] wird die automatische Vokalisierung des Arabischen betrachtet.

Es wird gezeigt, dass die Mehrdeutigkeit durch Wissen aus verschiedenen Quellen vermindert werden kann: zuerst durch die Morphologie, dann durch die Syntax und am Schluss durch die Semantik. Sogar auf dieser Ebene kann es sein, dass die Mehrdeutigkeit nicht gelöst ist. Dann muss subjektiv entschieden werden, welches die richtige Interpretation ist.

Die Aufgabe der Vokalisierung wird hier aber nur theoretisch betrachtet, es werden weder konkrete Anwendungen noch Ergebnisse angegeben.

All diese Arbeiten kümmern sich zwar um das Problem der Mehrdeutigkeit arabischer Wörter und der Wiederherstellung der diakritischen Zeichen, sie sind aber morphologieorientiert und setzen ein breites Spektrum an Wissen über die Sprache voraus.

3.2 Akzentuierung von Texten in europäischen Sprachen

Viele europäischen Sprachen haben auch diakritische Zeichen: z.B. Französisch, Spanisch, Rumänisch. Diese Zeichen sind dort nicht so häufig wie im Arabischen anzutreffen und auch nicht so wichtig für den Leser. Da sie z.B. in Emails oft nicht geschrieben werden, haben sich einige Forscher für die Frage interessiert, wie man diese Zeichen wiederherstellen kann.

Einige Arbeiten verwenden wortbasierte, statistische Methoden.

In [Simard, 1996] und [Simard, 1998] wird gezeigt, wie ein stochastisches Sprachmodell verwendet werden kann, um diakritische Zeichen in einem französischen Text zu finden. Das verwendete Sprachmodell ist ein Hidden Markov Modell (HMM) auf sogenannten „morpho-syntaktischen Tags" basierend. Dies ist ein klassenbasiertes HMM, dessen Emissionen die akzentuierten Wörter und die versteckten Zustände die morphologischen und syntaktischen Kategorien der Wörter sind. Durch Übergangs- und Emissionswahrscheinlichkeiten kann die Wahrscheinlichkeit eines Satzes berechnet werden. Dann liefert die Suche den akzentuierten Satz mit der höchsten Wahrscheinlichkeit.

In [Spriet and El-Bèze, 1997] wird ebenfalls die automatische Akzentuierung von französischen Texten behandelt. Ihr Ansatz besteht darin, den besten Pfad im Graph der möglichen Interpretationen zu wählen. Aus Speicher- und Korpusgründen wird kein n-Gramm Sprachmodell, sondern ein n-Klassen Sprachmodell verwendet (und zwar ein 3-Klassen Modell). Jedem Wort wird eine syntaktische Klasse zugewiesen. Um die wahrscheinlichste Folge von syntaktischen Tags zu finden, wird der Viterbi Algorithmus verwendet.

[Yarowsky, 1999] vergleicht verschiedene Verfahren für die Wiederherstellung von Akzenten, besonders in Spanisch. Die Wiederherstellung von diakritischen Zeichen wird hier als ein Beispiel des größeren Problems der Lösung von syntaktischen und semantischen Mehrdeutigkeiten betrachtet.

Das erste Verfahren (*baseline*) wählt einfach die häufigste Akzentuierung, wenn ein Wort mehrere Akzentuierungen haben kann.

Das zweite Verfahren, *n-gram tagger*, ist ein Verfahren, das schon für *POS*[1] *tagging* benutzt wurde. Es benutzt Informationen über Suffixe und *function words* in einem kleinen Kontext des Wortes.

Das dritte Verfahren ist ein *bayesian classifier*. Hier wird ein ziemlich breiter Kontext von Wörtern in der Umgebung des mehrdeutigen Wortes betrachtet.

Das vierte Verfahren verwendet „Entscheidungslisten" (*decision lists*), die in [Yarowsky, 1994] genauer erklärt werden. Es werden *collocations* von Wörtern betrachtet, d.h. Wörter, die sich in der Nähe befinden. In der Entscheidungsliste werden diese Merkmale nach Relevanz für die Lösung der Mehrdeutigkeit sortiert. Dieses Verfahren ist so effizient wie der *n-gram tagger* für lokale syntaktische Verbindungen und der *bayesian classifier* für breite, semantische Verbindungen.

Die bisher erwähnten Verfahren basieren auf Wörtern, und können unbekannte Wörter, d.h. Wörter, die nicht im Lexikon stehen, nicht akzentuieren.

In [Zweigenbaum and Grabar, 2002] interessiert man sich dafür, unbekannte Wörter zu akzentuieren, da im Bereich der Medizin viele Wörter nur sehr selten vorkommen. Es werden zwei Methoden vorgestellt, die jeweils auf Buchstaben basieren.

Die erste Methode basiert auf einem *POS tagger*. Regeln werden gelernt, um aus dem Kontext eines Buchstabens auf die Akzentuierung zu schließen.

Bei der zweiten Methode wird der Kontext als „gemischter Kontext" (*mixed context*) dargestellt. Um die Akzentuierung zu finden, werden *finite-state transducers* verwendet.

In [Mihalcea, 2002] basiert das Lernen auch auf Buchstaben, um diakritische Zeichen in einem Text zu finden. Hierbei ist von Vorteil, dass man kein großes Lexikon und keine besonderen Werkzeuge für die morphologische oder syntaktische Analyse braucht. Man braucht nur einen relativ kleinen Korpus von Wörtern mit diakritischen Zeichen.

Es wird ein *instance based* Lernverfahren verwendet, welches Regeln erzeugt, in denen aus dem Kontext eines Buchstabens (Umgebung von anderen Buchstaben) zwischen mehreren diakritischen Varianten entschieden werden kann.

3.3 Statistische Methoden für die Vokalisierung des Arabischen

Die Vokalisierung des Arabischen bzw. Hebräischen mittels statistischer Methoden, bei denen kein Wissen über die Sprache benötigt wird, wurde bisher kaum erforscht.

[1] *Part of Speech*

[Gal, 2002] stellt Ergebnisse der Anwendung von Hidden Markov Modellen für die Vokalisierung von Texten in Semitischen Sprachen (Arabisch, Hebräisch) vor.

Es wird ein Bigramm-Modell verwendet. Die versteckten Zustände sind die vokalisierten Wörter und die Emissionen sind die unvokalisierten Wörter, wie man sie im Text findet. Die Wahrscheinlichkeiten werden anhand eines Korpus berechnet. Der gewählte arabische Korpus ist eine Transkription des Korans, bestehend aus ca. 90000 Wörter.

Da der Korpus ziemlich klein ist, ist es ein Problem, dass viele Wörter nicht im Korpus vorkommen. Um das Problem zu behandeln, werden Zustände für unbekannte Wörter modelliert. Man verwendet dazu ein Verfahren von Katz mit der Good-Turing Schätzung (siehe [Katz, 1987]), um die unbekannten Wahrscheinlichkeiten zu berechnen.

Im Arabischen erreicht man damit eine bis zu 86% korrekte Vokalisierung.

Kapitel 4

Korpus

Der erste Schritt in dieser Arbeit war, Daten zu sammeln, um einen Korpus aufzubauen. Nach einigen Vorverarbeitungsschritten konnte die Qualität des Korpus bewertet werden. Dabei wurden Fehler und Inkonsistenzen entdeckt, weshalb noch einige Veränderungen nötig waren.

In der Forschung gab es bisher sehr wenige Arbeiten mit vokalisierten Korpora auf arabisch. Deshalb war die Datensammlung und die Verarbeitung sehr aufwändig, da keine direkt verwendbaren Korpora zur Verfügung standen. Auch wenn diese Phase der Arbeit nicht die interessanteste war, war sie aber wichtig, weil der Korpus die Basis der ganzen Arbeit ist. Es lohnt sich also, so gute und vollständige Daten wie nur möglich zu sammeln. Die vorgenommenen Analysen können einen Eindruck von der Schwierigkeit der Sammlung vokalisierter arabischer Daten geben.

4.1 Quellen

Es ist ziemlich schwierig, vokalisierte, arabische Texte in elektronischer Form zu finden. Es sind fast ausschließlich religiöse Texten vokalisiert. Im Internet wurden zwar auch einige Seiten gefunden, die vokalisiertes Material für Arabisch-Lernende anbieten, aber mit jeweils nur sehr wenig Daten. Außerdem ist die Vokalisierung auf solchen Seiten meistens nicht vollständig. Deshalb wurde für den Korpus fast ausschließlich die Bibel und der Koran verwendet. Ein weiteres Problem war, dass der Koran im Internet nahezu immer nur als Bitmap zur Verfügung stand.

Für die Bibel wurde die Version der *International Bibel Society* genommen, zu finden unter `http://www.gospelcom.net/ibs/bibles/arabic/`. Diese ist eine Übersetzung der Bibel in *Modern Standard Arabic*, die Sprache ist also die aktuelle Sprache. Der Text besteht aus dem Alten Testament (39 Kapitel) und dem Neuen Testament (27 Kapitel).

Für den Koran wurde die Version des *One Ummah Network* genommen, die un-

ter `http://www.oneummah.net/quran/quran.html` zu finden ist. In diesem Text ist die Sprache klassisches Arabisch und nicht die Sprache, die heute verwendet wird. Das Vokabular und die Grammatik sind aber ähnlich genug, so dass die Hinzunahme zum Korpus sinnvoll ist. Der Text besteht aus 114 Suren.

Als Ergänzung wurde ein kleiner Wortschatz hinzugefügt. Dieser Wortschatz besteht aus Ausdrücken des alltäglichen Arabisch (besonders aus dem Bereich des Tourismus), die von Hand vokalisiert wurden, mit dem Ziel, im Korpus auch Vokabular der modernen alltäglichen Sprache zu haben. Da der Aufwand für die Vokalisierung sehr hoch ist, war dieser Wortschatz aber nur gering (ca. 400 Sätze, 2000 Wörter).

4.2 Vorverarbeitung

Wie schon im Abschnitt 2.9 erwähnt, wurde für das Training der Korpus konvertiert. Die Vorverarbeitung der Daten wurde aber noch mit den UTF-8 Dateien gemacht, die erst am Ende mit der Buckwalter-Transliteration ins ASCII-Format umgewandelt wurden.

Der erste Schritt war, die erlaubten Zeichen zu bestimmen und die anderen zu löschen.

Die Zeichen, die erlaubt sind, sind die Zeichen, die in der Tabelle A.1 zu finden sind, ergänzt um Leerzeichen und Satzzeichen. Nur der *taṭwīl* wurde entfernt. Dies ist nämlich ein Zeichen, das nur für das schönere Aussehen der Wörter benutzt wird, aber keinen bedeutungstragenden Sinn hat.

In den Dateien waren einige fehlerhafte Zeichen, die entfernt wurden. Das *Soft Hyphen* Zeichen war noch zu finden und wurde mit einem Bindestrich ersetzt.

Der zweite Schritt war, die Sätze und Absätze zu trennen, um einen Korpus aus kleineren Einheiten zu bekommen. Das Ziel war, Zeilen zu bekommen, die als isolierten Einheiten Sinn machen.

Dafür wurden die Texte an der Versnummerierung und bei starken Satzzeichen (., !, ?, :) getrennt. Dabei wurden die Nummerierung sowie zusätzliche Leerzeichen entfernt. Anführungszeichen wurden auch gelöscht, da sie in den getrennten Zeilen keinen Sinn mehr hatten. Die Satzzeichen wurden durch Leerzeichen von den Wörtern getrennt.

Ein weiteres Problem war, dass Kapitelüberschriften und andere kleine Einführungen zu den Kapiteln nicht vokalisiert waren. Deshalb wurden alle Zeilen, die aus mindestens einem Wort ohne Vokalisierung bestehen, entfernt. Die Anzahl der Zeilen, die so entfernt werden mussten, entsprechen ca. 4% (Koran) bis 8% (Altes Testament) des originalen Textes.

Dabei wurden einige Zeilen entfernt, die tatsächlich vokalisiert waren, aber in denen ein Wort aufgrund eines Fehlers nicht vokalisiert war. Es kommt nämlich

ziemlich häufig vor, dass bei kleinen Wörtern wie z.B. لا *lā*, إلى *ʾilā*, في *fiy* die Vokalisierung vergessen wird. Die Anzahl der Zeilen, die so falsch entfernt wurden, ist aber sehr gering (14 Zeilen für das Alte Testament, 36 Zeilen für den Koran) und deshalb nicht problematisch.

In der Tabelle 4.1 ist eine Zusammenfassung der Größe des Korpus vor und nach der Verarbeitung dargestellt.

Text		Zeilen	Wörter	Byte (UTF-8)	Byte (ASCII)
Altes Testament	Original	10297	359421	6211486	3319738
	Nach Verarbeitung	31926	343807	6049382	3239877
Neues Testament	Original	3633	124131	2097698	1124008
	Nach Verarbeitung	12094	118523	2038682	1095735
Koran	Original	5676	85056	1324003	710827
	Nach Verarbeitung	6180	77025	1270737	673881
Wortschatz	Original	392	1955	30631	16308
	Nach Verarbeitung	391	1955	30601	16278
Insgesamt	Original	19998	570563	9663818	5170881
	Nach Verarbeitung	50591	541310	9389402	5025771

Tabelle 4.1: Größe des Korpus

Es ist zu beachten, dass der Korpus sehr klein ist (ca. 541000 Wörter, 5 MB) während für Computerlinguistik Korpora von üblicherweise Hunderten Megabyte bis einigen Gigabyte benutzt werden.

4.3 Bewertung des Korpus

Obwohl eine gedruckte Veröffentlichung religiöser Texten keinen Fehler enthalten darf (siehe 2.3.1), ist es leider bei einer Version, die heutzutage im Internet zu finden ist, ganz anders. Einerseits ist die Vokalisierung nicht so vollständig wie bei einer gedruckten Version, und andererseits gibt es eine hohe Anzahl von Fehlern.

4.3.1 Vokalisierung

Zuerst ist es zu beachten, dass im Korpus kein *waṣla* (Verbindungszeichen) zu finden ist. Der *ʾalif*, der als Verbindungs-*ʾalif* dient, trägt kein Zeichen (ا). (Dies rührt wahrscheinlich daher, dass dieses Zeichen in älteren Kodierungen als Unicode nicht definiert ist und sich auf arabischen Tastaturen nicht befindet). Dieses Zeichen kann also nicht gelernt werden.

Der *ʾalif maḥḏūfa*, eine abgeleitete Form des *ʾalif*, die normalerweise in alten Texten zu finden ist (im Besonderen bei dem Namen Gottes, اللّٰه), ist auch nicht im Korpus zu finden (aus den selben Gründen wie *ʾalif waṣla*). Man findet an der Stelle ein normales *ʾalif* oder einen kurzen Vokal *fatḥa*.

Sonst werden alle diakritischen Zeichen geschrieben. Die verwendete Vokalisierung ist also fast vollständig. Dies ist die Vokalisierung, die wir auch lernen wollen.

4.3.2 Fehler

Die Schreibweise des vokalisierten Arabischen, wie sie im Korpus anzutreffen ist, ist ziemlich einfach zu beschreiben. Ein Wort besteht nämlich aus einer Folge von Silben[1], und jede Silbe besteht aus einem Konsonant, eventuell einem Verdopplungszeichen und einem Vokalzeichen oder Vokallosigkeitszeichen. Es gibt noch lange Vokale und den Verbindungs-*ʾalif*. Außerdem gibt es einfache Regeln für Buchstaben, die z.B. nur am Ende eines Wortes vorkommen dürfen oder vor bzw. nach denen nur bestimmte Vokale stehen dürfen.

So kann man einen regulären Ausdruck zum Korrektheitstest eines arabischen Wortes schreiben. Mit der Perl-Syntax sieht dieser Ausdruck so aus: (?<= ist der Links-Kontext und ?= der Rechts-Kontext)

```
my $word = qr/(
                    $letter$vowel
                     # first syllab without shadda
                    |(?<=$char)$letter~?$vowel
                     # possible shadda on other syllabs
                    |(?<=(a|A|\|))p($short|F|N|K)(?=(\ |$))
                     # ta marbuta at the end
                    |>(a|u) |<i
                     # beginning hamza
                    |(?<=(a|u|i|o|\|))$hamza~?$vowel
                    |(?<=(aA|uw|iy))$hamza~?$vowel
                     # hamza in middle of words
                    |(A|li)$article
                     # article with verbindungs-alif
                    |A(?=$alif)
                     # verbindungs-alif
                    |(?<=uw)A(?=(\ |$)) |(?<=awo)A(?=(\ |$))
                     # silent alif
                    |\|
```

[1] Wir verwenden hier den Begriff von Silbe, obwohl es nicht genau der üblichen Bedeutung einer Silbe entspricht.

```
        |(?<=u)w |(?<=i)y |(?<=a)A |(?<=a)Y(?=(\ |$))
         # long vowels - Alef maqsura at the end
)+/;
```

- `$letter` ist ein Konsonant,
- `$vowel` ist ein Vokal (dabei ist zu beachten, dass *tanwīn* nur am Ende stehen darf),
- `$char` ist ein Buchstabe,
- `$short` ist ein kurzer Vokal (ohne *tanwīn*),
- `$hamza` ist ein *hamza* Zeichen mit seinem Träger,
- `$article` ist der Artikel mit dem erlaubten folgenden Kontext,
- `$alif` ist der Kontext eines Verbindungs-*ʾalif*.

Zwar beschreibt dieser Ausdruck nicht die ganze Orthographie des Arabischen, aber er erlaubt es, Unregelmäßigkeiten und Fehler im Korpus zu entdecken. Es lohnt sich, solche Fehler im Korpus zu suchen, um die Qualität des Korpus bewerten zu können. Sie hat nämlich einen starken Einfluss auf die Qualität der Vokalisierung.

Es wurden folgende Fehler entdeckt:

Leerzeichen

Manchmal gibt es ein zusätzliches Leerzeichen in der Mitte eines Wortes, das Wort wird also in zwei Wörtern getrennt. Dieses passiert oft nach den Buchstaben ر und ز und ist wahrscheinlich ein *Optical Character Recognition* Problem, da der Platz nach diesen Buchstaben oft etwas größer, als nach anderen Buchstaben ist. Vermutlich wurde zur Erzeugung des elektronischen Textes ein Buch gescannt und *OCR* Verfahren verwendet, dabei wurde der Platz zwischen den Buchstaben fälschlich als ein Leerzeichen interpretiert.

Namen

Bei manchen Namen: *al-lāh* (dem Namen von Gott), *ʾishāq* (Isaac), *hārūn* (Aaron), *ʾismāʿīl* (Ismail) fehlt der *a*-Vokal.

Besonderheiten des Korans

Im Koran ist die Rechtschreibung etwas anders, als die übliche Rechtschreibung

des Arabischen. Der Koran wird nämlich vorgelesen oder auswendig wie ein Gedicht aufgesagt. Deshalb wird manchmal eine besondere Vokalisierung verwendet, die dem Leser hilft, die Satzmelodie richtig wiederzugeben.[2]

Zum Beispiel folgt bei den Endungen وٰ´ und وا´ dem *ʾalif*, der nicht ausgesprochen wird, ein *sukūn* (also `awoAo` anstatt `awoA` und `uwAo` anstatt `uwA`).

Ein anderes Beispiel ist die Assimilation: wenn ein Wort mit bestimmten Buchstaben endet und das folgende Wort mit einem ähnlichen Buchstaben beginnt, findet eine Assimilation statt: der *sukūn* auf dem ersten Buchstaben verschwindet und der zweite Buchstabe wird verdoppelt (z.B. `yakun l~ahu` anstatt `yakuno lahu`). In anderen Texten ist die Assimilation nur beim Artikel zu finden (siehe 2.4).

Ein weiteres Beispiel ist, dass der *sukūn* auf dem *nūn* nicht geschrieben wird, wenn bestimmte Buchstaben folgen, wie z.B. `>anta` anstatt `>anota`.

Im Korpus kommen viele dieser Besonderheiten vor, aber nicht alle, die in der normalen schriftlichen Form des Korans (in Büchern) zu finden sind.

Verwechselung von Buchstaben

Ähnliche Buchstaben werden manchmal verwechselt, wie z.B. ى und ي, ة und ه.

Vokalisierungsfehler

Es gibt oft Fehler bei den kurzen Vokalen. Manchmal werden sie verdoppelt oder stehen an der falschen Stelle (z.B. nach einem langen Vokal, anstatt davor). Solche Fehler fallen nicht auf, wenn der Text in arabischer Schrift dargestellt wird (es werden nur zwei gleiche Zeichen an der gleichen Stelle dargestellt oder ein Zeichen etwas verschoben). Für unsere Arbeit sind sie aber problematisch, da sie zu zusätzlichen Formen von Wörtern führen, und somit die Mehrdeutigkeit erhöhen.

Manchmal ist auch die Vokalisierung falsch, was man z.B. bei der Vokalisierung von *hamza* bemerken kann, wo die Regeln einfach zu überprüfen sind.

Es kommt auch oft vor, dass die Vokalisierung einfach fehlt. Das kann man z.B. nach *hamza*, vor einem langen Vokal oder zwischen dem Artikel und einem Mondbuchstabe (siehe 2.4) feststellen. Im Koran fehlt sehr oft die Vokalisierung am Ende von Wörtern nach `A'`. Die Vokalisierung fehlt auch oft bei Präpositionen, die als Präfix von Wörtern vorkommen, z.B. `w>ana` ('und ich') anstatt `wa>ana`.

Reihenfolge der Buchstaben

In vielen Fällen stehen die Buchstaben nicht in der logischen Reihenfolge. Zum Beispiel befindet sich bei der `>F` Endung der *tanwīn* oft vor dem *hamza*: `F>`, obwohl er logischerweise danach stehen sollte.

[2]Eine ausführliche Beschreibung der Ausspracheregeln im Koran kann im Internet unter `http://quran.al-islam.com/Ahkam/Tree.asp?ID=1&t=TreeSub&l=eng&RecNo=2&Parnt=0` gefunden werden.

Dies sind keine Fehler bei der Darstellung in arabischer Schrift - ob das diakritische Zeichen vor oder nach dem Buchstaben steht macht hierbei keinen großen Unterschied. Aber wie schon erwähnt, ist es ein Problem für die Konsistenz des Korpus. Erstens ist es problematisch, weil verschiedene Reihenfolgen die Mehrdeutigkeit erhöhen. Zweitens wäre es wünschenswert, dass die Form der Wörter im Korpus immer die Gleiche ist, d.h. auf den Konsonant folgt immer die Vokalisierung, wie es logisch und auch im Korpus meistens der Fall ist.

Ligatur

Ein besonderes Problem ist die *lām-ʾalif* Ligatur.

Im Arabischen gibt es Ligaturen. Wenn bestimmte Buchstaben nebeneinander stehen, werden sie verbunden und etwas anders geschrieben, wie z.B. ثمّ anstatt ثمّ. Diese Ligaturen sind aber nicht obligatorisch, außer die besondere *lām-ʾalif* Ligatur: ا + ل ergibt immer ﻻ (oder ﻷ) und auf keinen Fall لا.

Die Vokalisierung bei dieser Ligatur ist problematisch. Nehmen wir das Beispiel von لَا ('nein'). Hier sollte ein Vokalzeichen zwischen dem *lām* und dem *ʾalif* stehen[3], also `laA`. Im Unicode Standard ist festgelegt, dass bei der Reihenfolge `laA` die Ligatur `lA` erkannt werden soll, die mit `a` vokalisiert ist. Das Darstellungsprogramm sollte also لَا darstellen. In anderen Kodierungen war der Zusammenhang zwischen Kodierung und Bildzeichen (*glyph*) aber nicht so eindeutig festgelegt und noch heute fehlen für Unicode oft die richtige Fonts, die diese vokalisierte Ligatur richtig darstellen können[4]. Deshalb wird meistens لَا dargestellt. Um das Problem zu vermeiden, wurde häufig das Zeichen nicht zwischen *lām* und *ʾalif* geschrieben, sondern direkt danach, also `lAa`. Dann ist das Ergebnis nicht perfekt (لَا anstatt لَا) aber auf jeden Fall besser, als لَا.

Im Korpus kommt diese Ligatur sehr oft vor: jedes Mal, wenn *ʾalif* (in einer der verschiedenen Formen: langer Vokal, Träger von *hamza* oder *madda*, Verbindungs-*ʾalif*) *lām* folgt. Das kommt im Besonderen beim Artikel (`Al`) vor und beim präpositionalen Präfix `li`, wenn das Wort mit *ʾalif* anfängt.

Man kann z.B. feststellen, dass fast immer `lAa` geschrieben wird. Hier ist es nicht so problematisch, aber trotzdem sollte aus logischen Gründen das Vokalzeichen vor dem langen Vokal stehen (wie bei `maA`). Bei `Al + <i`, das normalerweise `Alo<i` geschrieben werden sollte, wird oft `Aol<i` geschrieben, was لِإ anstatt إِل ergibt (hier kann das Vokalzeichen nicht nach der Ligatur geschrieben werden,

[3]Wie im Abschnitt 2.3.1 erwähnt, ist in diesem Fall das Zeichen überflüssig, es befindet sich aber in der Vokalisierung, die wir lernen wollen.

