Retrievalmethoden für historische Korpora mit nicht standardisierten Schreibweisen

Von der Fakultät für Ingenieurwissenschaften,
Abteilung Informatik und Angewandte Kognitionswissenschaft
der Universität Duisburg-Essen
zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
genehmigte Dissertation

von

Dipl.-Inform. Andrea Ernst-Gerlach

aus Wuppertal

1. Gutachter: Prof. Dr.-Ing. Norbert Fuhr

2. Gutachter: Prof. Dr. Gregory Ralph Crane

Tag der mündlichen Prüfung: 06. Juni 2013

Inhaltsverzeichnis

Danksagung

Ich danke Prof. Dr. Norbert Fuhr und Prof. Dr. Gregory Crane für ihre Unterstützung während der Dissertation. Sie standen mir mit ihrem Fachwissen stets zur Seite und haben durch ihre Anregungen und kritischen Kommentare zum guten Gelingen dieser Arbeit beigetragen.

Meinen Kollegen der Arbeitsgruppe Informationssysteme sowie den beteiligten studentischen Mitarbeitern und Abschlussarbeitern danke ich für den gedanklichen Austausch und die gute Zusammenarbeit. Insbesondere sind Henrik Nottelmann, Claus-Peter Klas und Dennis Korbar zu nennen.

Bei Rita Vock und Lars Hallmann möchte ich mich für Rat und Tat in der Endphase der Dissertation bedanken.

Mein besonderer Dank gilt meiner Familie. Ohne ihre Liebe und Geduld hätte ich diese Arbeit nicht fertigstellen können.

Abstract

Die Anzahl von digitalen Bibliotheken, die auch historische Volltexte enthalten, steigt immer weiter. Damit einhergehend wächst auch die Anzahl an digital verfügbaren historischen Dokumenten. Trotzdem gestaltet sich die Suche nach diesen Dokumenten immer noch schwierig. Aufgrund fehlender Standardisierung der Rechtschreibung ist es vielfach nicht möglich, mit Suchbegriffen in heutiger Sprache historische Texte zu finden. Diese Thematik ist vor allem bei Sprachen relevant, deren Rechtschreibung erst spät standardisiert wurde, wie z. B. Deutsch und Englisch.

In dieser Arbeit wird ein neuer Ansatz für Retrieval in Texten mit nicht standardisierter Rechtschreibung entwickelt. Es wird ein Algorithmus beschrieben, der den Benutzer bei der Suche in digitalen Bibliotheken unterstützt. Basierend auf Belegpaaren aus aktueller und historischer Schreibung generiert der Algorithmus probabilistische Regeln. Mit diesen werden Varianten eines Suchbegriffes in historischer Schreibung generiert.

Dargestellt wird die Gesamtarchitektur der Suchmaschine einschließlich der Evaluierung. Ausgehend von einem Suchbegriff in Grundform wird ein aktuelles deutsches Wörterbuch benutzt, um die zugehörigen Vollformen zu finden. Auf die gefundenen Vollformen werden die generierten Transformationsregeln angewendet, um die historischen Wortformen zu bilden. Die Experimente zeigen, dass sich die Retrievalqualität von historischen Kollektionen durch den vorgestellten Ansatz stark verbessert. Somit kann er den Benutzer in seiner täglichen Arbeit deutlich entlasten. Eine sehr große Anzahl historischer Dokumente, die bisher trotz ihrer Digitalisierung nicht sinnvoll durchsucht werden konnten, werden nun verschiedensten Benutzergruppen — vom Laien bis zum Historiker — besser zugänglich.

Mit Hilfe des im Anschluss entwickelten Verfahrens zur automatischen Erstellung der Belege ist es zusätzlich möglich, den Engpass bei der Regelerstellung aufzulösen. Das Verfahren wurde in den entwickelten RuleGenerator integriert. Dieser stellt eine Benutzeroberfläche zur Verfügung, die dem Anwender die Generierung und Bearbeitung von Belegen und Regeln ermöglicht.

Teil I.

Einleitung

1. Einleitung

In den vergangenen Jahren gab es viele Initiativen, die Bücher digitalisiert und sie im Internet verfügbar gemacht haben. Diese Initiativen folgten früheren kleineren Projekten wie z. B. dem Gutenberg Projekt[1], das bereits 1971 mit dem Aufbau einer digitalen Bibliothek begonnen hat [Stewart u. a., 2007]. Die US-Suchmaschine Google hat mittlerweile 15 Millionen Bücher digitalisiert und die EU-Mitgliedsstaaten immerhin 1,2 Millionen [Lischka, 2011]. Dadurch ist die Menge der zur Verfügung stehenden Texte exponentiell angestiegen.

Eine wachsende Zahl an digitalen Archiven soll die langfristige Erhaltung des kulturellen Erbes sichern und zukünftigen Generationen den Zugang zu den Dokumenten ermöglichen [Hedstrom und Ross, 2003; Rauber u. a., 2002]. Die Erstellung von Archiven wird von Chavez-Demoulin und Kollegen mit den Auswirkungen der Bibelübersetzungen aus dem Lateinischen verglichen [Chavez-Demoulin u. a., 2000].

Unter den digitalisierten Büchern sind immer mehr Kollektionen, die historische Dokumente enthalten. Zu Anfang wurden die Dokumente nur gescannt. Heutzutage werden die Texte mit Software zur Texterkennung bearbeitet und indexiert, um eine Volltextsuche bereitzustellen. Zusätzlich soll auf diese Weise einem breiten Nutzerkreis der Zugriff auf die Dokumente gewährt werden [Adriaans, 2005; Lischka, 2011].

1.1. Standardisierung der Rechtschreibung

Durch die große Anzahl von Dialekten und zeitlichen Variationen ist die Rechtschreibung in historischen Dokumenten sehr zeit- und ortsabhängig. Nach Braun werden als historische Dokumente alle Texte verstanden, die in einer historischen Form der modernen Sprache verfasst sind [Braun, 2002]. In einigen Ländern gab es zwar Institutionen, die schon früh die Standards für die Rechtschreibung festgelegt haben. Aber auch für Sprachen, deren Schreibung geregelt war, existieren frühe Schreibvarianten. Dies gilt z. B. für das Spanische, das bereits 1713 standardisiert wurde [Zaslavsky u. a., 2001]. Im Gegensatz dazu war die Rechtschreibung in England und Deutschland [Pilz, 2003] weitaus länger nicht festgelegt. Im Englischen wurde die Schreibweise allerdings bereits um 1800

[1] `http://www.gutenberg.org/` (letzter Aufruf 18.12.12)

Allgemeiner Theil.

Erſtes Capitel.

§ 11.

I. Von der Grenze der Civilproceßſachen.*)

Alle Juſtiz-Sachen ſind lediglich Gegenſtände des gerichtlichen Verfahrens; in Anſehung der Landesgerichte ſteht die ausſchließliche Competenz grundgeſetzlich feſt[1]). Als dahin gehörig werden die Sachen bezeichnet, welche contentiosae jurisdictionis ſind und worin die Parteien wegen ihres Privat-Intereſſe mit einander zu ſtreiten haben[2]), oder, wie es im § 394 des L. G. G. E. V. heißt, worin es auf einen recht- und richterlichen Ausſpruch ankommt[3]). Alle ſolche Wendungen können die eigentlichen Streitfragen hinſichtlich des Umfanges der Juſtiz-Sachen nicht abſchließen, und die ergangenen beſonderen Beſtimmungen gewähren zwar für die betreffenden Angelegenheiten genügenden Halt[4]), erſchöpfen aber den Gegenſtand nicht.

Abbildung 1.1.: Faksimile eines historischen Dokumentes [Trotsche, 1866]

herum standardisiert [Kranz, 1998, S. 13], während dies in Deutschland erst 1901/02 erfolgte. Der überwiegende Teil der 6000 Sprachen, die heutzutage gesprochen werden, wurde niemals offizielle Sprache und somit auch nicht verbindlich geregelt [Strunk, 2003].

Vor der Standardisierung galt die Regel "Schreibe wie Du sprichst", das sogenannte phonologische Prinzip [Modes, 2012, S. 20f.]. Allerdings können sich selbst bei gleicher Aussprache noch unterschiedliche Schreibungen ergeben [Pilz, 2003]. In Abbildung 1.1 ist ein Faksimile eines historischen Dokumentes dargestellt. Bereits in der Überschrift ist mit der historischen Schreibung *Theil* für die aktuelle Form *Teil* eine erste Schreibvariante enthalten. Zu Schreibvarianten kommt es nicht nur bei unterschiedlichen Autoren. Häufig sind auch verschiedene Variationen eines Wortes innerhalb eines Textes zu finden [O'Rourke u. a., 1997]. Teilweise wurde die Schreibweise auch angepasst, wenn sich dadurch die optische Gestaltung verbesserte. Da in den Texten oftmals Zitate aus älteren Dokumenten verwendet werden, verschärft sich das Problem weiter [Pilz u. a., 2005]. Selbst bei lateinischen Texten existieren Schreibvarianten. Diese kommen durch unterschiedliche Autoren und Editoren sowie durch den zeitlichen Wandel zustande [Jussen u. a., 2007].

Die Rechtschreibung wird trotz vorhandener Standardisierung weiterhin einem Wandel unterliegen. So gab es z. B. in den neunziger Jahren sowohl für das Niederländische [Reynaert, 2005, S. 89f.] als auch für das Deutsche [Agbaria, 2009, S. 23ff.] Rechtschreibreformen. Deswegen wird sich das Problem der Schreibvarianten zu einem späteren Zeitpunkt auch für Dokumente stellen, die den heutigen Regeln der Rechtschreibung entsprechen.

1.2. Suche in historischen Dokumenten

Die nicht standardisierte Schreibung führt zu Problemen bei der Suche in historischen Teilen von digitalen Bibliotheken. Der Großteil der Benutzer gibt die Suchbegriffe in der aktuellen Schreibweise ein, die sich von der Schreibweise in den historischen Dokumenten unterscheidet. Dabei ist der Benutzer sehr häufig auch an Texten interessiert, die Schreibvarianten enthalten. Die einfachste Möglichkeit zur Lösung des Problems stellt die Anfrageerweiterung dar. Obwohl es in den meisten Fällen auch für Laien problemlos möglich ist, die historischen Texte zu verstehen, fehlt einem Großteil der Benutzer das nötige Expertenwissen, um die Anfrageerweiterung vorzunehmen [Koolen, 2005]. Aufgrund der Vielzahl möglicher Schreibweisen kennen selbst erfahrene Spezialisten nicht alle Varianten [O'Rourke u. a., 1997; Biella u. a., 2005] und können deswegen die notwendige Erweiterung der Suchanfrage nicht vornehmen. Weil die meisten Suchmaschinen ihren Dokumentindex wortweise aufbauen, müssen die Suchbegriffe mit den indexierten Wörtern für eine erfolgreiche Suche übereinstimmen [Selbach, 2008]. Auch populäre Digitalisierungsinitiativen wie die Google-Buchsuche[2] oder die europäische digitale Bibliothek[3] unterstützen bisher keine Suche nach Schreibvarianten.

Mit steigender Anzahl an verfügbaren Dokumenten benötigt der Benutzer aber zugleich auch mehr Unterstützung bei der Suche. So werden in der quantitativ arbeitenden historischen Linguistik eine möglichst große Zahl von Hilfsmitteln wie historische Wörterbücher für die Arbeit mit den Korpora benötigt [Mehler u. a., 2009]. Neben der Volltextsuche stellen die Schreibvarianten auch in anderen Bereichen wie z. B. der Korpuslinguistik ein Problem dar. Die Korpuslinguistik befasst sich hauptsächlich mit der Erforschung und Darstellung sprachlicher Besonderheiten in Korpora [Stewart u. a., 2007; Lemnitzer und Zinsmeister, 2006, S. 5ff.]. Zu diesem Zweck werden sehr spezifische Korpora zusammengestellt [Lüdeling und Walter, 2010]. Dafür werden häufig extra historische Dokumente digitalisiert. Bei der anschließenden Erstellung von Häufigkeitsprofilen und Konkordanzen im Rahmen von quantitativen Analysen ergeben sich allerdings Probleme, weil Schreibvarianten als unterschiedliche Wörter gewertet werden [Baron und Rayson, 2008; Hockey, 2004].

[2] `http://books.google.com/` (letzter Aufruf 18.12.12)
[3] `http://www.europeana.eu/portal/` (letzter Aufruf 18.12.12)

Durch Schreibvarianten ergeben sich auch bei Kookkurrenzen falsche Werte. Ebenso entstehen bei Diskursanalysen auf Dokumenten mit Schreibvarianten Probleme, da die Werkzeuge zur semantischen Analyse nur für moderne Texte geeignet sind. Deswegen wurden von Archer und Kollegen Methoden zur Modernisierung von Dokumenten entwickelt, damit die vorhandenen Werkzeuge auch für historische Texte einsetzbar sind [Archer u. a., 2003].

Bei der Digitalisierung von Texten ist sowohl der Prozess der Digitalisierung selbst als auch deren Ergebnis Gegenstand wissenschaftlichen Arbeitens [TextGrid, 2004]. Damit umfasst der Kreis der Benutzer von digitalisierten Texten nicht nur Linguisten, sondern auch weitere Wissenschaftler aus den Humanwissenschaften, wie z. B. Historiker. Diese sind beispielsweise daran interessiert, zu einem Text alle Belegstellen zu finden [Mehler u. a., 2011]. Dabei sind sowohl lexikalische als auch syntaktische und textuelle Belege wichtig für die weitere Arbeit. Allerdings wird der Prozess der Vernetzung durch Schreibvarianten in den Texten ebenfalls deutlich erschwert.

Bei herkömmlichen Suchmaschinen ist eine falsche Schreibweise der häufigste Grund für eine leere Ergebnismenge [Hearst, 1999]. Einige Systeme bieten deswegen für Suchbegriffe mit geringer Häufigkeit Alternativen an. Diese werden oft nur auf der Basis von Wörterbüchern oder Ähnlichkeitsmaßen erzeugt. Erstere haben den Nachteil, dass sie nur bei Wörtern hilfreich sind, die auch im Wörterbuch enthalten sind. Dagegen haben Ähnlichkeitsmaße nur eine geringe Präzision, insbesondere für kurze Wörter. Außerdem können Ähnlichkeitsmaße nur eingesetzt werden, wenn die zu durchsuchenden Dokumente bereits vorliegen. Bei einer Suche im Internet kann deswegen eine Verwendung nur erfolgen, wenn Zugriff auf den entsprechenden Index besteht.

1.3. Ziel der Arbeit

Eine Suchmaschine soll nach Heller neben Verfahren aus der Mathematik und Informatik auch linguistische Eigenschaften berücksichtigen [Heller, 2004]. Dies gilt insbesondere bei der Suche nach historischen Dokumenten. Ziel der vorliegenden Arbeit ist es zu zeigen, dass ein regelbasierter Ansatz eine Suche nach Schreibvarianten ermöglicht. Dabei soll mit Anfragen in moderner Sprache eine Suche in Texten mit abweichender Schreibweise realisiert und so das Problem der Schreibvarianten reduziert werden.

Ein regelbasiertes Verfahren hat dabei die folgenden Vorteile [Ferber, 2003, S. 132]:

- Regeln sind auch für Laien verständlich.
- Jede Regel kann einzeln angewendet werden. Somit lässt sich durch jede zusätzliche Regel der Recall erhöhen.
- Regeln sind leicht zu generalisieren und zu spezialisieren.

Bei einer manuellen Erstellung von Regeln kann selbst ein Experte nur wenige Regeln pro Tag erstellen, während ein automatisches System eine größere Anzahl von Regeln in deutlich kürzerer Zeit generieren kann [Quinlan, 1986]. Deshalb soll die Erstellung der Regeln automatisch erfolgen.

Retrievalanwendungen sind immer durch eine Unsicherheit bei der Erstellung der Anfrage und einer Vagheit bezüglich des Informationsbedürfnisses gekennzeichnet [Fuhr, 2005]. Aufgrund der Vagheit des Informationsbedürfnisses ist das Ergebnis des Retrievalprozesses nur mit einer bestimmten Wahrscheinlichkeit korrekt. Demzufolge sind nicht alle generierten Varianten tatsächlich historische Schreibungen. Allerdings muss trotzdem die Retrievalqualität gewährleistet bleiben. Diese kann mit den Retrievalmaßen Recall und Precision bestimmt werden. Im Gegensatz zu Reynaert erfolgt dabei keine Korrektur der Texte [Reynaert, 2004b]. Das Ziel ist es, alle relevanten Dokumente zu finden. Somit liegt das Hauptaugenmerk auf dem Recall und nicht wie bei der automatischen Korrektur auf der Precision.

Wissenschaftler arbeiten häufig mit kleinen Korpora, die sie mit großem Aufwand zusammenstellen, den Text auszeichnen und mit Metadaten versehen [Audenaert u. a., 2006]. Die Suchmaschine soll die Recherche auf eigenen Datenbeständen ermöglichen, um anschließend eine genaue Analyse der lokal verfügbaren Korpora zu unterstützen. Es existiert eine Vielzahl von Initiativen zur Digitalisierung und Bestandserhaltung, die von einer Suchmaschine für historische Dokumente profitieren könnten. Die Grundlage der Suchmaschine soll die Generierung von Transformationsregeln bilden, die die Schreibweise für ein Wort in historischer Sprache generieren. Viele historische Bücher sind auch für Laien interessant [Hauser u. a., 2007]. Damit auch diese Benutzergruppe mit der Suchmaschine arbeiten kann, sollen sowohl bei der Generierung der Regeln als auch bei der Verwendung der Suchmaschine von den Benutzern keine besonderen Kenntnisse in den Bereichen Linguistik und Informatik benötigt werden. In der vorliegenden Arbeit soll eine Methode für die Suche nach Schreibvarianten entwickelt werden. Das Problem der Vokabellücke, bei dem keine moderne Form für eine historische Schreibung existiert, wird hier nicht weiter betrachtet.

Es wurden zwei Szenarien von Benutzertypen entworfen, um die verschiedenen Benutzerinteressen aufzuzeigen. Diese sollen in erster Linie die Spannweite des notwendigen Bedarfs an Unterstützung bei der Erstellung von Belegen und Regeln darstellen. Für einen Linguisten könnte beispielsweise bereits die Bildung der Belege ein interessantes Forschungsthema sein. Demzufolge möchte er die Belege nur mit semi-automatischer Unterstützung generieren, weil er auch an der Entwicklung der Sprache sowie an Regeln mit einer sehr hohen Precision interessiert ist. Er sucht oft nach allen Vorkommen eines Wortes in einer Kollektion und kann deswegen nur mit einem kompletten Regelsatz arbeiten. Im Gegensatz dazu möchte ein Historiker lediglich relevante Dokumente zu einem bestimmten Thema finden. Deswegen möchte er möglichst schnell eine Volltextsuche nutzen. Dementsprechend wird er

einen automatischen Ansatz bevorzugen. Auch wenn er deswegen zunächst einige Dokumente nicht findet, reicht es ihm aus, solange er die Möglichkeit hat, den Regelsatz später zu überarbeiten.

Das zu entwickelnde Werkzeug soll die Regelerstellung soweit wie möglich automatisieren. Zugleich soll der Benutzer aber die Möglichkeit bekommen, die erstellten Regeln weiter zu bearbeiten bzw. auch eigene zu erstellen. Ausgehend von seinen Bedürfnissen wird sich der Benutzer mehr auf den Recall oder die Precision seiner Suche konzentrieren. Das Werkzeug soll an dieser Stelle die notwendige Flexibilität bieten.

Die Deutsche Forschungsgemeinschaft (DFG) hat das Projekt *Regelbasierte Suche in Textdatenbanken mit nichtstandardisierter Rechtschreibung (RSNSR)* gefördert. Die vorliegende Dissertation ist im Rahmen dieses Projektes entstanden.

1.4. Aufbau der Arbeit

Im folgenden Kapitel wird zunächst der Begriff der Schreibvarianten definiert. Daran anschließend werden räumliche sowie zeitliche Auswirkungen auf die Schreibung erläutert. Anhand des Noisy Channel Model wird der Ablauf einer Suche nach Schreibvarianten vorgestellt. Anschließend werden die verwandten Themenbereiche dargestellt. Das dritte Kapitel erläutert die vorhandenen Ansätze zur Behandlung von Schreibvarianten. Hierbei erfolgt eine Unterscheidung zwischen wörterbuchbasierten Ansätzen, Ähnlichkeitsmaßen und regelbasierten Verfahren.

Eine Suchmaschine für historische Dokumente wird im dritten Teil entwickelt. Kapitel 4 stellt das Verfahren zur Regelgenerierung vor. Dazu folgt zunächst die Beschreibung von grundlegenden Lernverfahren. Anschließend wird die generelle Vorgehensweise erarbeitet. Darauf aufbauend wird die Generierung von Transformationsregeln entwickelt und evaluiert. Das anschließende Kapitel stellt zunächst die Architektur der entwickelten historischen Suchmaschine vor. Darauf folgt eine nähere Erläuterung der einzelnen Module. Die anschließende Evaluierung demonstriert die Einsatzfähigkeit des implementierten Systems. Die Entwicklung einer historischen Google-Buchsuche verdeutlicht die Übertragbarkeit des Ansatzes.

Teil IV erläutert die benutzergesteuerte Regelerstellung. Hierfür werden im sechsten Kapitel vorhandene Methoden zur Erstellung von Belegen beschrieben. Im Anschluss wird ein Verfahren zur automatischen Belegerstellung entwickelt. Nach erfolgter Evaluierung des Ansatzes wird ein Werkzeug zur benutzergesteuerten Regelentwicklung aufgebaut und die automatische Belegerstellung integriert. Anschließend folgt die Beschreibung und Auswertung der durchgeführten Benutzerevaluierung.

Der letzte Teil verdeutlicht zunächst in der Zusammenfassung in Kapitel 8 den Stellenwert der verbesserten Retrievalmöglichkeiten bei der Suche nach

historischen Dokumenten. Abschließend skizziert Kapitel 9 die weiteren Anwendungsmöglichkeiten. Dabei werden sowohl Erweiterungsmöglichkeiten des entwickelten Ansatzes als auch weitere Einsatzbereiche vorgestellt.

Teil II.