[4]Nur die *OpenType* Fonts mit der Verwendung im Darstellungsprogramm von *Uniscribe* (Microsoft's Unicode Script Processor) können diese vokalisierte Ligatur bei Windows Programmen richtig darstellen. Diese Technologie wurde von Microsoft für die Behandlung komplexer Skripte unter Windows entwickelt.

weil dort schon ein Vokalzeichen steht, deshalb wird es davor geschrieben).

Das Zeichen, das die Darstellung der Ligatur verhindert, wird aber oft einfach weggelassen, man findet also `lA` oder `Al<i`. Bei `lA` in der Mitte von Wörtern kommt es nur selten vor, aber beim Artikel und beim Präfix `li` (auch `la`) ist es fast immer der Fall. Das ist natürlich ein größeres Problem, da diese fehlende Vokalisierung gar nicht gelernt werden kann. Das Problem besteht nicht nur für Vokalzeichen, sondern auch für das Verdopplungszeichen, das auch oft weggelassen wird, wie z.B. beim Wort كُلَّ *kullan* ('alle, jeder'), das `kulAF` anstatt `kul~AF` geschrieben wird.

Die Tabelle 4.2 gibt für einige der oben genannten Fehler die Häufigkeit im Korpus an. Die Anzahl von Fehlern ist verhältnismäßig hoch, wobei nur die automatisch erkannten Fehler aufgeführt sind.

Im Alten Testament wurden insgesamt ca. 14450 Fehler innerhalb 32000 Wörter[5] (ohne die Fehler zu zählen, die nur die Reihenfolge betreffen) entdeckt, davon 78% aufgrund der *lām-ʾalif* Ligatur, 8,5% wegen Namen und 13,5% andere Fehler.

Fehler	Altes Test.	Neues Test.	Koran
Leerzeichen	66	34	0
Fehlende *sukūn* vor Mondbuchstabe	400	75	44
Fehlende Vokal am Ende	127	66	113
Fehlende Vokal nach Präposition و	175	24	20
`lA` anstatt `laA`	385	90	92

Tabelle 4.2: Einige Fehler im Korpus

Diese Fehler kann man zwar automatisch entdecken, aber die Identifizierung der Fehler ist nicht zu 100-prozentig sicher und die richtige Form kann man nicht immer automatisch finden. Deshalb ist es nicht möglich, diese Fehler automatisch zu korrigieren, denn das Risiko wäre zu hoch, neue Fehler hinzuzufügen. Aber trotzdem können einige Homogenisierungen gemacht werden.

4.4 Homogenisierung

Viele der Fehler, die oben erwähnt wurden, führen zu Inkonsistenzen im Korpus. Die folgenden Tabellen geben einige Beispiele dieser Inkonsistenzen wieder.

Tabelle 4.3 gibt die Anzahl verschiedener Formen für die Vokalisierung der *lām-ʾalif* Ligatur in den verschiedenen Texten an. Das erste Beispiel ist das Wort `Alo>aroD` (الْأَرْضَ), 'die Erde'. Das nächste Beispiel ist der allgemeine Fall, wenn

[5] Es können mehrere Fehler pro Wort sein.

ein *hamza* tragendes *ʾalif* dem Artikel folgt. Ein anderes Beispiel ist das Wort `salaAm` (سَلَام), 'Frieden'. Das letzte Beispiel beschreibt den allgemeinen Fall, dass auf den Buchstaben *lām* der lange Vokal *ʾalif* folgt.

Tabelle 4.4 beschreibt den Fall der ـً Endung, bei der die Reihenfolge der Buchstaben entweder `AF` oder `FA` sein kann.

Tabelle 4.5 beschreibt die وْا Endung, die im besonderen Fall des Korans mit *sukūn* geschrieben wird (siehe Abschnitt 4.3.2 S. 23).

Tabelle 4.6 gibt die Anzahl verschiedener Formen für Namen an. Der *a*-Vokal kann lang, kurz oder nicht vorhanden sein (siehe Abschnitt 4.3.2 S. 23).

Manchmal kommen verschiedene Formen von Wörtern gleich häufig vor, so dass es keine bevorzugte Vokalisierung gibt, was zur Bestimmung der korrekten Form problematisch ist. Das ist zum Beispiel oft im Koran der Fall: die Formen `Al + hamza` / `Alo + hamza`, `lAa` / `laA` (siehe Tabelle 4.3), `uwA` / `uwAo` (siehe Tabelle 4.5), `All~ah` / `All~h` (siehe Tabelle 4.6) kommen alle nahezu gleich häufig vor. Im Koran kann man sogar feststellen, dass die gewählte Form vom Kapitel abhängig ist.

Ein ähnliches Problem besteht darin, dass manchmal eine Form am häufigsten in einem Text vorkommt, aber eine andere Form am häufigsten in einem anderen Text vorkommt. Das ist z.B. der Fall für die `AF` Endung: in der Bibel findet man am häufigsten `AF`, während man im Koran am häufigsten `FA` findet.

Form	Altes Test.	Neues Test.	Koran
`Al>aroD`	1488	232	183
`Aol>aroD`	5	0	0
`Alo>aroD`	0	0	253
`Al + hamza`	6996	3065	587
`Aol + hamza`	21	0	0
`Alo + hamza`	0	0	703
`salAm`	10	5	2
`salaAm`	0	0	27
`salAam`	247	104	13
`lA`	400	92	93
`laA`	0	0	1561
`lAa`	8528	4052	1476

Tabelle 4.3: Inkonsistenzen bei der ﻵ Ligatur

Es wäre für die Arbeit mit dem Korpus sehr wünschenswert, diese Inkonsistenzen zu entfernen. Da die Inkonsistenzen die Mehrdeutigkeit der Trainingsdaten erhöhen, wird das Lernen schwieriger sein - zwei Varianten desselben Wortes werden nicht als eine einzige Form betrachtet, sondern als zwei verschiedenen

Form	Altes Test.	Neues Test.	Koran
AF	10219	4370	294
FA	3	4	2735

Tabelle 4.4: Inkonsistenzen bei der اً Endung

Form	Altes Test.	Neues Test.	Koran
uwA	6072	2693	1501
uwAo	0	0	1839

Tabelle 4.5: Inkonsistenzen bei der ـُوا Endung

Form	Altes Test.	Neues Test.	Koran
Allh	1043	1293	2
All~h	0	0	1444
All~ah	4	3	1075
<isoHq	94	5	0
<isoHaq	4	0	16
<isoHaAq	2	16	0

Tabelle 4.6: Inkonsistenzen bei Namen

Formen. Diese Inkonsistenzen werden auch beim Testen mit Daten aus dem Korpus zu Fehler führen, wenn die gewählte Form nicht die Form ist, die in der Testmenge zu finden ist, sondern eine andere Variante.

Einige der Inkonsistenzen können automatisch korrigiert werden. Deshalb wurde eine Homogenisierung des Korpus durchgeführt. Diese Homogenisierung besteht aus folgenden Änderungen:

Besonderheiten des Korans

Die Besonderheiten des Korans (siehe Abschnitt 4.3.2 S. 23) wurden korrigiert, damit die Vokalisierung des Korans der Vokalisierung der Bibel möglichst nahe kommt. Die Vokalisierung, die in der Bibel verwendet wurde, ist die Vokalisierung, die als „normale" Vokalisierung betrachtet wird. Es ist zu beachten, dass vielleicht nicht alle Besonderheiten des Korans entdeckt und korrigiert wurden.

Altes Testament

Im Alten Testament wurde das Wort `walakin` ('aber'), das sehr oft fälschlich `walkin` geschrieben wurde, korrigiert.

Allgemeine Homogenisierungen

In allen Texten wurden folgende Änderungen vorgenommen:

- Die Namen wurden standardisiert: `All~ah` für den Namen Gottes; ein langes `aA` für die anderen Namen, bei denen der *a*-Vokal nicht konsistent ist (siehe Beispiel in Tabelle 4.6).
- Die Reihenfolgen von Buchstaben, die nicht eindeutig waren, wurden standardisiert.
- Bei der *lām-ʾalif* Ligatur wurde das Vokalzeichen, wenn überhaupt vorhanden, immer zwischen beide Buchstaben verschoben.

Nach dieser Homogenisierung ist es aber immer noch ein Problem, dass bei der *lām-ʾalif* Ligatur die Vokalisierung oft fehlt.

4.5 Vereinfachung

Einige Vokalisierungen können durch einfache Regeln bestimmt werden. Das ist der Fall bei:

- dem Verdopplungszeichen der Assimilation des Artikels (siehe 2.4);
- dem Vokallosigkeitszeichen nach dem Artikel wenn er nicht assimiliert wird (wenn nach ihm ein Mondbuchstabe oder ein *hamza* folgt);

- der Hilfsvokal zwischen dem Artikel und einem Verbindungs-*ʾalif* (siehe [Brockelmann, 1965], S. 14).

Diese diakritischen Zeichen können also vom Korpus entfernt werden. Sie brauchen nicht gelernt zu werden und können mit einfachen Regeln gefunden werden. Dabei ist von Vorteil, dass einige Inkonsistenzen verschwinden (im Besonderen bei der *lām-ʾalif* Ligatur). Ein weiterer Vorteil ist, dass die entfernten Zeichen bei der *lām-ʾalif* Ligatur oft nicht im Korpus waren. Sie werden also mit einfachen Regeln gefunden, während sie mit der aus dem Korpus gelernten Vokalisierung nicht gefunden worden wären.

4.6 Mehrdeutigkeit im Korpus

4.6.1 Nicht vokalisierter Korpus

Aus dem vokalisierten Korpus kann man automatisch den nicht vokalisierten Korpus erzeugen. Die gewählte nicht vokalisierte Form ist die Form, die in 2.3.2 beschrieben wurde und bei der die morphologischen Verdopplungszeichen vorhanden sind. Da die anderen Verdopplungszeichen (von der Assimilation) mit Regeln gefunden werden, wird die Aufgabe auf die Wiederherstellung der Vokalzeichen beschränkt.

Zur Erzeugung des nicht vokalisierten Korpus müssen also nur alle Vokalzeichen entfernt werden.

In 6.2.3 werden die Ergebnisse angegeben, wenn das System auch die Verdopplungszeichen finden soll. In diesem Fall werden auch die Verdopplungszeichen vom nicht vokalisierten Korpus entfernt.

4.6.2 Mehrdeutigkeit

Man kann im Korpus für jede nicht vokalisierte Form zählen, wieviel vokalisierte Formen es gibt. Aus diesen Statistiken erhält man die Verteilung der Mehrdeutigkeit im Korpus (siehe Abb. 4.1) und die durchschnittliche Anzahl von Vokalisierungen pro Wort. Man kann diese Formen anhand des Wortschatzes zählen, der aus dem Korpus erzeugt wurde (siehe Tabelle 4.8) oder direkt anhand des Korpus (die Zahlen werden dann mit der Häufigkeit der Wörter gewichtet, siehe Tabelle 4.7). Folgende Ergebnisse erhält man auf den Trainingsdaten (90% des Korpus).

Diese Zahlen erlauben es, die Schwierigkeit der Aufgabe der Vokalisierung abzuschätzen.

Nur ca. 30% der Wörter sind eindeutig und es gibt im Durchschnitt ca. 3 mögliche Vokalisierungen pro Wort. Im Vergleich mit anderen Sprachen, wie z.B. dem Französischen, wo die Verhältnisse umgekehrt sind (ca. 70% eindeutige Wörter) und es im Durchschnitt nur 1,3 Akzentuierungen pro Wort gibt (nach

	Vor Verarbeitung	Nach Homogenisierung	Nach Vereinfachung
Prozentsatz eindeutiger Wörter	28,15	30,08	30,35
Prozentsatz mehrdeutiger Wörter	71,85	69,92	69,65
Durchschnittliche Anzahl Vokalisierungen	3,29	3,19	3,13

Tabelle 4.7: Mehrdeutigkeit des Korpus

	Vor Verarbeitung	Nach Homogenisierung	Nach Vereinfachung
Prozentsatz eindeutiger Wörter	78,83	79,82	79,98
Prozentsatz mehrdeutiger Wörter	21,17	20,18	20,02
Durchschnittliche Anzahl Vokalisierungen	1,33	1,32	1,31

Tabelle 4.8: Mehrdeutigkeit des Wortschatzes

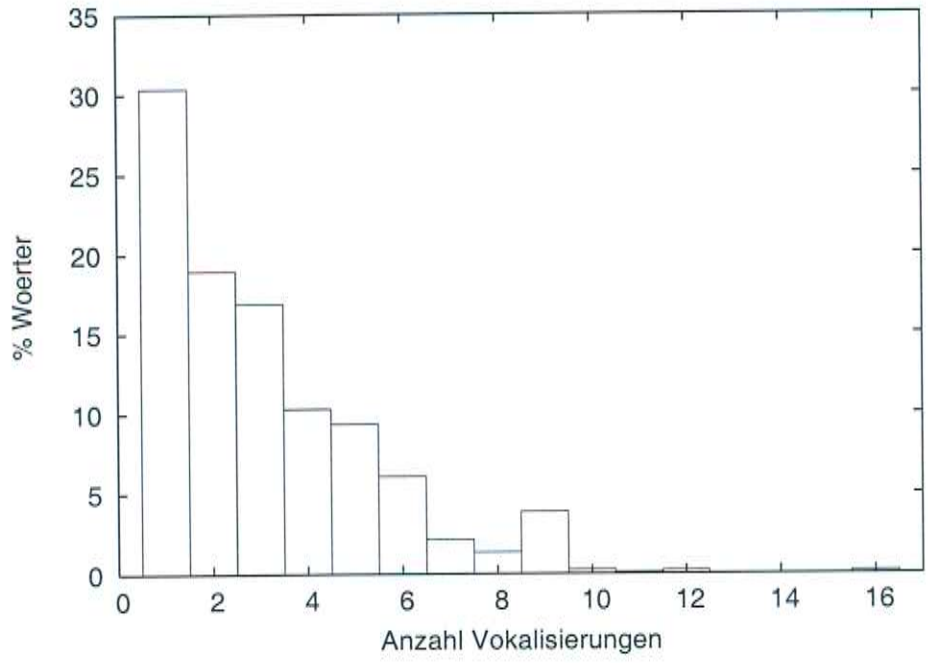


Abbildung 4.1: Verteilung der Mehrdeutigkeit im Korpus

[Debili and Achour, 1998]) ist das Arabische viel mehrdeutiger. Theoretisch ist die maximale Anzahl der möglichen Vokalisierungen für ein arabisches Wort sehr hoch, wenn man bedenkt, dass ein arabisches Wort im Durchschnitt 3 bis 5 Konsonanten hat[6] und es für jeden Konsonanten 4 bis 7 mögliche Vokalisierungen gibt (4 innerhalb des Wortes, 7 für den letzten Konsonanten). Zum Glück ist die durchschnittliche Anzahl möglicher Vokalisierungen praktisch nicht so hoch, wie man erwarten würde.

Es ist zu beachten, dass die Mehrdeutigkeit im Wortschatz niedriger ist. In der Praxis kommen eindeutige Formen seltener vor, als mehrdeutige.

Nach der Verteilung der Mehrdeutigkeit sollte die Wortfehlerrate bei trainierten Wörtern nicht höher als 48% sein (30% der Wörter können eindeutig vokalisiert werden, d.h. mit einer erwarteten Fehlerrate von 0%, bei 20% der Wörter sind nur zwei Möglichkeiten d.h. die erwartete Fehlerrate ist 50% usw.)

Diese Zahlen kann man mit Messungen anderer Arbeiten vergleichen.

In [Beesley, 1996] werden 5 möglichen morphologischen Analysen (also Vokalisierungen) pro Wort erwähnt. In [Debili and Achour, 1998] werden sogar 11,5 möglichen Vokalisierungen pro Wort gefunden. Der Unterschied zu unseren Zahlen liegt teilweise darin, dass in unserem Fall das morphologische Verdopplungszeichen in der nicht vokalisierten Form vorhanden ist, während bei den zitierten Ergebnissen das Wort ganz ohne diakritische Zeichen ist.

Die Zahlen auf den Wortschatz bezogen, unterscheiden sich auch von den Zahlen aus [Debili and Achour, 1998]. Hier liegt der Unterschied sicher darin, dass der Wortschatz in unserer Arbeit aus dem Korpus gebaut wurde, und somit nur sehr klein ist, im Vergleich mit dem Wortschatz, der im zitierten Artikel benutzt wird.

Die Verteilung der Mehrdeutigkeit ist mit den Statistiken in [Gal, 2002] mit ca. 30% eindeutiger Wörter vergleichbar, wo sie ausschließlich auf dem Koran gemessen worden sind.

[6] Die Wurzel bestehen meistens aus 3 Konsonanten und dazu kommen noch Präfixe, Suffixe und Infixe.

Kapitel 5

Untersuchte Verfahren

5.1 Vorgehensweise

5.1.1 Aufteilung des Korpus

Der Korpus wurde in drei Mengen aufgeteilt:

- Eine Trainingsmenge, bestehend aus 90% des Korpus, die für das Training benutzt wurde;
- Eine Entwicklungsmenge, bestehend aus 5% des Korpus, die für Experimente und zur Leistungsmessung während der Entwicklung verwendet wurde;
- Eine Testmenge, bestehend aus 5 % des Korpus, für die Experimente mit dem endgültigen System.

Die drei Mengen wurden homogen und disjunkt aus den verschiedenen Quellen herausgenommen.

Die Ergebnisse in diesem Kapitel basieren auf der Entwicklungsmenge.

5.1.2 Fehlerrate

Um die verschiedenen untersuchten Verfahren zu bewerten, wird die Fehlerrate durch den Vergleich des vom System vokalisierten Textes mit dem originalen Text berechnet.

Drei Fehlerraten wurden jeweils ermittelt:

Wortfehlerrate Die Wortfehlerrate (oder *Word Error Rate*, WER) ist der Prozentsatz von Wörtern, die nicht richtig vokalisiert wurden.

Diese Fehlerrate wird am häufigsten benutzt, um die Verfahren zu bewerten.

Ein Problem ist aber, dass sie nicht betrachtet, wieviel Fehler in einem Wort gemacht wurden - ein Wort wird gleich bewertet, ob es einen Fehler enthält oder mehrere.

Silbenfehlerrate Die Silbenfehlerrate (*Syllab Error Rate*, SER) ist der Prozentsatz von Silben, die nicht richtig vokalisiert wurden. Dabei wird als Silbe ein Konsonant mit den entsprechenden diakritischen Zeichen betrachtet.

Bei dieser Fehlerrate kann es so viele Fehler wie Konsonanten in einem Wort geben. Wenn nur die Vokalzeichen zu finden sind, gibt diese Rate eigentlich den Prozentsatz der falschen Vokalzeichen an.

Buchstabenfehlerrate Die Buchstabenfehlerrate (*Letter Error Rate*, LER) ist der Prozentsatz von Buchstaben, die falsch sind. Dabei wird die minimale Editierdistanz zwischen dem vokalisierten Wort und dem Referenzwort berechnet.

Der Unterschied zur Silbenfehlerrate ist derselbe, wie der Unterschied zwischen Wortfehlerrate und Silbenfehlerrate. Wenn für jeden Konsonant mehrere Zeichen zu finden sind (Verdopplungszeichen und Vokalzeichen), gibt die Buchstabenfehlerrate eine bessere Schätzung des Prozentsatzes der falschen Zeichen (es wird ein Fehler pro Zeichen gezählt und nicht pro Konsonant). Wenn nur die Vokalzeichen zu finden sind, gibt diese Rate nicht mehr Information über die Fehler an, als die Silbenfehlerrate.

5.1.3 Allgemeines Verfahren

Aus den Trainingsdaten wird ein Lexikon erzeugt. Das Vokabular dieses Lexikons besteht aus allen Wörtern, die im nicht vokalisierten Korpus vorkommen. Für jedes Wort wird eine Liste der möglichen Vokalisierungen erzeugt.

Das allgemeine Verfahren für die Vokalisierung ist also:

- Wenn das Wort im Vokabular ist, wird eine Vokalisierung aus den bekannten möglichen Vokalisierungen gewählt;
- Wenn das Wort nicht im Vokabular ist (d.h. nicht in den Trainingsdaten war), muss für jeden Buchstaben des Wortes eine Vokalisierung gefunden werden.

Im Abschnitt 5.3 werden Verfahren vorgestellt, die sich mit bekannten Wörtern beschäftigen. Verfahren für unbekannte Wörter werden im Abschnitt 5.4 behandelt.

Es ist zu erwarten, dass die Fehlerrate bei unbekannten Wörtern höher ist. Deshalb wäre es wünschenswert, dass das Lexikon so vollständig wie möglich ist. Da der Korpus aber klein ist,

- kommen viele Wörter nicht im Korpus vor und
- bei den Wörtern, die im Korpus vorkommen, kommen nicht alle mögliche Vokalisierungen vor.

Bei den Entwicklungsdaten ist zum Beispiel der Prozentsatz unbekannter Wörter (auch OOV, *Out Of Vokabulary* Rate) 8,48%. Bei den bekannten Wörtern ist in 13,5% der Fällen die erwartete Vokalisierung nicht bekannt. Das heißt, dass die Wortfehlerrate nicht unter 12,4% liegen kann.

Es ist also zu erwarten, dass diese Probleme einen Einfluss auf die Ergebnisse haben werden.

5.2 *Baseline*-Algorithmus

Um neue Verfahren einordnen zu können, wurde ein einfacher *Baseline*-Algorithmus entwickelt. Mittels dieses Algorithmus wird für ein bekanntes Wort (trainiertes Wort) die vokalisierte Form gewählt, die am häufigsten im Korpus vorkommt. Wenn das Wort unbekannt ist (nicht im Korpus vorkommt), gibt es drei Möglichkeiten, den Vokal zu einem Konsonanten zu finden:

- gar nicht vokalisieren;
- einen Vokal zufällig wählen;
- den Vokal wählen, der am häufigsten mit dem Konsonant vorkommt.

Die Tabelle 5.1 gibt die Ergebnisse dieses Algorithmus an. Die Ergebnisse der Vokalisierung wurden auf der Entwicklungsmenge gemessen, die 8,48% unbekannte Wörter enthält.

	Anteil	WER	SER	LER
Bekannte Wörter	91,52	21,78	5,91	3,29
(Bekannte) eindeutige Wörter	24,21	8,79	2,29	1,29
(Bekannte) mehrdeutige Wörter	67,31	26,45	7,51	4,18
Unbekannte Wörter	8,48			
- nicht vokalisiert		100	79,52	43,36
- zufällig vokalisiert		100	93,58	71,52
- nach häufigsten Vokalen vokalisiert		96,28	46,54	25,37
Insgesamt (unbekannte Wörter nach häufigsten Vokalen vokalisiert)	100	28,10	10,60	5,89

Tabelle 5.1: Ergebnisse des *Baseline*-Algorithmus

Es ist zu beachten, dass die zufällige Vokalisierung für unbekannte Wörter schlechter ist, als keine Vokalisierung. Als Referenz auf der Buchstabenebene werden wir die Vokalisierung nach dem häufigsten Vokal betrachten. Die Referenzfehlerrate ist also 21,78 % für bekannte Wörter und 96,28% für unbekannte.

Man kann hier auch sehen, dass die Fehlerrate für eindeutige Wörter ziemlich hoch ist (8,79%), da die korrekte Vokalisierung nicht im Korpus vorkommt.

5.3 Verfahren auf Wortebene

Mit dem *Baseline*-Algorithmus wird der Kontext von Wörtern nicht betrachtet. Wenn ein Wort mehrdeutig ist, muss dies aber getan werden, um aus verschiedenen möglichen Vokalisierungen die richtige wählen zu können. Zu diesen Zweck wurden zwei Verfahren untersucht: zuerst ein Verfahren mit Kollokationen und dann ein Verfahren mit einem Sprachmodell. Diese Verfahren können auf im Trainingskorpus vorhandene mehrdeutige Wörter angewendet werden.

5.3.1 Kollokationsmodell

Prinzip

Dieses Verfahren basiert auf dem *collocation*-Modell, das in [Yarowsky, 1999] beschrieben wird. In diesem Artikel ist das Ziel, die Mehrdeutigkeit der Akzentuierung von französischen und spanischen Wörtern zu lösen. Man muss hier unter zwei möglichen Formen die richtige finden. Das Verfahren besteht aus folgenden Schritten:

- Zählen der vokalisierten Formen und Sammlung der Kontexte

  Es werden die verschiedenen akzentuierten Formen für jede unakzentuierte Form im Korpus gezählt. So werden mehrdeutige Formen identifiziert. Für jede mehrdeutige Form werden verschiedene Kontexte gesammelt, die mit dieser Form assoziiert sind. Zu jedem Kontext wird die entsprechende Akzentuierung assoziiert.

- Messung der Verteilung der Kollokationen

  Eine Kollokation ist eine Menge von Wörtern, die nebeneinander im Korpus vorkommen. Solche Kollokationen können einen guten Hinweis für die Wahl einer Akzentuierung geben. Es können verschiedene Arten von Kollokationen betrachtet werden, z.B.:

  - das Wort direkt links von dem zu akzentuierenden Wort;
  - das Wort direkt rechts;
  - zwei Wörter links;
  - zwei Wörter rechts;
  - ein Wort links und ein Wort rechts;
  - ein Wort, das sich innerhalb der k nächsten Wörter befindet;

– ...

Für jede Kollokation wird gezählt, wie häufig jede akzentuierte Form vorkommt.

- Entscheidungslisten

  Aus den Messungen werden Entscheidungslisten (*decision lists*) gebaut.

  Man berechnet eine sogenannte *log-likelihood*, die der Kraft der Kollokationen für die Entscheidung der Akzentuierung entspricht. Die *log-likelihood* einer Kollokation k wird nach folgender Formel berechnet:

$$\left|\log\left(\frac{p(a_1|k)}{p(a_2|k)}\right)\right|$$

  wobei a_1 und a_2 die zwei möglichen Akzentuierungen sind.

  Die Kollokationen werden nach dieser Zahl sortiert: je höher die *log-likelihood* ist, desto besser ist die Kollokation für die Entscheidung geeignet. Bei der Akzentuierung wird die Form gewählt, die von der passenden Kollokation mit der größten Entscheidungskraft vorhergesagt wird.

Einsatz und Ergebnisse

In dieser Arbeit wurde eine vereinfachte Version des Kollokationsmodells benutzt.

In einem ersten Schritt werden auch die verschiedenen möglichen vokalisierten Formen für jede nicht vokalisierte Form im Korpus gezählt. In Tabelle 5.2 findet man ein Beispiel der Zählung für das Wort `ktb` (Wurzel von 'schreiben').

Nicht vokalisierte Form	Vokalisierte Form	Anzahl	Bedeutung
`ktb`	`kataba`	18	er schrieb
	`kutiba`	79	es ist geschrieben
	`kutubK`	3	Bücher (indet., gen./präp.)
	`kutubN`	1	Bücher (indet., nom.)
	`kutubi`	2	Bücher (det., gen./präp.)

Tabelle 5.2: Zählung der vokalisierten Formen für das Wort `ktb`

In einem zweiten Schritt werden für die mehrdeutigen Wörter Kollokationen gesammelt und die Verteilung der Vokalisierungen für diese Kollokationen gezählt. Es werden zuerst lediglich folgende Kollokationen betrachtet:

- ein Wort links (*m1*);
- ein Wort rechts (*p1*);

- ein Wort links und ein Wort rechts (*m1p1*).

Im folgenden Satz:

```
kmA ktb fy ktAb <$EyA' :
```

ist zum Beispiel für das Wort `ktb` die Kollokation *m1* `kmA ktb`, die Kollokation *p1* ist `ktb fy` und die Kollokation *m1p1* ist `kmA ktb fy`.

Die Tabelle 5.3 gibt ein Beispiel der Verteilung der Vokalisierungen für das Wort `ktb` im Trainingskorpus. Man kann schon bei diesem Beispiel sehen, dass die Kollokationen gute Hinweise auf die Vokalisierung geben können.

Art von Kollokation	Kollokation	kataba	kutiba	kutubK	kutubN	kutubi
m1	, ktb	3	0	0	0	0
	qd ktb	1	45	0	0	0
	mA ktb	2	5	0	0	0
	mn ktb	0	1	2	0	0
	fy ktb	0	0	0	0	2
p1	ktb lnA	2	0	0	0	0
	ktb All~h	4	0	0	0	0
	ktb fy	1	7	0	0	0
	ktb Enh	1	5	0	0	0
m1p1	qd ktb :	0	33	0	0	0

Tabelle 5.3: Verteilung der Vokalisierungen nach Kollokationen für das Wort `ktb`

Die Frage ist jetzt, wie man die Kollokationen verwenden kann, um die Vokalisierung vorherzusagen.

Man kann zuerst untersuchen, wie gut einzelne Kollokationen die Vokalisierung vorhersagen. Wenn man ein Wort vokalisieren will, sucht man, ob die benötigte Kollokation bekannt ist. Wenn sie bekannt ist, wird die häufigste Vokalisierung für diese Kollokation gewählt, sonst wird die häufigste Vokalisierung ohne Kontext gewählt.

Mit der Kollokation *p1* würde das Wort `ktb` im vorigen Beispiel als `kutiba` vokalisiert (das ist die häufigste Vokalisierung für die Kollokation `ktb fy`). Mit der Kollokation *m1* würde das Wort auch `kutiba` vokalisiert, aber dieses Mal, weil die Kollokation `kmA ktb` nicht bekannt ist, und man auf die häufigste Vokalisierung ohne Kontext zurück fallen muss.

Die Tabelle 5.4 zeigt die Ergebnisse (Fehlerrate bei bekannten Wörtern), wenn man so Wörter nach einzelnen Kollokationen vokalisiert. Die Abdeckung gibt an, wie oft die Kollokation bekannt war.

Art von Kollokation	Abdeckung	WER
m1	60,24	16,55
p1	60,02	18,31
m1p1	27,54	18,78

Tabelle 5.4: Ergebnisse der Vokalisierung mit einzelnen Kollokationen

Man kann also sehen, dass Kollokationen schon zu besseren Ergebnissen führen, als das *Baseline*-Verfahren. Um die Abdeckung zu verbessern, sollten die Hinweise von allen Kollokationen gleichzeitig betrachtet werden.

Es werden hier keine Entscheidungslisten verwendet. Da es in unserem Fall mehr als zwei mögliche Formen gibt, ist es nicht möglich, die Entscheidungskraft der Kollokationen mit der *log-likelihood* Formel zu berechnen, wie sie in [Yarowsky, 1999] gegeben wird. Außerdem ist in unserem Fall problematisch, dass die Wahrscheinlichkeiten, die für die Berechnung der *log-likelihood* benutzt werden, im Korpus oft null sind.

Ein weiteres Problem mit Entscheidungslisten ist, dass Information verloren geht. Nur der beste Hinweis wird betrachtet und andere Hinweise, die auch gut sein könnten, werden auf Seite gelassen. Yarowsky beweist, dass die Entscheidung dabei nicht schlechter ist. In unserem Fall ist der Trainingskorpus aber zu klein und es ist deshalb wahrscheinlich besser, alle Hinweise zu betrachten, die zur Verfügung stehen.

Bezeichnen wir v_i als die möglichen Vokalisierungen für ein Wort v und $f_k(v_i)$ als die Häufigkeit der Vokalisierung v_i bei der Kollokation k (wobei k *m1*, *p1* oder *m1p1* ist, allerdings ist sie nur definiert, wenn die Kollokation bekannt ist). Die verschiedenen Verfahren, die untersucht wurden, sind:

1. Wie bei Entscheidungslisten wird nur die „beste" zutreffende Kollokation verwendet. Hier nimmt man an, dass die Kollokation *m1p1* die stärkste Entscheidungskraft hat (der Kontext ist breiter). Die zweitbeste Kollokation ist dann *m1*, da laut Tabelle 5.4 diese besser ist, als *p1*. Das Verfahren vokalisiert mit der ersten verfügbaren Kollokation in der Reihenfolge: *m1p1*, *m1*, *p1* und sonst wird ohne Kontext vokalisiert.

2. Man wählt die Vokalisierung, die bei der Mehrheit der Kollokationen die häufigste ist. Man berechnet

$$n_{v_i} = \sum_k c_k(v_i)$$

wobei

$$c_k(v_i) = \begin{cases} 1 & \text{wenn } i = \text{argmax}_j\, f_k(v_j) \\ 0 & \text{sonst} \end{cases}$$

D.h., $c_k(v_i)$ ist gleich 1 wenn die Vokalisierung v_i die häufigste bei der Kollokation k ist. n_{v_i} ist die Anzahl der Kollokationen, bei denen die Vokalisierung v_i die häufigste ist. Man wählt dann $\hat{v} = \text{argmax}_{v_i} n_{v_i}$.

3. Ein anderes Verfahren betrachtet die Häufigkeit der Vokalisierungen bei den verschiedenen zutreffenden Kollokationen. Man berechnet also

$$n_{v_i} = \sum_k f_k(v_i)$$

n_{v_i} ist die Häufigkeit der Vokalisierung v_i, wenn man den Kontext, der von den Kollokationen gegeben ist, kennt. Man wählt $\hat{v} = \text{argmax}_{v_i} n_{v_i}$, d.h. die häufigste Vokalisierung in diesem Kontext.

4. Anstatt der Häufigkeit betrachtet man die Wahrscheinlichkeit der Vokalisierungen. Man berechnet also

$$n_{v_i} = \sum_k p(v_i) = \sum_k \frac{f_k(v_i)}{\sum_j f_k(v_j)}$$

n_{v_i} ist die Summe der Wahrscheinlichkeiten der Vokalisierung v_i bei allen zutreffenden Kollokationen. Man wählt $\hat{v} = \text{argmax}_{v_i} n_{v_i}$, d.h. die Vokalisierung, die im Durchschnitt bei den gegebenen Kollokationen die höchste Wahrscheinlichkeit hat.

Betrachten wir zum Beispiel wieder den Satz

kmA ktb fy ktAb <$EyA' :

Nehmen wir an, dass die Kollokation *m1p1* (kmA ktb fy) unbekannt ist und dass die Vokalisierungen bei den anderen Kollokationen wie in Tabelle 5.5 angezeigt verteilt sind (nur zwei Vokalisierungen kommen bei diesen Kollokationen vor).[1]

Art von Kollokation	Kollokation	kataba	kutiba
m1	kmA ktb	2	0
p1	ktb fy	5	8

Tabelle 5.5: Verteilungsbeispiel

Welche Vokalisierung soll man in diesem Fall wählen?

[1] Im Gegensatz zu den Zahlen der Tabelle 5.3 sind die Zahlen hier nicht die wirklichen Werte im Korpus, sondern nur hypothetische, um den Sachverhalt zu veranschaulichen.

Mit dem ersten Verfahren würde man kataba wählen: da die Kollokation *m1p1* nicht bekannt ist, fällt man auf *m1* zurück und die häufigste Vokalisierung wäre kataba.

Mit dem zweiten Verfahren würde man berechnen:

$$n_{\texttt{kataba}} = c_{m1}(\texttt{kataba}) + c_{p1}(\texttt{kataba}) = 1 + 0 = 1$$
$$n_{\texttt{kutiba}} = c_{m1}(\texttt{kutiba}) + c_{p1}(\texttt{kutiba}) = 0 + 1 = 1$$

Dieses Verfahren ist zu grob, da nur wenige Kollokationen betrachtet werden, kann man oft keine Entscheidung treffen.

Mit dem dritten Verfahren würde man folgende Zahlen erhalten:

$$n_{\texttt{kataba}} = m1(\texttt{kataba}) + p1(\texttt{kataba}) = 2 + 5 = 7$$
$$n_{\texttt{kutiba}} = m1(\texttt{kutiba}) + p1(\texttt{kutiba}) = 0 + 8 = 8$$

Man würde also kutiba wählen, weil diese Form bei den betrachteten Kollokationen häufiger vorkommt.

Mit dem vierten Verfahren würde man berechnen:

$$n_{\texttt{kataba}} = \frac{m1(\texttt{kataba})}{m1(\texttt{kataba})+m1(\texttt{kutiba})} + \frac{p1(\texttt{kataba})}{p1(\texttt{kataba})+p1(\texttt{kutiba})} = \frac{2}{2} + \frac{5}{13} = \frac{36}{26}$$
$$n_{\texttt{kutiba}} = \frac{m1(\texttt{kutiba})}{m1(\texttt{kataba})+m1(\texttt{kutiba})} + \frac{p1(\texttt{kutiba})}{p1(\texttt{kataba})+p1(\texttt{kutiba})} = \frac{0}{2} + \frac{8}{13} = \frac{16}{26}$$

Man würde also kataba wählen. Man kann hier feststellen, dass dieses Verfahren feiner ist, da auch die Aussagekraft der Kollokationen betrachtet wird. Kollokationen mit einer größeren Aussagekraft haben bei diesem Verfahren ein größeres Gewicht in der Entscheidung.

Tabelle 5.6 zeigt die Ergebnisse der verschiedenen Verfahren.

Verfahren	WER
1	15,11
2	15,41
3	15,63
4	14,87

Tabelle 5.6: Wortfehlerraten der 4 Kollokationsverfahren

Das vierte Verfahren scheint also das Beste zu sein. Die Ergebnisse sind deutlich besser, wenn man Hinweise von allen Kollokationen betrachtet (Verfahren 4), als wenn nur der beste Hinweis betrachtet wird (Verfahren 1). Die Abdeckung ist auf jeden Fall besser, als wenn man nur einzelne Kollokationen betrachtet: bei 79,88% der Wörter ist hier mindestens eine Kollokation bekannt. Durchschnittlich werden bei der Wahl der Vokalisierung 1,5 Kollokationen betrachtet.

Die Hinweise der Kollokationen (*m1*, *p1*, *m1p1*) sind nach Tabelle 5.4 aber nicht alle gleich gut. *m1* gibt bessere Hinweise, als *p1*. *m1p1* gibt auch gute Hinweise, da

die Abdeckung dieser Kollokation aber sehr klein ist, gibt es das Risiko des *overfittings*: seltene Ereignisse haben einen zu großen Einfluss. Einige Experimente (siehe Tabelle 5.7) zeigen, dass Gewichte die Ergebnisse verbessern können.

Gewicht *m1*	Gewicht *p1*	Gewicht *m1p1*	WER
1	1	1	14,87
2	1	1	14,83
1	2	2	15,14
1	1	2	14,87
3	1	1	15,00
3	2	1	14,80
3	1	2	14,89
3	3	1	14,85

Tabelle 5.7: Gewichtung von Kollokationen

Desweiteren können noch andere Arten von Kollokationen betrachtet werden.

Die Kollokation *m2m1* besteht aus zwei Wörtern links von dem zu vokalisierenden Wort, z.B. `@ kmA ktb` im Beispiel von S. 38 (wobei `@` das Zeichen für den Anfang des Satzes ist).

Diese Kollokation allein ergibt eine Fehlerrate von 19,15%, welche besser ist, als die Referenzfehlerrate des Baseline Verfahrens. Diese Kollokation könnte also auch ein guter Hinweis auf die Vokalisierung sein. Die Abdeckung beträgt aber nur 22,53%.

Bei einer Gleichgewichtung aller Kollokationen inklusiv *m2m1* beträgt die Wortfehlerrate 14,72%. Mit den Gewichten $m1 = 4$, $p1 = 3$, $m1p1 = 2$, $m2m1 = 1$ fällt die Wortfehlerrate geringfügig auf 14,71%.

Man kann in ähnlicher Weise die Kollokation *p1p2* betrachten (zwei Wörter rechts, also `ktb fy ktAb` im Beispiel), die allein eine Fehlerrate von 19,69% ergibt (Abdeckung 28,88%).

Mittels Gleichgewichtung beträgt die Wortfehlerrate dann 14,73%, und mit den Gewichten $m1 = 5$, $p1 = 4$, $m1p1 = 3$, $m2m1 = 2$ und $p1p2 = 1$ beträgt die Wortfehlerrate 14,61%. Die Abdeckung ist natürlich immer noch 79,88% und es werden in diesem Fall bei der Wahl der Vokalisierung durchschnittlich 2 Kollokationen pro Wort benutzt.

Die letzte Art von Kollokation, die untersucht wurde, ist die „Fenster-Kollokation“: man betrachtet die Wörter, die in einem Fenster der Größe k in der Umgebung des zu vokalisierenden Wortes vorkommen. Diese Kollokationen wären im Beispiel von S. 38 also `@`, `kmA`, `fy`, `ktAb` bei einem Fenster der Größe 2.

Der Unterschied zwischen dieser Kollokation und den anderen ist, dass man hier nicht festlegt, wo genau sich das Kollokations-Wort in Bezug auf das zu

vokalisierende Wort befindet. Die Reihenfolge und die Distanz sind beliebig. Im Beispiel von Tabelle 5.3 kann es zum Beispiel sein, dass das Wort `qd` einen guten Hinweis auf die Vokalisierung `kutiba` gibt, auch wenn es nicht direkt vor `ktb`, sondern etwas weiter entfernt steht.

Ein Fenster der Größe 4, ohne andere Kollokationen zu betrachten, ergibt eine Fehlerrate von 17,02% mit einer sehr hohen Abdeckung von 91,99%, da die Chance ein Wort in solch einem breiten Kontext zu sehen sehr hoch ist.

Mittels Gleichgewichtung beträgt die Wortfehlerrate 15,26%, mit den Gewichten $m1 = 6$, $p1 = 5$, $m1p1 = 4$, $m2m1 = 3$, $p1p2 = 2$ und 1 für die Fenster-Kollokationen beträgt die Wortfehlerrate 14,40%. Die Fenster-Kollokation mit den richtigen Gewichten bringt also eine deutliche Verbesserung gegenüber den anderen Kollokationen. Es werden durchschnittlich 6,5 Kollokationen pro Wort als Hinweise auf die Vokalisierung benutzt.

Die beste Performance erreicht man mit einem Fenster der Größe 3 (14,34% Fehlerrate). Ein Fenster der Größe 2 ist schlechter (14,41% Fehlerrate).

Fazit

Das Kollokationsmodell erzielt gute Ergebnisse bei der Akzentuierung von Texten in europäischen Sprachen. Die Ergebnisse der Vokalisierung des Arabischen sind im Vergleich zum *Baseline*-Algorithmus deutlich besser.

Allerdings ist das Problem der Vokalisierung der arabischen Sprache wesentlich schwieriger. Die Mehrdeutigkeit tritt fast bei jedem Wort auf, während in europäischen Sprachen nur wenige Wörter mehrdeutig sind. Deshalb ist es ein größerer Aufwand für Arabisch Kollokationen zu bauen, denn für das Auftreten fast jeden Wortes im Korpus muss man verschiedene Kollokationen erzeugen. Deswegen braucht dieses Verfahren viel Speicher und viel Zeit für die Vokalisierung.

Das Kollokationsmodell ist vielleicht auch für Arabisch nicht so gut geeignet, da hierbei die beste Vokalisierung für jedes Wort einzeln aus dem Kontext gewählt wird. Dabei wird aber nicht berücksichtigt, wie die andere Wörter im Kontext vokalisiert werden, sondern nur welche Wörter sich im Kontext befinden. Für europäische Sprachen, bei denen nur einige Wörter in einem Satz akzentuiert werden müssen, ist es gut geeignet. Im Arabischen muss aber jedes Wort in einem Satz vokalisiert werden. Es wäre also angemessener, die gesamte Folge der Wörter und Vokalisierungen auf einmal zu betrachten und die Mehrdeutigkeit für den ganzen Satz zu lösen.

5.3.2 Sprachmodell

Ein Sprachmodell ist besser geeignet, die Mehrdeutigkeit einer Wortfolge zu lösen. Sprachmodelle wurden bisher erfolgreich in mehreren Bereichen angewandt, z.B. Spracherkennung, maschinelle Übersetzung, ...

Sprachmodelle erlauben es, nach einem Training auf einem Korpus die Wahrscheinlichkeit eines Satzes einer Sprache zu schätzen. Sätzen, die viele Fehler aufweisen, wird eine niedrigere Wahrscheinlichkeit zugewiesen. Ein Sprachmodell ist also sehr leistungsfähig, da aus Statistiken über eine Sprache die Wahrscheinlichkeit eines Satzes gegeben werden kann, ohne explizites Wissen über Syntax oder Semantik zu benötigen - dieses Wissen ist implizit im Modell enthalten.

Theoretische Grundlagen

Sei U ein nicht vokalisierter Satz auf Arabisch. Was wir suchen ist der vokalisierte Satz $\hat{V}$, der die größte Wahrscheinlichkeit hat, die vokalisierte Form von U zu sein. Wir suchen also:

$$\hat{V} = \underset{V}{\operatorname{argmax}}\, p(V|U)$$

was mit Bayes Regel umgeformt werden kann in:

$$\hat{V} = \underset{V}{\operatorname{argmax}} \frac{p(U|V)p(V)}{p(U)} = \underset{V}{\operatorname{argmax}}\, p(U|V)p(V) \tag{5.1}$$

Da im Arabischen die nicht vokalisierte Form eindeutig aus der vokalisierten Form hervorgeht, hat man einen besonderen Fall. Sei u die Funktion, die die unvokalisierte Form eines vokalisierten Satzes ergibt. Dann ist entweder $p(U|V) = 1$ (wenn $u(V) = U$), oder $p(U|V) = 0$ (wenn $u(V) \neq U$). Die Formel kann hier also vereinfacht werden zu:

$$\hat{V} = \underset{V,u(V)=U}{\operatorname{argmax}}\, p(V) \tag{5.2}$$

Die Wahrscheinlichkeit eines vokalisierten Satzes $p(V)$ wird von dem sogenannten Sprachmodell (oder *Language Model*) gegeben. Eine ausführliche Einführung über Sprachmodelle kann man in [Chen and Goodman, 1998] finden.

Sei V eine Wortfolge $v_1 v_2 \dots v_l$. Um die Wahrscheinlichkeit des Satzes zu berechnen, kann man die Historie der Wörter betrachten, als:

$$p(V) = p(v_1 v_2 \dots v_l) = p(v_1)p(v_2|v_1)p(v_3|v_1 v_2)\dots p(v_l|v_1 \dots v_{l-1}) = \prod_{i=1}^{l} p(v_i|v_1 \dots v_{i-1})$$

Da es selten ist, dass eine lange Historie im Korpus vorkommt, werden in sogenannten n-Gramm Modellen die Historien auf $n-1$ Wörtern eingeschränkt, so dass:

$$p(v_i|v_1 \dots v_{i-1}) \approx p(v_i|v_{i-n+1} \dots v_{i-1})$$

Z.B. werden häufig Bigramm Modelle: $p(v_i|v_1 \dots v_{i-1}) \approx p(v_i|v_{i-1})$ und Trigramm Modelle: $p(v_i|v_1 \dots v_{i-1}) \approx p(v_i|v_{i-2}v_{i-1})$ verwendet.

Für die Berechnung von solchen n-Gramm Wahrscheinlichkeiten kann man im Korpus zählen, wie häufig Wortfolgen vorkommen:

$$p_{ML}(v_i|v_{i-n+1}\dots v_{i-1}) = \frac{c(v_{i-n+1}\dots v_{i-1})}{\sum_{v_j} c(v_{j-n+1}\dots v_{j-1})}$$

wobei c die Anzahl einer Wortfolge im Korpus ist. Diese Wahrscheinlichkeit wird als *maximum likelihood* bezeichnet.

Dabei ist aber problematisch, dass manche Wortfolgen nicht im Korpus vorkommen, was zu null-n-Gramm Wahrscheinlichkeiten führt und nicht der Wirklichkeit entspricht. Um dieses Problem zu vermeiden, wurden sogenannte Glättungsverfahren entwickelt (*smoothing*). Die Idee dabei ist, die Anzahl gesehener Wortfolgen zu vermindern, um die Wahrscheinlichkeit ungesehener Wortfolgen besser schätzen zu können.

Diese Verminderung wird als *discounting* bezeichnet. Dabei wird oft die sogenannte Good-Turing Schätzung verwendet, die einem n-Gramm, das r-mal im Korpus vorkommt, den reduzierten Wert r^* zuordnet:

$$r^* = (r+1)\frac{n_{r+1}}{n_r}$$

wobei n_r die Anzahl von n-Grammen ist, die genau r-mal im Korpus vorkommen.

Um die Wahrscheinlichkeit ungesehener n-Gramme zu schätzen, sind zwei Verfahren möglich:

- Rückfall (*backoff*): Die Wahrscheinlichkeiten gesehener n-Gramme werden mit *discounted* Verteilungen geschätzt und die Wahrscheinlichkeiten ungesehener n-Gramme werden mit der Verteilung der niedrigeren Ordnung ($n-1$-Grammen) geschätzt:

$$p(v_i|v_{i-n+1}\dots v_{i-1}) = \begin{cases} \alpha(v_i|v_{i-n+1}\dots v_{i-1}) & \text{wenn } c(v_{i-n+1}\dots v_i) > 0 \\ \gamma(v_{i-n+1}\dots v_{i-1})p(v_i|v_{i-n+2}\dots v_{i-1}) & \text{wenn } c(v_{i-n+1}\dots v_i) = 0 \end{cases}$$

- Interpolation: Die Glättung wird rekursiv durch eine Interpolation vom *maximum likelihood*-Modell der Ordnung n mit dem Modell der Ordnung $n-1$ erzielt:

$$\begin{aligned} p(v_i|v_{i-n+1}\dots v_{i-1}) &= \lambda_{v_{i-n+1}\dots v_{i-1}} p_{ML}(v_i|v_{i-n+1}\dots v_{i-1}) \\ &\quad +(1-\lambda_{v_{i-n+1}\dots v_{i-1}})p(v_i|v_{i-n+2}\dots v_{i-1}) \end{aligned}$$

Es gibt verschiedene Glättungsverfahren, unter den bekanntesten sind die Katz-Glättung, die Witten-Bell-Glättung und die Kneser-Ney-Glättung. Diese Verfahren werden in [Chen and Goodman, 1998] erklärt und verglichen.