Schreibvarianten in Dokumenten

2. Schreibvarianten

Dieses Kapitel konkretisiert zunächst den Begriff der Schreibvariante. Dabei wird auch auf die regionalen und zeitlichen Veränderungen der Schreibung eingegangen. Anhand des Noisy Channel Models wird verdeutlicht, wie das zu entwickelnde System arbeiten muss, um die Veränderung der Schreibweisen bei der Suche zu überwinden. Im Anschluss wird ein Überblick über die verwandten Themenbereiche gegeben und eine Abgrenzung zu der behandelten Problematik vorgenommen.

2.1. Definition

Bei Schreibvarianten handelt es sich um real vorkommende Wörter aus Texten, deren Schreibung von der aktuell gültigen Rechtschreibung für die jeweilige Sprache abweicht. Der Fokus dieser Arbeit liegt auf den historischen Varianten, die in Texten vorkommen, welche vor einer Standardisierung der Rechtschreibung entstanden sind. Allerdings existieren auch bei vorhandener Standardisierung oftmals mehrere Schreibvarianten. Dies gilt insbesondere nach der Rechtschreibreform aus den neunziger Jahren. So wird im Duden bei den Hinweisen zur Wörterbuchbenutzung explizit auf die Behandlung von Schreibvarianten eingegangen. Dies gilt z. B. für *Geograf* und *Geograph* [Wermke u. a., 2009, S. 10]. Unter Schreibvarianten können auch dialektbedingte Varianten verstanden werden. Als weitere Form von Schreibvarianten können Tippfehler angesehen werden.

Braun hat beim Retrieval in historischen Korpora die beiden Problembereiche der Schreibvarianten und der Vokabellücke identifiziert [Braun, 2002]. Für Schreibvarianten gilt demnach eine ähnliche Definition wie bei Synonymen:

- Die Bedeutung ist dieselbe.
- Sie haben eine unterschiedliche, aber erlaubte Schreibweise.
- Sie sind in derselben grammatikalischen Form. Beispielsweise sind beide im Nominativ Plural.

Schreibvarianten sind Varianten eines Wortes (z. B. *Kaiser* - *Kayser*), Synonyme (z. B. *Apfelsine* - *Orange*) hingegen nicht. Zu jeder Schreibvariante existiert ein Wort in aktueller Schreibung. Dabei unterscheiden sich die beiden Wörter lediglich in der Schreibweise. Im Gegensatz dazu sind Synonyme Wortpaare, bei denen mit verschiedenen Wörtern dasselbe ausgedrückt wird.

Aktuelle Schreibweise	**Schreibweise 19. Jh.**	**Regeln**	
wiedergaben	widergaben	*wieder* → *wider* *ie* → *i*	(1)
akzeptieren	acceptieren	*kz* → *cc* k → *c* ∧ *z* → c	(2)
überall	ueberall	*ü* → *ue*	(3)
seht	sehet	*t* → *et*	(4)

Tabelle 2.1.: Beispielregeln für Deutsch

Das Problem der Schreibvarianten bei Braun entspricht der in der vorliegenden Arbeit beschriebenen Fragestellung. Unter einer Vokabellücke wird hingegen die Problematik verstanden, dass für ein historisches Wort kein entsprechendes aktuelles Wort in moderner Schreibung existiert. Entweder wird es in der heutigen Sprache nicht mehr benutzt, oder die Bedeutung des aktuellen Wortes hat sich im Vergleich zum historischen Wort deutlich verändert [Koolen, 2005]. Dies gilt z. B. für *marcken*, das eine historische Form für *handeln* darstellt [Hauser u. a., 2007]. Diese Aufgabe ließe sich z. B. mit der Hilfe eines Thesaurus bearbeiten. Dafür sind Methoden erforderlich, mit denen sich Paare aus modernen und historischen Wörtern mit derselben Bedeutung automatisch bilden lassen. Koolen geht davon aus, dass sich das Problem der Vokabellücke deutlich leichter automatisch lösen lässt, wenn zuvor die Thematik der Schreibvarianten behandelt wurde [Koolen, 2005]. Eine Möglichkeit dazu wären parallele Korpora. Beispielsweise könnten verschiedene Bibelübersetzungen genutzt werden, um mit Hilfe von Kookkurrenzen oder von Fußnoten mit Worterklärungen einen Thesaurus zu erstellen.

In unterschiedlichen Editionen eines Textes kommen wegen der früher üblichen manuellen Vervielfältigung durch Abschreiben der Dokumente auch verschiedene Schreibvarianten vor [Stewart u. a., 2007]. Die Erfindung des Buchdrucks und die darauf folgende weite Verbreitung der Luther-Bibel haben wesentlich zu einer Vereinheitlichung der Schreibung beigetragen [Heller, 2006]. Die Entwicklung einer standardisierten Schriftsprache war zudem mit einer neuen Sichtweise verknüpft [Braun, 2002]. In den Niederlanden verringerte sich z. B. im 16. Jahrhundert der Einfluss der Aussprache auf die Schreibweise. Dadurch ließen auch die regionalen Einflüsse nach, die sich durch unterschiedliche Dialekte ergeben hatten. Auf diese Weise wurde die Schreibweise einheitlicher. Diese Entwicklung wurde durch die zunehmenden Kontakte zwischen Städten und Regionen weiter unterstützt.

Tabelle 2.1 zeigt Beispiele für manuell generierte Regeln für Belege aus dem 19. Jahrhundert. Das erste Beispiel zeigt zwei Regeln verschiedener Spezialisierungsgrade. Die Regel *wieder* → *wider* transformiert einen Präfix, während *i* → *ie* nur einen Allograph transformiert. Das nächste Beispiel bietet ebenfalls zwei Möglichkeiten. In diesem Fall kann die Transformation aus einer Regel (*kz* → *cc*) oder aus der Konkatenation von zwei Regeln (*k* → *c* und *z* → *c*)

Aktuelle Schreibweise	Schreibweise 16. Jh.	Regeln	
always	alwaies	$y \rightarrow ie$	(1)
sudden	suddain	$e \rightarrow ai$	(2)
publicly	publikely	$c \rightarrow ke$	(3)

Tabelle 2.2.: Beispielregeln für Englisch

bestehen. Die Precision der ersten Regel wird offensichtlich deutlich höher sein als die Kombination von zwei Regeln. Der dritte Fall zeigt mit $\ddot{u} \rightarrow ue$ eine sehr häufige Regel für Umlaute. Das letzte Beispiel enthält mit $t \rightarrow et$ eine sehr generelle Regel, die allerdings eine deutlich höhere Precision erreichen kann, wenn sie mit Kontextinformationen verknüpft wird. Obwohl sich diese Arbeit schwerpunktmäßig mit der deutschen Sprache beschäftigt, lassen sich auch Beispiele für die englische Sprache finden [Rayson u. a., 2005], bei denen ein regelbasiertes Verfahren anwendbar ist (s. Tabelle 2.2).

Beispiele gibt es nicht nur für den europäischen Sprachraum, sondern auch für das asiatische Sprachgebiet. So unterscheiden sich z. B. modernes und traditionelles Mongolisch neben den Schreibvarianten auch noch durch die Schriftart [Khaltarkhuu und Maeda, 2008]. Diese wurde 1946 von traditionellem Mongolisch auf Kyrillisch umgestellt. Mongolische Buchstaben haben in Abhängigkeit von der Position im Wort (Isoliert, Initial, Mitte und Ende) zwischen drei und zwölf unterschiedliche Schreibungen. Nicht alle Buchstaben der historischen Schreibung sind im Unicode definiert.

2.2. Räumliche und zeitliche Unterschiede

Bei Schreibvarianten treten sowohl räumliche als auch zeitliche Unterschiede auf. So ist z. B. der prozentuale Anteil der Varianten in alten Texten deutlich höher als in neueren Dokumenten. Bei einer Untersuchung von Pilz mit Texten aus der Testkollektion (s. Abschnitt 4.3.4) sinkt er von 70 % im 13. Jahrhundert auf einen Anteil von unter 5 % gegen Ende des 19. Jahrhunderts (s. Abbildung 2.1) [Pilz, 2009, S. 67ff.]. Die beiden Regressionsgeraden für X und Y in der Abbildung minimieren dabei den quadratischen Fehler bezüglich der Zeit bzw. der Varianten. Im Laufe der Zeit wurde zudem eine Vielzahl von Schreibungen verwendet. Einige Beispiele für *geheiligt* und *Himmel* sind in Abbildung 2.2 dargestellt. Mit zunehmendem Alter der Texte steigt sowohl die Anzahl der Buchstabenersetzungen als auch die Anzahl der Stellen, an denen diese Ersetzungen auftreten [Pilz u. a., 2008b]. Darüber hinaus lassen sich auch sprachliche Varianten je nach Klasse oder Status unterscheiden. Downie und Kollegen haben dies exemplarisch an Textbeispielen aus der Gutenberg-Kollektion gezeigt [Downie u. a., 2006].

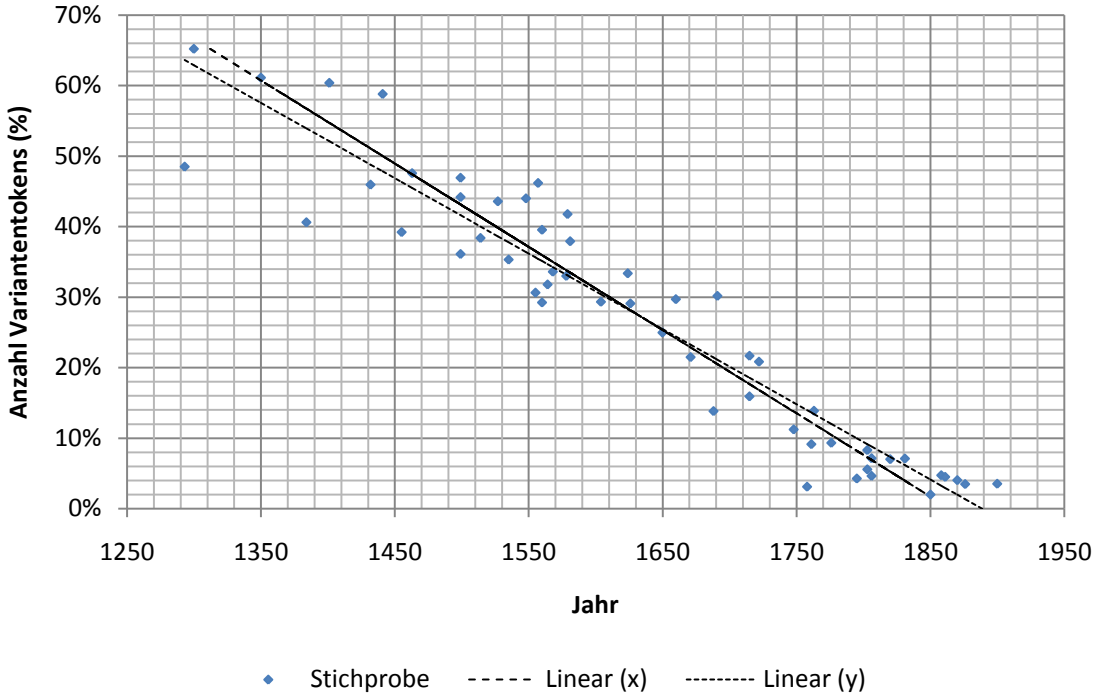


Abbildung 2.1.: Zeitliche Verteilung der Varianten [Pilz, 2009, S. 69]

Die genaue räumliche Einteilung der Varianten ist häufig schwierig. Zunächst einmal müsste geklärt werden, ob der Ort (an dem der Text entstanden ist) oder der sprachliche Hintergrund des Autors eine größere Rolle spielt. Bei geklärter Herkunft der Schreibvariante bleibt das Problem der genauen räumlichen Zuordnung, da sich sowohl Ländergrenzen als auch Städtegrenzen durch politische Entwicklung im Laufe der Zeit häufig ändern [Mostern, 2006]. Außerdem sind Städtenamen ohne weiteren Namenszusatz nicht immer eindeutig, wie z. B. bei Neustadt, Mülheim und Hausen. Bei der räumlichen Zuordnung der Schreibvarianten handelt es sich um ein Problem des geographischen Information Retrieval. Dementsprechend lässt es sich nur mit Hilfe eines geographischen Informationssystems in Kombination mit einem Retrievalsystem lösen. Geoinformationssysteme stellen Daten mit räumlichem Bezug in Karten dar und ermöglichen eine entsprechende Analyse. Es gibt viele Methoden und Werkzeuge, mit denen die räumliche Veränderung der Länder und Regionen untersucht werden kann [Jessop, 2006]. So können z. B. mehrere Karten übereinandergelegt werden, um Informationen aus den Karten miteinander zu kombinieren [Koster, 2006]. Allerdings sind historische Karten nicht sehr präzise; obwohl die Kartographen bereits sehr exakt gearbeitet haben, sind die Daten nicht alle mit der gleichen Genauigkeit erhoben worden. So ist die Lage von wichtigen Gebäuden, wie z. B. Kirchen, sehr genau abgebildet worden. Die Bereiche

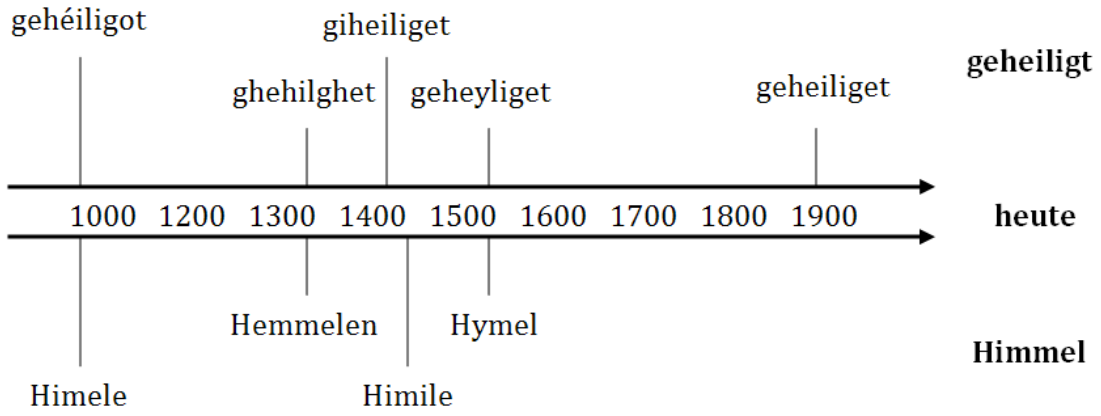


Abbildung 2.2.: Varianten für *geheiligt* und *Himmel* [Pilz, 2009, S. 147]

dazwischen sind jedoch ungenau dargestellt. Die notwendige Georeferenzierung der Karten wird dadurch deutlich erschwert.

Die räumliche Zuordnung hat keinen Einfluss auf die generelle Behandlung des Problems der Schreibvarianten. Laut Pilz ist zudem das Problem der zeitlichen Variation größer als das der räumlichen Variation [Pilz, 2009, S. 83]. Deswegen wird die räumliche Variation im Folgenden nicht explizit behandelt. Um dies bei einer entsprechenden Suchmaschine mit einzubeziehen, wäre lediglich eine zusätzliche Abbildung notwendig, die den angegebenen Namen für die Suchregion um die historischen Namen ergänzt. Beispielsweise wurde die Stadt Wuppertal 1929 u. a. aus den Städten Barmen und Elberfeld gegründet. Eine entsprechende Suchanfrage müsste also letztendlich alle drei Städtenamen berücksichtigen.

2.3. Noisy Channel Model

Shannon beschreibt mit dem Noisy Channel Model (s. Abbildung 2.3) den Ablauf der Kommunikation einer Nachricht [Shannon, 2001]. Dabei wird die Mitteilung, die eine Person übermitteln möchte, durch Nebengeräusche verfälscht und stattdessen eine veränderte Nachricht ausgegeben. Die von einer Informationsquelle eingegebene Nachricht wird vom Sender in ein Signal umgewandelt und an den Empfänger übertragen. Der Empfänger dekodiert die Nachricht und leitet sie an den Adressaten weiter. Während der Datenübertragung wird das Signal durch die Störquelle verändert.

Das Noisy Channel Model kommt u. a. bei der maschinellen Übersetzung, in der Spracherkennung und der Rechtschreibkorrektur zum Einsatz. Bei der Anwendung auf die Rechtschreibkorrektur weiß der Benutzer, welche Wörter er schreiben möchte. Allerdings kommt bei der Texteingabe ein Rauschen

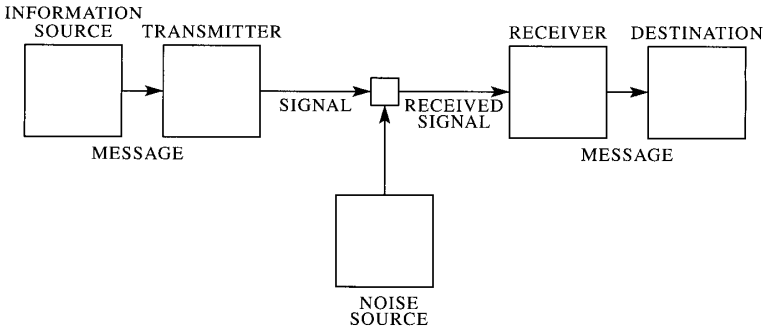


Abbildung 2.3.: Schema der Kommunikationsübertragung [Shannon, 2001]

in Form von Rechtschreib- und Tippfehlern hinzu [Kernighan u. a., 1990]. Das Korrekturprogramm gibt für eine Liste von Fehlern die zugehörigen Korrekturvorschläge mit den entsprechenden Wahrscheinlichkeiten aus. Da die Rechtschreibung vor der Standardisierung dem phonologischen Prinzip (s. Abschnitt 1.1) folgte, können vor allem Variationen der Aussprache durch räumliche und zeitliche Spracheinflüsse als Nebengeräusche gemäß dem Noisy Channel Model betrachtet werden.

Im Folgenden soll ein Modell entwickelt werden, das es erlaubt, von der modernen Schreibweise auf die potentiellen Schreibvarianten zu schließen und somit die Störquelle auszuschalten. Gemäß dem Korrektur-System von Shannon in Abbildung 2.4 müsste dazu ein Beobachter sowohl die eingegebenen Suchbegriffe als auch die zugehörigen Varianten kennen und die Suchbegriffe über einen Korrekturkanal anpassen. Für die Korrektur werden Paare aus moderner und historischer Schreibung benötigt. Aus diesen Belegen sollen Regeln abgeleitet werden, damit der Korrekturkanal auch unbekannte moderne Formen korrigieren kann. Sprachliche Regeln können sowohl in großer Anzahl als auch sehr spezifisch vorkommen. Daher würden wichtige Regeln nicht erstellt, wenn der Benutzer sie nur auf Basis seiner Intuition verfasst [Mangu und Brill, 1997]. Deswegen muss ein System entwickelt werden, das die Regeln zur Simulation des Rauschens automatisch erzeugen kann.

2.4. Verwandte Themen

Zu dem Problem der Schreibvarianten gibt es mit der Rechtschreibkorrektur, der Texterkennung, der Namensidentifikation, dem Audioretrieval und dem mehrsprachigen Retrieval einige verwandte Themen. Diese werden im Folgenden näher erläutert.

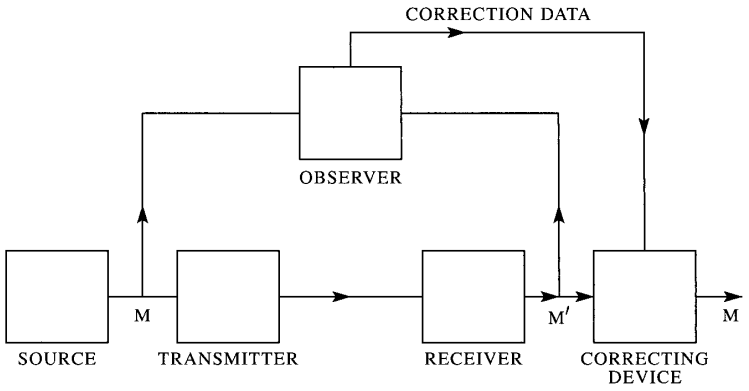


Abbildung 2.4.: Schema eines Korrektursystems für die Kommunikationsübertragung [Shannon, 2001]

2.4.1. Rechtschreibkorrektur

Bei der Rechtschreibkorrektur soll zu einem falsch geschriebenen Wort die korrekte Schreibung gefunden werden. Damit handelt es sich um eine 1:1-Relation. Somit reicht es aus, wenn ein Programm zur Rechtschreibkorrektur das ähnlichste Wort aus dem Wörterbuch zurückliefert. Letztendlich wird aber eine Liste mit den m besten Ergebnissen zurückgegeben, aus der der Benutzer passende Vorschläge auswählen kann. Bei der Suche nach Schreibvarianten steht die Größe der Ergebnismenge nicht vorab fest, weil zu einem Suchbegriff mehrere Schreibvarianten existieren können. Die Suche in Texten mit Schreibvarianten stellt somit angenähert eine 1:n-Relation dar.

Das Problem der Rechtschreibkorrektur ist stark verwandt mit der Suche nach Schreibvarianten. Ein Unterschied besteht zwar in der theoretisch eindeutigen Lösung bei der Rechtschreibkorrektur. In der Praxis werden allerdings ebenfalls Ergebnislisten zurückgegeben, die z. T. auch gewichtet sind. Somit spiegelt sich dieser Unterschied bei der Nutzung nur sehr eingeschränkt wider.

2.4.2. Namensidentifikation

Bei der Namensidentifikation müssen Namen, insbesondere solche mit fehlerhafter Schreibung, in einer Namensliste gefunden werden [Pfeifer u. a., 1996]. Dabei können die Teilprobleme phonetische Ähnlichkeit, Tippfehler und einfache Wortähnlichkeiten unterschieden werden. Zu Schwierigkeiten kommt es neben Tippfehlern besonders häufig bei der Übertragung von Namen aus anderen Alphabeten. In der Literatur gibt es z. B. für den russischen Namen *Tschebyschew* über 20 Varianten in lateinischer Schrift (u. a. *Tschebischeff* und *Chebychev*) [Fuhr, 1999; Ferber, 2003, S. 274]. Des Weiteren existieren unterschiedliche

Vorgehensweisen in Bezug auf die Abkürzung von Namen (z. B. *Anne Sophie Müller*, *Anne S. Müller* und *A. S. Müller*) [Fuhr, 1999]. Namen unterscheiden sich normalerweise nur in der Schreibweise und nicht in der Aussprache. Darin besteht der Hauptunterschied zum Problem der Schreibvarianten. Wörter in (historischen) Texten können sich im Gegensatz zu Namen durchaus in ihrer Aussprache von heutigen Wörtern unterscheiden. Dazu kommt es hauptsächlich durch örtliche Dialekte [Strunk, 2003]. Diese regionalen Unterschiede haben auch Effekte auf die Schreibung (s. Abschnitt 2.2). Pfeifer und Kollegen haben verschiedene Standardähnlichkeitsmaße verglichen und die Retrievalergebnisse durch Kombination der untersuchten Maße verbessert [Pfeifer u. a., 1996].

2.4.3. Audioretrieval

Beim Audioretrieval soll auf Transkriptionen von Audiodokumenten gesucht werden. Die Transkriptionen, auf denen letztendlich die Suche erfolgt, sollen möglichst die Originalaussprache wiedergeben. Diese unterscheidet sich im Normalfall deutlich von der standardisierten Schreibweise. Im Englischen werden z. B. häufig Wörter, die mit *-ing* enden, mit der Endung *-in* gesprochen und entsprechend transkribiert [Kendall und French, 2006].

2.4.4. Mehrsprachiges Retrieval

Beim mehrsprachigen Retrieval versucht ein Benutzer, Texte in einer anderen Sprache zu finden [Peters, 2001]. Verfügt er über genügend passives Wissen in der Sprache, benötigt er lediglich Hilfe bei der Formulierung der Anfrage, weil sein aktiver Wortschatz für die Erstellung zu gering ist. Benutzer ohne Kenntnis der Fremdsprache wollen oftmals zunächst die Relevanz der gefundenen Texte beurteilen, um abzuschätzen, ob eine Übersetzung der Texte sinnvoll ist [Ogden u. a., 1999].

Die Themenstellung dieser Arbeit kann auch als Variante des mehrsprachigen Retrieval aufgefasst werden. Beim mehrsprachigem Retrieval erfolgt die Suche allerdings zwischen Sprachen aus dem gleichen Zeitraum, während dies bei der Suche nach Varianten meistens nicht der Fall ist. Das Problem der Schreibvarianten kann auf dem Graphem-Level gelöst werden. Dagegen lässt sich mehrsprachiges Retrieval analog zu den Vokabellücken meistens nur mit Thesauri, Wörterbüchern, Kookkurrenzen und parallelen bzw. vergleichbaren Korpora durchführen. Dies erfordert entweder eine Übersetzung der Anfragen in die Sprache der Dokumente oder umgekehrt [Adriaans, 2005]. Bei stark verwandten Sprachen wie z. B. Norwegisch und Schwedisch können auch regelbasierte Ansätze eingesetzt werden [Järvelin u. a., 2006]. Dies gilt ebenso für technische Begriffe, die oftmals auf demselben lateinischen oder griechischen Ursprungswort basieren [Toivonen u. a., 2005]. Um in diesem Fall die Regeln zu bilden, werden von Pirkola und Kollegen aus

einem Lexikon äquivalente Paare aus zwei Sprachen extrahiert und daraus Regeln mit der Edit-Distanz gebildet [Pirkola u. a., 2003]. Nach der Regelanwendung werden in einem zweiten Schritt die ähnlichsten N-Gramme gesucht.

2.4.5. Texterkennung

Texterkennung (OCR[4]) wird benutzt, um automatisch Texte in Bildern zu erkennen. Sie ist immer noch fehlerbehaftet. Dies gilt insbesondere für historische Dokumente, bei denen die Papierqualität nicht mit der heutigen zu vergleichen ist [Hauser, 2007]. Außerdem kommt es durch den Einsatz von Scannern zu Neigungen der Zeilen zur Buchinnenseite und Schattenbildung. Mischke und Luther beschreiben ein Verfahren, das Neigungen automatisch erkennen und das Bild entsprechend entzerren kann [Mischke und Luther, 2005]. Historische Dokumente sind häufig in Frakturschrift geschrieben. Dies erschwert die automatische Texterkennung deutlich, weil OCR-Programme auf aktuelle Schriftarten ausgerichtet sind. Bei der Frakturschrift unterscheiden sich z. B. die Buchstaben $\mathfrak{B}$ und $\mathfrak{V}$ nur anhand eines Querstrichs [Höhn, 2008]. Außerdem benutzten verschiedene Setzer auch unterschiedliche Fraktursätze [Pilz, 2009, S. 42]. Weitere Probleme sehen Mihov und Kollegen bei Texten in Bildern und Tabellen sowie der Spalteneinteilung [Mihov u. a., 2005]. Geringer Kontrast und schlechte Druckqualität führen ebenfalls zu Fehlern bei der Texterkennung. Die Anzahl der Fehler steigt bei der Verwendung von kleinen Schriftgrößen und Kursivschrift laut Strohmaier und Kollegen signifikant an [Strohmaier u. a., 2003].

Die meisten Ansätze zur Behandlung von OCR-Fehlern benötigen entweder Wörterbücher (z. B. OCRSpell [Taghva und Stofsky, 2001]), die vor allem für historische Wörter nur selten zur Verfügung stehen, oder für einen Teil des Textes eine manuell erstellte Digitalisierung. Teilweise lässt sich die Software anhand von Beispielen trainieren [Holley, 2009]. Dies gilt z. B. für den ABBYY FineReader[5]. Allerdings muss der Benutzer dafür die erkannten Wörter bestätigen bzw. korrigieren. Weil die Qualität der Texterkennung stark von der Kollektion abhängt, ist der manuelle Trainingsaufwand, insbesondere für kleine Kollektionen, vergleichsweise hoch. Höhn hat einen Ansatz entwickelt, der ohne diese Ressourcen auskommt [Höhn, 2008]. Dazu werden die digitalisierten Wörter anhand ihrer Ähnlichkeit geclustert. Für das Clustering wird die Ähnlichkeit von Wörtern, die sich nur wenig unterscheiden, durch eine erweiterte gewichtete Levenshtein-Distanz bestimmt. Das Verfahren gibt als Ergebnis die Verwechslungswahrscheinlichkeiten für Buchstaben und Buchstabengruppen aus. Die höhere Wahrscheinlichkeit für die korrekte Erkennung einzelner Zeichen im Vergleich zu der Wahrscheinlichkeit für eine falsche Erkennung stellt die Grundannahme des Verfahrens dar. Es wird davon ausgegangen, dass das häufigste Wort innerhalb eines Clusters richtig erkannt wird.

[4] Optical Character Recognition
[5] `http://finereader.abbyy.de/` (letzter Aufruf 18.12.12)

Allerdings kommen Wörter nicht immer mehrfach in einem Text vor. Außerdem sind unterschiedliche Schreibvarianten auch innerhalb eines Textes möglich. Diese beiden Punkte berücksichtigt der Ansatz nicht.

Schreibvarianten weisen größtenteils eine phonetische Ähnlichkeit auf. Dagegen geht es bei der Korrektur von Fehlern in der Texterkennung eher um graphische Ähnlichkeiten zwischen Buchstaben. Das Problem ist im Gegensatz zu den Schreibvarianten vergleichsweise sprachunabhängig. Bei der Zusammenstellung der Trainingspaare ist es deswegen notwendig, zwischen historischen Schreibvarianten und Fehlern bei der Texterkennung zu unterscheiden [Stewart u. a., 2007].

OCR-Fehler lassen sich nach Pilz und Luther in drei Gruppen aufteilen [Pilz und Luther, 2009]:

- Falsche Erkennung von einzelnen Zeichen (z. B. *Barbaren - Rarbaren*)
- Auslassen bzw. Hinzufügen von Buchstaben (z. B. *kommen - kommenm*)
- Fehlerhafte Aufteilung von Zeichen (z. B. *reinblütigen - remblütigen*)

Liu und Kollegen haben ein weiteres Verfahren zur Korrektur von OCR-Fehlern vorgestellt [Liu u. a., 1991]. Dabei werden die Zeichenersetzungen in Abhängigkeit von der Umgebung (z. B. einer Texterkennungssoftware oder einer Sprache) gelernt. Dazu markiert zunächst eine Rechtschreibprüfung potentiell falsch geschriebene Wörter. Bei dieser Vorgehensweise bleiben allerdings Echtwortfehler unberücksichtigt. Für jedes unbekannte Wort wird eine Liste mit Korrekturvorschlägen und den zugehörigen Konfidenzwerten ausgegeben. Aus dieser Liste kann der Benutzer die richtigen Vorschläge auswählen. Falls der Konfidenzwert hoch genug ist, kann auch eine automatische Korrektur erfolgen. Mit den Zuordnungen durch den Benutzer können anhand der Häufigkeit, mit der sie falsch erkannt werden, oder einer Matrix aus Bigrammen sowie deren Häufigkeitsverteilung Korrekturvorschläge gemacht werden. Sind die korrigierten Wörter im Wörterbuch der Rechtschreibkorrektur enthalten und erreichen einen vom Benutzer festgelegten Konfidenzwert, werden die falsch erkannten Wörter korrigiert. Findet der Benutzer bei der manuellen Bearbeitung in der Liste ein korrektes Wort, das der Rechtschreibkorrektur unbekannt ist, kann er es einem Wörterbuch hinzufügen. Auf diese Weise wird das entsprechende Wort beim nächsten Mal nicht als Fehler eingeordnet.

2.5. Zusammenfassung

In diesem Kapitel wurde das Problem der Schreibvarianten dargestellt, der Begriff definiert und es wurden Bezüge zu verwandten Themenbereichen erläutert. Nach Braun lassen sich herkömmliche Retrievalmethoden nicht ohne Weiteres auf Retrieval in historischen Dokumenten übertragen [Braun, 2002]. Es existieren weitere Forschungsfelder, die verwandte Problemstellungen aufweisen, wie z. B.

mehrsprachiges Retrieval und die Namensidentifikation. Im folgenden Abschnitt werden die relevanten Verfahren zur Behandlung von Schreibvarianten vorgestellt.

3. Methoden zur Suche nach Schreibvarianten

In dem vorgestellten Forschungsbereich gibt es zur Zeit viele Aktivitäten. Dabei ist die Trennung zwischen Wörterbuch-, Ähnlichkeitsmaß- und Regel-basierten Verfahren nicht immer eindeutig, weil einige Ansätze auch mehrere Verfahren kombinieren. Die Entwicklung der vorgestellten Ansätze erfolgte für die Sprachen Englisch, Niederländisch und Deutsch, die alle zur westgermanischen Sprachfamilie gehören und somit auch Teil der indoeuropäischen Sprachen sind [Baker, 2007, S. 4]. Im Folgenden werden die wichtigsten Arbeiten auf diesem Gebiet vorgestellt.

3.1. Wörterbücher

Ein wörterbuchbasierter Ansatz schlägt Schreibvarianten in einem Wörterbuch nach. Die Suchfunktionen der elektronischen Versionen des Deutschen Rechtswörterbuchs (DRW)[6] und des Deutschen Wörterbuchs von Jacob und Wilhelm Grimm (DWB)[7] behandeln das Problem der Schreibvarianten beide auf diese Weise.

Pilz hat anhand von 100 zufällig ausgewählten Belegen aus der Testkollektion (s. Abschnitt 4.3.4) untersucht, wie viele Schreibvarianten über die Eingabe in der lexikoneigenen Suchmaske der beiden Lexika zu finden sind [Pilz, 2009, S. 100ff.]. Dabei wurden keine Anfragen nach in Wörterbüchern nicht üblichen Flexionsformen gestellt. Außerdem wurden Varianten bereits als gefunden gewertet, wenn zumindest ein Wort mit der charakteristischen Ersetzung erkannt wurde. Trotzdem waren beim DRW 66 % und beim DWB 69 % der gesuchten Schreibvarianten nicht zu finden. Weil jeweils zwei Drittel der Anfragen nicht erfolgreich sind, scheint ein allein wörterbuchbasierter Ansatz wenig erfolgversprechend. Für die meisten historischen Sprachen sind zudem keine elektronischen Wörterbücher verfügbar [Gotscharek u. a., 2009b]. Sie müssten deswegen für jeden einzelnen Dialekt bzw. jede Kollektion erstellt werden. Allerdings sind Wörterbücher nie vollständig. Dies gilt vor allem für stark flektierte Sprachen wie Deutsch. Durch die vielen Schreibvarianten wird das Problem für historische Wörterbücher noch deutlich größer [Reffle, 2011]. Um

[6] `http://drw-www.adw.uni-heidelberg.de/drw/` (letzter Aufruf 18.12.12)
[7] `http://germazope.uni-trier.de/Projekte/DWB` (letzter Aufruf 18.12.12)

effektiv beim Retrieval auf historischen Texten zum Einsatz zu kommen, müssen die Wörterbücher zudem zeitliche und örtliche Varianten enthalten.

3.2. Ähnlichkeitsmaße

Das generelle Ziel von Ähnlichkeitsmaßen ist es, zu einem gegebenen Term in einem Index den ähnlichsten Begriff zu finden. Falls kein Grenzwert für die Ähnlichkeit vorgegeben ist, wird ein Begriff mit sehr großer Distanz zurückgegeben, wenn kein anderes Wort eine größere Ähnlichkeit zu dem Suchbegriff aufweist. Trainierbare Maße können ihre Vorteile nach Kempken und Kollegen nur ausspielen, wenn ihre Entwicklung auf Basis von homogenen Trainingsdaten erfolgt [Kempken u. a., 2006].

3.2.1. Levenshtein-Distanz

Die Levenshtein-Distanz wurde zum Vergleich von Zeichenketten entwickelt [Levenshtein, 1966]. Sie berechnet die minimale Anzahl von Transformationen, um eine Zeichenkette auf eine andere abzubilden. Die Kosten für die Transformationen wie einfügen, löschen und ersetzen können einzeln festgelegt werden. Abbildung 3.1 zeigt, wie die Levenshtein-Distanz von *Himmel* und *Hymel* berechnet wird. Die Kosten sind dabei für einfügen und löschen auf 1 sowie für ersetzen auf 2 gesetzt. Das Ergebnis (in diesem Fall 3) ist jeweils unten rechts in der Matrix zu sehen.

Camps und Daudé haben auf Basis der Levenshtein-Distanz ein Ähnlichkeitsmaß vorgestellt, bei dem die Bestimmung der Gewichte anhand von Trainingsdaten erfolgt [Camps und Daudé, 2003]. Neben der Art der Ersetzung (einfügen, löschen, substituieren) werden dabei noch die Position der Ersetzung und die ersetzten Buchstaben bei der Berechnung der Kosten miteinbezogen.

In Abbildung 3.2 ist die Berechnung der Kosten am Beispiel des Wortpaares *Avery - Garvey* dargestellt. Für das Einfügen bzw. Löschen der Buchstaben A, G und R gelten die folgenden Kosten:

$$\begin{aligned} \delta_c(A,\varepsilon) &= \delta_c(\varepsilon,A) = 1,1 \\ \delta_c(G,\varepsilon) &= \delta_c(\varepsilon,G) = 1,3 \\ \delta_c(R,\varepsilon) &= \delta_c(\varepsilon,R) = 1,08 \end{aligned}$$

Alle anderen Kosten sind auf 1 gesetzt. Bei den Positionen wird zwischen Wortanfang und -ende sowie allgemeinen Positionen unterschieden. Vorteilhaft an dem Verfahren ist außerdem die Nutzung eines Grenzwertes, der von der Wortlänge abhängig ist (s. Tabelle 3.1). Wird dieser während der Berechnung des Wortabstandes überschritten, kann das Verfahren für diese Wörter abgebrochen werden.

		H	y	m	e	l
	0	1	2	3	4	5
H	1	0	1	2	3	4
i	2	1	2	3	4	5
m	3	2	3	2	3	4
m	4	3	4	3	4	5
e	5	4	5	4	3	4
l	6	5	6	5	4	3

Abbildung 3.1.: Levenshtein-Distanz für *Himmel* - *Hymel*

		G	a	r	v	e	y
	0	1,3	2,4	3,48	4,48	5,48	6,48
A	1,1	1	1,3	2,38	3,38	4,38	5,38
v	2,1	2	2	2,3	2,38	3,38	4,38
e	3,1	3	3	3	3,3	2,38	3,38
r	4,18	4,08	4	3	4	3,46	3,38
y	5,18	5,08	5	4	4	4,46	3,46

Abbildung 3.2.: Distanz für *Avery* - *Garvey* nach Camps und Daudé [Camps und Daudé, 2003]

3.2.2. N-Gramme

N-Gramme zerlegen Wörter in alle möglichen Folgen von n Buchstaben. Das Wort *Himmel* besteht z. B. aus den Bi-Grammen {**h, hi, im, mm, me, el, *l*} und den Trigrammen {***h, *hi, him, imm, mme, mel, el*, l***}. Am häufigsten kommen Bi- und Trigramme zum Einsatz. Die Ähnlichkeit zweier Zeichenketten a und b kann mit Hilfe des Dice-Koeffizienten (s. Gleichung 3.1) bestimmt werden.

$$d(a,b) = \frac{2|T(a) \cap T(b)|}{|T(a)| + |T(b)|} \tag{3.1}$$

Die Grundidee des Verfahrens basiert auf der Annahme, dass die Ähnlichkeit von zwei Zeichenketten umso größer ist, je mehr N-Gramme übereinstimmen.

Länge der Anfrage	≤ 3	4	5	6	7	8	9	10	11	12/13	≥ 14
Schwellenwert	1,9	2,1	2,16	2,6	2,8	3	3,1	3,5	4	4,5	5

Tabelle 3.1.: Schwellenwerte bei Camps und Daudé [Camps und Daudé, 2003]

Code	Zeichen
1	B, F, P, V
2	C, G, J, K, Q, S, X, Z
3	D, T
4	L
5	M, N
6	R

Tabelle 3.2.: Soundex-Codes

3.2.3. Phonetische Ähnlichkeitsmaße

Beispiele für phonetische Ähnlichkeitsmaße sind der Soundex- und der Phonix-Algorithmus, von denen ersterer hier kurz vorgestellt werden soll [Russel und Odell, 1918; Gadd, 1990]. Er bildet ein Wort auf einen phonetischen Code ab. Dabei wird davon ausgegangen, dass Wörter einander ähnlich sind, wenn sie denselben phonetischen Code haben. Somit kann der Algorithmus nur zwischen ähnlichen und nicht-ähnlichen Wörtern unterscheiden und keine Wahrscheinlichkeit für den Grad der Ähnlichkeit angeben. Der Algorithmus arbeitet dabei auf die folgende Weise:

1. Der erste Buchstabe wird nie verändert.
2. Der Algorithmus entfernt alle anderen Vokale, die Konsonanten, *H*, *W*, *Y*, doppelte Buchstaben sowie direkt aufeinander folgende Buchstaben, die auf denselben Code abgebildet würden.
3. Der Soundex-Code wird durch eine Kombination des ersten Buchstabens mit den zugehörigen Ersetzungen (s. Tabelle 3.2) der nächsten Buchstaben gebildet.

Beispielsweise haben die beiden Wörter *Himmel* und *Hymel* jeweils den Soundex-Code H540 und sind demzufolge als ähnlich einzustufen. Phonetische Algorithmen erreichen meistens einen hohen Recall. Allerdings ist im Gegenzug die Precision in den meisten Fällen niedrig, da häufig auch relativ unterschiedliche Wörter denselben Soundex-Code aufweisen [Baron, 2011, S. 27]. Dies gilt z. B. für *Kanton* und *Kundendienst*, die beide den Soundex-Code K535 haben. Der Soundex-Code basiert letztendlich auf dem ersten Buchstaben, gefolgt von drei codierten Buchstaben. Bei kurzen Wörtern werden zusätzliche Nullen hinzugefügt.

Die Entwicklung des Soundex-Algorithmus erfolgte für die englische Sprache. Somit ist eine Modifizierung der phonetischen Klassen notwendig, bevor der Ansatz für eine andere Sprache verwendet werden kann. Außerdem verhindert die Beschränkung des Codes auf 4 Zeichen bei langen Wörtern eine Berücksichtigung der Wortenden. Für die deutsche Sprache gibt es mit der Kölner Phonetik [Postel, 1969] ebenfalls einen phonetischen Algorithmus.

Die phonetische Ähnlichkeit zwischen Schreibvarianten ist größer als die orthographische. Dies konnte für die niederländische Sprache mit Hilfe des Soundex-Algorithmus nachgewiesen werden [Adriaans, 2005]. Allerdings war zugleich die Precision sehr niedrig. Deswegen ist der Soundex-Algorithmus für das Problem der Schreibvarianten nicht geeignet.

3.2.4. Editex

Experimente von Zobel und Dart haben eine geringe Effektivität für Soundex und Phonix nachgewiesen. Trotzdem waren beide Verfahren in der Lage, ähnliche Zeichenketten zu finden, die mit der Levenshtein-Distanz nicht zu finden waren [Zobel und Dart, 1996]. Deswegen wurden die Verfahren im Editex-Algorithmus kombiniert. Die Buchstaben werden auf Grundlage ihrer Phonetik in Klassen eingeteilt (s. Tabelle 3.3). Dabei ist auch eine mehrfache Zuordnung von Buchstaben erlaubt. Die Buchstaben *H* und *W* sowie doppelte Buchstaben finden keine Berücksichtigung, weil sie oftmals bei der Aussprache nicht relevant sind. Anschließend erfolgt wie bei der Levenshtein-Distanz eine Suche nach dem Wort mit dem geringsten Abstand. Den einzelnen Bearbeitungsoperationen werden separate Kosten zugewiesen. Identische Zeichen verursachen daher keine Kosten. Sind die Zeichen in derselben Buchstabenklasse enthalten, werden die Kosten auf 1, sonst auf 2 gesetzt. In Abbildung 3.3 ist beispielhaft die Berechnung des Abstandes des Wortpaares *Himmel - Hymel* dargestellt. Die deutliche Überlegenheit des Verfahrens gegenüber herkömmlichen Abstandsmaßen wurde von Kempken gezeigt [Kempken, 2005].