Bei all diesen Modellen werden oft *cutoff*-Faktoren eingesetzt. *cutoff*-Faktoren geben einen Schwellwert für die Häufigkeit der n-Gramme im Korpus an. n-Gramme, die seltener als dieser Schwellwert vorkommen, werden als ungesehen betrachtet. Ihre Wahrscheinlichkeit wird wie für n-Gramme, die wirklich nicht im Korpus vorkommen, mittels Rückfall oder Interpolation berechnet.

Die Frage ist jetzt, wie das Sprachmodell benutzt werden kann, um die beste Vokalisierung eines Satzes zu finden. Die Gleichung 5.2 umfasst nämlich zwei Probleme: die Definition des Modells selbst und dann die Suche nach der optimalen Lösung.

Das Problem aus Gleichung 5.1 kann mit einem *Hidden Markov Modell* (HMM) modelliert werden, bei dem die versteckten Zustände die vokalisierten Wörter und die Ausgaben (Emissionen) die nicht vokalisierten Wörter sind. Die Übergangswahrscheinlichkeiten werden anhand eines Sprachmodells berechnet. Die Ausgabewahrscheinlichkeiten sind immer 1 (wenn man nur Wörter betrachtet, die die passende nicht vokalisierte Form haben).

Eigentlich müssen die Zustände für ein n-Gramm Modell die $n-1$ Wörter enthalten, die als Historie für den nächsten Zustand gelten. Nehmen wir das einfache Beispiel des Satzes `mn >nt ?`. Zuerst müssen die vokalisierten Wörter bestimmt werden, die diese nicht vokalisierten Formen haben können. Nehmen wir an, dass es für `mn` zwei Möglichkeiten gibt: `mano` ('wer') und `mino` ('von'). Für `>nt` gibt es auch zwei Möglichkeiten: `>anota` ('du', männlich) und `>anoti` ('du', weiblich). Es müssen noch die Zeichen für den Beginn `<s>` und das Ende `</s>` des Satzes hinzugefügt werden. Bei einem Bigramm Modell erhält man nur ein Wort pro Zustand, siehe Abb. 5.1. Bei einem Trigramm Modell erhält man das HMM von Abb. 5.2.

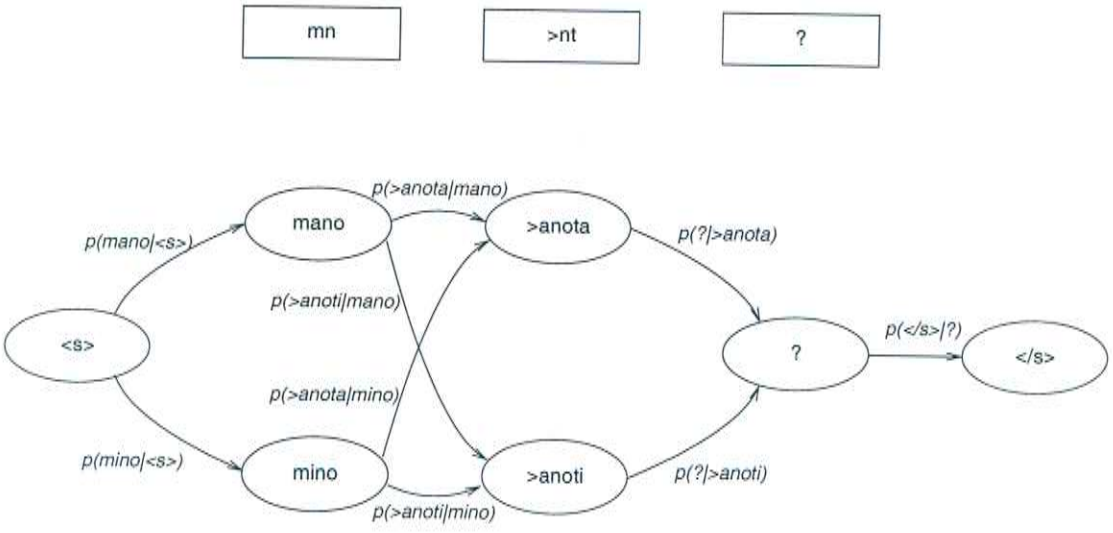


Abbildung 5.1: HMM mit Bigramm Modell

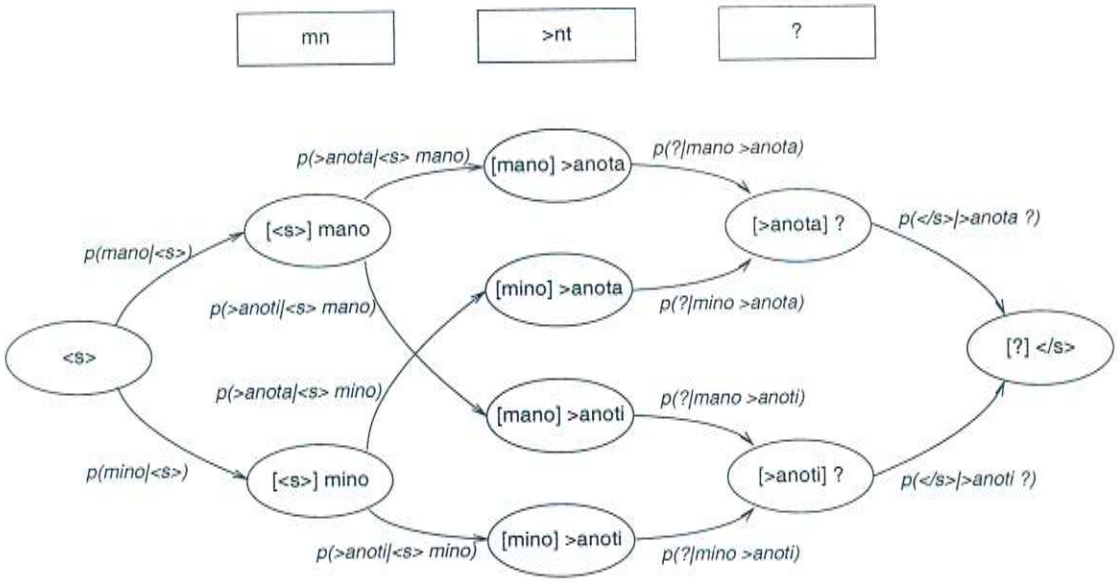


Abbildung 5.2: HMM mit Trigramm Modell

Die Suche nach der besten Vokalisierung entspricht also der Suche nach dem besten Pfad im HMM. Dafür kann man den Viterbi Algorithmus verwenden. Dieser Algorithmus führt eine Breitensuche aus. Bei jedem Schritt werden alle mögliche Übergänge vom aktuellen Zustand zum nächsten Zustand betrachtet, bis man zum Endzustand kommt. Die Wahrscheinlichkeit eines Pfades ist das Produkt aller Übergangswahrscheinlichkeiten des Pfades. Zu jedem Zustand wird der beste Vorgänger gespeichert. So kann am Ende der Suche vom letzten Zustand aus mittels *backpointer* der beste Pfad rekonstruiert werden. In Abb. 5.3 wird die Suche für das Beispiel veranschaulicht. Über jedem Zustand ist die log-Wahrscheinlichkeit des besten Pfades geschrieben, welcher in diesem Zustand endet. Die *backpointer* sind mit dicken Pfeilen gekennzeichnet. Der beste Pfad, und somit auch die gewählte Vokalisierung, ist: `mano >anota ?`, mit einer log-Wahrscheinlichkeit von ca. -6.

Einsatz und Ergebnisse

Zur Erzeugung des Sprachmodells wurde das *Language Modeling Toolkit* SRILM benutzt, das vom *SRI Speech Technology and Research Laboratory* entwickelt wurde (siehe [Stolcke, 2002]). Das Toolkit wurde gewählt, weil es aktuell und gut dokumentiert ist, über eine große Auswahl unterstützter Verfahren verfügt und auf vielen Plattformen (einschließlich Windows) verfügbar ist.

Das Toolkit enthält das Programm `ngram-count`, um Sprachmodelle zu berechnen. Mit verschiedenen Optionen können die Ordnung des Modells, die Behandlung unbekannter Wörter sowie Glättungsverfahren und deren Parameter bestimmt werden.

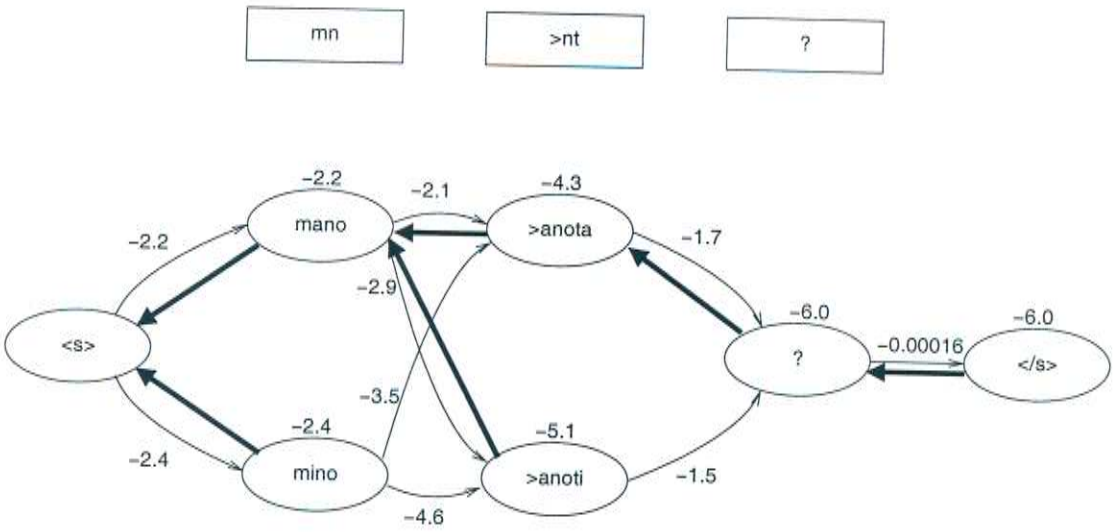


Abbildung 5.3: Viterbi Suche

Das Toolkit stellt auch das Programm `disambig` zur Verfügung, das gut für die Vokalisierung geeignet ist. Gegeben eine Wortfolge aus einem Vokabular $V1$, deren Wörter mehrere mögliche Übersetzungen im Vokabular $V2$ haben, und gegeben ein Sprachmodell über $V2$, dann kann die wahrscheinlichste Wortfolge aus $V2$, die der Wortfolge aus $V1$ entspricht, gesucht werden.

Eine typische Anwendung des Programms ist, mit ASCII Wörtern als $V1$ und akzentuierten nicht-ASCII Wörtern als $V2$ nach einer Akzentuierung, beispielsweise auf Spanisch, zu suchen. Wenn man als $V1$ nicht vokalisiertes Arabisch und als $V2$ vokalisiertes Arabisch nimmt, sucht das Programm die wahrscheinlichste Vokalisierung eines nicht vokalisierten Satzes.

Das Programm führt eine Viterbi Suche in einem HMM, wie auf S. 47 beschrieben, aus. Assoziationen zwischen nicht vokalisierten Formen und den entsprechenden möglichen vokalisierten Formen müssen in einer *mapping*-Datei definiert werden. Eine solche Assoziation sieht z.B. wie folgt aus:

```
ktb     kataba kutiba kutubK kutubN kutubi
```

Abb. 5.4 zeigt den gesamten Prozess: aus der Trainingsmenge werden die *mapping*-Datei und mit `ngram-count` das Sprachmodell erzeugt. Aus diesen Daten und dem nicht vokalisierten Text wird mit `disambig` der vokalisierte Text erzeugt.

Tabelle 5.8 zeigt die Ergebnisse (auf der Entwicklungsmenge getestet). Nur die Fehlerrate auf bekannten (trainierten) Wörtern wird angegeben, da unbekannte Wörter ohne Vokalisierung gelassen werden.

Im Vergleich zum *Baseline*-Verfahren ist die Fehlerrate bei mehrdeutigen Wörtern schon viel niedriger, aber immer noch ziemlich hoch.

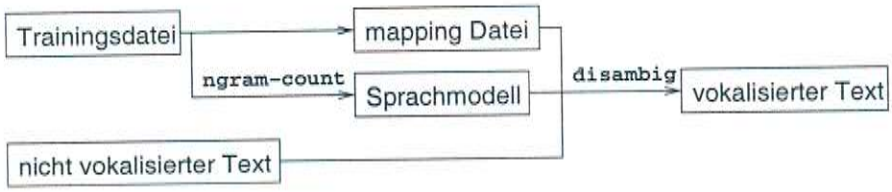


Abbildung 5.4: Vokalisierungsprozess mit Sprachmodell

	WER	SER	LER
Bekannte Wörter	14,72	4,07	2,27
Eindeutige Wörter	8,79	2,29	1,29
Mehrdeutige Wörter	16,86	4,86	2,70

Tabelle 5.8: Ergebnisse des Sprachmodellverfahrens

Optimierung des Sprachmodells

Die Erzeugung des Sprachmodells kann durch die Einstellung mehrerer Parameter geändert werden. Man kann verschiedene Parameter untersuchen, um die beste Einstellung des Sprachmodells zu erhalten.

Der Korpus ist im Vergleich mit den Korpora, die üblicherweise für die Sprachmodellierung benutzt werden, sehr klein. Das Sprachmodell, das mit den Standardparametern erzeugt wurde, enthält 95513 Unigramme, 333046 Bigramme und 39861 Trigramme.

Perplexität

Die Güte eines Sprachmodells wird oft anhand der Perplexität gemessen. Die Perplexität eines Sprachmodells auf einem Text T erhält man nach der Formel:

$$ppl(T) = 2^{-\frac{1}{|T|}\log_2 p(T)}$$

wobei $|T|$ die Länge des Textes ist. Man nimmt an, dass je kleiner die Perplexität ist, desto besser die Aussagekraft des Sprachmodells ist.

Man kann untersuchen, ob es in unserem Fall einen direkten Zusammenhang zwischen der Perplexität des Sprachmodells auf der Entwicklungsmenge und der Fehlerrate auf derselben Menge gibt. Ein solcher Zusammenhang würde ermöglichen, als Maß für die Optimierung die Perplexität statt der Fehlerrate zu nehmen, da es einfacher ist, die Perplexität als die Fehlerrate zu berechnen. Abb. 5.5 zeigt die Ergebnisse für mehrere Sprachmodelle, bei denen die Ordnung, die *cutoff*- und *discounting*-Parameter geändert wurden.

Man kann aus Abb. 5.5 schließen, dass es hier keinen direkten Zusammenhang zwischen der Perplexität und der Güte des Sprachmodells für die Vokalisierung

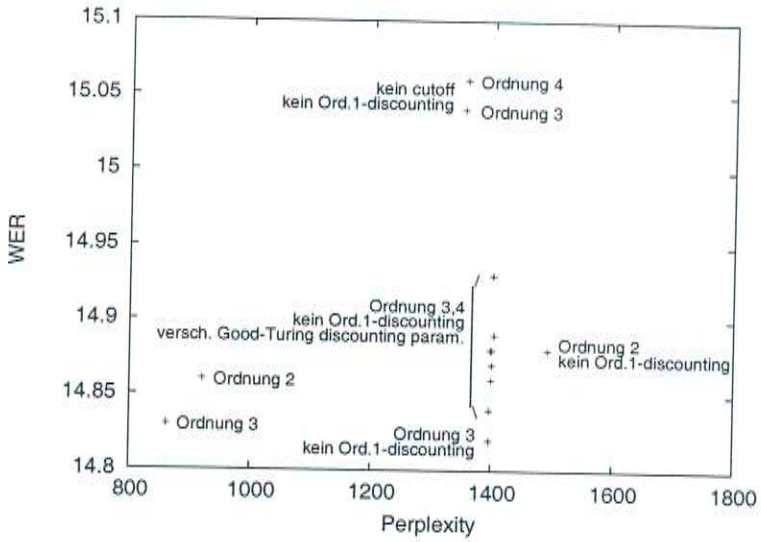


Abbildung 5.5: Zusammenhang zwischen Perplexität und Fehlerrate

gibt. Die Perplexität ist fast immer dieselbe, während die Fehlerrate stark variiert. Nur wenn bei der Ordunung 1 geglättet wird, sinkt die Perplexität, aber die Fehlerrate ist dadurch nicht immer niedriger.

Die Perplexität ist also kein gültiger Maß für die Güte des Sprachmodells im Bezug auf die Vokalisierung. Um die Vokalisierung zu optimieren, werden wir das Sprachmodell anhand der Fehlerrate bewerten.

Satzzeichen

Bisher wurden die Satzzeichen für das Sprachmodell wie andere Wörter betrachtet. Man kann untersuchen, ob die Ergebnisse besser wären, wenn die Satzzeichen aus dem Korpus gelöscht würden. Die Ergebnisse aus Tabelle 5.9 zeigen, dass dies aber nicht der Fall ist. Anscheinend sind Satzzeichen gute Hinweise für die Vokalisierung.[2]

	WER	SER	LER
Korpus mit Satzzeichen	14,72	4,07	2,27
Korpus ohne Satzzeichen	14,85	4,11	2,29

Tabelle 5.9: Einfluss der Satzzeichen im Sprachmodell

Ordnung

Die bisherigen Experimente wurden mit einem Sprachmodell der Ordnung 2

[2]Das kann man auch beim Beispiel in Tabelle 5.3 sehen.

(Unigrammen und Bigrammen) gemacht. Tabelle 5.10 zeigt, dass ein Sprachmodell der Ordnung 3 besser ist. Ein Sprachmodell der Ordnung 4 ist aber wiederum schlechter.

Der Unterschied ist aber sehr gering. Er ist auch nur bei der Wortfehlerrate zu sehen, weil es sich nur um einige Wörter handelt, die besser vokalisiert werden.

	WER	SER	LER
Ordnung 2	14,72	4,07	2,27
Ordnung 3	14,70	4,06	2,27
Ordnung 4	14,72	4,07	2,27

Tabelle 5.10: Vergleich verschiedener Ordnungen des Sprachmodells

Unbekannte Wörter

Mit der Standardeinstellung wird ein *closed-vocabulary*-Sprachmodell erzeugt. Das heißt, dass Wörter, die im Vokabular nicht vorkommen (als Vokabular werden alle Wörter des Trainingskorpus betrachtet) im Sprachmodell nicht betrachtet werden. Bei der Vokalisierung kommen aber unbekannte Wörter vor, die nicht im Trainingskorpus waren. Mit einem *closed-vocabulary*-Sprachmodell ist die Übergangswahrscheinlichkeit zu solchen Wörtern theoretisch 0 (in der Praxis wird sie als sehr klein angenommen) und die Wahrscheinlichkeit eines Satzes mit unbekannten Wörtern wird somit schlecht geschätzt.

Man sollte also ein *open-vocabulary*-Sprachmodell erzeugen. Bei einem solchen Modell wird das Zeichen `<unk>` eingeführt, das für unbekannte Wörter steht. Diesem Zeichen wird eine Unigramm-Wahrscheinlichkeit zugewiesen, was durch eine Verminderung der anderen Unigramm-Wahrscheinlichkeiten erreicht wird. Eine Sprache, in der unbekannte Wörter vorkommen, wird so besser modelliert. Die Ergebnisse in der Tabelle 5.11 zeigen, dass ein *open vocabulary*-Sprachmodell für die Vokalisierung bessere Ergebnisse erzielt.

	WER	SER	LER
closed-vocabulary	14,70	4,06	2,27
open-vocabulary	14,65	4,05	2,26

Tabelle 5.11: *open-vocabulary*- im Vergleich zum *closed-vocabulary*-Sprachmodell

Im oben erwähnten *open-vocabulary*-Sprachmodell wird das `<unk>` Zeichen nur als Unigramm betrachtet, es tritt aber nicht in n-Grammen höherer Ordnungen auf. Standardmäßig werden nämlich alle Wörter des Trainingkorpus als Vokabular betrachtet. Im Korpus werden somit keine Beispiele von n-Grammen mit

unbekannten Wörtern gefunden. Wenn bei der Vokalisierung ein n-Gramm mit einem unbekannten Wort gefunden wird, wird auf n-Gramme kleinerer Ordnung zurückgefallen, bis das n-Gramm im Sprachmodell gefunden wird.

Man könnte annehmen, dass es besser wäre, das <unk> Zeichen schon im Korpus zu simulieren, damit die Wahrscheinlichkeit von n-Grammen mit unbekannten Wörtern besser geschätzt wird (ohne Rückfall). Dazu muss man ein beschränktes Vokabular für die Erzeugung des Sprachmodells verwenden. Alle Wörter, die im Vokabular nicht enthalten sind, werden mit dem <unk> Zeichen ersetzt. Als Vokabular kann man die häufigsten Wörter nehmen, z.B. alle Wörter, die mehr als einmal im Korpus vorkommen. Somit werden alle Wörter, die nur einmal vorkommen, als unbekannte Wörter betrachtet.

Die Ergebnisse mit einem solchen begrenzten Vokabular sind aber schlechter: die Wortfehlerrate steigt dann auf 15,21%. Da der Korpus sehr klein ist, existieren zu viele Wörter, die nur ein Mal vorkommen, und infolgedessen aus dem Vokabular ausgeschlossen werden. Man hat nur noch 37627 Unigramme, was die Verschlechterung erklären könnte, auch wenn die Anzahl der Trigramme bei *cutoff*-Faktor 2 höher ist.

Glättung

Glättungsverfahren sind wichtig, da sie bestimmen, wie die Wahrscheinlichkeit ungesehener n-Gramme geschätzt wird. Mit `ngram-count` kann für jede Ordnung ein Glättungsverfahren gewählt werden: entweder Katz-Glättung (mit Good-Turing *discounting*), Witten-Bell-Glättung, Ristad's *natural discounting* oder die von Chen und Goodman veränderte Kneser-Ney-Glättung (siehe [Chen and Goodman, 1998]). Bei der Witten-Bell-Glättung und der Kneser-Ney-Glättung kann man auch ein interpoliertes Modell (siehe S. 45) anstatt eines *backoff*-Modells verwenden.

Beim Standardverfahren der Katz-Glättung scheint es Probleme mit der Berechnung der Wahrscheinlichkeiten für die Ordnung 1 (Unigrammen) zu geben, wahrscheinlich wegen des sehr kleinen Korpus. Tabelle 5.12 zeigt, dass für Ordnung 1 die Katz-Glättung keinen großen Einfluss hat. Nur die veränderte Kneser-Ney-Glättung (nicht interpoliert) ist deutlich besser, als die anderen Verfahren.

Man kann auch für höhere Ordnungen andere Glättungsverfahren verwenden. Nach [Chen and Goodman, 1998] ist die veränderte Kneser-Ney-Glättung oft am besten. Tabelle 5.13 zeigt, dass für Ordnung 2 die Kneser-Ney-Glättung tatsächlich die beste ist (aber nur mit Interpolation). Die Witten-Bell- und Ristad-Verfahren sind fast so gut. (Man verwendet hierbei Kneser-Ney-Glättung für die Ordnung 1 und Standard-Katz-Glättung für die Ordnung 3).

Man kann noch weitere Experimente durchführen um das beste Glättungsverfahren für die Ordnung 3 zu bestimmen (mit Kneser-Ney-Glättung für die Ordnung 1 und der interpolierten Kneser-Ney-Glättung für die Ordnung 2). Die interpolierte Kneser-Ney-Glättung ist hier wieder die beste.

	WER	SER	LER
Keine Glättung	14,66	4,05	2,26
Katz	14,65	4,05	2,26
Witten-Bell	14,64	4,05	2,26
Witten-Bell interp.	14,69	4,06	2,27
Ristad	14,64	4,05	2,26
Kneser-Ney	14,21	3,93	2,19
Kneser-Ney interp.	14,32	3,96	2,21

Tabelle 5.12: Glättungsverfahren für die Ordnung 1

	WER	SER	LER
Katz	14,21	3,93	2,19
Witten-Bell	14,11	3,91	2,18
Witten-Bell interp.	14,10	3,91	2,18
Ristad	14,11	3,91	2,18
Kneser-Ney	14,28	3,95	2,20
Kneser-Ney interp.	14,09	3,90	2,18

Tabelle 5.13: Glättungsverfahren für die Ordnung 2

	WER	SER	LER
Katz	14,09	3,90	2,18
Witten-Bell	14,08	3,90	2,17
Witten-Bell interp.	14,05	3,89	2,17
Ristad	14,12	3,91	2,18
Kneser-Ney	14,09	3,90	2,18
Kneser-Ney interp.	14,02	3,88	2,16

Tabelle 5.14: Glättungsverfahren für die Ordnung 3

Ähnliche Experimente zeigen, dass ein Sprachmodell der Ordnung 4 mit der interpolierten Kneser-Ney-Glättung für 4-Gramme die Wortfehlerrate noch leicht verbessert (14,01%). Allerdings bleibt die Buchstabenfehlerrate 2,16%. Die Ordnung 5 bringt keinen Gewinn mehr, und ist sogar leicht schlechter. Da der Aufwand für die Erzeugung des Sprachmodells mit der Ordnung wächst, und die Ergebnisse nicht deutlich besser sind, lohnt es sich allerdings nicht, ein Sprachmodell der Ordnung 4 zu erzeugen.

Wir werden also für die Ordnung 1 das *backoff*-Kneser-Ney-Verfahren benutzen und für die Ordnungen 2 und 3 das interpolierte Kneser-Ney-Verfahren.

cutoff-Faktoren

Die *cutoff*-Faktoren (siehe S. 45) sind ursprünglich dafür gedacht, die Größe des Sprachmodells zu verringern, wenn viele Daten vorliegen. In unserem Fall ist aber nicht zu erwarten, dass man auf dem kleinen Korpus bessere Ergebnisse durch *cutoff*-Faktoren größer als 1 erzielt.

Die Standardparameter sind ein *cutoff*-Faktor von 1 für die Ordnungen 1 und 2 (kein *cutoff*) und 2 für höhere Ordnungen. Für die Ordnungen 1 und 2 bringen höhere *cutoff*-Faktoren tatsächlich eine Verschlechterung der Vokalisierung (14,62% Fehlerrate mit *cutoff*-Faktor 2 für Unigramme, 15,82% Fehlerrate mit *cutoff*-Faktor 2 für Bigramme), wie auch zu erwarten war. Wie die Tabelle 5.15 zeigt, ist es bei Ordnung 3 auch leicht besser, einen *cutoff*-Faktor 1 zu verwenden und alle Trigramme zu behalten.

cutoff-Faktor	WER	SER	LER
1	14,00	3,87	2,16
2 (Standard)	14,02	3,88	2,16
3	14,09	3,90	2,18

Tabelle 5.15: Einfluss der *cutoff*-Faktoren für Trigramme

Varianten

Der Endvokal spielt bei arabischen Substantiven und Adjektiven eine besondere Rolle (siehe 2.6). Tabelle 5.16 zeigt, dass die meisten Fehler bei den Endvokalen gemacht werden.

Deshalb kann man überlegen, die Vokalisierung des Endvokals von der Vokalisierung des Stammes zu trennen. Die Motivierung dafür ist, dass der Endvokal oft von den anderen Endvokalen in der Umgebung abhängt, aber nicht von den anderen Wörtern selbst. Ein Adjektiv wird zum Beispiel nach einem mit *u* vokalisierten Substantiv auch mit *u* vokalisiert, unabhängig von der Bedeutung des Adjektivs und des Substantivs selbst.

		WER	SER	LER
Mit Endvokal	Bekannte Wörter	14,00	3,87	2,16
	Eindeutige Wörter	8,79	2,29	1,29
	Mehrdeutige Wörter	15,87	4,57	2,54
Ohne Endvokal	Bekannte Wörter	4,11	1,23	0,76
	Eindeutige Wörter	3,99	1,08	0,65
	Mehrdeutige Wörter	4,15	1,29	0,81

Tabelle 5.16: Anteil der Endvokalfehler an der Gesamtfehlerrate

Diese Vorgehensweise ähnelt der Idee bei morphembasierten Sprachmodellen, die in der Spracherkennung schon erforscht wurden [Byrne et al., 2001]. In unserem Fall ist das Ziel aber nicht, die OOV-Rate zu vermindern, sondern die Abhängigkeiten bei der Vokalisierung der Endungen besser zu modellieren.

Alle Wörter werden zwischen Stamm und Endvokal zerlegt. `AlkitaAbu Alkabiyru` ('das große Buch') wird zum Beispiel als `AlkitaAb #u Alkabiyr #u` abgetrennt. Das Sprachmodell wird wie in der Abb. 5.6 dargestellt mit diesen zerlegten Wörtern trainiert. Nach der Vokalisierung müssen die Stämme und Endungen wieder verbunden werden.

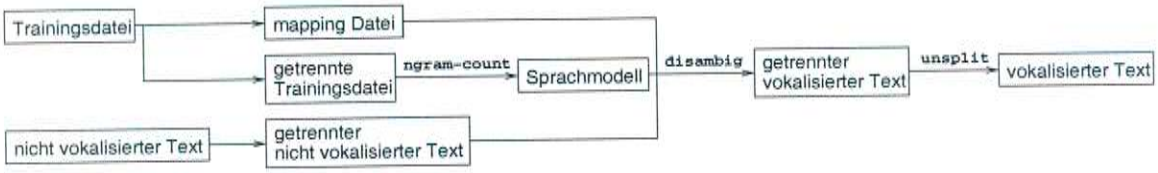


Abbildung 5.6: Prozess des Sprachmodellverfahrens mit abgetrenntem Endvokal

Die Ergebnisse, wie in der Tabelle 5.17 zu sehen, sind nicht sehr gut. Die Vokalisierung der Stämme ist etwas besser, allerdings ist gegenüber dem Standardsprachmodell eine Verschlechterung bei der Vokalisierung der Endvokale zu verzeichnen.

Problematisch ist, dass man bei der Trennung nicht weiß, um welche Wortart es sich handelt. Verben werden z.B. auch getrennt, obwohl bei Verben der Endvokal nicht getrennt von der Vokalisierung des Stammes gefunden werden sollte. Darüberhinaus ist bei Wörtern, die mit einem langen $\bar{a}$ (ا oder ى) enden der Endvokal nicht direkt am Ende des Wortes und es ist somit nicht klar, wo das Wort getrennt werden sollte.

Ein weiteres Problem ist, dass die Kontexte nicht mehr so breit sind - bei einem Trigramm hat man nur noch das vorige Wort satt der vorigen zwei Wörter. Die besten Ergebnisse erzielt man mit Ordnung 4 (anstatt mit Ordnung 3 ohne

		WER	SER	LER
Mit Endvokal	Bekannte Wörter	15,01	4,11	2,29
	Eindeutige Wörter	10,34	2,63	1,47
	Mehrdeutige Wörter	16,69	4,77	2,65
Ohne Endvokal	Bekannte Wörter	4,05	1,22	0,75
	Eindeutige Wörter	3,99	1,08	0,65
	Mehrdeutige Wörter	4,07	1,28	0,80

Tabelle 5.17: Ergebnisse des Sprachmodellverfahrens mit abgetrenntem Endvokal

abgetrennten Endvokal), die Kontexte von Wörtern werden aber nicht mehr so gut modelliert.

Eine weitere Möglichkeit wäre zwei Durchgänge zu machen. In einem ersten Durchgang werden die Stämme mit einem Sprachmodell über die Stämme (der Ordnung 3) vokalisiert. In einem zweiten Durchgang werden die Endvokale gefunden (mit einem Sprachmodell über die Stämme und Endvokale, der Ordnung 4). Der zweite Durchgang kann entweder mit dem `disambig` Programm gemacht werden oder mit dem `hidden-ngram` Programm, dessen Aufgabe es ist, versteckte *tags* (hier Vokale) zwischen Wörtern zu finden.

Wie die Tabelle 5.18 zeigt sind die Ergebnisse aber schlechter.

		WER	SER	LER
Mit Endvokal	Bekannte Wörter	15,12	4,14	2,31
	Eindeutige Wörter	10,44	2,64	1,48
	Mehrdeutige Wörter	16,81	4,80	2,67
Ohne Endvokal	Bekannte Wörter	4,08	1,22	0,75
	Eindeutige Wörter	3,99	1,08	0,65
	Mehrdeutige Wörter	4,11	1,28	0,80

Tabelle 5.18: Ergebnisse mit abgetrenntem Endvokal und zwei Durchgängen (mit `disambig`)

Bisher wurden für jedes Wort alle Vokale als mögliche Endvokale betrachtet. Dieses kann bei Wörtern die im Korpus eindeutig sind aber deren Endvokal nicht der Vokal ist, der bei der Vokalisierung erwartet wurde, zu besseren Ergebnissen führen. Das kann passieren, da viele Wörter selten sind und nur mit einer Endvokalisierung im Korpus gesehen wurden, obwohl auch andere Endvokalisierungen möglich wären. Der Nachteil bei der Endvokalabtrennung ist aber, dass für viele Wörter Vokalisierungen betrachtet werden, die gar nicht möglich sind - insbesondere für Verben. Das Ergebnis aus Tabelle 5.17 zeigt, dass die Vokalisierung der eindeutigen Wörter noch schlechter ist, als mit dem bisherigen Sprachmodell.

Eine weitere Variante betrachtet nur die Endvokalisierungen, die für das Wort im Korpus schon gesehen wurden. Ob man alle Endvokalisierungen betrachtet oder nur die bekannten kann auch nach der Häufigkeit des Wortes im Korpus bestimmt werden. Die Ergebnisse dieser Variante werden in der Tabelle 5.19 gezeigt. Mit zwei Durchgängen ist die Wortfehlerrate in diesem Fall 14,57%.

		WER	SER	LER
Mit Endvokal	Bekannte Wörter	14,50	3,98	2,22
	Eindeutige Wörter	8,79	2,29	1,29
	Mehrdeutige Wörter	16,55	4,72	2,63
Ohne Endvokal	Bekannte Wörter	4,06	1,22	0,75
	Eindeutige Wörter	3,99	1,08	0,65
	Mehrdeutige Wörter	4,08	1,28	0,80

Tabelle 5.19: Ergebnisse mit abgetrennten, beschränkten Endvokalen

Die Ergebnisse sind besser, als wenn alle Endvokale erlaubt werden, aber immer noch schlechter, als die Ergebnisse mit dem Standardsprachmodell.

Ein Vorteil dieser Verfahrensvariante könnte sein, dass es die Endvokale unbekannter Wörter besser findet. Mit dieser Verfahrensvariante werden nämlich Endvokale auch für unbekannte Wörter gefunden. Es könnte sein, dass diese Endvokale besser geschätzt werden, als mit den Verfahren für die Vokalisierung unbekannter Wörter auf Buchstabenebene (siehe 5.4). Man kann also diese Variante als Vokalisierungsverfahren für bekannte Wörter mit Vokalisierungsverfahren auf Buchstabenebene für unbekannte Wörter kombinieren, und bei dieser Kombination untersuchen, welche Endvokalisierung am besten ist.

Die Ergebnisse aus Tabelle 5.20 zeigen aber, dass die beste Endvokalisierung die Vokalisierung ist, die auf Buchstabenebene gefunden wird.

	WER	SER	LER
wortbasiert	69,08	17,20	9,38
buchstabenbasiert	49,80	12,85	7,01

Tabelle 5.20: Vokalisierung der Endvokale unbekannter Wörter

Dieser Versuch, die Vokalisierung mit dem Einsatz von morphologischem Wissen über die Sprache zu verbessern, war also nicht erfolgreich. Das Wissen, das hier benutzt wurde, ist aber nur sehr grob. Man würde bestimmt mit feinerem und weiterem Wissen bessere Ergebnisse erreichen.

5.4 Verfahren auf Buchstabenebene

Wie in [Mihalcea, 2002] gezeigt wird, kann man gute Ergebnisse beim Lernen auf der Buchstabenebene erreichen und insbesondere dann, wenn man nur einen kleinen Korpus und kein weiteres externes Wissen über die Sprache hat.

Da solche Verfahren die Wortkontexte ziemlich schlecht modellieren können, scheinen sie für bekannte Wörter, bei denen die Umgebung von anderen Wörtern für die Bestimmung der richtigen Vokalisierung entscheidend ist, nicht so gut geeignet zu sein.

Sie lassen es aber zu unbekannte Wörter zu vokalisieren. Wenn aus dem Korpus kein Wissen über das Wort selbst verfügbar ist, ist aber Wissen über die Buchstaben, aus denen das Wort besteht, verfügbar. Manchmal ist dieses Wissen ausreichend, um richtig zu vokalisieren.

Im *Baseline*-Algorithmus (siehe 5.2) wird für jeden Konsonant die häufigste Vokalisierung dieses Konsonanten gewählt. Ein besseres Verfahren wäre, wie auf der Wortebene auch die Kontexte der Buchstaben zu betrachten.

Es wurden zwei Verfahren untersucht: ein Verfahren mit sogenannten „gemischten Kontexten" und wieder ein Verfahren unter Verwendung eines Sprachmodells.

5.4.1 Gemischte Kontexte

Prinzip

Dieses Verfahren wurde in [Zweigenbaum and Grabar, 2002] vorgestellt (siehe S. 16). Das Prinzip ist die Ausnützung der Buchstabenkontexte für die Akzentuierung der Buchstaben.

Die Kontexte werden als „gemischte Kontexte" (*mixed context*) dargestellt. Diese Repräsentation wurde ursprünglich in [Theron and Cloete, 1997] vorgestellt. In gemischten Kontexten werden wechselweise die Buchstaben rechts und links des zu akzentuierenden Buchstabens betrachtet. Jeder Kontext wird mit der entsprechenden Akzentuierung assoziiert. Man hat also einen *transducer*, dessen Eingabe der Kontext und dessen Ausgabe die Akzentuierung ist.

Alle diese *transducer* werden als eine *trie*-Struktur repräsentiert, in der die gemeinsamen Präfixe der Kontexte faktorisiert werden. Die *trie*-Struktur ist ein lexikographisches Baum, bei dem jeder Knoten ein Buchstabe ist und jeder Pfad von der Wurzel bis zu einem Blatt ein Wort ist. Wenn ein Wort zur *trie*-Struktur hinzugefügt wird, wird zuerst der längste Präfix in der *trie*-Struktur gesucht und dann werden die benötigten weiteren Knoten hinzugefügt. In so einem Baum ist die Suche einer Zeichenkette optimal.

Mittels dieser *trie*-Struktur für die Kontexte kann man anhand des minimalen Kontexts entscheiden, welches die richtige Akzentuierung ist. Für einen gegebenen Kontext muss man nur in der *trie*-Struktur den Pfad suchen, der ein Präfix des Kontextes ist und zu einem Endzustand führt. Wenn der Kontext zu keinem

Endzustand führt, ist die Akzentuierung unentscheidbar.

Einsatz und Ergebnisse

Zuerst wurde eine vereinfachte Version verwendet.

Für jeden Buchstaben wird zuerst der gemischte Kontext bestehend aus zwei Buchstaben jeder Seite betrachtet, dann der gemischte Kontext bestehend aus einem Buchstaben jeder Seite, dann der Buchstabe selbst, bis ein bekannter Kontext gefunden wird. Die Akzentuierung, die am häufigsten in diesem Kontext vorkommt, wird gewählt. Beim Training muss man dafür nur für jeden möglichen Kontext die möglichen Vokalisierungen zählen.

Nehmen wir zum Beispiel das (nicht vokalisierte) Wort `AlktAb` ('das Buch'), bei dem wir den Buchstaben `l` vokalisieren wollen. Um den gemischten Kontext zu erzeugen, werden die Buchstaben wie folgt nummeriert: (wobei `@` und `#` den Anfang bzw. das Ende des Wortes kennzeichnen)

```
@ A l k t A b #
4 2 0 1 3 5 6 7
```

Man fängt mit dem zu vokalisierenden Buchstaben an und ordnet ihm die Zahl 0 zu, dann kommt der erste Buchstabe rechts, ihm ordnet man die Zahl 1 zu, dann der erste Buchstabe links, dann der zweite Buchstabe rechts usw. bis allen Buchstaben des Wortes eine Nummer zugewiesen wurde. Der vollständige gemischte Kontext ist also: `lkAt@Ab#`.

In der vereinfachten Version wird zuerst gesucht, ob `lkAt@` trainiert wurde. Wenn nicht, wird auf `lkA` zurückgefallen und dann auf `l` (was dem *Baseline*-Verfahren entspricht).

Bei dieser vereinfachten Version ist die Wortfehlerrate 69,21% für unbekannte Wörter und somit um einiges besser als beim *Baseline*-Verfahren. Es ist zu beachten, dass 75,56% der Buchstaben mit größerem Kontext und 21,99% mit kleinerem Kontext vokalisiert wurden und nur in 1,45% der Fälle musste man auf den Buchstaben selbst zurückfallen. Man kann also gute Ergebnisse von diesem Verfahren erwarten, wenn größere Kontexte betrachtet werden.

Eine weitere Version des Verfahrens diesmal mit *trie*-Struktur wurde entwickelt. Jeder Knoten dieser *trie*-Struktur enthält den Buchstaben an der aktuellen Position des Wortes und die Menge der Vokalisierungen, die für die Kontexte, die in diesem Zustand enden, möglich sind und ihrer Anzahl.

Beim Training wird für jeden Buchstaben eines Wortes der entsprechende gemischte Kontext erzeugt. Dieser Kontext wird zur *trie*-Struktur hinzugefügt und zur Menge der Vokalisierungen, die mit dem letzten Zustand dieses Kontextes in der *trie*-Struktur assoziiert ist, wird die Vokalisierung hinzugefügt.

Zum Beispiel ist die *trie*-Struktur, die in Abb. 5.7 dargestellt wird, ein Auszug aus der *trie*-Struktur, die man nach dem Training des Satzes `AlokitaAbu Alokabiyru lika` ('das große Buch gehört dir') bekommen würde.[3]

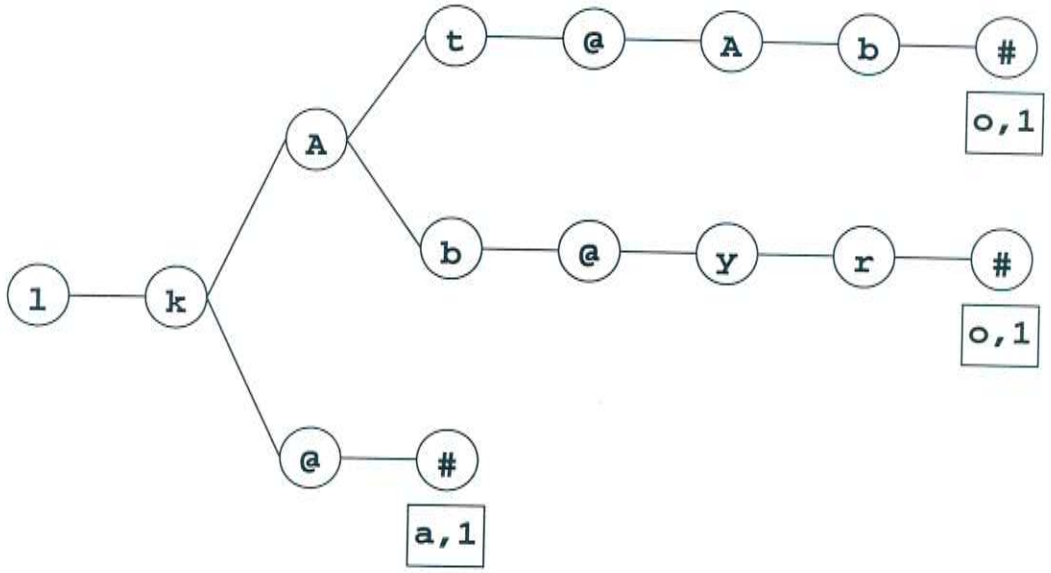


Abbildung 5.7: Repräsentation von gemischten Kontexten in einer *trie*-Struktur

Dann wird diese *trie*-Struktur ergänzt und vereinfacht. Bisher war nur zu den Endzuständen eine Menge Vokalisierungen assoziiert. Jetzt wird jedem Zustand eine solche Menge zugewiesen. Diese Menge wird aus der Vereinigung aller Mengen gebildet, die zu den aus diesem Zustand erreichbaren Endzuständen assoziiert sind. Sie kann rekursiv von den Blättern zur Wurzel berechnet werden. Dabei werden die Pfade, die eindeutig sind, abgeschnitten (nach einem Zustand, dessen Vokalisierung eindeutig ist, sind weitere Zustände nicht mehr nötig, d.h. man braucht keinen längeren Kontext).

In der Abb. 5.8 wird die *trie*-Struktur dargestellt, die man aus dem vorherigen Beispiel erhalten würde.

Nehmen wir an, dass wir jetzt den Buchstaben `l` des Wortes `Alkml` vokalisieren wollen. Der entsprechende gemischte Kontext ist `lkAm@l#`. Der Präfix `lkA` ist in der *trie*-Struktur zu finden und führt zu einem Zustand, bei dem die Vokalisierung `o` eindeutig ist.

Wenn wir den Buchstabe `l` des Wortes `mlk` vokalisieren wollen, ist der gemischte Kontext `lkm#@`. Dann ist nur der Präfix `lk` in der *trie*-Struktur zu finden. Bei dem

[3]Dies ist nur ein einfaches Beispiel. Es wurde in 4.5 erwähnt, dass die Vokalisierung des Artikels nicht gelernt wird, sondern mit Regeln gefunden wird. Man würde in der Wirklichkeit im Korpus die Vereinfachung `AlkitaAbu Alkabiyru lika` finden.

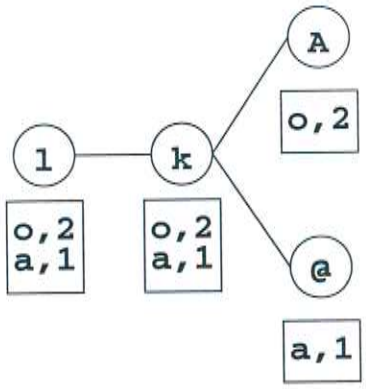


Abbildung 5.8: Ergänzte und vereinfachte *trie*-Struktur

erreichten Zustand ist die Vokalisierung nicht eindeutig. Man kann trotzdem die Vokalisierung o wählen, da sie in diesem Kontext am häufigsten vorkommt.

In der Praxis wird die *trie*-Struktur sehr groß und es wäre deshalb wünschenswert, die maximale Größe der Kontexte zu begrenzen. Experimente zeigen, dass es sich nicht lohnt, Kontexte größer als 9 Buchstaben zu betrachten.

Tabelle 5.21 enthält die Ergebnisse dieses Verfahrens auf den Entwicklungsdaten. Die Wortfehlerrate ist noch hoch. Wenn man auf Buchstabenebene die Vokalisierung bestimmt, ist eher die Silbenfehlerrate zu betrachten. Eine deutliche Verbesserung ist bei der Silbenfehlerrate im Vergleich zu den Ergebnissen des *Baseline*-Verfahrens zu sehen. In über 76% der Fällen kann die Vokalisierung eindeutig gewählt werden. In Abb. 5.9 wird die Verteilung der Kontextgrößen, die bei der Vokalisierung benutzt werden und von der *trie*-Struktur abgedeckt sind, dargestellt.

	WER	SER	LER
Unbekannte Wörter	63,21	16,23	8,85

Tabelle 5.21: Ergebnisse des Verfahrens mit gemischten Kontexten

5.4.2 Sprachmodell

Die gemischten Kontexte sind gut geeignet, wenn man einen einzelnen Buchstaben in einem Wort vokalisieren will. Wenn man aber alle Buchstaben eines Wortes vokalisieren will, könnte wie auf der Wortebene ein Sprachmodell besser sein. Die gemischten Kontexte haben gegenüber dem Sprachmodell den Nachteil, dass man nicht berücksichtigt, wie die anderen Buchstaben vokalisiert werden. Mit einem Sprachmodell wird nicht die Wahrscheinlichkeit der Vokalisierung eines einzelnen

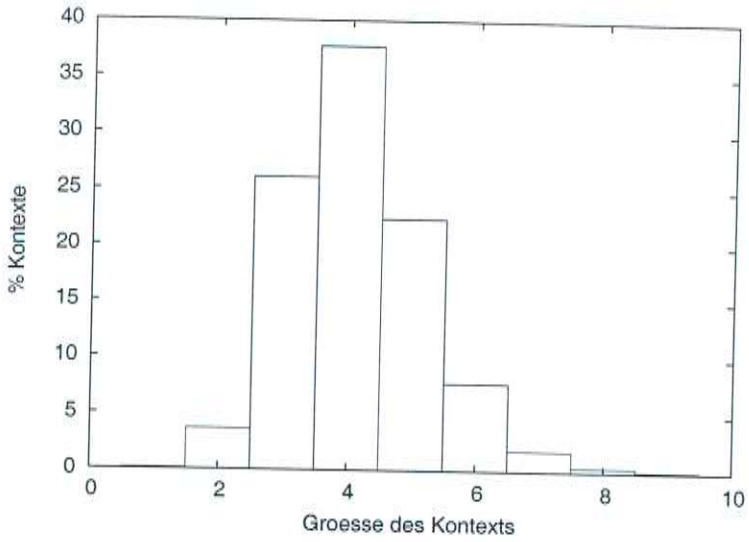


Abbildung 5.9: Größe der verwendeten Buchstabenkontexte

Buchstabens berechnet, sondern die Wahrscheinlichkeit der Vokalisierung einer Folge von Buchstaben.

Einsatz

Aus dem Korpus kann man ein buchstabenbasiertes Sprachmodell erzeugen, indem in diesem Sprachmodell die arabischen Wörter als Sätze genommen werden und die arabischen Silben (Konsonant + Vokalisierung) als Wörter des Vokabulars genommen werden. Der Trainingskorpus wird für die Erzeugung des Sprachmodells also umgewandelt. Die Wörter werden in Silben getrennt und jede Zeile enthält nur ein Wort. Der Satz `AlkitaAbu kabiyrN` ('Das Buch ist groß') ergibt zum Beispiel die zwei Zeilen:

```
A l ki ta A bu
ka bi y rN
```

Die *mapping*-Datei definiert eine Assoziation zwischen einem Buchstabe und seinen möglichen Vokalisierungen, z.B. :

```
j     j jF jK jN ja ji jo ju
```

Sonst ist das Prinzip dasselbe, wie auf der Wortebene.