Code	**Zeichen**
0	A, E, I, O, U, Y
1	B, P
2	C, K, Q
3	D, T
4	L, R
5	M, N
6	G, J
7	F, P, V
8	S, X, Z
9	C, S, Z

Tabelle 3.3.: Editex-Codes

3.2.5. Approximative Indexierungstechniken

Heller präsentiert einen Ansatz für approximative Indexierungstechniken für historische Texte, der den Phonet 2-Algorithmus in Kombination mit der

		H	y	m	e	l
	0	0	2	4	6	8
H	0	1	0	2	4	6
i	2	0	0	1	3	5
m	4	2	1	1	1	3
m	4	2	1	1	1	3
e	6	4	3	1	1	1
l	8	6	5	3	1	1

Abbildung 3.3.: Editex-Distanz für *Himmel* - *Hymel*

Levenshtein-Distanz verwendet [Heller, 2006]. Der Phonet 2-Algorithmus bildet zwei phonetische Repräsentationen mit unterschiedlicher Präzision für ein gegebenes Wort [Michael, 1999]. Heller benutzt diese Repräsentationen zum Aufbau eines Indexes. Zeitgleich mit der Erstellung des Indexes wird ein Levenshtein-Kontrollautomat generiert, der alle Wörter des Korpus speichert. Dieser meldet auftretende Fehler sowie ein Überschreiten eines gesetzten Schwellenwertes für Fehler während eines Durchlaufs. Beim Aufbau des Indexes kommt der Phonet 2-Algorithmus zum Einsatz, um phonetische Repräsentationen zu bilden. Die Codierung der Suchterme erfolgt ebenfalls mit dem Phonet 2-Algorithmus. Jeder Term aus dem Index wird mit dem Suchterm verglichen. Es werden alle Indexterme angezeigt, deren Levenshtein-Distanz zum Suchterm nicht größer als 3 ist. Eine diachrone Angleichung der Phonet 2-Regeln ist nicht erfolgt.

3.2.6. FlexMetric

Kempken hat verschiedene Distanzmaße für ihre Verwendung bei Schreibvarianten geprüft [Kempken, 2005]. Dabei werden drei Kategorien unterschieden: Phonetische Distanzmaße, Ansätze zur Korrektur von Tippfehlern und Zeichenketten-Ähnlichkeitsmaße. Basierend auf den Ergebnissen dieser Evaluierung hat Kempken mit der FlexMetric eine trainierbare Edit-Distanz entwickelt. Als Eingabe für die Berechnung der Transformationskosten eines Buchstabens in einen anderen dienen Wortpaare aus moderner und historischer Form. Wie bei der Levenshtein-Distanz hat das jeweils ähnlichste Wort die geringsten Transformationskosten. Alternativ können die Transformationskosten auch vom Benutzer festgelegt werden. Evaluierungen haben ergeben, dass zum Lernen der Gewichte etwa 4.500 Trainingsbeispiele notwendig sind [Pilz u. a., 2008b]. Der Nachteil des Verfahrens ist der fehlende Einbezug des Kontextes, weil nur Ersetzungskosten für einzelne Buchstaben gelernt werden. Bei einer Ausweitung des Verfahrens auf Bi- oder Trigramme würde die Anzahl der Bearbeitungsoperationen exponentiell ansteigen [Kempken, 2005].

3.3. Regelbasierte Methoden

Der einfachste Weg für die Suche in historischen Texten ist eine Anfrageerweiterung mit den Schreibvarianten. Allerdings ist es bei dieser Verfahrensweise selbst bei einem großen Fachwissen unmöglich, alle Schreibvarianten abzudecken. Braun geht davon aus, dass Regeln für Schreibvarianten das Retrievalproblem für historische Dokumente grundsätzlich lösen [Braun, 2002]. Allerdings entwickelt er manuelle Regeln, weil er das Problem für zu komplex hält, um Regeln automatisch zu generieren. Ziel seiner Regeln ist nicht die Bildung moderner Wörter, sondern lediglich eine Modernisierung der Wörter. Auf diese Weise sollen sie mit Hilfe eines Algorithmus zur Stammformreduktion für aktuelle Sprachen zu finden sein. Esser fasst manuell Muster in phonetischen Gruppen zusammen [Esser, 2004]. Für jede mögliche Kombination innerhalb der Gruppen werden Gewichte festgelegt. Giusti und Kollegen bilden hingegen mit Hilfe von manuell erstellten Regeln Cluster, um die Veränderung der Häufigkeitsverteilung durch Schreibvarianten in einem historischen Wörterbuch für brasilianisches Portugiesisch zu verhindern [Giusti u. a., 2007].

Pilz beschreibt die manuelle Konstruktion von zwei Regelsätzen für deutsche Texte aus dem 19. Jahrhundert [Pilz, 2003]. Beim ersten Regelsatz kann jede Regel höchstens einmal an einer bestimmten Position im Wort eingesetzt werden (z. B. $\ddot{a} \rightarrow e$). Im Gegensatz dazu kann die Anwendung der Regeln des zweiten Regelsatzes beliebig oft erfolgen (z. B. $aa \rightarrow a$). Weil Schreibvarianten oft auf Unterschiede in der Aussprache zurückzuführen sind, wurde ein Teil der Regeln durch Aussprachevergleiche entwickelt. Weitere Regeln sind durch Literaturrecherche entstanden. Für jedes in Frage kommende Wort werden die Regeln sowohl auf die aktuelle als auch auf die historische Wortform angewendet. Wenn die Ergebnisse der Transformationen übereinstimmen, wird von einer Schreibvariante ausgegangen.

Der von Biella und Kollegen beschriebene Ansatz des Variantgraph hat Ähnlichkeiten mit der phonetischen Namensidentifikation [Biella u. a., 2003]. Für jedes Graphem werden die entsprechenden Allographen generiert. Beispielsweise hat das Graphem f die Allographen $\{u, v, f, ff, pf\}$. Ist eine Menge von Allographen für jedes Graphem gegeben, so kann für jedes Eingabewort in aktueller Sprache ein Variantgraph gebildet werden. Abbildung 3.4 zeigt z. B. einen Variantgraph für das Wort *Himmel*.

Mittlerweile gibt es einige Beispiele, bei denen Regeln automatisch generiert werden, da eine manuelle Konstruktion von Regeln sehr zeitaufwendig und zudem stark auf die zugrunde liegende Kollektion abgestimmt ist [Adriaans, 2005]. Exemplarisch folgt dazu die Beschreibung eines Ansatzes für historisches Dokumenten-Retrieval, des Error-Models von Brill und Moore für Regeln zur Rechtschreibkorrektur und eines regelbasierten Verfahrens zur Normalisierung von Texten.

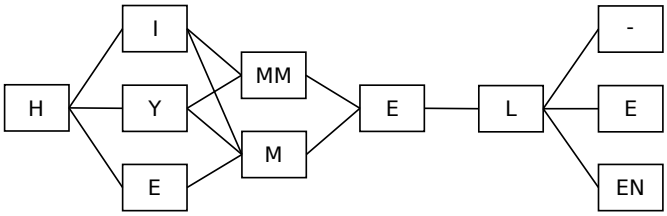


Abbildung 3.4.: Variantgraph für *Himmel* [Biella u. a., 2005]

3.3.1. Historisches Dokumenten-Retrieval

Koolen und Kollegen beschreiben die Anforderungen der Umschreibung von historischen Texten für die niederländische Sprache [Koolen u. a., 2006]. Während die Aufgabe in dieser Arbeit als innersprachliches Problem aufgefasst wird, betrachtet die niederländische Gruppe sie als Gegenstand des mehrsprachigen Retrieval. Die Entscheidung zwischen diesen unterschiedlichen Betrachtungsweisen ist allerdings mehr eine Frage der zugrunde liegenden Definition. Sie entscheidet, wie stark die Unterschiede in der Schreibung sein müssen, damit es sich um unterschiedliche Sprachen handelt. Die Sichtweise der niederländischen Arbeitsgruppe kann an der gleichzeitigen Betrachtung des Vokabelproblems liegen. Diese Perspektive betrachtet größere sprachliche Unterschiede als eine ausschließliche Behandlung der Schreibvarianten. Ziel des Ansatzes ist es analog zu [Braun, 2002], historische Varianten zu modernisieren, um einen modernen Algorithmus zur Stammformreduktion einzusetzen.

Die folgenden drei Ansätze wurden zur Generierung der Regeln entwickelt:

- *Phonetische Sequenzähnlichkeit*: Diese Methode nutzt die Transkriptionen von eins-zu-eins Zuordnungen von historischen und modernen Wörtern. Neue Regeln werden gebildet, wenn die Sequenzen des modernen und des historischen Wortes unterschiedliche Schreibungen und basierend auf der Transkription dieselbe Aussprache haben. Nachfolgend ist ein Beispiel für das historische Wort *klaghen* dargestellt:

historisches Wort:	klaghen
modernes Wort:	klagen
historische Sequenzen:	kl, a, gh, e, n
moderne Sequenzen:	kl, a, g, e, n

  Die Evaluierungsergebnisse zeigen deutlich die direkte Abhängigkeit von der phonetischen Transkription. Diese wurde allerdings nur für modernes Niederländisch entwickelt.

- *Relative Sequenzhäufigkeit*: Wie der vorherige Ansatz unterteilt die relative Sequenzhäufigkeit Wörter der modernen und historischen Korpora in Sequenzen aus Vokalen und Konsonanten. Die Entwicklung von Regeln erfolgt auf Basis von Wörtern mit Sequenzen, die im Vergleich zu modernen

Korpora außerordentlich häufig in historischen Korpora vorkommen. Die Sequenzen von Vokalen und Konsonanten werden durch eine Wildcard ersetzt. Anschließend folgt eine Suche nach modernen Schreibweisen.

- *Relative N-Gramm-Häufigkeit*: Die Relative N-Gramm-Häufigkeit ist eine Variante des letzten Ansatzes, die N-Gramme anstelle von Sequenzen aus Vokalen und Konsonanten benutzt. Dabei gibt es keine Beschränkung der Sequenzen. Der Algorithmus kann Kontextinformationen auf diese Weise besser einbeziehen.

Bei allen drei Verfahren kommt ein sogenannter Beschneidungs-Schwellenwert zum Einsatz, der die Anzahl der falschen Zuordnungen minimieren soll. Die besten Ergebnisse erzielte eine Kombination der Verfahren. Dabei wurde zunächst die relative N-Gramm-Häufigkeit, dann die relative Frequenzhäufigkeit und anschließend die phonetische Sequenzhäufigkeit angewendet.

Bei der Einteilung in Sequenzen von Vokalen und Konsonanten wird die moderne Einteilung der Buchstaben zugrunde gelegt. Die beiden letzten Ansätze verwenden nur typisch historische Sequenzen. Etliche Schreibvarianten enthalten Sequenzen, die auch im modernen Niederländisch sehr häufig sind. Für diese Wörter ist es demzufolge nicht möglich, auf Basis der Sequenzhäufigkeiten Regeln zu generieren.

Im niederländischen Ansatz fanden Werkzeuge für die phonetische Transkription Verwendung, die für das Deutsche nicht in zuverlässiger Qualität zur Verfügung stehen [Pilz, 2009, S. 117]. Zudem werden im Deutschen auch Regeln benötigt, die Konsonanten durch Vokale ersetzen (z. B. $v \rightarrow u$), sodass die Unterteilung in Sequenzen und Konsonanten für die deutsche Sprache nicht sinnvoll ist.

3.3.2. Error Model zur Rechtschreibkorrektur

Bisherige Modelle zum Einsatz des Noisy Channel Models zur Rechtschreibkorrektur basierten auf der Levenshtein-Distanz und erlaubten nur eine Editoperation pro Wort. Außerdem bezogen sie nur teilweise den Kontext mit ein, um die Wahrscheinlichkeit für die einzelnen Operationen zu berechnen. Brill und Moore erweitern das Channel Model für Rechtschreibfehler um ein sog. Error Model, bei dem das Wort w aus dem Wörterbuch D gefunden werden soll, das mit der größten Wahrscheinlichkeit die geplante Eingabe zu einer Zeichenkette s war [Brill und Moore, 2000].

Die Erweiterung des Modells lernt generische Operationen zur Überführung von Zeichenketten. Zu den Operationen gehören jeweils spezifische Editkosten, die sich im Unterschied zu den Levenshtein-Ansätzen nicht nur auf einzelne Buchstaben, sondern auf Buchstabensequenzen beziehen. Es sind alle Editoperationen der Form $\alpha \rightarrow \beta$ möglich. $P(\alpha \rightarrow \beta)$ gibt die Wahrscheinlichkeit an, dass der Benutzer eine Zeichenkette α anstatt der Zeichenkette β eingibt. Zusätzlich wird die Stelle

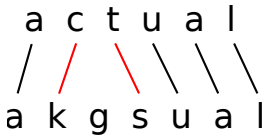


Abbildung 3.5.: Beispiel Alignment von Brill und Moore [Brill und Moore, 2000]

c	$\rightarrow$	k
ac	$\rightarrow$	ak
c	$\rightarrow$	kg
ac	$\rightarrow$	akg
ct	$\rightarrow$	kgs

Tabelle 3.4.: Beispiel für Ersetzungen bei Brill und Moore [Brill und Moore, 2000]

der Positionsänderung im Wort (Wortanfang, -mitte und -ende) berücksichtigt, da Fehler z. B. am Wortende häufiger auftreten als am Wortanfang.

Zum Training des Modells werden Wortpaare (s, w) als Eingabe benutzt. Beim Aligning wird versucht, die Anzahl der Editoperationen für die einzelnen Buchstaben zu minimieren. Für jede Editoperation können bis zu N benachbarte Zeichen mit einbezogen werden. In Abbildung 3.5 ist das Alignment am Beispiel des Trainingspaares *akgsual* - *actual* dargestellt. Daraus ergeben sich für $N = 2$ die in Tabelle 3.4 enthaltenen Ersetzungen für nicht identische Editoperationen. In der Trainingsphase wird für jede Editoperation eine Wahrscheinlichkeit anhand der Häufigkeit der gebildeten Substitutionen bestimmt. Die Anwendung gibt eine gerankte Liste mit den n besten Kandidaten für die Korrektur zurück.

3.3.3. Regelbasierte Normalisierung von Texten

Ein regelbasiertes Verfahren zur Normalisierung von historischen Dokumenten haben Bollmann und Kollegen vorgestellt [Bollmann u. a., 2011]. Ein erweiterter Levenshtein-Algorithmus gibt die notwendigen Editoperationen zur Abbildung eines Wortes auf ein anderes aus. Dabei wird immer der unmittelbare Kontext, jeweils ein Zeichen oder eine Wortgrenze, berücksichtigt. Es wird zwischen Identitätsregeln und Nichtidentitätsregeln unterschieden. Letztere sind Regeln im herkömmlichen Sinn. Identitätsregeln dagegen bilden Buchstaben auf sich selbst ab. Da ein Großteil eines Wortes unverändert bleibt, werden sie eingesetzt, wenn sie ein höheres Ranking haben als Nichtidentitätsregeln. In einem zweiten Schritt werden aufeinander folgende Nichtidentitätsregeln zu sequenzbasierten Regeln zusammengefasst, wenn an einer Position mehrere Regeln zum Einsatz kommen. Auf diese Weise werden z. B. Regeln für Affixe gebildet. Für das Beispiel *jrem* - *ihrem* ergeben sich so die folgenden Regeln:

Eingabe	j	r	e	m
Operation	s	=	=	=
Ausgabe	ih	r	e	m

Da Einfügeregeln oft zu Fehlern führen, werden zusätzlich sogenannte Epsilon-Identitätsregeln eingeführt. Diese geben an, in welchem Kontext eine Verwendung von Einfügeregeln untersagt ist. So gibt z. B. die Regel $\varepsilon \rightarrow \varepsilon \ / \ e \ \_ \ n$ an, dass zwischen e und n kein Buchstabe eingefügt werden darf.

Für die Normalisierung wird überprüft, ob die gebildeten Varianten in einem modernen Lexikon enthalten sind, damit nur existierende Wörter zum Einsatz kommen. Die gelernten Regeln werden auf Basis ihrer Häufigkeit gewichtet. Letztendlich wird das Wort mit dem höchsten Ranking zur Normalisierung verwendet.

3.4. Kombinierte Verfahren

Hauser und Kollegen kombinieren spezielle Lexika und einen regelbasierten Ansatz mit einem Verfahren zum Abgleich von Wortähnlichkeiten, um modernen Schreibungen ihre historischen Varianten zuzuordnen [Hauser u. a., 2007]. Die Lexika werden vor allem verwendet, um das Problem der Vokabellücke zu lösen und lateinische Varianten zu finden. Der regelbasierte Ansatz senkt die Kosten für die entsprechenden Editoperationen. Bei der Bestimmung der Wortdistanz kommt das Verfahren von Brill und Moore zum Einsatz (s. Abschnitt 3.3.2), bei dem auch Sequenzen von Buchstaben ersetzt werden können. Im Anschluss erfolgt eine Gewichtung der Kandidaten aller drei Schichten. Dazu werden Wortähnlichkeiten, Häufigkeitsinformationen und geeignete Heuristiken für die Einbeziehung der Präferenzen des Benutzers bezüglich der ersten beiden Schichten verwendet.

3.5. Zusammenfassung

Dieses Kapitel hat verschiedene Verfahren zur Behandlung von Schreibvarianten vorgestellt. Es wurden die wichtigsten wörterbuchbasierten Methoden, Ähnlichkeitsmaße, regelbasierte Verfahren sowie kombinierte Vorgehensweisen beschrieben. Kommen Wörterbücher zum Einsatz, wird automatisch für jedes neue Korpus ein passendes Wörterbuch benötigt. Dies steht häufig nicht zur Verfügung. Bei der Verwendung von Ähnlichkeitsmaßen zur Suche ist immer ein Zugriff auf den Index erforderlich. Allerdings ist dies insbesondere bei der Suche im Internet oft nicht möglich. Zwar erfordert ein regelbasiertes Verfahren für ein neues Korpus gegebenenfalls auch eine Erstellung neuer Regeln. Jedoch lässt sich der Recall durch eine weitere Regel deutlich schneller steigern als durch einen zusätzlichen

Lexikoneintrag. Demzufolge ist auch die Wahrscheinlichkeit höher, dass selbst mit einem nicht optimalen Regelsatz trotzdem noch gute Retrievalergebnisse erzielt werden.

Ein regelbasierter Ansatz erfordert zu Beginn einen recht hohen Aufwand bei der Erstellung der Lernmenge, die aus Belegen von moderner Schreibweise und zugehöriger Variante besteht. Auch bei ständig wachsenden Kollektionen wird im Folgenden nur wenig zusätzliche Arbeit erwartet, weil auf dem Graphemlevel gearbeitet wird.

Nach Witten und Frank zeigen manuell erstellte Regeln bereits gute Ergebnisse. Allerdings sollten maschinelle Lernverfahren zum Einsatz kommen, wenn die Erstellung von manuellen Regeln zu aufwendig ist [Witten und Frank, 2000, S. 25]. Die manuelle Erstellung eines Regelsatzes aus 338 Belegen benötigte 3 Tage und die Erstellung der Regeln für VARD (s. Abschnitt 6.3) dauerte sogar über ein Jahr [Pilz u. a., 2008b]. Diese Zeitaufwände verdeutlichen die Notwendigkeit einer automatischen Erstellung von Regelsätzen, damit sich die Wissenschaftler mehr mit den Dokumenten selbst und nicht mit der Bereitstellung der Suche nach den Dokumenten befassen können. Die vorliegende Arbeit bearbeitet das Problem demzufolge mit einem automatischen Ansatz zur Regelerstellung. Dadurch wird es möglich, einen breiteren Wortschatz abzudecken und so den Recall zu erhöhen. Auf der anderen Seite sollte die Precision der generierten Regeln hoch genug sein, um zwischen Schreibvarianten des Suchbegriffes und anderen Wörtern zu unterscheiden.

Teil III.

Entwicklung einer Suchmaschine für historische Dokumente

4. Verfahren zur Regelgenerierung

Die vorangegangenen Kapitel haben die Problematik bei der Suche nach Schreibvarianten verdeutlicht und die bisher eingesetzten Verfahren diskutiert. Der nachfolgende Abschnitt befasst sich zunächst grundlegend mit Lernverfahren, bevor sich auf Basis der aus den verwandten Arbeiten gewonnenen Erkenntnisse die Entwicklung eines eigenen Lernverfahrens zur Generierung von Regeln anschließt.

4.1. Lernverfahren

Nach Mitchell lernt ein Programm aus einer Erfahrung unter Einbeziehung einer Menge von Aufgaben und eines Bewertungsmaßes. Dabei muss sich die Leistung des Systems bei den Aufgaben, bezogen auf das Bewertungsmaß, durch die Nutzung des Gelernten verbessern [Mitchell, 2001, S. 2f.]. Entsprechend besteht bei der Suche in historischen Dokumenten die Aufgabe darin, zu einem Suchbegriff die in einem Korpus enthaltenen Schreibvarianten zu finden. Um dies zu erreichen, soll eine dem Noisy Channel Model (s. Abschnitt 2.3) entsprechende Korrektur der Suchanfrage in Form einer Anfrageerweiterung durchgeführt werden. Als Bewertungsmaße kommen dabei Recall und Precision zum Einsatz. Die Erfahrung wird aus der Trainingsmenge in Form von Belegen gewonnen.

Bei der Entwicklung eines Lernsystems hat die Auswahl einer repräsentativen Trainingsmenge einen großen Einfluss auf den Lernerfolg [Mitchell, 2001, S. 6]. Stammen die Belege z. B. nur von einem Autor oder aus einem bestimmten Zeitraum, so ist nicht von einer repräsentativen Verteilung auszugehen, wenn das Korpus mehrere Autoren oder andere Zeiträume mit abdeckt. In der Theorie des maschinellen Lernens sollte die Verteilung der Trainingsinstanzen identisch mit der Verteilung der Testinstanzen sein. Ein weiterer wichtiger Aspekt ist der Automatisierungsgrad des Systems: Übernimmt das System die komplette Kontrolle, oder bekommt der Benutzer die Gelegenheit, das Ergebnis zu beeinflussen? Kann der Benutzer z. B. jede gelernte Regel verändern, oder hat er nur die Möglichkeit, durch indirektes Feedback in Form von genutzten Varianten Einfluss zu nehmen?

Generell unterscheidet man bei den Lernverfahren deduktive, abduktive und induktive Prozesse [Ferber, 2003, S. 106ff.]. Beim deduktiven Prozess wird eine neue Regel aus bereits bekannten Regeln abgeleitet. Damit muss bereits vor dem

Einsatz des eigentlichen Lernverfahrens eine Basis von Regeln zur Verfügung stehen. Sollten in den Ausgangsregeln Fehler enthalten sein, übertragen sich diese auch auf die abgeleiteten Regeln. Beim abduktiven Prozess werden auf Basis von Beispielen und vorhandenen Regeln Rückschlüsse auf die Zusammenhänge gezogen. Diese Vorgehensweise setzt allerdings die Möglichkeit zur Umkehrung vorhandener Regeln voraus. Bei einem induktiven Prozess werden Regeln aus Beispielen abgeleitet. Die Korrektheit der vorhandenen Beispiele beeinflusst dabei entscheidend die Qualität der Regeln. Die gelernten Regeln werden dabei meistens anhand einer von der Trainingsmenge unabhängigen Testmenge überprüft. Bei einer großen Anzahl von Beispielen ist es nicht immer möglich, allgemeingültige Regeln zu generieren. Dann ist es das Ziel, Regeln zu erstellen, die für eine möglichst große Anzahl von Beispielen gelten. Bei der Entwicklung des regelbasierten Verfahrens soll eine Ableitung aller Regeln aus den vorhandenen Texten bzw. Beispielen erfolgen. Deswegen wird im Folgenden ein induktives Verfahren zur Erstellung der Regeln entwickelt.

Bei der Erstellung von Regeln lassen sich des Weiteren der Top-down- und der Bottom-up-Ansatz unterscheiden [Ferber, 2003, S. 135f.]. Die Top-down-Methode beginnt mit der allgemeinsten Regel. Diese wird in den darauf folgenden Bearbeitungsschritten durch speziellere Regeln verfeinert. Die Bottom-up-Methode beginnt hingegen mit sehr speziellen Regeln, und versucht durch Generalisierung mehrere Beispiele mit allgemeineren Regeln abzudecken.

Damit erlernte Regelsätze auch unabhängig von den Trainingsdaten gute Ergebnisse erzielen, ist eine Überanpassung an die Daten zu vermeiden [Petersohn, 2005, S. 149]. Würden z. B. alle einzelnen Belege aus moderner und historischer Schreibung direkt in spezifische Regeln umgewandelt, ließe sich auf Basis der Trainingsdaten auf perfekte Retrievalergebnisse schließen.

Ein weit verbreiteter Algorithmus im Data Mining ist der ID3[8] [Quinlan, 1986]. Dabei sind die Beispiele in einem Vektor mit nominalen Attributen repräsentiert. Darauf aufbauend entwickelt der Algorithmus, basierend auf dem Teile- und Herrsche-Prinzip, einen Entscheidungsbaum. Ausgehend vom Wurzelknoten wird bei jedem Knoten ein Attribut zur Klassifizierung der Beispiele verwendet. Ziel ist jeweils die Auswahl des Attributes mit dem größten Informationsgewinn. Der Algorithmus endet, wenn die Blätter des Baumes erreicht sind. Ein Blatt gibt für das jeweilige Tupel die Kategorie an. Oftmals erreichen einfachere Entscheidungsbäume bessere Resultate als kompliziertere. Deswegen erfolgt eine Reduzierung mit Hilfe von Beschneidungsverfahren. Dabei werden Knoten mit geringer statistischer Signifikanz aus dem Baum entfernt bzw. nicht mit aufgenommen [Petersohn, 2005, S. 137]. Es wird grundsätzlich zwischen Pre- und Post-Beschneidung unterschieden [Witten und Frank, 2000, S. 174ff.]. Die Pre-Beschneidung entscheidet bereits während des Aufbaus des Baumes darüber, ob eine Weiterentwicklung der Unterbäume sinnvoll ist. Es ist nicht möglich, Vorteile, die sich durch die Kombination von Attributen ergeben, mit in den Entscheidungsprozess

[8] Iterative Dichotomiser 3

einzubeziehen. Bei der Post-Beschneidung wird der Entscheidungsbaum vor der Beschneidung zunächst komplett aufgebaut. Dabei wird weiter zwischen der Teilbaum-Ersetzung, die Unterbäume durch einzelne Blätter ersetzt, und dem Teilbaum-Hochziehen, das Unterbäume nach oben zieht, unterschieden. Ein Entscheidungsbaum kann einfach in eine Regelmenge überführt werden, indem man für jedes Blatt eine Regel generiert, die dem Pfad von der Wurzel zum Blatt entspricht.

Dagegen entwickeln Abdeckungsalgorithmen unmittelbar Regeln. Sie betrachten einzelne Klassen nacheinander und generieren in jedem Schritt eine Regel, die einen Teil der Instanzen abdeckt [Witten und Frank, 2000, S. 104]. Die nicht von der gebildeten Regel abgedeckten Instanzen werden dabei zugleich ausgeschlossen. Im nächsten Schritt werden nur noch die bisher nicht abgedeckten Instanzen betrachtet.

ID3 wählt das Attribut mit dem maximalen Informationsgewinn aus. Dabei prüft er allerdings nur die Blätter. Dagegen suchen Abdeckungsalgorithmen dasjenige Attribut/Wert-Paar, mit dem sich die Wahrscheinlichkeit einer bestimmten Klassifikation maximieren lässt.

4.2. Vorgehensweise

In der Regel ist der Benutzer nicht nur an Suchergebnissen interessiert, die den Suchbegriff selbst beinhalten, sondern auch an Dokumenten, die die zugehörigen morphologischen Varianten enthalten [Braun, 2002]. Die Computerlinguistik fasst verschiedene Flexionsformen deswegen als zusammengehörig, teilweise sogar als identische Wörter auf [Ferber, 2003, S. 40f.]. Beim Aufbau eines Indexes werden deshalb Wortformen häufig zusammengefasst, um den Speicherbedarf zu minimieren und den Recall zu erhöhen.

Bei der Bildung der Grundform wird zwischen Stammformreduktion und Grundformreduktion unterschieden. Die Stammformreduktion bildet Wörter auf ihren Wortstamm zurück. Dieser ist im Allgemeinen kein Wort der zugrunde liegenden Sprache. Bei der Stammformreduktion kann es durchaus vorkommen, dass sehr unterschiedliche Wörter fälschlicherweise auf denselben Stamm zurückgeführt und somit als zusammengehörig betrachtet werden. So gibt es für den Wortstamm *comput* z. B. die folgenden Flexions- und Derivationsformen (sog. Vollformen): *compute*, *computer*, *computation*.

Dagegen führt die Grundformreduktion Vollformen auf Lemmata und somit auf existierende Wörter zurück. Die Grundform für Nomen ist der Nominativ Singular und bei Verben der Infinitiv Präsens [Hausser, 2000, S. 267.]. So hat die Grundform *compute* unter anderem die zugehörigen Vollformen *compute*, *computes*, *computed*.

Im Deutschen ändert sich bei der Bildung von Flexionsformen häufig auch der Wortstamm [Ferber, 2003, S. 44f.]. So werden Umlaute eingefügt (z. B. *Haus*

- *Häuser*) oder Präfixe hinzugefügt (z. B. *backen - gebacken*). Präfixe können innerhalb des Satzes auch abgetrennt werden (z. B. *er ging mit*).

Weil sich die Vollformen stark von ihren Wortstämmen und Grundformen unterscheiden, müssen bei der Suche in historischen Texten die Regeln auch auf die einzelnen Vollformen angewendet werden.

Experimentelle Suchmaschinen für die englische Sprache arbeiten meist mit Algorithmen zur Stammformreduktion, um die Vollform eines Suchbegriffs zu finden. Bei der im folgenden schwerpunktmäßig untersuchten deutschen Sprache handelt es sich um eine stark flektierte Sprache. Deswegen ist es sinnvoller, mit Grundformen anstelle von Stammformen zu arbeiten.

Analog zur Suche nach morphologischen Formen [Moreau u. a., 2007] gibt es zwei mögliche Ansätze für die Suche in Texten mit Schreibvarianten:

1. *Grundformreduktion zur Indexierungszeit*: Diese üblicherweise im Information Retrieval eingesetzte Methode erfordert einen Satz von Regeln zur Grundformreduktion, der zur Indexierungszeit bekannt sein muss. Im Kontext von Schreibvarianten müssen darüber hinaus auch Regeln für die Standardisierung der verschiedenen Schreibweisen vorhanden sein. Allerdings erfordert eine regelbasierte Grundformreduktion für heutiges Deutsch aufgrund des hohen Flexionsgrades einen sehr komplexen Regelsatz. Noch schwieriger sind Regelsätze zu bilden, die die historische Schreibung auf die zugehörige aktuelle Grundform abbilden. In diesem Fall existiert kein fester Regelsatz für die Standardisierung der Schreibweisen. Aufgrund der Zeit- und Ortsabhängigkeit der Varianten wird dieser vermutlich auch nie vorliegen. Auch bei Sprachen, für die sehr gute Methoden zur Grundformreduktion vorhanden sind (wie z. B. Englisch), stellt sich die Frage, ob diese bei historischen Dokumenten zum Einsatz kommen können.
2. *Generierung von Suchtermvarianten zur Retrievalzeit*: Für diese Anfrageerweiterung werden ebenfalls Regeln für Flexionen und Derivationen sowie auch für die Schreibvarianten benötigt, allerdings diesmal in der anderen Richtung; somit ist die folgende Abbildung notwendig:

   $$\textit{Suchterm} \rightarrow \textit{moderne Flexionsform} \rightarrow \textit{Schreibvariante}$$

   Dieser Ansatz ist deutlich flexibler, weil eine Integration neuer Regelsätze leichter erfolgen kann.

Zwar ist der erste Ansatz deutlich schneller als der zweite, weil die zeitkritische Generierung der Transformationen einmalig bei der Indexierung einer neuen Kollektion abläuft. Allerdings ist zu erwarten, dass die Regelsätze häufig über längere Zeit nicht endgültig entwickelt sind. Außerdem eignet sich der erste Ansatz nur für eigene Korpora, bei denen Zugriff auf den Index besteht. Eine Übertragung des Ansatzes auf die Suche mit einer Internetsuchmaschine wäre nur in Ausnahmefällen möglich, wenn ein entsprechender Zugriff besteht. Deswegen bietet

nur der zweite Ansatz die notwendige Flexibilität. Bei diesem ist zunächst eine Behandlung der morphologischen Variationen erforderlich, bevor die Regeln für Schreibvarianten eingesetzt werden können. Eine direkte Anwendung von modernen Algorithmen zur Stammform- bzw. Grundformreduktion ist nicht möglich [Braun, 2002]. Adriaans hat für das Niederländische gezeigt, dass bei der Anwendung von Stammformreduktion das Suffix entfernt wird, während sich bei Schreibvarianten häufig das Präfix oder Infix ändert [Adriaans, 2005].

4.3. Generierung von Transformationsregeln

Das Problem der Schreibvarianten lässt sich grundsätzlich auf das Problem der Schreibvarianten für Silben reduzieren. Es ist davon auszugehen, dass die Variationen in der Schreibung sich nicht über Silbengrenzen hinweg erstrecken. Leider stand dem Projekt kein zuverlässiges Werkzeug zur Silbentrennung für modernes Deutsch zur Verfügung. Weil sich die Silbentrennung ebenfalls im Verlauf der Zeit geändert haben kann, wäre darüber hinaus die Entwicklung eines entsprechenden Werkzeugs zur Silbentrennung speziell für die Schreibvarianten erforderlich. Der nachfolgend dargestellte Ansatz ermöglicht allerdings eine direkte Anwendung auf Silben, so dass hier keine Anpassung des Ansatzes notwendig wäre, wenn ein entsprechendes Werkzeug zur Verfügung stehen sollte.

Für die Generierung der Transformationsregeln können die folgenden Eingabedaten genutzt werden:

- die Belege aus 1:1-Abbildungen von modernen Formen auf die zugehörige Schreibvariante,
- die historischen Dokumente, auf deren Grundlage die Bildung der Belege erfolgt ist, sowie die Termhäufigkeit der Wörter, basierend auf den Dokumenten.

Analog zu anderen Verfahren zur Verarbeitung natürlicher Sprache [Abels und Hahn, 2005] wird im Folgenden davon ausgegangen, dass keine Unterscheidung zwischen Groß- und Kleinschreibung erforderlich ist. Deswegen erfolgt eine Umwandlung der Belege in Kleinschreibung.

Wie bereits beschrieben, startet die Regelgenerierung mit der Erstellung der Belege. Die Vorgehensweise ist dabei ähnlich wie bei Liu und Kollegen (s. Abschnitt 2.4.5) [Liu u. a., 1991]. Dazu markiert zunächst ein Programm zur Rechtschreibprüfung die unbekannten Wörter und somit auch die potentiellen Varianten. Der Grundgedanke der Rechtschreibprüfung basiert auf der Annahme, dass ein nicht im Lexikon enthaltenes Wort falsch geschrieben wurde [Robertson und Willett, 1998]. Abweichend wird in der vorliegenden Arbeit davon ausgegangen, dass es sich bei den markierten Wörtern um Schreibvarianten handeln könnte. Bei Schreibfehlern kann es aber auch zur Bildung von gültigen Wörtern, den sogenannten Echtwortfehlern,

kommen [Fliedner, 2001]. Dies gilt auch für einige historische Varianten, die demzufolge auf diese Weise nicht zu finden sind. Die Basis für das vorgestellte Verfahren ist die Regelhaftigkeit. Deswegen wird angenommen, dass es möglich ist, die für diese Wörter notwendigen Regeln anhand von anderen Belegen zu erstellen. Für Schreibvarianten wird manuell die aktuelle Schreibweise hinzugefügt. Zusätzlich wird die Kollektionshäufigkeit der Terme in dem historischen Text bestimmt. Dadurch erhält man ein Menge von Tripeln $H(\mathbf{a},\ \mathbf{h},\ cf)$, bestehend aus der aktuellen Wortform **a**, der historischen Vollform **h** und der Kollektionshäufigkeit cf.

Nachfolgend wird die Buchstabenfolge **a** auch als Sequenz $a_1 \dots a_n$ bezeichnet. Für $n = 0$ wird $a_1 \dots a_n$ auch als leere Folge ε bezeichnet. Des Weiteren bezeichnet $a \mid b$ die Konkatenation von Buchstabenfolgen. Die Definitionen gelten ebenfalls für die Buchstabenfolge **h**. Zusätzlich beginnt und endet jede Wortform mit einem Leerzeichen. Für Listen wird die Notation $[l_1,\ \dots,\ l_k]$ verwendet. Zur Aufteilung von Listen in Kopf h und Rumpf t wird die Schreibweise $[h \mid t]$ verwendet.

4.3.1. Generierung von Regelkernen

Bei der Generierung der Transformationsregeln wird die Menge H aus Belegen und Kollektionshäufigkeit zum Training genutzt. Zunächst werden jeweils die beiden Wörter eines Belegs verglichen und mit den sogenannten Regelkernen die notwendigen Transformationen identifiziert. Der zweite Schritt bestimmt die Regelkandidaten, die auch den Regelkontext des Wortes **a** mit einbeziehen. Anschließend werden die nützlichen Regeln durch Beschneidung der Menge der Regelkandidaten ausgewählt. Zur Beschreibung der verschiedenen Schritte werden die zugrunde liegenden Funktionen näher erläutert.

Für die Regelkerne folgt die Bestimmung der notwendigen Transformationen sowie des zugehörigen Kontextes. Zunächst wird eine Funktion $rcg1()$ definiert, die eine gemischte Liste aus Transformationen und Kontext erzeugt. Dazu benötigt sie als Eingabe einen Beleg sowie eine leere Zeichenkette:

$$
rcg1(a_1 \dots a_n,\ h_1 \dots h_m,\ p) =
\begin{cases}
[p] & \text{, wenn } n = m = 0 \\
[rcg1(a_2 \dots a_n,\ h_2 \dots h_m,\ p \mid a_1)] & \text{, wenn } a_1 = h_1 \\
[p,\ (a_1 \dots a_j,\ h_1 \dots h_l) \mid rcg1(a_{j+1} \dots a_n,\ h_{l+1} \dots h_m,\ \varepsilon)] & \\
\quad \text{so dass } a_{j+1} = h_{l+1} \text{ und } j + l \text{ ist minimal} & \text{, wenn } a_1 \neq h_1
\end{cases}
$$

Werden z. B. die Wörter **a**=' unnütz ' und **h**=' unnuts ' eingegeben, liefert, wie nachfolgend aufgeführt, $rcg1(\mathbf{a},\ \mathbf{h},\ \varepsilon)$ = [' unn', ('ü', 'u'), 't', ('z', 's'), ' '].

$$
\begin{aligned}
rcg1(\text{' unnütz ', ' unnuts ', } \varepsilon) &= rcg1(\text{'unnütz ', 'unnuts', } \varepsilon \mid \text{' '}) \\
&= rcg1(\text{'nnütz ', 'nnuts ', ' ' | 'u'}) \\
&= rcg1(\text{'nütz ', 'nuts ', ' u' | 'n'}) \\
&= rcg1(\text{'ütz ' ,'uts ', ' un' | 'n'}) \\
&= [\text{' unn', ('ü', 'u') } \mid rcg1(\text{'tz ', 'ts ', } \varepsilon)] \\
&= [\text{' unn', ('ü', 'u') } \mid rcg1(\text{'z ', 's ', } \varepsilon \mid \text{'t'})] \\
&= [\text{' unn', ('ü', 'u'), 't', ('z', 's') } \mid rcg1(\text{' ', ' ', } \varepsilon] \\
&= [\text{' unn', ('ü', 'u'), 't', ('z', 's') } \mid rcg1(\varepsilon, \varepsilon, \varepsilon \mid \text{' '}] \\
&= [\text{' unn', ('ü', 'u'), 't', ('z', 's'), ' '}]
\end{aligned}
$$

Aus der mit $rcg1()$ berechneten Liste L werden die Regelkerne, also die Transformationen sowie der rechte und der linke Kontext, mit Hilfe der folgenden, rekursiven Funktion generiert:

$$
rcg(L) = \begin{cases} \emptyset & \text{, wenn } |L| = 1 \\ \{(l,\ t,\ r)\} \cup rcg(L') \text{ mit } L = [l,\ t,\ r\ \mid\ R] \text{ und } L' = [r \mid\ R] & \text{, sonst} \end{cases}
$$

Für das obige Beispiel werden die Regelkerne sowie der Kontext wie folgt berechnet:

$$
\begin{aligned}
rcg([\text{' unn', ('ü', 'u'), 't', ('z', 's'), ' '}]) &= \{(\text{' unn', ('ü', 'u'), 't'})\} \cup rcg([t, \text{ ('z', 's'), ' '}]) \\
&= \{(\text{' unn', ('ü', 'u'), 't'})\} \cup \{(\text{'t', ('z', 's'), ' '})\} \\
&= \{(\text{' unn', ('ü', 'u'), 't'}), (\text{'t', ('z', 's'), ' '})\}
\end{aligned}
$$

Somit ergibt sich die 2-elementige Menge aus Regelkernen und Kontext:

{(' unn', ('ü', 'u'), 't'), ('t', ('z', 's'), ' ')}

4.3.2. Bestimmung der Regelkandidaten

Für jeden Regelkern entsteht eine Menge von Regelkandidaten durch schrittweises Hinzufügen von rechtem und linkem Kontext zu der Transformationsregel. Neben den exakten Zeichen im Kontext werden als Abstraktion auch die Buchstabenklassen Vokale (V) und Konsonanten (K) entsprechend hinzugefügt. Zusätzlich bezeichnet B ein Leerzeichen. Folglich hat der linke Kontext einer

Transformation die generelle Syntax B?[K/V]*['a'..'z']* [9], und der rechte Kontext folgt der Grammatik ['a'..'z']*[K/V]*B?.

Für jedes Element $(l,\ (s,\ d),\ r)$ aus der Liste von Regelkernen und Kontext wird die Funktion $rg(l,\ r,\ (\varepsilon,\ s,\ \varepsilon,\ d))$ aufgerufen, die wie folgt definiert ist:

$$\begin{aligned} rg(l,\ r,\ t) &= rgcl(l,\ r,\ t) \cup rgcr(l,\ r,\ t) \cup rge(l,\ r,\ t) \\ rge(l,\ r,\ t) &= rgl(l,\ r,\ t) \cup rgr(l,\ r,\ t) \cup \{t\} \end{aligned}$$

$$rgcl(l_1 \ldots l_n,\ r_1 \ldots r_m,\ (f,\ s,\ b,\ d)) = \begin{cases} rg(l_1 \ldots l_{n-1},\ r_1 \ldots r_m,\ (l_n \mid f,\ s,\ b,\ d)) & \text{, wenn } n > 0 \\ \emptyset & \text{, sonst} \end{cases}$$

$$rgcr(l_1 \ldots l_n,\ r_1 \ldots r_m,\ (f,\ s,\ b,\ d)) = \begin{cases} rg(l_1 \ldots l_n,\ r_2 \ldots r_m,\ (f,\ s,\ b \mid r_1,\ d)) & \text{, wenn } m > 0 \\ \emptyset & \text{, sonst} \end{cases}$$

$$rgl(l_1 \ldots l_n,\ r_1 \ldots r_m,\ (f,\ s,\ b,\ d)) = \begin{cases} rge(l_1 \ldots l_{n-1},\ r_1 \ldots r_m,\ (cc(l_n) \mid f,\ s,\ b,\ d)) & \text{, wenn } n > 0 \\ \emptyset & \text{, sonst} \end{cases}$$

$$rgr(l_1 \ldots l_n,\ r_1 \ldots r_m,\ (f,\ s,\ b,\ d)) = \begin{cases} rge(l_1 \ldots l_n,\ r_2 \ldots r_m,\ (f,\ s,\ b \mid cc(r_1),\ d)) & \text{, wenn } m > 0 \\ \emptyset & \text{, sonst} \end{cases}$$

Die Funktion rg() vereinigt die Ergebnisse der Funktionen rgcl(), rgcr() und rge(). Hier rufen die Funktionen rgcl() bzw. rgcr() die Funktion rg() rekursiv auf. Bei jedem Aufruf wird dabei die notwendige Transformation links bzw. rechts mit weiteren Kontextinformationen in Form von Buchstaben ergänzt. Dagegen produziert rge() letztendlich die Regeln und ruft die Funktionen rgl() und rgr() für die oben beschriebenen Generalisierungen auf. rgl() generalisiert den nächsten Buchstaben des linken Kontextes und rgr() den nächsten Buchstaben des rechten Kontextes. In der praktischen Anwendung beschränkt eine maximale Länge für den zu betrachtenden Kontext die Anzahl der durch diese Funktionen generierten Regelkandidaten. Die Funktion cc(x) liefert die Buchstabenklasse (K bzw. V) des Zeichens x.