Wenn man zum Beispiel das Wort `AlktAb` vokalisieren will, wird für den Satz `<s> A l k t A b </s>` das entsprechende Hidden Markov Modell erzeugt, und die Vokalisierung mit der höchsten Wahrscheinlichkeit gesucht.

Ergebnisse

Hier scheint wie auf Wortebene das veränderte Kneser-Ney-Glättungsverfahren mit Interpolation das beste zu sein. Bessere Ergebnisse bekommt man wiederum, wenn man kein *cutoff* verwendet. Da die Wörter des Vokabulars (die Buchstaben) oft vorkommen, kann man ein Sprachmodell einer höheren Ordnung erzeugen. Die Ordnung 5 scheint am besten zu sein.

Tabelle 5.22 gibt die Ergebnisse des Sprachmodellverfahrens auf Buchstabenebene für unbekannte Wörter an. Das Sprachmodellverfahren erzielt also eine niedrigere Fehlerrate, als das Verfahren mit gemischten Kontexten.

Ordnung	WER	SER	LER
4	55,67	14,54	7,93
5	50,42	12,79	6,97
6	50,46	12,81	6,99

Tabelle 5.22: Ergebnisse des Sprachmodellverfahrens für unbekannte Wörter auf Buchstabenebene

Variante

Auf der Buchstabenebene wurden bisher die Kontexte auf ein Wort begrenzt. D.h., die Vokalisierung der Buchstaben eines Wortes berücksichtigt nur die Buchstaben dieses Wortes und nicht die anderer Wörter in der Umgebung. Um die Begrenzung aufzuheben kann das Sprachmodell auch auf ganzen Sätzen trainiert werden. In diesem Fall ergibt der Satz `AlkitaAbu kabiyrN` für das Training `A l ki ta A bu # ka bi y rN`, wobei `#` ein Kennzeichen für die Grenzen zwischen Wörtern ist. Bei der Vokalisierung wird die wahrscheinlichste Folge von Silben für den ganzen Satz gesucht.

Wie Tabelle 5.23 zeigt, ist die Fehlerrate mit diesem Verfahren etwas niedriger.

	WER	SER	LER
Unbekannte Wörter	49,80	12,85	7,01

Tabelle 5.23: Ergebnisse des mit Sätzen trainierten Sprachmodellverfahrens auf Buchstabenebene

In diesem Fall müssen aber ganze Sätze aus zehn bis Hunderten Silben betrachtet werden. Im Gegensatz dazu bestehen im wortbasierten Sprachmodell die Sätze meist nur aus weniger als 50 Wörtern. Im buchstabenbasierten Sprachmodell, bei dem die Wortgrenzen nicht überschritten werden, bestehen die zu disambiguierenden Sätze ebenso aus einigen Dutzend von Buchstaben. Die Suche nach dem

besten Pfad im HMM ist hier also viel aufwändiger. Es wird folglich mehr Zeit für die Vokalisierung benötigt, aber die kleine Verringerung der Fehlerrate berechtigt diesen Aufwand im Allgemeinen nicht.

Bei diesem Verfahren werden nicht nur unbekannte Wörter, sondern der ganze Satz auf Buchstabenebene vokalisiert. Man kann also die Vokalisierung der bekannten Wörtern mittels dieses Verfahrens mit der Vokalisierung auf Wortebene vergleichen. Auf Wortebene war die niedrigste Fehlerrate 14%. Hier beträgt die Fehlerrate bei bekannten Wörtern 17,30%.

Obwohl Verfahren auf Wortebene für bekannte Wörter besser geeignet sind, kann es sich lohnen, bekannte Wörter, die nur selten im Korpus vorkommen auf Buchstabenebene zu vokalisieren. Das heißt, dass man bei diesen Wörtern den gesehenen Vokalisierungen nicht vertraut, da es gute Chancen gibt, dass andere gültige Vokalisierungen nicht gesehen wurden.

Dazu kann man eine Schwelle bestimmen. Wörter, deren Häufigkeit im Trainingskorpus unter diesem Schwellwert liegt, werden wie unbekannte Wörter behandelt und auf Buchstabenebene vokalisiert. Wörter, die häufiger als der Schwellwert vorkommen, werden auf Wortebene vokalisiert. Mit dem Schwellwert 1 werden die bekannte Wörter immer auf Wortebene vokalisiert.

Tabelle 5.24 gibt die Ergebnisse für einige Schwellwerte an. Keine Verbesserung wird für eindeutige Wörter erreicht. Die Fehlerrate bei mehrdeutigen Wörtern kann aber geringfügig verringert werden.

Schwelle	Eindeutige Wörter	Mehrdeutige Wörter	Insgesamt
1	8,79	15,87	14,00
2	9,21	15,87	14,11
3	9,42	15,79	14,10
4	9,58	15,80	14,16
5	9,71	15,90	14,27
6	9,85	15,91	14,31
9	10,05	16,18	14,56

Tabelle 5.24: Wortfehlerrate der Vokalisierung bekannter Wörter auf Buchstabenebene

Kapitel 6

Abschließende Ergebnisse

Nach den Experimenten im vorigen Kapitel erbringen die Verfahren, die auf einem Sprachmodell basieren, sowohl wortbasiert als auch buchstabenbasiert, die besten Ergebnisse.

Die Ergebnisse wurden bisher nur auf den Entwicklungsdaten gemessen. Als Abschluss der Untersuchung soll die Leistung des Systems auf bisher ungesehenen Testdaten gemessen werden.

Zuerst soll die Performance der Vokalisierung von ungesehenen Testdaten aus dem Korpus gemessen werden. Danach wird die Vokalisierung von Texten aus anderen Bereichen bewertet. Schließlich werden wir die Ergebnisse mit den Ergebnissen anderer Arbeiten vergleichen.

6.1 Testdaten aus dem Korpus

Wie in 5.1.1 beschrieben, wurde ein Teil des Korpus beiseite gelassen, um abschließende Experimente ausführen zu können. So kann das System unabhängig von den Daten, die zur Entwicklung und Validierung gedient haben, bewertet werden.

Auf diesen Testdaten beträgt die Wortfehlerrate 17,06% und die Silbenfehlerrate 4,95%. Dieses Ergebnis ist in Tabelle 6.1 aufgelistet. Es gibt also keinen wesentlichen Unterschied zu den Ergebnissen auf den Entwicklungsdaten (Wortfehlerrate 17,09% und Silbenfehlerrate 4,90%).

Mit dem *Baseline*-Verfahren beträgt die Wortfehlerrate auf denselben Daten 27,76% und die Silbenfehlerrate beträgt 10,51%. Das heißt, dass das System eine Reduzierung der Wortfehlerrate bzw. der Silbenfehlerrate um 10,7% bzw. 5,56% erbracht hat. Bei den bekannten Wörtern beträgt diese Reduzierung der Wortfehlerrate 7,5% und bei den unbekannten Wörtern beträgt sie 45,93%.

Die vorigen Ergebnisse messen die Leistung des Systems auf dem Korpus. Es wäre aber auch interessant zu ermitteln, wie gut das System Texte, die unabhängig vom Korpus sind, vokalisieren würde.

Dazu kann man das System auf einem Teil des Korpus trainieren und auf einem anderen disjunkten Teil testen. Zum Beispiel kann das System auf der Bibel trainiert und auf dem Koran getestet werden. Tabelle 6.1 enthält die Ergebnisse (Wortfehlerrate / Silbenfehlerrate, in Prozent) dieser Experimente. Mit „religiöse Texte" wird die Summe aus Bibel bestehend aus Altem Testament und Neuem Testament und Koran bezeichnet.

Training / Test	Alles	Rel. Texte	Altes Test.	Neues Test.	Koran
Alles	17,06 / 4,95	18,14 / 5,38	28,46 / 9,19	35,86 / 11,90	51,10 / 18,83
Rel. Texte	17,04 / 4,95	17,04 / 4,94	28,37 / 9,16	35,86 / 11,89	51,13 / 18,82
Altes Test.	17,94 / 5,14	17,94 / 5,15	18,35 / 5,30	35,66 / 11,63	53,38 / 19,67
Neues Test.	15,75 / 4,51	15,76 / 4,51	25,58 / 8,05	17,62 / 5,14	45,08 / 16,53
Koran	14,77 / 4,75	14,77 / 4,74	34,21 / 11,61	37,01 / 13,17	15,48 / 5,14
Wortschatz	18,60 / 5,32	32,48 / 11,53	33,66 / 12,64	35,09 / 13,47	45,83 / 21,42

Tabelle 6.1: Training und Test des Systems auf Texten verschiedener Quellen

Je kleiner der Trainingskorpus ist, desto höher ist natürlich die Fehlerrate. Nach dem Training auf dem Koran sind zum Beispiel 52% der Wörter des Alten Testaments unbekannt.

Die Ergebnisse zeigen, dass zwischen 15% und 20% Fehlerrate zu erwarten ist, wenn die Testdaten den Trainingsdaten ähnelt. Wenn der Bereich der Trainingsdaten und der Bereich der Testdaten ganz unterschiedlich sind, ist eine Fehlerrate zwischen 25% und 50% zu erwarten.

6.2 Vokalisierung von Alltagstexten

Das System kann auch bei der Vokalisierung auf Alltagstexten evaluiert werden. Für dieses Experiment wurde ein Text aus einem Zeitungsartikel[1] und ein Text aus einem Lehrbuch[2] genommen. Der Zeitungsartikel besteht aus 9 Sätzen und 280 Wörtern. Der Text aus dem Lehrbuch besteht aus 10 Sätzen und 167 Wörtern. Es ist zu beachten, dass die Sätze in diesen Texten viel länger sind, als die Sätze im Korpus (im Korpus sind die Sätze durchschnittlich 11 Wörter lang).

Der Zeitungsartikel wurde von Hand von einem Araber vokalisiert. Die Vokalisierung wurd von mehreren Personen überprüft. Dabei hat sich herausgestellt, dass Araber auch selbst häufig Fehler bei der Vokalisierung eines Textes machen. Nur gebildete Leute, die die arabische Sprache studiert haben, können einen Text

[1] Der Artikel wurde auf der Seite `www.arabia.com` gefunden.

[2] Das Buch [Deheuvels, 2000] enthält einige kurze vokalisierte Texte.

ohne Fehler vokalisieren. Es wurde dabei auch festgestellt, dass sogar einige Fehler im Zeitungsartikel enthalten waren. Außerdem haben die *hamza* Zeichen im Originaltext der Zeitung meistens gefehlt und mussten hinzugefügt werden.

Für das Training wurden bei diesen Experimenten alle verfügbaren Daten genommen, d.h. der ganze Korpus.

6.2.1 Wiederherstellung der Vokalzeichen

Tabelle 6.2 und 6.3 zeigen die Ergebnisse der Vokalisierung (mit vorhandenen Verdopplungszeichen).

	Anteil	WER	SER	LER
Bekannte Wörter	80,24	20,74	6,81	3,60
Unbekannte Wörter	19,16	62,50	14,13	7,67
Insgesamt	100	28,74	8,70	4,63

Tabelle 6.2: Ergebnisse der Vokalisierung des Lehrbuchtextes

	Anteil	WER	SER	LER
Bekannte Wörter	64,29	28,89	9,55	5,09
Unbekannte Wörter	35,71	77,00	18,63	9,97
Insgesamt	100	46,07	13,80	7,37

Tabelle 6.3: Ergebnisse der Vokalisierung des Zeitungstextes

Die Fehlerrate beim Zeitungstest ist viel höher, weil weniger Wörter vom Trainingskorpus abgedeckt sind. Das Vokabular in einer Zeitung ist wahrscheinlich komplexer, als in einem Lehrbuch. Insbesondere gibt es in einer Zeitung mehr Namen, die im Trainingskorpus nicht vorkommen (zum Beispiel Namen von Ländern und Personen).

6.2.2 Charakterisierung der Fehler

Die Fehler, die bei der Vokalisierung dieser Texte gemacht wurden, können von Hand analysiert und charakterisiert werden.

Im Text aus dem Lehrbuch wurden 48 Wörter falsch vokalisiert. Davon sind:

- 20 unbekannte Wörter
- 12 bekannte Wörter, deren richtige Vokalisierung im Korpus nicht bekannt war. Die Ursache dieser Fehler ist also direkt der zu kleine Korpus

- 16 bekannte mehrdeutige Wörter, deren Vokalisierung hätte gefunden werden können.

Diese Verteilung der Fehler ist in Tabelle 6.4 zusammengefasst.

Wortkategorie	Unbekannt	Bekannt Vok. unbekannt	Bekannt Vok. bekannt
Falsch vokalisierte Wörter	41,7	25,0	33,3
Falsch vokalisierte Silben	41,9	30,6	27,4

Tabelle 6.4: Verteilung der Fehler nach Wortkategorien bei der Vokalisierung des Lehrbuchtextes

Übrigens sind unter den falsch vokalisierten Wörtern 10 Namen und auch einige Wörter wie 'Doktor' oder 'französisch', die keine arabische Wurzel haben.

Die Hälfte der falschen Vokalisierungen könnte direkt durch den Korpus oder durch den Stamm (Namen oder fremde Wurzel) erklärt werden. Bei der anderen Hälfte sind zwei Drittel der Fehler eine falsche Kasusvokalisierung am Ende des Wortes. Im Besonderen bei mehrdeutigen bekannten Wörtern, deren richtige Vokalisierung bekannt war, sind bis auf einen alle Fehler Kasusfehler.

Auf Zeichenebene (Silbenfehlerrate) sind 62 Zeichen falsch. Davon sind 32 falsche Kasusvokalisierungen bei Substantiven, Namen und Adjektiven.

Eine Besonderheit des Arabischen ist, dass es verschiedene Ebenen der Aussprache für diese Kasusvokalisierungen gibt (siehe [Deheuvels, 2000] S. 54). Im klassischen Arabisch werden alle diese Vokale ausgesprochen, außer am Ende eines Satzes, wo sie nie ausgesprochen werden. Je spontaner die Sprache wird, desto weniger Kasusvokale werden ausgesprochen. Beim Vorlesen eines Textes werden Kasusvokale vor einem Komma weggelassen. In der spontanen Sprache werden fast alle Kasusvokale weggelassen. Nur die Kasusvokale die zur Verbindung mit dem nächsten Wort dienen, werden immer ausgesprochen. Bei Namen ist es sehr selten, dass der Endvokal ausgesprochen wird. Araber selbst machen viele Fehler bei der Kasusvokalisierung, insbesondere bei Namen.

Die Fehler bei der Kasusvokalisierung sind zwar schlimm, wenn sie ausgesprochen werden, aber sie würden oft in spontaner Sprache nicht auffallen, da man sie einfach weglassen würde. Bei der Anwendung des Vokalisierungssystems für die Sprachsynthese, wo manche Kasusvokalen nicht ausgesprochen würden, je nachdem, was die Ebene der gewünschten Sprache ist, wären also viele Fehler des Systems tolerierbar.

Bei den 32 falschen Kasusvokalisierungen würden 7 nie ausgesprochen, 2 würden nur in der höchsten Sprache ausgesprochen, 18 würden nur bei der Vorlesung ausgesprochen, nur 5 müssen immer ausgesprochen werden. Das heißt, dass nur 5 dieser Fehler für die Sprachsynthese spontaner Sprache wirklich problematisch

wären. Diese Fehler sind grammatikalische Fehler, die den Sinn des Wortes aber nicht ändern.

Es ist noch zu beachten, dass 2 der Fehler bei der *lām-ʾalif* Ligatur auftreten. Die Ligatur konnte nicht vokalisiert werden, da diese Vokalisierung im Korpus meistens fehlt (siehe 4.3.2 S. 25).

Die Verteilung der Fehler bei der Vokalisierung des Zeitungsartikels ist in Tabelle 6.5 aufgeführt.

Wortkategorie	Unbekannt	Bekannt Vok. unbekannt	Bekannt Vok. bekannt
Falsch vokalisierte Wörter	59,7	20,2	20,2
Falsch vokalisierte Silben	63,2	17,4	19,5

Tabelle 6.5: Verteilung der Fehler nach Wortkategorien bei der Vokalisierung des Zeitungstextes

Beim Zeitungsartikel treten 14% der falschen Wörter und sogar 22,6% der falschen Zeichen bei Namen oder Fremdwörtern auf. Die Vokalisierung solcher Wörter ist besonders schlecht: es gibt durchschnittlich mehr als zwei Fehler.

Für den Zeitungsartikel wurden keine weiteren Fehler analysiert.

6.2.3 Wiederherstellung aller diakritischen Zeichen

Meistens sind in Zeitungsartikeln die Verdopplungszeichen nicht vorhanden. Das war unter anderem im Testartikel der Fall.

Man kann mit dem System alle diakritischen Zeichen (Vokalzeichen sowie Verdopplungszeichen) wiederherstellen. Dazu muss das System mit Trainingsdaten, bei denen die Verdopplungszeichen in der nicht vokalisierten Form auch nicht vorhanden sind, trainiert werden.

In diesem Fall ist die Mehrdeutigkeit im Korpus etwas höher. 72,85% der Wörter sind mehrdeutig und es gibt durchschnittlich 3,6 Vokalisierungen pro Wort. Mit schon vorhandenen Verdopplungszeichen sind 69,65% der Wörter mehrdeutig und die durchschnittliche Anzahl der Vokalisierungen pro Wort beträgt 3,13 (siehe 4.6.2). Die Aufgabe ist also etwas schwieriger.

Die Ergebnisse, in Tabelle 6.6 zusammengefasst, sind wie erwartet etwas schlechter. Der Unterschied der Fehlerrate bei der Wiederherstellung aller Zeichen und bei der Wiederherstellung der Vokalzeichen ist aber nicht groß.

Es ist zu beachten, dass die Fehlerrate, wenn die Verdopplungszeichen nicht gezählt werden, noch 47,50% beträgt. Das heißt, dass die zusätzlichen Fehler im

	Anteil	WER	SER	LER
Bekannte Wörter	65	32,42	10,74	6,29
Unbekannte Wörter	35	77,55	19,46	10,76
Insgesamt	100	48,21	14,74	8,34

Tabelle 6.6: Ergebnisse der Wiederherstellung aller diakritischen Zeichen

Vergleich mit der Wiederherstellung der Vokale (46,07% Fehlerrate, siehe Tabelle 6.3) nicht nur bei den Verdopplungszeichen auftreten, sondern auch bei den Vokalen.

6.2.4 Bewertung des Trainingskorpus

Zuerst kann man untersuchen, wie gut die verschiedenen Quellen für den Trainingskorpus sind. Dazu kann man das System mit bestimmten Quellen trainieren und die Fehlerrate bei der Vokalisierung auf dem Zeitungsartikel messen.

Die Ergebnisse, die in Tabelle 6.7 zusammengefasst sind, zeigen, dass der Wortschatz, der den Korpus mit Ausdrücken des alltäglichem Arabisch ergänzt, die Abdeckung verbessert. Die Wortfehlerrate wird dabei auch niedriger. Der Unterschied besteht aus einem einzigen Wort.

Es ist auf jeden Fall besser, wenn die Bibel sowie der Koran im Korpus enthalten sind. Der Koran allein ist viel zu klein um eine brauchbare Abdeckung zu gewährleisten. Seine Hinzunahme zum Korpus bringt aber eine Verbesserung.

	Abdeckung	WER
Alles	64,29	46,07
Religiöse Texte	62,86	46,43
Altes Testament	60,36	48,21
Koran	38,93	60,71

Tabelle 6.7: Performance auf verschiedenen Trainingskorpora

Man kann auch untersuchen, ob die Homogenisierung und die Vereinfachung des Korpus (siehe 4.4 und 4.5) tatsächlich die Ergebnisse verbessern.

Wenn der Korpus nicht vorverarbeitet wird, beträgt die Wortfehlerrate 53,21%. Mit dem homogenisierten Korpus beträgt die Wortfehlerrate 51,79%. Mit dem vereinfachten Korpus beträgt sie 46,07%.

Die Vorverarbeitungen des Korpus lohnen sich also.

6.3 Vergleich mit anderen Arbeiten

In [Gal, 2002] wurde ein ähnliches Verfahren zum Sprachmodellverfahren, das in unserer Arbeit vorgestellt wird, untersucht. In dieser Arbeit wird ein HMM mit Bigrammen für die Vokalisierung arabischer Texte verwendet. Als Korpus wird ausschließlich der Koran verwendet. Die Testmenge besteht aus 10% des Korpus. Dabei ist die Fehlerrate 14%. Unbekannte Wörter, die 8% der Testmenge ausmachen, werden nicht vokalisiert. Bei bekannten Wörtern beträgt die Fehlerrate ca. 6,5%. Die Fehlerrate in dieser Arbeit ist also viel niedriger, als in unserer Arbeit. Es gibt aber wesentliche Unterschiede zwischen beiden Arbeiten, die den Unterschied zwischen den Fehlerraten erklären können.

Die Fehlerrate des *Baseline*-Verfahrens das als Referenz dienen soll, ist in unserer Arbeit bereits höher (27,76% gegenüber 26% in [Gal, 2002]).

In [Gal, 2002] wird der Koran als Quelle für den Korpus genommen, während in dieser Arbeit auch andere Quellen genommen werden. Wenn das System, das in dieser Arbeit entwickelt wurde, nur auf dem Koran trainiert und getestet wird, beträgt die Gesamtfehlerrate 15,48% und die Fehlerrate bei bekannten Wörtern beträgt 9,38%. (siehe Tabelle 6.1).

Ein sehr großer Unterschied zwischen beiden Arbeiten ist, dass die Arbeit von Gal auf einer Transkription des Arabischen basiert, während unsere Arbeit auf einer Transliteration basiert. Der Unterschied zwischen Transkription und Transliteration wurde in Abschnitt 2.9 erläutert.[3]

Grundsätzlich kann das System, das in unserer Arbeit entwickelt wurde, auf wirkliche arabische Texte angewendet werden, wie sie z.B. im Internet zu finden sind. Es muss nur eine Umwandlung des Textes durch die Buckwalter-Transliteration in eine ASCII Datei vorgenommen werden. Da diese Umwandlung nach der Vokalisierung wieder rückgängig gemacht werden kann, ist das Ergebnis auch wieder ein arabischer Text, der mit arabischen Buchstaben dargestellt werden kann.

Im Gegensatz dazu kann das System von Gal eine Transkription von Arabisch vokalisieren aber keinen richtigen arabischen Text. Die vokalisierte Transkription kann nicht in einen Text mit arabischen Buchstaben zurückgewandelt werden, da sie mehrdeutig ist. Zum Beispiel kann die Transkription *at* entweder für تَ oder für ةَ stehen.

Aus dem Koran werden also in beiden Arbeiten unterschiedliche Daten erzeugt, in der Arbeit von Gal durch eine Transkription und in unserer Arbeit durch eine Transliteration gefolgt von einer Homogenisierung und Vereinfachung. Deshalb

[3] Gal verwendet die Bezeichnung „Transliteration". Der Unterschied zwischen beiden Begriffen ist in der Literatur und im Internet nicht immer sehr klar, aber wir verwenden hier die Bezeichnung mit der klaren Definition wie sie von Beesley gegeben wird (siehe 2.9).

sind die Ergebnisse auch nicht direkt vergleichbar. Die Fehlerrate beim System von Gal ist zwar niedriger, aber die Anwendung dieses Systems ist auf Transkriptionen beschränkt.

Eine direkte Folge der Benutzung einer Transkription ist auch, dass weniger Vokale aufgrund der Nunation (siehe S. 6 und S. 10) wiederhergestellt werden müssen.

Für كِتَابٌ steht in einer Transkription *kitābun*. Das *n* Zeichen wird also geschrieben, als ob es ein Konsonant wäre, obwohl es beim arabischen Wort im diakritischen Zeichen enthalten ist. Die nicht vokalisierte Form ist dann *ktābn*. Nur der Vokal *u* muss gefunden werden. Insgesamt müssen also nur 3 Vokale wiederhergestellt werden: *a*, *i*, *u*.

Die Buckwalter-Transliteration desselben Wortes lautet `kitaAbN`. Das Zeichen `N` steht also für das Zeichen ٌ (den Nunation-Vokal *un*), welches sich vom Zeichen `u` das für das Zeichen ُ (den Vokal *u*) steht, unterscheidet. Insgesamt müssen 6 Vokalzeichen wiederhergestellt werden. Für jeden Vokal gibt es auch den entsprechenden Nunation-Vokal: `a/F`, `i/K`, `u/N`.

Wenn die Nunation-Fehler nicht gezählt werden (d.h. `u` und `N` werden nicht unterschieden) beträgt die Gesamtfehlerrate in unserer Arbeit 14,85% und die Fehlerrate bei bekannten Wörtern beträgt 9,09% (beim Training und Testen auf dem Koran).

Außerdem ist durch die Darstellung der Nunation in einer Transkription die Mehrdeutigkeit niedriger. Die Wörter كِتَابُ und كِتَابٌ haben verschiedene nicht vokalisierte Formen: *ktāb* und *ktābn*. Bei einer Transliteration ist die nicht vokalisierte Form für beide Wörter `ktAb`. Dies entspricht besser der Wirklichkeit, da die nicht vokalisierte Form mit der arabischen Schrift auch für beide Wörter die gleiche ist: كتاب.