Für den ersten der soeben berechneten Regelkerne wird im folgenden die Berechnung der Regelkandidaten auszugsweise durchgeführt:

[9] Entsprechend der Schreibweise für reguläre Ausdrücke bezeichnet ? ein höchstens einmaliges Vorkommen und * bezeichnet gar kein oder mehrfaches Vorkommen

$$
\begin{array}{rcll}
rg(\text{'unn'}, \text{'t'}, (\varepsilon, \text{'ü'}, \varepsilon, \text{'u'})) & = & rgcl & (\text{'unn'}, \text{'t'}, (\varepsilon, \text{'ü'}, \varepsilon, \text{'u'})) \\
 & \cup & rgcr & (\text{'unn'}, \text{'t'}, (\varepsilon, \text{'ü'}, \varepsilon, \text{'u'})) \\
 & \cup & rge & (\text{'unn'}, \text{'t'}, (\varepsilon, \text{'ü'}, \varepsilon, \text{'u'})) \\
 & & & \\
rgcl(\text{'unn'}, \text{'t'}, (\varepsilon, \text{'ü'}, \varepsilon, \text{'u'})) & = & rg & (\text{'un'}, \text{'t'}, (\text{'n'}, \text{'ü'}, \varepsilon, \text{'u'})) \\
 & = & rgcl & (\text{'un'}, \text{'t'}, (\text{'n'}, \text{'ü'}, \varepsilon, \text{'u'})) \\
 & \cup & rgcr & (\text{'un'}, \text{'t'}, (\text{'n'}, \text{'ü'}, \varepsilon, \text{'u'})) \\
 & \cup & rge & (\text{'un'}, \text{'t'}, (\text{'n'}, \text{'ü'}, \varepsilon, \text{'u'})) \\
 & & & \\
rgcl(\text{'un'}, \text{'t'}, (\text{'n'}, \text{'ü'}, \varepsilon, \text{'u'})) & = & rg & (\text{'u'}, \text{'t'}, (\text{'nn'}, \text{'ü'}, \varepsilon, \text{'u'})) \\
 & = & rgcl & (\text{'u'}, \text{'t'}, (\text{'nn'}, \text{'ü'}, \varepsilon, \text{'u'})) \\
 & \cup & rgcr & (\text{'u'}, \text{'t'}, (\text{'nn'}, \text{'ü'}, \varepsilon, \text{'u'})) \\
 & \cup & rge & (\text{'u'}, \text{'t'}, (\text{'nn'}, \text{'ü'}, \varepsilon, \text{'u'})) \\
 & & & \\
rge(\text{'unn'}, \text{'t'}, (\varepsilon, \text{'ü'}, \varepsilon, \text{'u'})) & = & rgl & (\text{'unn'}, \text{'t'}, (\varepsilon, \text{'ü'}, \varepsilon, \text{'u'})) \\
 & \cup & rgr & (\text{'unn'}, \text{'t'}, (\varepsilon, \text{'ü'}, \varepsilon, \text{'u'})) \\
 & \cup & & \{(\varepsilon, \text{'ü'}, \varepsilon, \text{'u'})\} \\
 & & & \\
rge(\text{'un'}, \text{'t'}, (\text{'n'}, \text{'ü'}, \varepsilon, \text{'u'})) & = & rgl & (\text{'un'}, \text{'t'}, (\text{'n'}, \text{'ü'}, \varepsilon, \text{'u'})) \\
 & \cup & rgr & (\text{'un'}, \text{'t'}, (\text{'n'}, \text{'ü'}, \varepsilon, \text{'u'})) \\
 & \cup & & \{(\text{'n'}, \text{'ü'}, \varepsilon, \text{'u'})\} \\
 & & & \\
rgl(\text{'unn'}, \text{'t'}, (\varepsilon, \text{'ü'}, \varepsilon, \text{'u'})) & = & rge & (\text{'un'}, \text{'t'}, (K, \text{'ü'}, \varepsilon, \text{'u'})) \\
 & & & \\
rge(\text{'un'}, \text{'t'}, (K, \text{'ü'}, \varepsilon, \text{'u'})) & = & rgl & (\text{'un'}, \text{'t'}, (K, \text{'ü'}, \varepsilon, \text{'u'})) \\
 & \cup & rgr & (\text{'un'}, \text{'t'}, (K, \text{'ü'}, \varepsilon, \text{'u'})) \\
 & \cup & & \{(K, \text{'ü'}, \varepsilon, \text{'u'})\}
\end{array}
$$

Für das oben genannte Beispiel werden auf diese Weise unter anderem die folgenden Regelkandidaten generiert:

$$(\varepsilon, \text{ü}, \varepsilon, \text{u}), (\text{n}, \text{ü}, \varepsilon, \text{u}), (\varepsilon, \text{ü}, \text{t}, \text{u}), (\text{n}, \text{ü}, \text{t}, \text{u}),$$
$$(\text{K}, \text{ü}, \varepsilon, \text{u}), (\varepsilon, \text{ü}, \text{K}, \text{u}), (\text{K}, \text{ü}, \text{K}, \text{u}).$$

Für die Regelkandidaten einer aktuellen Wortform **a** und der zugehörigen historischen Wortform **h** wird ein Tupel $(\mathbf{a}, \mathbf{h}, cf, (f, s, b, d))$ für jeden Regelkern gebildet. Die Menge dieser Tupel für alle Tripel aus H bildet die Menge der Trainingsinstanzen E.

4.3.3. Beschneidung der Regeln

Die Generierung der endgültigen Transformationsregeln kann als Klassifizierungsaufgabe betrachtet werden, bei der zwischen "korrekten" und

"nicht korrekten" Regeln zu differenzieren ist. Die Menge E von Instanzen $e_i = (\mathbf{a}_i, \mathbf{h}_i, cf_i, (f_i, s_i, b_i, d_i)) \in E$ mit den Regelkandidaten enthält die positiven Beispiele.

Zum Beschneiden des Baumes muss die Fehlerrate abgeschätzt werden, die sich durch diesen Schritt ergibt. Dazu sind negative Beispiele notwendig, wofür fehlerhafte Belege manuell konstruiert werden müssten. Dabei wäre eine Abschätzung erforderlich, welche Schreibvarianten vorkommen können und welche zu abstrakt sind. Diese Entscheidung ließe sich nur schwer treffen. Deswegen ist es besser, die negativen Beispiele mit Hilfe der Regelkerne zu generieren. Die negativen Beispiele bestehen aus den Wörtern $\mathbf{a}$ in E, bei denen eine Regel angewendet werden kann, aber nicht generiert wurde. Dies ist nur eine Approximation. Allerdings entfällt hierbei der große manuelle Aufwand für die Erstellung der negativen Beispiele.

Um die negativen Beispiele zu generieren, werden die Regelkerne auf die Belege angewendet. Ausgangsbasis ist die Annahme, dass die Belege aus den Dokumenten der Testmenge vollständig vorliegen. Jedes Paar, das durch die Anwendung der Regelkerne auf die aktuelle Schreibungen entsteht und nicht Teil der Belege ist, stellt somit ein negatives Beispiel dar.

Beispielsweise erzeugen der folgende abstrakte Beleg

$$ab - aab$$

und die zugehörige Regel

$$a \rightarrow az$$

ein negatives Beispiel, falls das Wort az im Dokument enthalten ist, aber $ab - azb$ kein Beleg ist.

Um aus den Trainingsdaten einen guten Satz von Transformationsregeln zu generieren, wurde eine Erweiterung des PRISM-Algorithmus (s. Abbildung 4.1) aus dem Bereich des Data Mining entwickelt [Cendrowska, 1987]. PRISM gehört zu den Abdeckungsalgorithmen. Er nimmt an, dass eine Menge von Instanzen in eine Menge von Klassen einzuteilen ist. Im vorliegenden Fall würde das eine Unterteilung in korrekte und inkorrekte Transformationsregeln bedeuten. Eine feste Menge von Attributen mit Werten aus einer Nominalskala beschreibt die Instanzen. Für jede Klasse C versucht der Algorithmus, eine Menge von Regeln mit hohen Precision-Werten zu finden, um die Instanzen zu identifizieren, die zur Klasse C gehören.

Die Anwendung des PRISM Algorithmus ist aus zwei Gründen nicht direkt möglich:

1. Es werden in dieser Arbeit keine perfekten Regeln angestrebt, weil diese zu speziellen Regeln für jede einzelne Wortform führen würden. Somit könnten

```
Für jede Klasse C
  E mit der Instanzmenge initialisieren
  Solange E Instanzen der Klasse C enthält
    Eine Regel R mit leerer linker Seite anlegen, die Klasse C vorhersagt
    Bis R perfekt ist (oder keine weiteren Attribute vorliegen), do
      Für jedes nicht in R erwähnte Attribut A und jeden Wert v
        Prüfen, ob die Bedingung A=v in die linke Seite von R aufgenommen
        werden sollte
        Auswahl von A und v, um die Genauigkeit p/t zu maximieren
      in Zweifelsfällen Entscheidung durch Auswahl der Bedingung mit dem
      größten p)
    A=v zu R hinzufügen
  Die von R abgedeckten Instanzen aus E entfernen
```

Abbildung 4.1.: PRISM-Algorithmus [Cendrowska, 1987]

neue Wörter (mit Ausnahme von Komposita) nur bedingt gefunden werden, weil praktisch ein Wörterbuch erstellt würde. Auf diese Weise würde eine sehr hohe Precision, aber nur ein sehr niedriger Recall erreicht.

2. Anstelle einer Menge von Attributen mit einer festen Menge von nominalen Werten ist eine theoretisch unendliche Menge von Regeln gegeben, weil die Menge der Regeln von der Anzahl und der Komplexität der Trainingsdaten abhängt. Deswegen erfolgt die Entwicklung von Regelkandidaten auf Basis von Beispielen, während PRISM Regeln unabhängig von Beispielen generiert und testet. Das Hinzufügen von Bedingungen zu einer Regel führt bei PRISM zu einer spezielleren Regel. Im vorliegenden Fall existieren auch Generalisierungs- und Spezialisierungsbeziehungen zu den vorhergehenden Regeln, die man für die Suche auswerten kann. Diese würden bei einer Nutzung von PRISM unberücksichtigt bleiben.

Basierend auf diesen Überlegungen entstand ein Beschneidungs-Algorithmus, der aus der Menge E der Regelkandidaten die endgültige Regelmenge F generiert. Ähnliche Algorithmen wurden bereits für die Textkategorisierung vorgeschlagen (s. z. B. [Cohen und Singer, 1999]). Allerdings berücksichtigen sie nicht die Generalisierungshierarchie auf den Regelbedingungen.

Als zusätzliche Parameter für diesen Algorithmus ist die Spezifizierung von zwei Grenzwerten erforderlich: $q_{\min}$ bezeichnet auf Basis der Token die minimale Anzahl der korrekten Regelanwendungen und $p_{\min}$ gibt die minimale Precision für die zu berücksichtigenden Regeln an.

Angenommen, man hat eine boolsche Funktion $match(r, \mathbf{a})$, die prüft, ob ein Wort $\mathbf{a}$ die rechte Seite einer Regel r erfüllt. Basierend auf dieser Funktion ist der teuerste Schritt des Algorithmus die Suche nach allen Wortformen, bei denen $match()$ wahr ergibt. Um diesen Prozess zu beschleunigen, werden die Instanzen zunächst anhand

der Regeln sortiert. Dadurch ist die Durchführung der Suche lediglich einmal pro Regel notwendig (und nicht einmal pro Instanz).

Im Folgenden bestimmt E_i die Menge der Regelkandidaten zu einem Beleg. m_i zählt die Häufigkeit aller Anwendungen einer Regel r_i. q_i bezeichnet die Anzahl der positiven Vorkommen der Regel r_i und p_i gibt die Precision der Regel r_i an.

Für jede Trainingsinstanz $e_i \in E_i$ gilt

$$\begin{aligned} E_i &= \{r \mid r \in E \wedge r = (\mathbf{a},\ \mathbf{h},\ cf,\ (f_i,\ s_i,\ b_i,\ d_i))\} \\ m_i &= \sum_{e_j \in E_i \wedge match(e_i,\ \mathbf{a}_j)} cf_j \\ q_i &= \sum_{e_j \in E_i} cf_j \\ p_i &= \frac{q_i}{m_i} \end{aligned}$$

Entferne alle Instanzen e_i aus E mit $p_i < p_{\min} \vee q_i < q_{\min}$.

Sei $F = \emptyset$.

while $E \neq \emptyset$ do

1. Für Instanzen mit dem höchsten p-Wert wähle die mit den höchsten q-Werten und von diesen wird eine gewählt, für die keine allgemeinere Instanz in E_i enthalten ist. Sei e_i diese Instanz.
2. $F = F \cup (e_i,\ p_i)$.
3. Entferne alle Instanzen aus E_i, bei denen e_i angewendet werden kann: Sei

$$\begin{aligned} D = \{&e_j \mid e_j \in E_i \wedge e_j \neq e_i \wedge e_j = (\mathbf{a},\ \mathbf{h},\ cf,\ (f',\ s,\ b',\ d)) \\ &\wedge\ e_i = (\mathbf{a},\ \mathbf{h},\ cf,\ (f,\ s,\ b,\ d))\} \end{aligned}$$

   Dann setze $E := E - D$.

od.

Nachfolgend wird die Vorgehensweise des Beschneidungs-Algorithmus anhand der in Tabelle 4.1 dargestellten Regelkandidaten verdeutlicht. Zu Beginn des Beschneidungs-Verfahrens legt der Benutzer die Grenzwerte z. B. auf $p_i = 0,45$ und $q_i = 5$ fest. Daraufhin werden die Regelkandidaten $\varepsilon \rightarrow h$ und $y \rightarrow hy$ entfernt, die diese Grenzwerte nicht erfüllen. Danach werden die ersten beiden Kandidaten betrachtet, da sie die höchste Precision haben. Beide weisen die gleiche Häufigkeit auf. Deswegen bekommt mit dem Regelkandidaten $\ddot{u} \rightarrow u$ der allgemeinere den Vorzug. Der Regelkandidat $n\ddot{u} \rightarrow nu$ wird daraufhin entfernt, weil die erste Regel bei ihm angewendet werden kann. Im nächsten Durchlauf wird der Regelkandidat $t \rightarrow th$ ausgewählt, da er nun die höchste Precision hat. Infolgedessen können die Regelkandidaten $e \rightarrow he$ und $a \rightarrow ha$ gelöscht werden, wenn die ihnen zugrundeliegenden Belege durch die zuvor ausgewählte Regel bereits abgedeckt sind.

Regelkandidat			p_i	q_i	
ü	→	*u*	0,69	21	(1)
nü	→	*nu*	0,69	21	(2)
ε	→	*h*	0,13	251	(3)
t	→	*th*	0,67	116	(4)
e	→	*he*	0,64	23	(5)
a	→	*ha*	0,65	34	(6)
y	→	*hy*	0,50	2	(7)

Tabelle 4.1.: Beispiel Regelkandidaten für die Beschneidung

4.3.4. Testkollektion

Der experimentelle Teil dieser Arbeit basiert größtenteils auf einer im *RSNSR*-Projekt (s. Abschnitt 1.3) erstellten Testkollektion. Diese besteht zum größten Teil aus Dokumenten der *Nietzsche Rezeption*[10] aus den Jahren 1865 - 1945. Ergänzt wurden sie durch Texte aus den nachfolgenden kleineren Kollektionen: Die *Bibliotheca Augustana*[11] wurde an der Hochschule Augsburg zusammengestellt. Sie umfasst historische Dokumente aus dem europäischen Sprachraum. Darin sind auch Dokumente aus der deutschen Sprache vom 8. bis 20. Jahrhundert vertreten. Die Texte entstammen aus unterschiedlichen Bereichen wie Politik und Literatur. Volltexte aus den archivpädagogischen Veröffentlichungen des Hessischen Staatsarchives sind im *Digitalen Archiv Hessen-Darmstadt*[12] enthalten. Es stehen Texte aus verschiedenen Themenbereichen wie z. B. der Zeit des Absolutismus zur Verfügung. Im *documentArchiv.de*[13] sind historische Dokumente der deutschen Geschichte ab 1800, wie z. B. des Heiligen Römischen Reiches deutscher Nation und des Rheinbundes, enthalten.

Die aus den Kollektionen ausgewählten 95 Texte sind zwischen dem 15. und 19. Jahrhundert entstanden. Die Kollektion (s. Tabelle 4.2) enthält 534.827 Token und 51.721 Types. Auf Basis der Testkollektion wurde eine Belegdatenbank mit 12.385 Wortpaaren aufgebaut.

Dokumente	15. - 19. Jahrhundert
Tokens	534827
Types	51721
Trainingsmenge	478
Regeln	65

Tabelle 4.2.: Daten zur Testkollektion

[10] `http://www2.inf.uni-due.de/Studienprojekte/Nietzsche/pp2001/die_cd/die_cd.htm` (letzter Aufruf 27.08.10)
[11] `http://www.hs-augsburg.de/~harsch/augustana.html` (letzter Aufruf 18.12.12)
[12] `http://www.digada.de/index.html` (letzter Aufruf 18.12.12)
[13] `http://www.documentarchiv.de` (letzter Aufruf 18.12.12)

4.3.5. Evaluierung

Zur Evaluierung wird der entwickelte Ansatz mit den manuell gebildeten Regeln von Pilz und dem Variantgraph von Biella und Kollegen (s. Abschnitt 3.3) verglichen [Ernst-Gerlach und Fuhr, 2006]. Die manuellen Regeln werden nur auf Basis der Nietzsche-Kollektion erstellt, während die Regeln der anderen beiden Ansätze für die ganze Testkollektion (s. Abschnitt 4.3.4) entwickelt wurden. Zum Zeitpunkt der Evaluierung war die Testkollektion noch im Aufbau. Sie enthält 64.290 Tokens mit 11.326 Types. Nach einer Rechtschreibprüfung der Types, gefolgt von einer manuellen Überprüfung der markierten Wörter, sind 717 Wörter in historischer Schreibung.

Zwei Drittel der Belege finden als Trainingsinstanzen Verwendung. Das verbliebene Drittel bildet die Testmenge. Die Recall- und Precision-Werte basieren auf der Kollektionshäufigkeit der gefundenen Vollformen. Die Ergebnisse der Anwendung der drei Verfahren auf die Testmenge zeigt Tabelle 10.1. Abbildung 4.2 enthält den zugehörigen Recall-Precision-Graph.

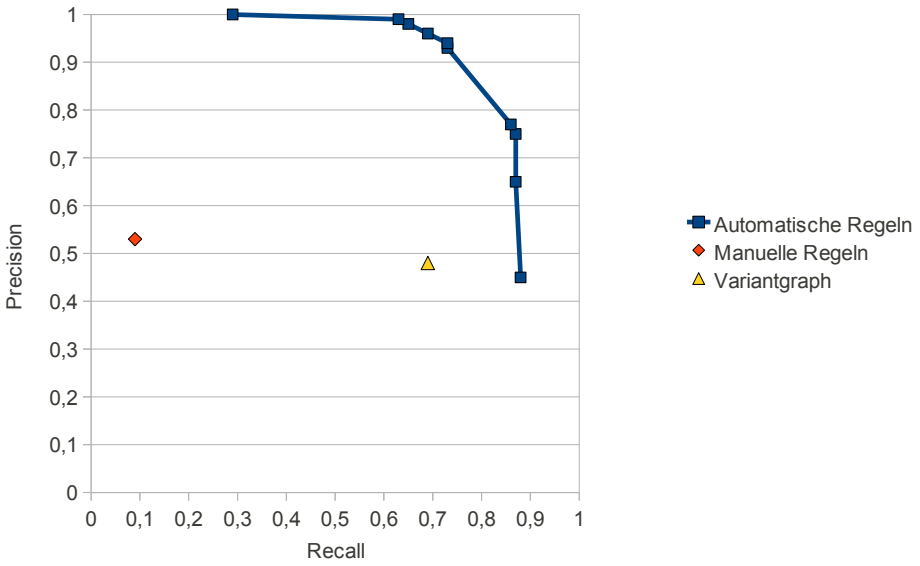


Abbildung 4.2.: Recall-Precision-Graph für automatische und manuelle Regeln sowie den Variantgraph

Die manuell gebildeten Regeln erzielen schlechte Ergebnisse, da sie eine Precision von 0,53, aber nur einen Recall von 0,09 erreichen. Erschwerend zu den schlechten Ergebnissen kommt der hohe intellektuelle Aufwand bei der Regelerstellung hinzu. Da nicht einmal die erwarteten hohen Precision-Werte für die zur Verfügung stehenden Regeln erreicht werden, ist dieser Ansatz für Retrieval in historischen Texten wenig geeignet.

Der Variantgraph erreicht eine vergleichbare Precision, aber der Recall-Wert ist mit 0,69 deutlich höher. Dieser Ansatz findet aber immer noch ungefähr jedes dritte Wort nicht. Das ist kein zufriedenstellendes Ergebnis, auch wenn nur ein Teil der Wörter in nicht standardisierter Schreibung vorkommt. Ein weiterer Nachteil beim Einsatz des Variantgraph ist die hohe Anzahl an generierten Schreibvarianten, z. B. 18 in Abbildung 3.4, wodurch die Retrievalzeit wesentlich erhöht wird.

Der erste Versuch mit automatisch generierten Regeln erreicht eine Precision von 0,45 und einen Recall von 0,88. Im Vergleich zu den anderen beiden Ansätzen ist die Precision etwas schlechter. Dafür ist der Recall 28 % höher als mit dem Variantgraph.

Mit den automatisch generierten Regeln ist es einfach, die Precision-Werte für einzelne Regeln zu betrachten, weil sie auf der Kollektionshäufigkeit der Vollwortformen und den falschen positiven Ergebnissen, bei denen die Regeln zum Einsatz kommen, basieren.

Die Tabelle 4.3 zeigt einige häufig eingesetzte Regeln mit ihren zugehörigen Precision-Werten.

Regeln	**Kontext**	**Häufigkeit**	**Precision**	**Beispiel**
$t \rightarrow th$		116	0,67	Einteilung - Eintheilung
$\ddot{a} \rightarrow ae$	post: K	42	0,98	Ämter - Aemter
$s \rightarrow ß$		35	0,62	aus - auß
$k \rightarrow c$		32	0,8	Kollegien - Collegien
$\ddot{u} \rightarrow ue$		20	0,69	Übertragung - Uebertragung
$\ddot{a} \rightarrow ai$		18	1,0	souverän - souverain

Tabelle 4.3.: Häufig eingesetzte Regeln

Der erste Regelsatz wurde mit den Parametern $p_{\min} = 0,0$ und $q_{\min} = 1$ generiert. Durch die Verwendung von höheren Schwellenwerten für $p_{\min}$ im Beschneidungs-Algorithmus ist es möglich, die Precision-Werte für die Testmenge zu verbessern. Tabelle 4.4 und Abbildung 4.2 zeigen die entsprechenden Ergebnisse. Ein sinnvoller Schwellenwert scheint 0,4 zu sein, weil der Recall-Wert sich nur leicht von 0,88 auf 0,86 verschlechtert, während die Precision von 0,45 auf 0,77 steigt. Der Precision-Wert ist somit deutlich höher als bei den anderen beiden Ansätzen.

$\mathbf{p}_{\min}$	0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0
Recall	0,88	0,87	0,87	0,87	0,86	0,73	0,73	0,69	0,65	0,63	0,29
Precision	0,45	0,65	0,65	0,75	0,77	0,93	0,94	0,96	0,98	0,99	1,00

Tabelle 4.4.: Recall und Precision für unterschiedliche Schwellenwerte $p_{\min}$

Der vorgestellte Ansatz zur automatischen Regelgenerierung übertrifft in der Evaluierung die Resultate für die bisherigen Methoden deutlich. Das entwickelte

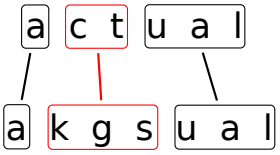


Abbildung 4.3.: Beispiel 1 zur Bildung des Regelkerns

Verfahren ist somit der richtige Weg, um eine hohe Qualität bei der Generierung von historischen Schreibungen zu erreichen.

4.4. Zusammenfassung

Die Entwicklung eines Verfahrens zur automatischen Regelgenerierung hat die Grundlage für einen schnellen Einsatz der regelbasierten Suchmaschine bei neuen Kollektionen geschaffen. Dabei werden dem Top-down-Ansatz (s. Abschnitt 4.1) entsprechend aus dem Regelkern speziellere Regelkandidaten gebildet, bevor in der Beschneidungsphase die Auswahl der geeigneten Regeln erfolgt.

Die Unterschiede bei der Bildung von Ersetzungsregeln zwischen dem Ansatz von Koolen und Kollegen (s. Abschnitt 3.3.1) und der im Rahmen dieser Arbeit entwickelten Vorgehensweise können auf die jeweils bearbeiteten Sprachen zurückzuführen sein. Im Niederländischen kommen deutlich häufiger lange Sequenzen von Vokalen oder Konsonanten vor. Ein weiterer Unterschied in den Ergebnissen kann an der unterschiedlichen Einordnung der Belege liegen, weil einige der niederländischen 1:1-Ersetzungen im vorgestellten Ansatz als Vokabellücke eingeschätzt werden.

Im Gegensatz zum Alignment bei Brill und Moore (s. Abbildung 3.5) werden beim regelbasierten Ansatz nicht einzelne Editoperationen betrachtet, sondern größtmögliche Wortteile zusammengefasst (s. Abbildung 4.3). Dabei bilden jeweils Wortteile ohne direkte Entsprechung in der Variante den Regelkern. Für das Trainingspaar *akgsual* - *actual* bildet $kgs \rightarrow ct$ den Regelkern. Dieser ist auch in den Beispielersetzungen bei Brill und Moore (s. Tabelle 3.4) enthalten. Dabei geht der regelbasierte Ansatz, im Unterschied zu Brill und Moore, davon aus, dass der Regelkern nicht im Kontext anderer Regelkerne enthalten ist (s. Abbildung 4.4 und 4.5). Die anderen Regelkerne stellen somit neben Wortanfang und Wortende eine weitere Grenze für den Regelkern dar.

Sowohl der regelbasierte Ansatz als auch der Ansatz von Brill und Moore lernen die Ersetzungen auf Basis von Trainingsdaten und Gewichten anhand der Häufigkeit der Ersetzungen. Der wesentliche Unterschied besteht in den Substitutionen bzw. Regeln, die aus den jeweiligen Erstellungsmethoden resultieren.

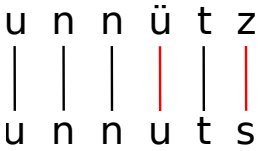


Abbildung 4.4.: Beispiel 2 Alignment von Brill und Moore

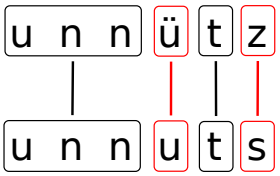


Abbildung 4.5.: Beispiel 2 zur Bildung des Regelkerns

Bei Bollmann und Kollegen (s. Abschnitt 3.3.3) werden durch die Verwendung von Identitätsregeln viele Regeln gebildet, die das Ausgangswort unverändert lassen. Deswegen modifiziert keine der Regeln mit dem höchsten Ranking das Eingabewort [Bollmann u. a., 2011]. Die starre Festlegung des Kontextes auf ein Zeichen ist bei der Verwendung des Ansatzes zur Erstellung von historischen Varianten hinderlich. Auf diese Weise können auch sehr häufige Substitutionsregeln wie beispielsweise $ss \rightarrow ß$ nicht ohne Kontext erstellt werden. Stattdessen muss für jeden Kontext eine eigene Regel erstellt werden. Auf der anderen Seite kann die Precision der Regeln nicht durch die Verwendung von mehr Kontextinformationen gesteigert werden. Der in dieser Arbeit vorgestellte Ansatz passt dagegen die Größe des verwendeten Kontextes flexibel an den Bedarf an. Es ist auch möglich, zu einem Regelkern mehrere Regeln mit unterschiedlichem Kontext zu bilden. Eine Bearbeitung der Regeln durch den Benutzer ist von Bollmann und Kollegen zwar nicht vorgesehen, würde aber durch die fehlende Berücksichtigung der Verknüpfungen zwischen den Regeln erschwert.

Die Evaluierung hat gezeigt, dass der vorgestellte regelbasierte Ansatz eine Korrektur des Suchbegriffs gemäß dem Noisy Channel Model ermöglicht. Im folgenden Abschnitt wird darauf aufbauend eine probabilistische Suchmaschine für historische Dokumente entwickelt. In diese wird das Modul zur Generierung von historischen Varianten für Suchbegriffe integriert. Nur durch Retrievalexperimente mit historischen Kollektionen kann die eigentliche Qualität der vorgestellten Methode beurteilt werden.

5. Suchmaschine für historische Dokumente

Nachdem der Benutzer Regeln für sein Korpus generiert hat, sollen diese Retrieval auf den eigenen Dokumenten ermöglichen. Dazu wird im Kontext dieser Arbeit eine Suchmaschine entwickelt. Eine Evaluierung soll die Retrievalqualität des zu erstellenden Systems verdeutlichen. Am Beispiel der Google-Buchsuche wird nachfolgend gezeigt, dass der regelbasierte Ansatz auch bei einer Internetsuchmaschine einsetzbar ist.

5.1. Suchmaschine

Die geschichtete Architektur der entwickelten Suchmaschine, die sich aus der angestrebten Vorgehensweise ergibt (s. Abschnitt 4.2), ist in Abbildung 5.1 dargestellt. Die vierte Schicht bildet der eingegebene Suchbegriff. Er wird in die speziell für die Suche nach historischen Dokumenten entwickelte Benutzeroberfläche in der Grundform eingegeben. Die darauf folgende Ebene erzeugt die Flexionsformen mit Hilfe eines geeigneten Werkzeugs. Die zweite Schicht, der Kern der Suchmaschine, bildet jede einzelne moderne Flexionsform durch die Anwendung der generierten Regeln auf die historischen Varianten ab. Die gebildeten Wortformen der beiden Schichten werden von der Benutzeroberfläche angezeigt. Die erste Schicht benutzt anschließend eine Standardsuchmaschine, um mit der um die historischen Formen erweiterten Anfrage Dokumente in der digitalen Bibliothek zu finden. Die Resultatliste wird anschließend in der Benutzeroberfläche angezeigt. Durch Filterung der Varianten kann die Suchanfrage entsprechend modifiziert werden und die Resultatliste wird aktualisiert.

Ein Beispiel für den Ablauf der Suche zeigt Abbildung 5.2. Der Benutzer gibt den Suchterm *ruhen* in die Benutzeroberfläche ein. Die nächste Schicht generiert für diesen Suchbegriff unter anderem die Wortformen *geruht*, *ruhte*, *ruhn*, *ruhen*, *ruhten*. Für jede Wortform werden die historischen Varianten generiert. Beispielsweise wird durch Anwendung der Regel $t \rightarrow et$ auf die Flexionsform *geruht* die Variante *geruhet* generiert. Mit den gebildeten historischen Varianten wird die Anfrage an die Suchmaschine (Ebene 1) erweitert, in der digitalen Bibliothek nach Dokumenten gesucht und die Resultatliste zurückgegeben.

Aufgrund der flexiblen Struktur kann eine Integration des vorgestellten Ansatzes durch den Aufruf der Schichten 2 und 3 auch in andere Suchmaschinen erfolgen. Nachstehend werden die einzelnen Schichten genauer erläutert.

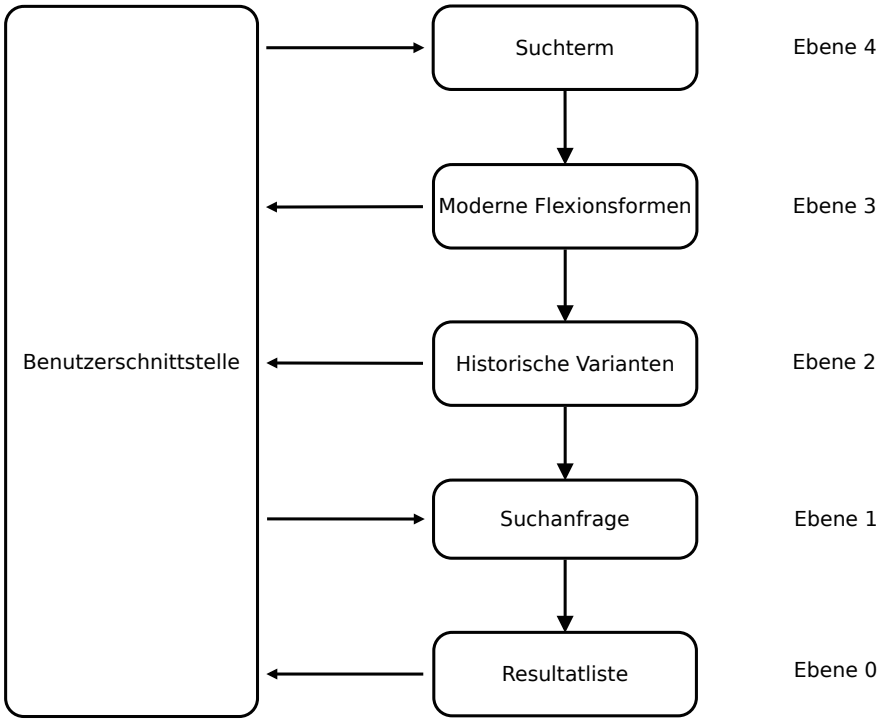


Abbildung 5.1.: Architektur der Suchmaschine

5.1.1. Benutzeroberfläche

Viele Suchmaschinen unterstützen den Benutzer bei der Eingabe des Suchbegriffs durch eine Rechtschreibkontrolle. So entwickelte z. B. Jordan eine proaktive Anfrageunterstützung für das Daffodil-System (s. Abbildung 5.3) [Jordan, 2005]. Analog zur Darstellung von möglichen Fehlerstellen in Textverarbeitungsprogrammen werden wahrscheinlich fehlerhaft geschriebene Suchterme rot unterstrichen. Darüber hinaus bietet das System Vorschläge für alternative Schreibweisen an.

Bei der Erstellung der Benutzeroberfläche ist auf eine einfache Bedienung zu achten, weil Benutzer aus dem Bereich der Humanwissenschaften oftmals Probleme mit komplexeren Oberflächen haben und diese demzufolge nicht nutzen würden [Bates, 2002]. Abbildung 5.4 zeigt eine Benutzeroberfläche für eine unscharfe Suche in einer pharmazeutischen Enzyklopädie. Sie stellt dar, nach welchen Varianten gesucht

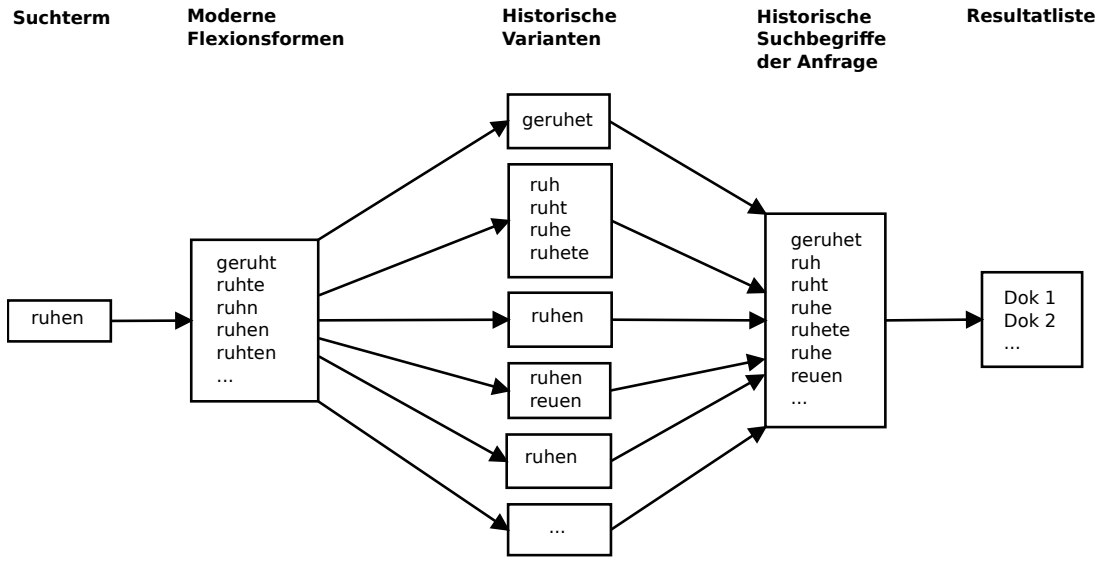


Abbildung 5.2.: Beispiel für den Ablauf der Suche

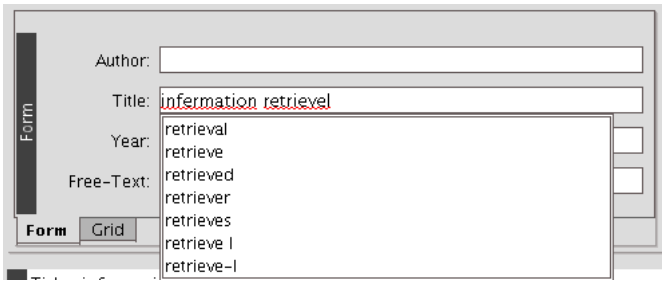


Abbildung 5.3.: Proaktive Vorlagefunktion in Daffodil [Klas, 2007, S. 76]

wurde und ermöglicht eine nachträgliche Filterung der Ergebnisse anhand der Varianten [Esser, 2004].

Bei der Entwicklung von Benutzeroberflächen ist sicherzustellen, dass der Benutzer die Kontrolle behält und sich das System wie erwartet verhält [Bates, 1990; Shneiderman und Maes, 1997]. Der Benutzer weiß oftmals nicht genau, wie er seine Ziele erreichen kann. Deswegen muss die Benutzeroberfläche ihn dabei unterstützen, z. B. indem die Formulierung der Anfrage durch das System erleichtert wird [Hearst, 1999].

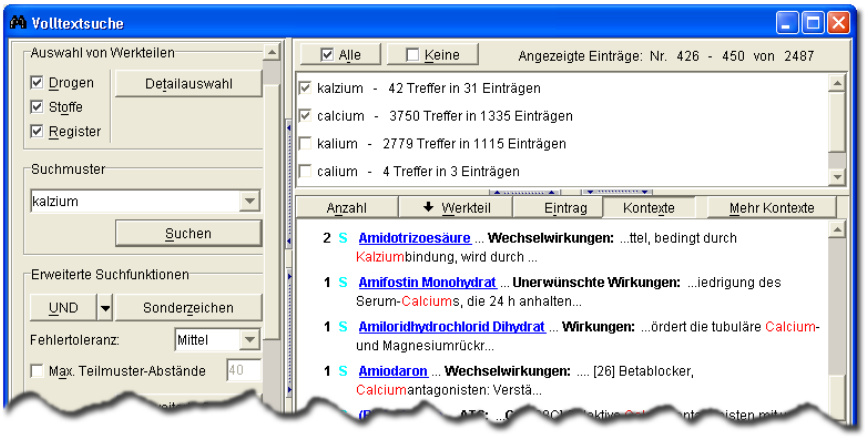


Abbildung 5.4.: Benutzeroberfläche für fehlertolerante Suche in digitalen multilingualen Enzyklopädien [Esser, 2004]

Bates unterscheidet die folgenden Stufen bei der Systembeteiligung [Bates, 1990]:

0. Keine Systembeteiligung
1. Auflistung von möglichen Vorgehensweisen
2. Ausführung von Vorgehensweisen auf Befehl
3. Beobachten der Suche und empfehlen
   a) Nur wenn der Suchende Vorschläge anfordert
   b) Immer, wenn der Bedarf gesehen wird
4. Automatische Ausführung
   a) Mit Information des Suchenden
   b) Ohne Information des Suchenden

Weil es sich um eine spezielle Suchmaschine für historische Dokumente handelt, wird in der entwickelten Benutzeroberfläche (s. Abbildung 5.5) die automatische Ausführung (Stufe 4 nach Bates) eingesetzt. Der Benutzer wird über die generierten Varianten, mit denen die Suchanfrage erweitert wurde, informiert. Er bekommt wie bei Esser die Gelegenheit, das Suchergebnis auf bestimmte Varianten einzuschränken und wird so, wie von Hearst empfohlen, bei der Formulierung der Anfrage unterstützt. Auf diese Weise behält er die Kontrolle über das Suchergebnis.

Die Benutzeroberfläche teilt sich in eine einfache Suche und eine erweiterte Suche auf. Dadurch können die unterschiedlichen Bedürfnisse der Benutzer berücksichtigt werden. Die einfache Suchoberfläche erlaubt eine Suche nach Titel, Autor, Jahr und Freitext. Bei den erweiterten Suchoptionen kann der Benutzer zusätzlich das minimale Regelgewicht, die maximale Anzahl an gebildeten Formen und

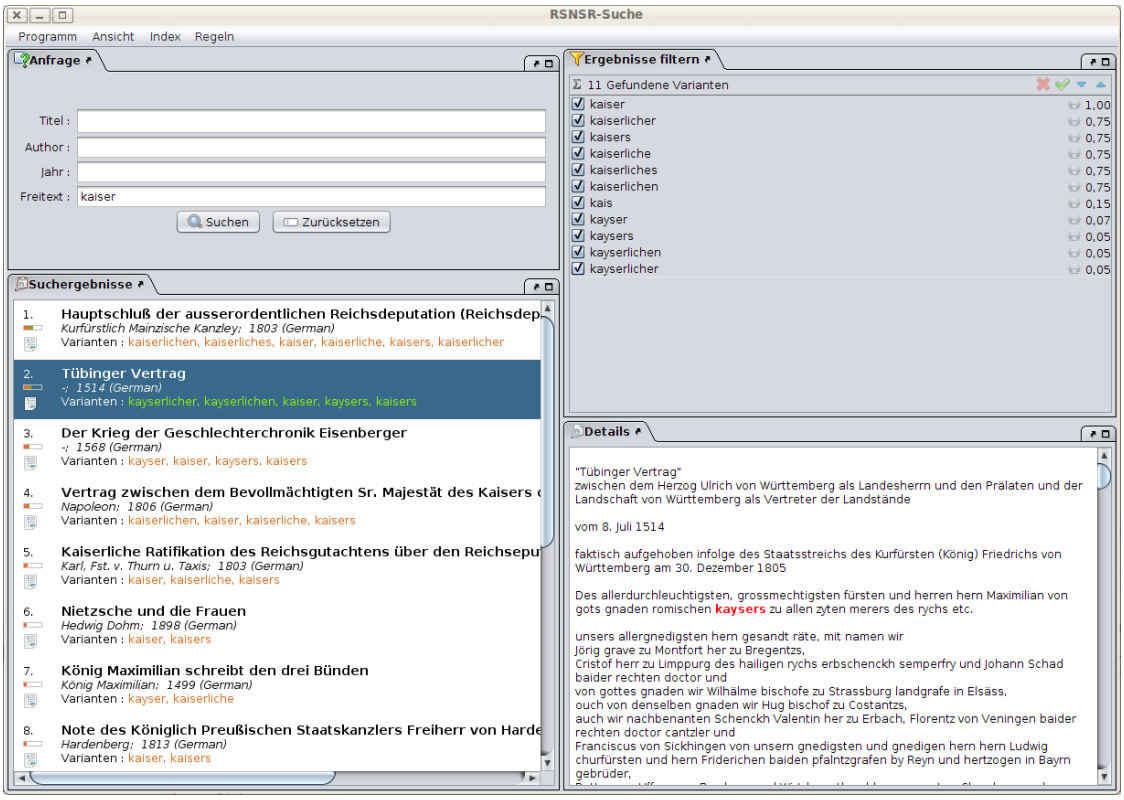


Abbildung 5.5.: Benutzeroberfläche der Suchmaschine

die maximale Anzahl an Regeldurchläufen angeben. Darüber hinaus kann der Benutzer entscheiden, ob er nach Flexionsformen oder in und nach Komposita suchen möchte. Die Benutzeroberfläche der Suchmaschine bietet dem Benutzer die Option, selbst generierte Regeln hochzuladen. Um die Suche auch auf eigenen Volltextkollektionen des Benutzers zu ermöglichen, wurde der Suchmaschine eine Indexierungskomponente hinzugefügt. Mit ihr ist sowohl eine Indexierung kompletter Verzeichnisse als auch von zusammengestellten Texten inklusive Metadaten auf Basis einer CSV-Datei möglich.