Die Korpora und dadurch die Schwierigkeit der Vokalisierung in beiden Arbeiten sind also verschieden, weshalb die Ergebnisse nicht direkt vergleichbar sind.

Kapitel 7

Zusammenfassung und Ausblick

In dieser Arbeit wurden verschiedene Verfahren für die Wiederherstellung der diakritischen Zeichen (Vokalisierung) arabischer Texte untersucht.

Als Maß für die Schwierigkeit dieser Aufgabe kann die Mehrdeutigkeit innerhalb der Texte verwendet werden. Im Korpus, der in dieser Arbeit verwendet wurde, sind ca. 70% der Wörter mehrdeutig mit einer durchschnittlichen Anzahl von etwas mehr als 3 Vokalisierungen pro Wort.

Arbeiten im Bereich NLP des Arabischen haben sich bisher eher mit Morphologie beschäftigt und die Wiederherstellung der diakritischen Zeichen wurde hauptsächlich bei europäischen Sprachen mit einer nur geringen Mehrdeutigkeit untersucht. Statistische Verfahren für das Arabische wurden bisher kaum untersucht.

Der erste Schritt dieser Arbeit, die Datensammlung, war aufwändiger als erwartet. Es zeigte sich, dass es nur sehr wenige vokalisierte arabische Texte gibt, wobei diese für statistische Verfahren aber unbedingt notwendig sind. Außerdem sind die vokalisierten Texte auf Arabisch, die heutzutage in elektronischer Form zu finden sind, von schlechter Qualität. Es gibt noch viele Kodierungsprobleme der arabischen Schrift, insbesondere werden vokalisierte Texte oft auf dem Bildschirm schlecht dargestellt. Mit der Verbreitung von Unicode sollten diese Probleme hoffentlich bald der Vergangenheit angehören.

Bei allen untersuchten Verfahren wird zuerst aus dem Korpus ein Lexikon mit nicht vokalisierten Wörtern und ihren möglichen Vokalisierungen erzeugt. Die Wörter, die im Lexikon stehen, werden auf Wortebene vokalisiert, wobei der Wortkontext berücksichtigt wird. Die Wörter, die nicht im Lexikon stehen, werden auf Buchstabenebene vokalisiert.

Auf Wortebene wurde zuerst ein kollokationsbasiertes Verfahren untersucht. Trotz der guten Ergebnissen leidet dieses Verfahren hauptsächlich unter zwei Nachteilen: es ist ziemlich aufwändig und nur die nicht vokalisierten Wörter werden im Kontext berücksichtigt. Es hat aber den Vorteil, dass beliebige Zusammenhänge zwischen Wörtern dargestellt werden können.

Ein weiteres Verfahren mit besseren Ergebnissen basierte auf einem Sprachmodell. Verschiedene Einstellungen wurden für die Optimierung des Sprachmodells untersucht, vornehmlich verschiedene Glättungsverfahren. Die Kneser-Ney-Glättung hat sich als besonders geeignet erwiesen. Versuche, die Vokalisierung der Endungen der Wörter besser zu modellieren, waren allerdings nicht erfolgreich.

Auf Buchstabenebene wurden zuerst „gemischte Kontexte" mit einer *trie*-Struktur untersucht. Damit wurde eine deutliche Verbesserung der Fehlerrate auf unbekannten Wörtern erreicht. Der Einsatz eines Sprachmodells auf Buchstabenebene erzielte aber noch bessere Ergebnisse.

Die durchgeführten Experimente zeigen also die Stärke der Sprachmodellierung für die Wiederherstellung von diakritischen Zeichen. Abb. 7.1 und Abb. 7.2 fassen die Ergebnisse der verschiedenen untersuchten Verfahren nochmals zusammen.

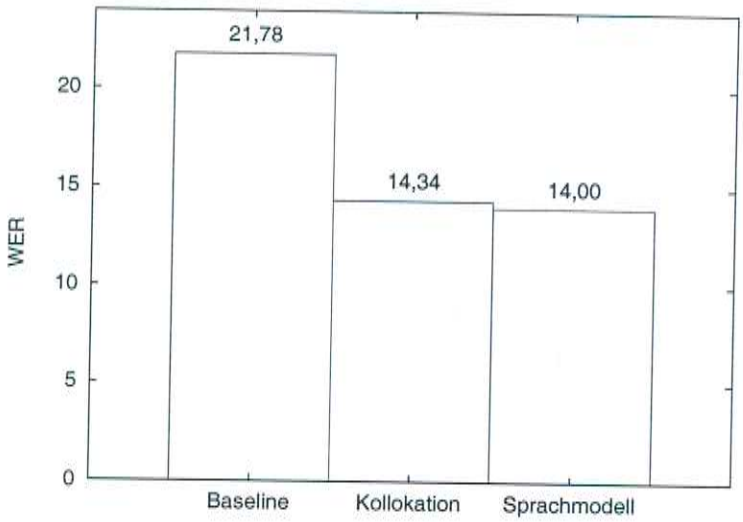


Abbildung 7.1: Wortfehlerrate bei bekannten Wörtern auf Wortebene

Beim Testen des besten Systems (sprachmodellbasierte Verfahren auf Wort- und Buchstabenebene) auf einer Testmenge aus dem Korpus beträgt die Gesamtfehlerrate ca. 17% und die Silbenfehlerrate ca. 5%. Bei Tests mit Alltagstexten hat sich herausgestellt, dass bei Texten, die vom Trainingskorpus weiter entfernt sind, eine höhere Wortfehlerrate von 25% bis 50% zu erwarten ist. Auf Zeichenebene liegt die Fehlerrate zwischen 8% und 15%. Die meisten Fehler werden bei der Kasusvokalisierung gemacht. Als Hauptursache der Fehler ist der zu kleine Umfang des Trainingskorpus zu nennen.

Es wurde auch gezeigt, dass das System bei einer ähnlichen Fehlerrate alle diakritischen Zeichen (Vokalzeichen sowie Verdopplungszeichen) wiederherstellen kann.

Das Vokalisierungssystem, das in dieser Arbeit entwickelt wurde, kann auf un-

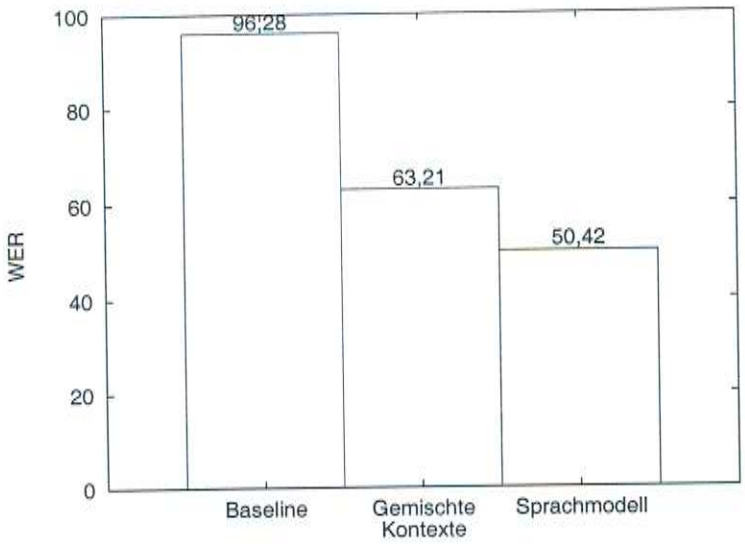


Abbildung 7.2: Wortfehlerrate bei unbekannten Wörtern auf Buchstabenebene

terschiedliche Art und Weise verwendet werden. Es kann für die arabische Sprachsynthese benutzt werden, zum Beispiel in Sprachübersetzungssystemen. Die automatische Vokalisierung arabischer Texte kann auch für Arabisch-Lernende sehr hilfreich sein, da sie einen nicht vokalisierten Text oft nicht lesen und nur schlecht verstehen können.

Von Hand ist die Vokalisierung eines arabischen Textes sehr aufwändig. Obwohl die Fehlerrate der automatischen Vokalisierung noch ziemlich hoch ist, kann sie als Ausgangsbasis sehr hilfreich sein.

In dieser Arbeit wurde versucht alle Wörter zu vokalisieren, auch wenn die Vokalisierung nicht sicher ist. Man könnte auch nur die Vokale bestimmen, bei denen man sich mit hoher Wahrscheinlichkeit sicher ist.

Diese Arbeit hat gezeigt, dass die Verwendung statistischer Verfahren, ohne externe Wissensquellen, für die Vokalisierung arabischer Texte für manche Anwendungen akzeptable Ergebnisse bringt. Die Korrektheit der Vokalisierung mittels solcher Verfahren kann sicher noch verbessert werden.

Eine erste Verbesserung wäre, die diakritische Zeichen, die für die Eindeutigkeit des Sinnes in nicht vokalisierten Texten manchmal vorhanden sind, mit zu berücksichtigen. Im aktuellen System werden diese Zeichen bei der Vokalisierung nicht beachtet.

Wortbasierte und buchstabenbasierte Verfahren könnten auch besser kombiniert werden. Diese Arbeit hat die Beschränkung des reinen wortbasierte Verfahrens für bekannte Wörter gezeigt, wenn der Korpus zu klein ist. Außerdem ist durch die Verkettung mancher Präpositionen und Pronomen ein Teil des Kontextes in Wörtern selbst enthalten. Auf Buchstabenebene könnten nicht nur die

umgebende Buchstaben, sondern auch die umgebenden Wörter mitberücksichtigt werden.

Auch bei Sprachmodellen wären Erweiterungen denkbar. Eine wertvolle Erweiterung bei einem kleinen Korpus wäre wahrscheinlich ein klassenbasiertes Sprachmodell. Das würde unter anderem das Problem der unbekannten (OOV) Wörter lösen. Es bleibt noch zu untersuchen, was mögliche Klassen wären. Es könnten zum Beispiel Vokalisierungsschablonen oder grammatische Klassen sein. Andere Varianten wie Cache Sprachmodelle oder Sprachmodelle mit Lücken-n-Grammen könnten auch untersucht werden.

Andere Lernverfahren wie neuronale Netze wären auch für die Vokalisierung arabischer Texte denkbar.

Eine erste Herausforderung für weitere Arbeiten wäre auf jeden Fall, einen größeren vokalisierten Korpus manuell zu erzeugen, welcher sich evtl. näher am aktuellen Wortschatz orientiert.

Desweiteren gibt es einen sehr engen Zusammenhang zwischen der Morphologie und der Vokalisierung arabischer Wörter. In den Arbeiten von [Itai and Segal, 2000] (mit Hebräisch) und [Ramsay and Mansur, 2001] werden die möglichen morphologischen Analysen für ein Wort erzeugt. Dann wird versucht, anhand morphologischer Informationen die richtige Analyse zu finden. Von dieser kann man dann auf die Vokalisierung schließen.

In dieser Arbeit werden die möglichen Vokalisierungen erzeugt und die beste wird je nach Kontext des Wortes gewählt. Dabei wird im Vergleich zu den zitierten Arbeiten die Morphologie nicht betrachtet, sondern es werden nur Statistiken der Wortformen verwendet. Von der gewählten Vokalisierung kann man nicht direkt auf die morphologische Analyse schließen. Die Vokalisierung ermöglicht es aber, zwischen den möglichen morphologischen Analysen, die ein Analysator liefert, zu entscheiden, da eine Vokalisierung meistens einer eindeutigen morphologischen Analyse entspricht.

Eine mögliche Erweiterung wäre, die Morphologie und die Vokalisierung gleichzeitig zu betrachten. Nicht nur Wortformen sondern auch Vokalisierungsschablonen sollten betrachtet werden. Es könnten Statistiken über die Trennung von Wörtern in Präfixe, Suffixe, Infixe, Konsonantenwurzel und Vokalschablone aufgestellt werden, ähnlich wie in [Goldsmith, 2001], [Déjean, 1998] oder [Schone and Jurafsky, 2000] bei europäischen Sprachen. Man könnte so ein statistisches System entwickeln, das mittels eines unüberwachten Lernverfahrens gleichzeitig die wahrscheinlichste Vokalisierung und die beste morphologische Analyse der Wörter bestimmen könnte. Ein solches System würde keinen externen Analysator benötigen, um die Struktur der Wörter zu bestimmen.

Auf jeden Fall sollten morphologische und syntaktische Informationen für die Wahl der richtigen Vokalisierung berücksichtigt werden. Diese könnten gelernt werden oder aus externen Wissensquellen stammen.

Anhang A

Die Buckwalter-Transliteration

Buchstabe	Name	Unicode	Buckwalter
ء	*hamza* on the line	U+0621	'
آ	*madda* on *ʾalif*	U+0622	|
أ	*hamza* on *ʾalif*	U+0623	>
ؤ	*hamza* on *wāw*	U+0624	&
إ	*hamza* under *ʾalif*	U+0625	<
ئ	*hamza* on *yāʾ*	U+0626	}
ا	*ʾalif*	U+0627	A
ب	*bāʾ*	U+0628	b
ة	*tāʾ marbūṭa*	U+0629	p
ت	*tāʾ*	U+062A	t
ث	*ṯāʾ*	U+062B	v
ج	*ǧīm*	U+062C	j
ح	*ḥāʾ*	U+062D	H
خ	*ẖāʾ*	U+062E	x
د	*dāl*	U+062F	d
ذ	*ḏāl*	U+0630	*
ر	*rāʾ*	U+0631	r
ز	*zāy*	U+0632	z
س	*sīn*	U+0633	s
ش	*šīn*	U+0634	$
ص	*ṣād*	U+0635	S
ض	*ḍād*	U+0636	D
ط	*ṭāʾ*	U+0637	T
ظ	*ẓāʾ*	U+0638	Z

ع	*ʿayn*	U+0639	E
غ	*ġayn*	U+063A	g
ـ	*taṭwīl*	U+0640	_
ف	*fāʾ*	U+0641	f
ق	*qāf*	U+0642	q
ك	*kāf*	U+0643	k
ل	*lām*	U+0644	l
م	*mīm*	U+0645	m
ن	*nūn*	U+0646	n
ه	*hāʾ*	U+0647	h
و	*wāw*	U+0648	w
ى	*ʾalif maqṣūra*	U+0649	Y
ي	*yāʾ*	U+064A	y
ً	*fatḥa tawīla*	U+064B	F
ٌ	*ḍamma tawīla*	U+064C	N
ٍ	*kasra tawīla*	U+064D	K
َ	*fatḥa*	U+064E	a
ُ	*ḍamma*	U+064F	u
ِ	*kasra*	U+0650	i
ّ	*šadda*	U+0651	~
ْ	*sukūn*	U+0652	o
ٰ	*ʾalif maḥḏūfa*	U+0670	`
آ	*waṣla* on *ʾalif*	U+0671	{

Tabelle A.1: Die Buckwalter-Transliteration

Anhang B

Arabische Ressourcen im Internet

B.1 Arabische Sprache

Webarabic

`http://www.webarabic.com/`
Gateway auf Französisch für die arabische Sprache und Kultur.
Einige Ressourcen für Arabisch-Lernende findet man unter
`http://www.webarabic.com/choix-apprendre.html`, unter anderem Vokabellisten (nicht vokalisiert), einige vokalisierte Gedichte und Sprichwörter. Allerdings sind die arabischen Texte als Bild (gif) dargestellt.

Arabic Software, Tutorials & More

`http://www.geckil.com/~harvest/arabic/`
Umfangreiche Links zu arabischen Ressourcen, unter anderem

- Zeitungen
- Einführungen in die arabische Sprache
- Vokabular
- Grammatik
- Transliterationen

Arabe-Français

`http://pince31.free.fr/lang/arabic/liens.htm`
Links zu Seiten für Arabisch-Lernende.

Vokabular und Grammatik unter `http://pince31.free.fr/lang/arabeindex.htm` (Dateien im XML-Format, UTF-8 kodiert, keine Sätze sondern einzelne Wörter, Endungen nicht vokalisiert)

Textarab

`http://www.textarab.org/`
Magazin für Arabisch-Lernende mit arabischen Texten in französisch.
Bietet PDF-Dateien an, unter denen vokalisierte Sprichwörter unter `http://www.textarab.org/Proverbesarabes.htm` (Vokalisierung nicht vollständig).

Arabic for Non Arab

Arabische Ressourcen für Nicht-Araber unter `http://www.lootah.sch.ae/ArabicTutor/MenuEng.htm`, darunter ca. 120 Sätze und 9 Dialoge, teilweise vokalisiert, als gif-Dateien.

Arabic2000.Com

`http://www.arabic2000.com/`
Bietet Arabisch Kurse an, enthält einige kostenlose online Ressourcen.
Ca. 20 einfache Ausdrücke, vokalisiert, als gif-Dateien, unter `http://www.arabic2000.com/arabic/public/common.html` (mit englischer Übersetzung und Aussprache in RealAudio-Format).

The Arab World

`http://cecilmarie.web.prw.net/arabworld/arabic/`
Vokabular und einfache Ausdrücke, als gif-Datei oder nur als Transkription.

Babel : Arabic

`http://www.i-cias.com/babel/arabic/index.htm`
Einführung in die arabische Schrift, 5 Listen bestehend aus einfachen Ausdrücken (als Transkription) mit Aussprache (RealAudio), 9 kurze Dialoge mit arabischem Text als gif-Datei (nicht vokalisiert), Transkription und Aussprache (RealAudio).

Muttaqun OnLine

`http://muttaqun.com/arabic/`
Vokabular, meistens nur Transkription, manchmal auch gif-Darstellung, nicht vokalisiert, mit Aussprache in RealAudio-Format.

Islamic Studies

http://www.geocities.com/Athens/Thebes/8206/hkrausen/studies.htm
10 Lektionen für nicht-Araber um den Koran lesen zu können. Gescannte Texte und Vokabular als gif-Dateien.

Transparent Language

http://www.transparent.com/
Die Seite bietet Sprachensoftware an.
Arabische Ressourcen unter http://www.transparent.com/languagepages/arabic/FSArabic.htm (einfache Ausdrücke, als Transkription, mit Aussprache).

Travlang

http://www.travlang.com/
Sprachressourcen für Reisende.
Arabische Ausdrücke unter http://www.travlang.com/languages/cgi-bin/langchoice.cgi?lang1=french&lang2=arabic (Transkription, Aussprache, nicht vokalisiertes Arabisch als gif-Datei).

al-bab.com

http://www.al-bab.com/
Gateway zur arabischen Welt.
Einführung in die arabische Sprache unter
http://www.al-bab.com/arab/language/lang.htm
Sprichwörter unter http://www.al-bab.com/arab/language/lang.htm#PROVERBS
(nicht vokalisiert, als gif-Datei).

UK INDIA

http://www.ukindia.com/
Sprachressourcen für orientalische Sprachen.
Einführung in die arabische Schrift unter http://www.ukindia.com/zar1.htm
Wörter, teilweise vokalisiert, als gif-Datei.

Basic words & Expressions

http://mlang1.osaka-gaidai.ac.jp/~tagengo/languages/Arabic.html
Einige Ausdrücke, teilweise vokalisiert, UTF-8 Kodierung, mit Aussprache, englische und japanische Übersetzung.

Ahlul Bayt Digital Islamic Library Project (DILP) - Common Arabic Expressions

`http://www.al-islam.org/about/education/arabic.doc`
Ca. 35 religiöse Ausdrücke, vokalisiert, als Word-Datei.

Ajeeb - Learn Arabic

Ressourcen für Arabisch Lernende.
Kostenlose online Kurse unter `http://afl.ajeeb.com/freetour/menu/menu.html`

B.2 Koran

Multimedia Quran with transliteration

`http://www.geckil.com/~harvest/quran/`
Koran als gif-Bilder mit Transkription und Aussprache (RealAudio).

Al-Islam.org

`http://www.al-islam.org/default.asp`
Seite vom Ahlul Bayt Digital Islamic Library Project (DILP). Sammlung von Texten, Artikeln über den Islam.
Koran (arabischer Text, verschiedene englische Übersetzungen) unter `http://www.al-islam.org/quran/`.

Al-Islam - The Holy Qur'an

`http://quran.al-islam.com/`
Ressourcen über den Koran.
Koran als gif-Bilder mit Aussprache und Übersetzungen in verschiedene Sprachen unter `http://quran.al-islam.com/Index/indexe1.asp`. Praktische Suche nach Suren- und Versen-Nummern.

About Islam and Muslims

`http://www.unn.ac.uk/societies/islamic/`
Seite der studentischen islamischen Arbeitsgemeinschaft der Universität Northumbria.
Koran als gif-Bilder und englische Übersetzung unter
`http://www.unn.ac.uk/societies/islamic/quran/naeindex.htm`.

Supreme Council for Islamic Affairs

http://www.alazhr.com/
Verschiedene Resourcen über den Islam.
Übersetzungen des Korans in verschiedene Sprachen, arabische Version (gif, cp-1256 nicht vokalisiert, Aussprache) unter http://www.alazhr.com/Quran/.

Quraan.com

http://www.quraan.com/
Sammlung von Artikeln, Texten usw. über den Koran.
Verschiedene Übersetzungen des Korans, arabische Version unter http://www.quraan.com/Arabic/ (als Text, mit Aussprache).

Sure Guidance

http://mukhtar.homedns.org/
Kommentare über den Koran.
Arabischer Text (UTF-8, Windows oder Macintosh Kodierung nach Wahl) mit Kommentaren, englischer Übersetzung und Transkription unter
http://mukhtar.homedns.org/SureGuidance/ Yusufali.do?action=tableOfContents.

OneUmmah.Net

http://www.oneummah.net/
Seite vom One Ummah Netzwerk.
Koran als arabischer Text und englische Übersetzung unter
http://www.oneummah.net/quran/quran.html.

IslamiCity.com

http://islamicity.com/
Gateway für die Islamische Gemeinschaft.
Koran als gif-Bilder, Aussprache, Transkription und verschiedenen Übersetzungen unter
http://islamicity.com/mosque/arabicscript/Ayat/1/1_1.htm.
Arabischer Text (gif) mit Aussprache unter
http://islamicity.com/mosque/ArabicScript/sindex.htm.

IslamArabe

http://www.islamarabe.com/
Seite für die französische islamische Gemeinschaft.

Koran als jpg-Bilder, französische Übersetzung und Transkription unter `http://www.islamarabe.com/ia/souratesdossier.htm`.

The Internet Sacred Text Archive

`http://www.sacred-texts.com/`
Sammlung von religiösen Texten verschiedener Religionen.
Islamische Ressourcen unter `http://www.sacred-texts.com/isl/`.
Transkription des Korans unter
`http://www.sacred-texts.com/isl/quran/`.

Islamic Server of MSA-USC

`http://www.usc.edu/dept/MSA/`
Transkription des Korans (dieselbe wie bei Sacred Texts) unter
`http://www.usc.edu/dept/MSA/quran/transliteration/`.

B.3 Bibel

International Bibel Society

`http://www.gospelcom.net/ibs/`
Arabische Version (PDF oder MS Word Format) unter
`http://www.gospelcom.net/ibs/bibles/arabic/`.

Arabic Bible Outreach Ministry

`http://www.arabicbible.com/`
Arabische Bibel in verschiedenen Formaten verfügbar (MS Word, gif, HTML, Audio, PDF).

The Good Way

`http://www.the-good-way.com/`
Sammlung von Texten und Artikeln über die Bibel.
Arabische Bibel in PDF Format unter
`http://www.the-good-way.com/arab/pdff/abible/`.

CopticChruch.Net

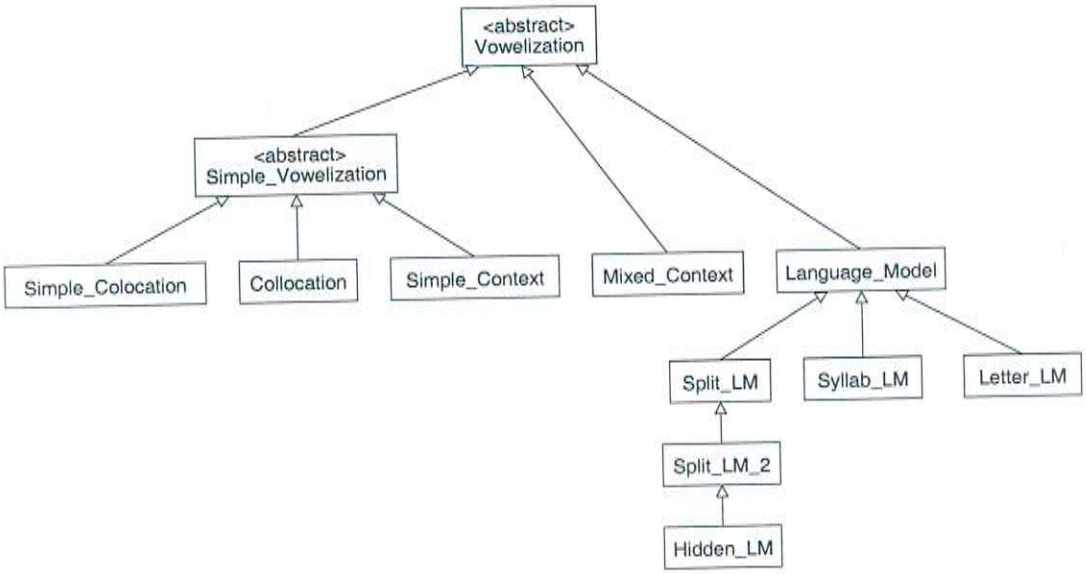
`http://www.copticchurch.net/`
Seite vom *Coptic Orthodox Chruch Network*.

Online Version der Bibel (Version vom Arabic Bible Outreach Ministry) unter `http://www.copticchurch.net/cgibin/bible/`, mit Suchmöglichkeiten.

Anhang C

Klassenhierarchie des Systems

Das folgende Diagramm stellt die Klassenhierarchie der Implementierung der verschiedenen Verfahren dar (die Implementierung erfolgte in C++).



Die einzelnen Klassen sind im Source-Code dokumentiert.
Die Abstrakte Klasse `Vowelization` zum Beispiel definiert eine Schnittstelle für alle Vokalisierungsverfahren:

```
virtual void write_data(ostream &os) const;
     Writes the data needed by the algorithm to an output stream.

virtual void read_data(istream &is);
     Reads the data needed by the algorithm from an input stream.

virtual void train(ifstream& infile_v, ifstream& infile_nv) = 0;
```

```
     Trains the algorithm.

void compute_stats();
     Computes statistics about the training corpus.

virtual void vocalize(ifstream& infile, ofstream& outfile,
                      bool keep_unk = true) = 0;
     Vocalizes an input file giving an output file.
     If keep_unk is true words that can not be vocalized (unknown
     words for word-based algorithm and known words for letter-based
     algorithm) are left as found in the input file.
     Else they are replaced with <unk>.

void compare(ifstream& infile1, ifstream& infile2, ifstream& infile3,
             bool categ, bool details);
     Compares the vocalized file infile1 with the reference vocalized
     file infile2, given the reference unvocalized file infile3.
     If categ is true results are given for each category of words.
     If details is true a list of false words is given.
```

Anhang D

Programmdokumentation

Die folgenden Seiten erläutern einige entwickelte Programme und Skripte.
Das Programm `combined1` liefert die besten Ergebnisse. Es verwendet ein Verfahren mit einem Sprachmodell auf Wortebene für bekannte Wörter und auf Buchstabenebene für unbekannte Wörter.
Für die Ausführung der Programme werden Perl und das SRILM Toolkit benötigt. Hilfe zur Installation kann in der beigefügten README-Datei gefunden werden.

Vorgang

Das folgende Beispiel erläutert den Vorgang bei der Vokalisierung.
Folgende Dateien müssen verfügbar sein:

`training_nv.txt`: Trainingsdaten ohne diakritische Zeichen.

`training_v.txt`: Trainingsdaten mit diakritischen Zeichen.

`test_nv.txt`: Testdaten ohne diakritische Zeichen.

`test_v.txt`: Testdaten mit diakritischen Zeichen (als Referenz um die Fehlerrate zu berechnen).

Die Namen der Dateien sind beliebig. Falls die Dateien sich nicht da befinden, wo das Programm ausgeführt wird, muß der vollständige Pfad angegeben werden.
Die Dateien müssen ASCII-Dateien, in der Buckwalter-Transliteration, sein.
Im zu vokalisierende Text `test_nv.txt` können Verdopplungszeichen (aus der Morphologie, nicht aus der Assimilation, siehe 2.3.2) vorhanden sein, in diesem Fall müssen sie auch in den Trainingsdaten `training_nv.txt` vorhanden sein.
Dann kann die Vokalisierung wie folgt ausgeführt werden:

```
combined1 training_nv.txt training_v.txt test_nv.txt test.txt test_v.txt
```

Die Datei `test.txt` enthält die gefundene Vokalisierung für den Text, der in `test_nv.txt` enthalten ist.
Die Datei `combined1.log` enthält Informationen über die Vokalisierung (Statistiken, Fehlerrate).
Das Verhalten des Programms kann mittels verschiedener Optionen verändert werden, insbesondere um die Namen der erzeugten Dateien oder die Parameter der Verfahren zu bestimmen. Die Standartwerte für die Parameter liefern die besten Ergebnisse.
Die verschiedenen Schritte des Vorgangs können getrennt werden. Die folgenden Abschnitte erläutern diese Schritte und die Dateien, die bei jedem Schritt erzeugt werden.

Training

Das Training kann wie folgt einzeln ausgeführt werden:

```
combined1 -v -c training_nv.txt training_v.txt
```

Es werden folgende Dateien erzeugt:

`combined1.log`: Enthält Statistiken über den Korpus.

`lm.dat`: Enhält die Daten, die beim Training für das Verfahren auf Wortebene berechnet wurden.

`syllab_lm.dat`: Enhält die Daten, die beim Training für das Verfahren auf Buchstabenebene berechnet wurden.

`lm.lm.gz`: Enthält das Sprachmodell auf Wortebene.

`syllab_lm.lm.gz`: Enthält das Sprachmodell auf Buchstabenebene.

`lm.map`: *Mapping*-Datei auf Wortebene.

`syllab_lm.map`: *Mapping*-Datei auf Buchstabenebene.

Vokalisierung

Die Vokalisierung kann wie folgt einzeln ausgeführt werden:

```
combined1 -c test_nv.txt test.txt
```

Die Daten, die für die Vokalisierung benötigt werden, werden aus den Dateien `lm.dat`, `syllab_lm.dat`, `lm.lm.gz`, `syllab_lm.lm.gz`, `lm.map`, `syllab_lm.map`, die beim Training erzeugt wurden, gelesen.
In `combined1.log` wird die Anzahl der unbekannten Wörter geschrieben.

Vergleich

Der Vergleich zwischen dem vom System vokalisierten Text und dem Referenztext mit der richtigen Vokalisierung kann wie folgt einzeln ausgeführt werden:

```
combined1 -t -v test_nv.txt test.txt test_v.txt
```

In `combined1.log` wird die Fehlerrate und ggf. die Liste der falschen Wörter geschrieben.
Wenn die `-t` Option weggelassen wird, werden die Dateien, die beim Training erzeugt wurden, gelesen, um die Fehlerrate nach den Kategorien der Wörter zu geben.

Vokalisierungsskripte

Die diakritische Zeichen, die nicht gelernt werden, können mittels einigen Vokalisierungsskripten (siehe entsprechende Seite im Manual) gefunden werden.
Wenn diese Zeichen sich im Referenztext befinden, muss dieser Schritt zwischen dem Vokalisierungsschritt und dem Vergleich wie folgt gemacht werden:

```
restore_shadda test.txt
restore_o_alif test.txt
restore_o_hamza test.txt
restore_o_moon test.txt
```

Arabische Schrift

Um den vokalisierten Text in arabischer Schrift zu bekommen, kann das `buck-2-utf8` Skript, das einen Text in der Buckwalter Transliteration in einen Text in der UTF-8 Kodierung für Arabisch umwandelt, verwendet werden. Das Skript kann wie folgt aufgerufen werden:

```
buck-2-utf8 test.txt
```

Manual Pages

Programs

version1
: vowelization program using the baseline algorithm

version2
: vowelization of unknown words using simple letter-based context

version3
: vowelization of known words using a language model

version4
: vowelization of unknown words using letter-based mixed contexts

version5
: vowelization of known words using a language model with splitted word endings

version6
: vowelization of known words using a language model with a second pass with hidden-ngram for endings

version7
: vowelization of known words using a language model with a second pass with disambig for endings

version8
: vowelization of unknown words using a syllab-based language model (limited to words)

version9
: vowelization of known words using a simple collocation model

version10
: vowelization of known words using collocations

version11
: vowelization of unknown words using a letter-based language model (sentence-wide)

combined1
: vowelization program using a word-based language model for known words and a syllab-based language model for unknown words

combined2
: vowelization program combining a word-based language model with splitted endings for known words and a letter-based language model for unknown words

combined3
: vowelization program using a word-based language model for known words and a letter-based language model for unknown words

Utility Scripts

Preprocessing scripts
: preprocessing tasks to build the corpus.

Statistics scripts
: statistics about the vowelization of the corpus.

Program scripts
: utility scripts used by vowelization programs.

Vowelization scripts
: scripts using simple rules to find some of the vowels.

Miscellaneous scripts
: miscellaneous utilities.

version3

NAME

version3 - vowelization of known words using a language model

SYNOPSIS

version3 *general_options files specific_options*

DESCRIPTION

version3 generates in a first step from the training files a language model, with the **ngram-count** program, and a mapping between vowel-less words and diacriticized words.

In the vowelization step, a HMM is used to find the vowelization of the sentences with the highest probability, with the **disambig** program.

The comparison step compares the vowelized file with the reference file and gives the error rate.

GENERAL OPTIONS

-h|--help
Print option summary.
-z|--details
Give a list of the words with false vowels when comparing the vocalized file with the reference vocalized file.
-s|--stats
Turn off training set statistics computing.
-l|--log *log_file*
Specifies the name of the file where the results will be printed. The default name is *lm.log*.
-d|--data *data_file*
Specifies the name of the file containing the data needed by the program. When training files are given, data are written to this file. When no training files are given, data are read from this file. The default name is *lm.dat*.
-t|--training
Turns off training step.
-v|--vocalize
Turns off vowelization step.
-c|--compare
Turns off comparison step.
-g|--categ
Turns off categorization of the results.

FILES

non_voc
Training file without vowels.
voc
Training file with vowels.
in
Input file (without vowels).
out
Output file (with vowels).
ref
Reference file (with vowels).

ALGORITHM SPECIFIC OPTIONS

-k|--keep
Keep temporary files.

-f|--freq *freq*
Sets minimum frequency for words included in vocabulary of the language model. Words with a frequency below this threshold will be considered as unknown words by the language model. If the threshold is higher as 1, discounting parameters should be determined from the full vocabulary. For this the discounting options given by the **-p** option should specify files with the **-gt*n*** or **-kn*n*** options of **ngram-count**. The default value is 1 (all words are included in the vocabulary).

-c|--closed
Use closed-vocabulary language model. The default is to build an open-vocabulary language model, i.e. the unknown-word token is considered as a regular word.

-l|--lm *lm file*
Specifies the name of the language model file. The default name is *lm.lm.gz* (compressed file).

-m|--map *map file*
Specifies the name of the mapping file. The default name is *lm.map*. It can be a compressed file (.gz).

-d|--debug *level*
Sets the debugging level for the **disambig** program. Possible values are:
0: no debug
1: debug zero probabilities
2: debug transitions
3: debug treillis
4: debug context length
The default value is 0.

-o|--order *order*
Sets the order of the language model. When no training files are given, this order is only used for the vowelization step. It must be lower than the order of the previously computed language model read from the file. The default order is 3.

-p|--discount *file*
Specifies a file containing discounting options for **ngram-count**.
The default options are: *-kndiscount1 -kndiscount2 -interpolate2 -kndiscount3 -interpolate3 -gt3min 1*

AUTHOR

Amélie Deltour <adeltour@netcourrier.com>

version8

NAME

version8 - vowelization of unknown words using a syllab-based language model (limited to words)

SYNOPSIS

version8 *general_options files specific_options*

DESCRIPTION

version8 generates in a first step from the training files a language model with syllabs as vocabulary (the sentences are limited to word boundaries), with the **ngram-count** program, and a mapping between vowel-less consonants and diacriticized syllabs.

In the vowelization step, a HMM is used to find the vowelization of the words with the highest probability, with the **disambig** program.

The comparison step compares the vowelized file with the reference file and gives the error rate.

GENERAL OPTIONS

-h|--help
Print option summary.
-z|--details
Give a list of the words with false vowels when comparing the vocalized file with the reference vocalized file.
-s|--stats
Turn off training set statistics computing.
-l|--log *log_file*
Specifies the name of the file where the results will be printed. The default name is *syllab_lm.log*.
-d|--data *data_file*
Specifies the name of the file containing the data needed by the program. When training files are given, data are written to this file. When no training files are given, data are read from this file. The default name is *syllab_lm.dat*.
-t|--training
Turns off training step.
-v|--vocalize
Turns off vowelization step.
-c|--compare
Turns off comparison step.
-g|--categ
Turns off categorization of the results.

FILES

non_voc
Training file without vowels.
voc
Training file with vowels.
in
Input file (without vowels).
out
Output file (with vowels).
ref
Reference file (with vowels).

ALGORITHM SPECIFIC OPTIONS

-k|--keep
Keep temporary files.

-c|--closed
Use closed-vocabulary language model. The default is to build an open-vocabulary language model, i.e. the unknown-word token is considered as a regular word.

-l|--lm *lm file*
Specifies the name of the language model file. The default name is *syllab_lm.lm.gz* (compressed file).

-m|--map *map file*
Specifies the name of the mapping file. The default name is *syllab_lm.map*. It can be a compressed file (.gz).

-d|--debug *level*
Sets the debugging level for the **disambig** program. Possible values are:
0: no debug
1: debug zero probabilities
2: debug transitions
3: debug treillis
4: debug context length
The default value is 0.

-o|--order *order*
Sets the order of the language model. When no training files are given, this order is only used for the vowelization step. It must be lower than the order of the previously computed language model read from the file. The default order is 5.

-p|--discount *file*
Specifies a file containing discounting options for **ngram-count**.
The default options are: *-kndiscount1 -kndiscount2 -interpolate2 -kndiscount3 -interpolate3 -gt3min 1 -kndiscount4 -interpolate4 -gt4min 1 -kndiscount5 -interpolate5 -gt5min 1*

AUTHOR

Amélie Deltour <adeltour@netcourrier.com>

combined1

NAME

combined1 - vowelization program using a word-based language model for known words and a syllab-based language model for unknown words

SYNOPSIS

combined1 *general_options files specific_options*

DESCRIPTION

combined1 trains a word-based language model as in **version3** and a syllab-based language model as in **version8**.

In the vowelization step, the vowelization of known words given by the word-based LM and the vowelization of unknown words given by the syllab-based LM are combined to give a fully vowelized text.

The comparison step compares the vowelized file with the reference file and gives the error rate.

GENERAL OPTIONS

-h|--help
Print option summary.
-z|--details
Give a list of the words with false vowels when comparing the vocalized file with the reference vocalized file.
-s|--stats
Turn off training set statistics computing.
-l|--log *log_file*
Specifies the name of the file where the results will be printed. The default name is *combined1.log*.
-d|--data *data_file*
Specifies the name of the file containing the data needed by the word-based method. When training files are given, data are written to this file. When no training files are given, data are read from this file. The default name is *lm.dat*.
-D|--Data *data_file*
Specifies the name of the file containing the data needed by the letter-based method. When training files are given, data are written to this file. When no training files are given, data are read from this file. The default name is *syllab_lm.dat*.
-t|--training
Turns off training step.
-v|--vocalize
Turns off vowelization step.
-c|--compare
Turns off comparison step.
-g|--categ
Turns off categorization of the results.
-k|--keep
Keep temporary files used for merging.

FILES

non_voc
Training file without vowels.
voc
Training file with vowels.
in

Input file (without vowels).

out

Output file (with vowels).

ref

Reference file (with vowels).

ALGORITHM SPECIFIC OPTIONS

-k|--keep

Keep temporary files of the word-based method.

-K|--Keep

Keep temporary files of the syllab-based method.

-f|--freq *freq*

Sets minimum frequency for words included in vocabulary of the word-based language model. Words with a frequency below this threshold will be considered as unknown words by the language model. If the threshold is higher as 1, discounting parameters should be determined from the full vocabulary. For this the discounting options given by the **-p** option should specify files with the **-gt*n*** or **-kn*n*** options of **ngram-count**. The default value is 1 (all words are included in the vocabulary).

-c|--closed

Use closed-vocabulary for the word-based language model. The default is to build an open-vocabulary language model, i.e. the unknown-word token is considered as a regular word.

-l|--lm *lm file*

Specifies the name of the word-based language model file. The default name is *lm.lm.gz* (compressed file).

-L|--Lm *lm file*

Specifies the name of the syllab-based language model file. The default name is *syllab_lm.lm.gz* (compressed file).

-m|--map *map file*

Specifies the name of the word-based mapping file. The default name is *lm.map*. It can be a compressed file (.gz).

-M|--Map *map file*

Specifies the name of the syllab-based mapping file. The default name is *syllab_lm.map*. It can be a compressed file (.gz).

-d|--debug *level*

Sets the debugging level for the **disambig** program. The first digit gives the level for the word-based method, the second digit gives the level for the syllab-based method. Possible values are:
0: no debug
1: debug zero probabilities
2: debug transitions
3: debug treillis
4: debug context length
The default value is 00.

-o|--order *order*

Sets the order of the language models. The first digit gives the order for the word-base LM and the second digit gives the order for the syllab-based LM. When no training files are given, this order is only used for the vowelization step. It must be lower than the order of the previously computed language model read from the file. The default order is 35.

-p|--discount *file*

Specifies a file containing discounting options for **ngram-count**. The first line is for the word-based LM, the second line is for the syllab-based LM.
The default options for the word-based LM are: *-kndiscount1 -kndiscount2 -interpolate2 -kndiscount3 -interpolate3 -gt3min 1*
The default options for the syllab-based LM are: *-kndiscount1 -kndiscount2 -interpolate2 -kndiscount3 -interpolate3 -gt3min 1 -kndiscount4 -interpolate4 -gt4min 1 -kndiscount5 -interpolate5 -gt5min 1*

AUTHOR

Amélie Deltour <adeltour@netcourrier.com>

Vowelization scripts

NAME

restore_shadda, restore_o_moon, restore_o_hamza, restore_o_alif

SYNOPSIS

restore_shadda *files...*
restore_o_moon *files...*
restore_o_hamza *files...*
restore_o_alif *files...*

DESCRIPTION

These scripts find some of the vowels with simple rules. They take as input texts that were already vowelized by the programs.

restore_shadda finds shaddas (doubling signs) by the assimilation of the article.

restore_o_moon finds the vowel (sukun) between the article and a word beginning with a moon-letter.

restore_o_hamza finds the vowel (sukun) between the article and a word beginning with a hamza.

restore_o_alif finds the vowel (kasra) between the article and a word beginning with alif.

AUTHOR

Amélie Deltour <adeltour@netcourrier.com>

Miscellaneous scripts

NAME

utf8-2-buck, buck-2-utf8, cp1256-2-utf8, bidi, sets

SYNOPSIS

utf8-2-buck *files...*
buck-2-utf8 *files...*
cp1256-2-utf8 *files...*
bidi *files...*
sets *infile training verif test*

DESCRIPTION

utf8-2-buck performs a transliteration from a UTF-8 arabic file to an ASCII file using the Buckwalter transliteration.

buck-2-utf8 performs the reverse transliteration (from ASCII to UTF-8).

cp1256-2-utf8 converts an arabic file with cp1256 encoding to a file with UTF-8 encoding.

bidi inserts Unicode tags at the beginning and end of each line, to be recognized by the BiDi (bi-directional) algorithm so as to represent correctly texts mixing arabic and roman letters.

sets extracts lines (in an homogenous manner) from a corpus file so as to give a training set with 90% of the data, a validation set with 5% of the data and a test set with 5% of the data.

AUTHOR

Amélie Deltour <adeltour@netcourrier.com>

Literaturverzeichnis

[Beesley, 1996] Beesley, K. R. (1996). Arabic finite-state morphological analysis and generation. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96). Copenhagen, August 5-6, 1996*, volume 1, pages 89–94.

[Beesley, 1998] Beesley, K. R. (1998). Arabic morphological analysis on the internet. In *Proceedings of the 6th International Conference and Exhibition on Multi-lingual Computing. Cambridge, 17-18 April, 1998.*

[Brockelmann, 1965] Brockelmann, C. (1965). *Arabische Grammatik: Paradigmen, Literatur, Übungsstücke und Glossar.* VEB Verlag Enzykopädie. Besorgt von Manfred Fleischhammer.

[Byrne et al., 2001] Byrne, W., Hajic, J., Ircing, P., Jelinek, F., Khudanpur, S., Krbec, P., and Psutka, J. (2001). On large vocabulary continuous speech recognition of highly inflectional language - Czech. Eurospeech 2001.

[Chen and Goodman, 1998] Chen, S. F. and Goodman, J. T. (1998). An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Computer Science Group, Harvard University.

[Darwish, 2002] Darwish, K. (2002). Building a shallow arabic morphological analyzer in one day. ACL-2002 Workshop on 'Computational Approaches to Semitic Languages'. University of Pennsylvania, Philadelphia, PA, July 2002.

[Debili and Achour, 1998] Debili, F. and Achour, H. (1998). Voyellation automatique de l'arabe. In *Proceedings of the COLING-ACL'98 Workshop on 'Computational Approaches to Semitic Languages'. Université de Montreal, 16th August 1998.*

[Deheuvels, 2000] Deheuvels, L.-W. (2000). *Manuel d'arabe moderne*, volume 1. L'Asiathèque, 6 edition.

[Déjean, 1998] Déjean, H. (1998). Morphemes as necessary concept for structures discovery from untagged corpora. In Powers, D., editor, *Proceedings of CoNLL-98 (Workshop on Paradigms and Grounding in Language Learning)*, pages 295–298. ACL.

[Gal, 2002] Gal, Y. (2002). A HMM approach to vowel restoration in Arabic and Hebrew. ACL-2002 Workshop on 'Computational Approaches to Semitic Languages'. University of Pennsylvania, Philadelphia, PA, July 2002.

[Gharieb, 1996] Gharieb, G. M. (1996). *Arabisch für Volkshochschulen.* Dr. Gharieb M. Gharieb Verlag.

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

[Itai and Segal, 2000] Itai, A. and Segal, E. (2000). A corpus based morphological analyzer for hebrew undotted texts. Department of Computer Science, The Technion, Haifa, Israel.

[Karboul, 1999] Karboul, O. (1999). Die hocharabische Sprache und Romanisierung ihrer Schrift. Studienarbeit, Insitut für Logik, Komplexität und Deduktionssysteme, Fakultät für Informatik der Universität Karlsruhe.

[Katz, 1987] Katz, S. M. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP35(3):400–401.

[Kiraz, 1995] Kiraz, G. A. (1995). Computational analyses of arabic morphology. University of Cambridge.

[Koskenniemi, 1984] Koskenniemi, K. (1984). A general computational model for word-form recognition and production. In *Proceedings of COLING-84, 2-4 July 1984, Stanford University, California*, pages 178–181.

[Lagally, 1999] Lagally, K. (1999). ArabTeX, a system for typesetting arabic. User manual version 3.09. Technical Report 1998/09, Universität Stuttgart, Fakultät Informatik.

[Levinger et al., 1995] Levinger, M., Ornan, U., and Itai, A. (1995). Learning morpho-lexical probabilities from an untagged corpus with an application to hebrew. *Computational Linguistics*, 21(3):383–404.

[Mihalcea, 2002] Mihalcea, R. (2002). Diacritics restoration: Learning from letters versus learning from words. In *Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics (CICLING 2002). Mexico City, Mexico, February 2002*, pages 339–348.

[Ramsay and Mansur, 2001] Ramsay, A. and Mansur, H. (2001). Arabic morphology: a categorial approach. ACL-2001 Workshop on 'Arabic Language Processing: Status and Prospects'. Toulouse, 2001.

[Schone and Jurafsky, 2000] Schone, P. and Jurafsky, D. (2000). Knowledge-free induction of morphology using latent semantic analysis. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 67–72. Lisbon, Portugal, September 2000.

[Simard, 1996] Simard, M. (1996). Réaccentuation automatique de textes français. Technical report, Centre d'Innovation en Technologies de l'Information (CITI), Laval, Canada.

[Simard, 1998] Simard, M. (1998). Automatic insertion of accents in french text. In *Proceedings of the Third Conference on Epirical Methods in Natural Language Processing (EMNLP-3). Grenade, Spain, June 1998.*

[Spriet and El-Bèze, 1997] Spriet, T. and El-Bèze, M. (1997). Réaccentuation automatique de textes. FRACTAL 97. Besançon, France, décembre 1997.

[Stolcke, 2002] Stolcke, A. (2002). SRILM - An extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing. Denver, Colorado, September 2002.*

[The Unicode Consortium et al., 2000] The Unicode Consortium, Aliprand, J., Allen, J., McGowan, R., Becker, J., Everson, M., Ksar, M., Moore, L., Suignard, M., Whistler, K., Davis, M., Freytag, A., and Jenkins, J. (2000). *The Unicode Standard, Version 3.0.* Addison-Wesley.

[Theron and Cloete, 1997] Theron, P. and Cloete, I. (1997). Automatic acquisition of two-level morphological rules. In Grishman, R., editor, *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP). Washington, DC, 31 March - 3 April 1997*, pages 103–110. ACL.

[Yarowsky, 1994] Yarowsky, D. (1994). Decision lists for lexical ambiguity resolution: Application to accent restoration in spanish and french. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics. Las Cruces, NM*, pages 88–95.

[Yarowsky, 1999] Yarowsky, D. (1999). A comparison of corpus-based techniques for restoring accents in spanish and french text. In *Natural Language Processing Using Very Large Corpora*, pages 99–120. Kluwer Academic Publishers.

[Zweigenbaum and Grabar, 2002] Zweigenbaum, P. and Grabar, N. (2002). Accenting unknown words in a specialized language. ACL-2002 Workshop on 'Natural Language Processing in the Biomedical Domain' (BioNLP). University of Pennsylvania, Philadelphia, PA, July 2002.