Die Anzeige der Schreibvarianten zu einer Suchanfrage erfolgt in einer Liste. Sie können zur Filterung des Suchergebnisses verwendet werden. Bei der Darstellung der Suchergebnisse werden, neben den Metadaten der Dokumente, die in den Dokumenten enthaltenen Schreibvarianten mit angezeigt. Wie bei Suchmaschinen mit Volltexten üblich [Hearst, 1999], werden bei der Anzeige der Dokumente die gefundenen Varianten darüber hinaus hervorgehoben dargestellt. Für das gefundene Dokument stehen die Optionen ”speichern”, ”kopieren” und ”drucken” zur Verfügung.

5.1.2. Bildung der Flexionsformen

Zur Bildung der Flexionsformen existieren im Deutschen einige Werkzeuge, wie z. B. Morphy und die Wortschatz-Datenbank. Das Lexikon von Morphy enthält über 300.000 Wortformen, die auf über 50.000 Wortstämmen basieren [Lezius u. a., 1998]. Das Wörterbuch ist sehr kompakt, weil es nicht die einzelnen Flexionsformen zu jeder Grundform speichert, sondern nur die Flexionsklasse, auf deren Grundlage die Flexionsformen gebildet werden.

Die Wortschatz-Datenbank[14] [Quasthoff, 1998] wurde automatisch aus Texten aufgebaut, die in einem maschinenlesbaren Format vorliegen, wie z. B. aus Zeitungen, wissenschaftlichen Journalen, Fachlexika und Monographien aus verschiedenen Wissensgebieten [Quasthoff und Wolff, 1999]. Die ausgewählten Texte decken sowohl thematisch als auch sprachlich eine große Bandbreite ab. Die Datenbank enthält über 3,7 Millionen Types, die zusammen mit Kollokationen und grammatischen Informationen wie Flexionsformen und Stopwörtern in der Datenbank gespeichert sind. Verweise auf die Wortformen sind ebenfalls enthalten. Obwohl Algorithmen zur Fehlerkorrektur verwendet werden, enthält die Liste mit den Wortformen eine Fehlerrate von 1 - 2 % [Quasthoff, 1998]. Diese Fehler basieren auf den Schreibfehlern in den Quelldokumenten und falsch interpretierten Layoutinformationen bei der Textübernahme.

Die Bildung der Flexionsformen basiert bei der Wortschatz-Datenbank auf zwei Methoden:

- Die erste Methode nimmt an, dass alle Flexionsformen in der Datenbank enthalten sind. Demzufolge werden die möglichen Wortformen für ein Lemma u. a. mit typischen Suffixen gebildet. Wenn das Ergebnis im Wörterbuch enthalten ist, wird davon ausgegangen, dass eine neue Flexionsform gefunden wurde.
- Die Annahme, dass bei Wörtern, die eine ähnliche Struktur haben, die Flexionsformen auch auf ähnliche Weise gebildet werden, bildet die Basis für die zweite Methode. Diese Vorgehensweise wird vor allem für Komposita benutzt. Die Klasse der Flexionsformen eines Kompositums basiert auf dem letzten Wort. Wenn es möglich ist, diesen Teil eines Kompositums korrekt zu identifizieren, können die Flexionsformen leicht aus der Datenbank abgeleitet werden.

Der gesamte Prozess der Bildung von Flexionsformen wird bei der Wortschatz-Datenbank von Algorithmen zur Fehlererkennung begleitet.

Eine Alternative zu diesen Werkzeugen könnte die Nutzung des Suchdienstes canoo[15] sein, der auch Vollformen zur Verfügung stellt. Kurze Tests mit dem Onlineservice sahen vielversprechend aus. Allerdings war es nicht möglich,

[14] `http://wortschatz.uni-leipzig.de/` (letzter Aufruf 18.12.12)
[15] `http://www.canoo.net/` (letzter Aufruf 18.12.12)

eine Forschungslizenz zu bekommen. Morphy kann nur auf Computern mit Windows-Betriebssystemen eingesetzt werden. Weil eine plattformunabhängige Suchmaschine entwickelt werden soll, wird die Wortschatz-Datenbank zur Bildung der Vollformen eingesetzt. Für diese wird zudem ein Webservice[16] angeboten, der einen direkten Zugriff auf die vorhandenen Daten ermöglicht.

Ein besonderes Problem bei der Bildung der Vollformen im Deutschen ist die häufige Verwendung von Komposita. Sie entstehen durch die Kombination verschiedener Wörter (Substantive, Verben, Adjektive oder Adverbien). Häufig werden Fugenlaute zur Verknüpfung von Wörtern genutzt [Wegener, 2005]. Die Bildung von Komposita folgt dabei keinen bestimmten Regeln [Ferber, 2003, S. 45]. Beispiele für Komposita mit Fugenlauten sind *Hundehütte*, *Bootsfahrt* und *Brillenglas*. Aufgrund der hohen Anzahl an Kombinationsmöglichkeiten bei zusammengesetzten Wörtern ist nur ein sehr geringer Teil der Komposita in der Wortschatz-Datenbank enthalten. Deswegen kann die Datenbank in diesem Fall häufig keine Flexionsformen zurückgeben. Demzufolge ist eine Zerlegung des Kompositums in seine Bestandteile erforderlich, um darauf aufbauend die Flexionsformen zu bilden. Außerdem soll es auch möglich sein, einen Suchbegriff zu finden, der in Komposita enthalten ist.

Dafür können geeignete Werkzeuge wie z. B. TAGH[17] und der JWordSplitter[18] eingesetzt werden. TAGH dient der automatischen Bestimmung von deutschen Wortformen und kann in dem Zusammenhang auch Komposita zerlegen [Geyken und Hanneforth, 2006]. Es basiert auf einem Lexikon für Stammformen und Regeln, mit deren Hilfe Wörter auf ihre Wortstämme abgebildet werden. TAGH ist in C++ implementiert und steht sowohl für Windows als auch für Linux zur Verfügung. Der JWordSplitter ist in Java implementiert. Er trennt Komposita in drei Schritten [Abels und Hahn, 2005]. Zunächst wird versucht, eine direkte Dekomposition vorzunehmen, indem mit einer rekursiven Methode eine Suche nach den Morphemen des Wortes erfolgt. Ist eine vollständige Dekomposition des Wortes nicht möglich, wird durch eine schrittweise Trunkation von links nach rechts durch Entfernung des jeweils nächsten Buchstabens nach weiteren Morphemen gesucht. Falls in diesem Schritt auch keine komplette Dekomposition durchgeführt werden kann, wiederholt sich der Ablauf mit einer Trunkation von rechts nach links. Als Eingabe benötigt der WordSplitter eine Wortliste. Anhand dieser Liste wird überprüft, ob eine weitere Zerlegung möglich ist. Aufgrund der sprachunabhängigen Programmierung des JWordSplitter ist lediglich ein Austausch der Wortliste erforderlich, wenn ein Einsatz für eine andere Sprache erfolgen soll.

Die Suchmaschine soll in Java entwickelt werden. Deswegen wird im Folgenden der ebenfalls in Java implementierte JWordSplitter für die Kompositazerlegung eingesetzt. Bei der Indexierung einer neuen Kollektion wird, neben einem herkömmlichen Index, ein zweiter Index aufgebaut. In diesem sind die Terme der

[16] `http://wortschatz.uni-leipzig.de/Webservices/` (letzter Aufruf 18.12.12)
[17] `http://www.tagh.de/` (letzter Aufruf 18.12.12)
[18] `http://sourceforge.net/projects/jwordsplitter/` (letzter Aufruf 18.12.12)

Dokumente nach einer Kompositazerlegung enthalten. Weil bei einem Kompositum das letzte Teilwort für die Bildung der Flexionsform ausschlaggebend ist, können nach der Zerlegung die Flexionsformen in der Wortschatz-Datenbank nachgeschlagen werden. Im Anschluss kann die Bildung der entsprechenden Vollformen erfolgen. Bei einem Suchbegriff wird zusätzlich überprüft, ob es sich um ein Kompositum handelt. Lässt er sich zerlegen, erfolgt auch für die Teilwörter eine Bildung der Varianten und eine entsprechende Erweiterung der Suchanfrage.

5.1.3. Regelanwendung

Nach der Generierung der probabilistischen Regeln kann der Regelsatz in der Suchmaschine eingesetzt werden. Für ein aktuelles Wort **a** sollen alle historischen Schreibungen generiert werden.

Für jedes Element $(r_i, p_i) \in F$ ergibt sich das Wort $\mathbf{a}_i$, wenn die Funktion $match(r_i, \mathbf{a})$, bei der r_i auf **a** angewendet wird, zum Einsatz kommt. Auf diese Weise erfolgt eine Generierung einer Menge von historischen Schreibungen für ein einzelnes Wort **a** durch die Anwendung von Regeln. Offensichtlich haben diese Wortformen nicht alle die gleiche Precision. Deswegen haben die Wörter Gewichte, die die Precision p_i der zur Generierung verwendeten Regeln widerspiegeln [Zobel und Dart, 1996].

Das Verfahren der Regelanwendung startet mit der allgemeinsten Regelwurzel, die keine Elternregeln hat. Kann sie bei dem Suchbegriff angewendet werden, wird geprüft, ob speziellere Regeln existieren, die ebenfalls anwendbar sind. In diesem Fall wird die speziellere Regel verwendet und die entsprechende Precision für das Ranking berücksichtigt.

Die Retrievalmaschine berechnet das Ranking auf die folgende Weise: Für eine Schreibvariante w, die aus einem Suchbegriff t gebildet wurde, wird p als die Wahrscheinlichkeit $p = P(t \rightarrow w)$ interpretiert, mit der t die Schreibvariante w impliziert. Da die Suchmaschine auf Retrieval, und somit auf unsicherem Schließen, basiert, können diese Wahrscheinlichkeiten sehr einfach in den Retrievalprozess integriert werden. Im einfachsten Fall kann die Integration durch binäre Indexierung und Einzeltermanfragen erfolgen. Dann ist p das Gewicht eines Dokumentes, das w enthält.

5.1.4. Suchmaschinenebene

Für die Suchmaschinenebene wurde die weit verbreitete Standardsuchmaschine Apache Lucene[19] eingesetzt. Lucene ist eine quellenoffene Bibliothek für Suchmaschinen. Sie ist in Java implementiert und plattformunabhängig. Das

[19] `http://lucene.apache.org/` (letzter Aufruf 18.12.12)

Framework ermöglicht es, Volltextsuche und -indexierung in eigene Applikationen zu integrieren [Dollinger, 2005].

5.1.5. Zusammenfassung

In Kapitel 4 lag der Fokus auf der automatischen Generierung von Schreibvarianten auf Basis von Belegen, die aus den modernen Flexionsformen und den zugehörigen Schreibvarianten bestehen. Hier wurde der ganze Retrievalprozess beginnend mit der Eingabe des Lemmas in die Benutzeroberfläche bis zur Suchmaschinenebene modelliert und entsprechend implementiert. Die von der Suchmaschine verwendeten Werkzeuge sind in Abbildung 5.6 dargestellt. Nachfolgend wird das Verfahren der Generierung der Schreibvarianten bei Eingabe eines Lemmas evaluiert.

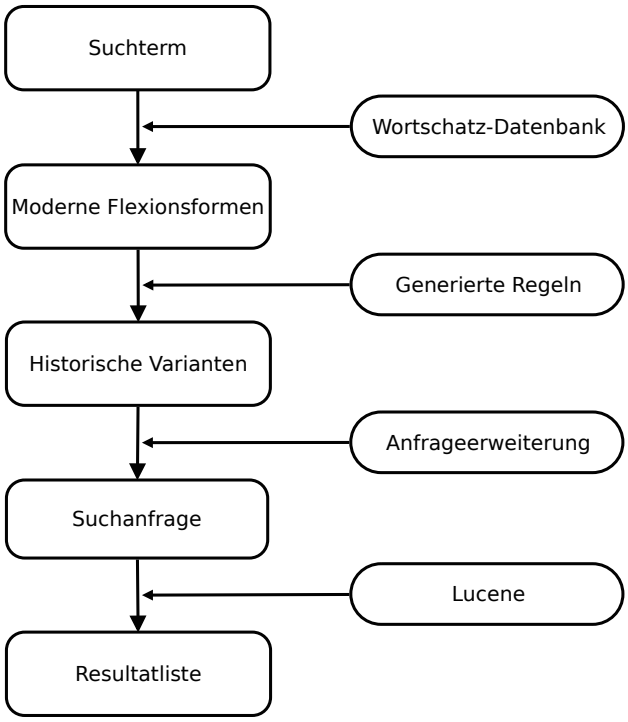


Abbildung 5.6.: Eingesetzte Werkzeuge in der Suchmaschine

5.2. Evaluierung des regelbasierten Ansatzes

Mit dem regelbasierten Ansatz verbessert sich die Retrievalqualität bei der Suche in historischen Dokumenten deutlich. Um dies zu zeigen, erfolgt eine Evaluierung des regelbasierten Ansatzes [Ernst-Gerlach und Fuhr, 2007]. Dabei wird auch die Kombination mit der Bildung von Flexionen überprüft.

Mit der folgenden Evaluierung wird getestet, wie gut die historischen Wortformen für einen Suchbegriff (gegeben als Lemma) zu finden sind. Basierend auf einer kleinen Trainingsmenge von 478 Belegen aus der Belegdatenbank (s. Abschnitt 4.3.4) werden 65 Regeln generiert.

Um die Qualität des Ansatzes zu bestimmen, werden keine klassischen Retrievalexperimente durchgeführt. Diese würden einen relativ hohen Aufwand bedeuten und gleichzeitig wäre es schwierig, Schlussfolgerungen für den Fokus der vorliegenden Forschungsarbeit zu ziehen. Stattdessen werden Einwortanfragen genutzt und dann zum einen der Anteil der gefundenen Wörter, die zum Anfrageterm gehören, bestimmt und zum anderen der Anteil der gefundenen Wörter an vorkommenden Wörtern. Im Folgenden wird der Anteil der Wortformen, die zu der Anfrage gehören, als "relevante Token" bezeichnet. Die zugehörigen Retrievalmaße sind wie folgt definiert:

$$Precision = \frac{|relevante\ Token \cap gefundene\ Token|}{|gefundene\ Token|} \tag{5.1}$$

$$Recall = \frac{|relevante\ Token \cap gefundene\ Token|}{|relevante\ Token|} \tag{5.2}$$

$$F_1 = \frac{2 \cdot p \cdot r}{p + r} \tag{5.3}$$

Wie beim herkömmlichen Recallmaß ist es auch hier schwierig, alle relevanten Token in der Kollektion zu identifizieren. Um dieses Problem zu vermeiden, wird eine ähnliche Technik wie bei der Dokument-Source-Methode [Ferber, 2003, S. 92] verwendet. Für eine zufällige Menge von Token aus der Kollektion wird das Lemma manuell gebildet und als Suchbegriff eingegeben. Anschließend erfolgt eine Überprüfung, ob das Originaltoken gefunden wird. Der Anteil der gefundenen Token ist dann die Recallabschätzung. Beim Retrieval erreicht ein Verfahren häufig den höheren Recall-Wert (r), während ein anderes den höheren Precision-Wert (p) erzielt. Das F-Maß kombiniert beide Werte und erleichtert so den Vergleich von verschiedenen Verfahren [Carstensen u. a., 2009]. Beim hier verwendeten F_1-Maß fließen Recall und Precision gleichmäßig in die Bewertung mit ein.

Als Vergleichsbasis für die Evaluierung kommt eine einfache Lucene-Suchmaschine zum Einsatz. Zur Bildung des Indexes mit Lucene wird die deutsche Variante des Snowball Stammform-Algorithmus[20] verwendet. Diese Kombination wird als Vergleichsbasis für die Wortschatz-Datenbank eingesetzt. Für den zweiten Schritt, die Suche nach Schreibvarianten, dient die Levenshtein-Distanz als Vergleichsbasis. Um die Vergleichbarkeit der Ansätze zu erhöhen, wird sowohl der Stammform-Algorithmus als auch die Wortschatz-Datenbank mit dem

[20] `http://snowball.tartarus.org/` (letzter Aufruf 18.12.12)

regelbasierten Ansatz und der Levenshtein-Distanz getestet. Für die Experimente wird jeweils eine maximale Levenshtein-Distanz von 1 bzw. 2 betrachtet. Die zugehörigen Retrieval-Werte sind in Tabelle 10.2 zu finden.

Für die Evaluierung werden zwei Testmengen gebildet. Für die erste Menge werden zufällig Types aus der Kollektion ausgewählt. Dieser Test simuliert den normalen Suchprozess für eine historische Suchmaschine. Für die zweite Testmenge werden nur Wörter in historischer Schreibung ausgewählt, um die Qualität bezogen auf die Schreibvarianten zu bestimmen. Die folgenden Abschnitte stellen die Ergebnisse für die verschiedenen Testmengen vor.

5.2.1. Evaluierung von Wörtern der gesamten Kollektion

Bei der Auswahl der Wörter für die gesamte Kollektion werden Wörter, die in der Stopwortliste der Wortschatz-Datenbank enthalten sind, nicht berücksichtigt. Die verbliebenen Wörter haben eine Gesamthäufigkeit von 300.532. Für die Evaluierung wird eine repräsentative Auswahl von 200 Wörtern gebildet. Dabei ist die Häufigkeitsverteilung der Types vergleichbar zu denen der gesamten Kollektion, wenn die Vorkommenshäufigkeit der Terme in den Anfragen vergleichbar mit ihrer Kollektionshäufigkeit ist.

Die Lemmata für die Wörter der Testmenge werden manuell gebildet. Dieser Schritt kann nicht mit Hilfe der Datenbank durchgeführt werden, weil darin die historischen Wortformen nicht enthalten sind. Für die Lemmata werden alle zugehörigen Wortformen in der Wortschatz-Datenbank nachgeschlagen und manuell überprüft. Dabei werden die Wortformen nur falsch markiert, wenn sie zu einem anderen Lemma gehören und nicht, wenn sie Schreibfehler enthalten, die z. B. auch historische Varianten sein könnten. Anschließend werden die historischen Formen gebildet. Für jede der gebildeten Varianten, die in der Kollektion enthalten ist, folgt eine Überprüfung, ob sie zum selben Lemma gehört. Die Ergebnisse werden mit denen bei der Verwendung des Stammform-Algorithmus verglichen. Abbildung 5.7 enthält den zugehörigen Recall-Precision-Graph.

Herkömmliche Suchmaschine

Um die Evaluierungsergebnisse für Wörter der gesamten Kollektion besser einordnen zu können, werden zusätzlich die Retrieval-Werte für eine herkömmliche Suchmaschine berechnet. Diese Suchmaschine findet nur die Types, die als Lemma im Text vorkommen. Weil nur Suchbegriffe gefunden werden, die das Dokument ohne Veränderung enthält, wird offensichtlich Precision 1 erreicht. Dagegen wird nur ein Recall von 0,36 erzielt — das bedeutet, es werden zwei von drei Vorkommen eines Suchbegriffes nicht gefunden.

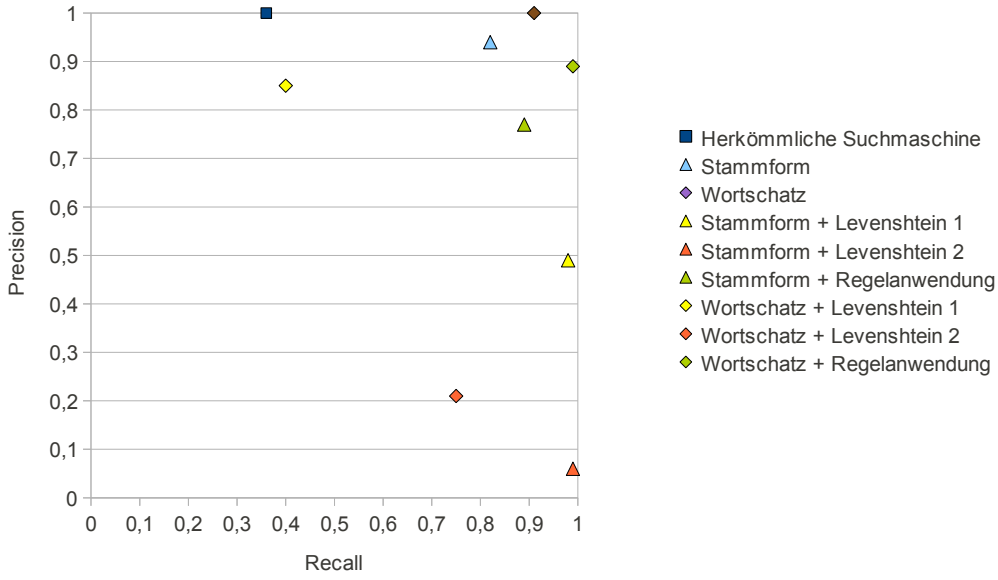


Abbildung 5.7.: Recall-Precision-Graph für die Suche nach Types aus der gesamten Kollektion

Stammformbildung

Wird der Algorithmus zur Stammformbildung benutzt, werden 3.394 verschiedene Wortformen für die Lemmata gebildet. Dabei wird ein Recall von 0,82 und eine Precision von 0,94 erreicht. Obwohl immer noch jedes fünfte Wort nicht gefunden wird, sind dies schon recht gute Ergebnisse.

Wortschatz-Datenbank

Bei der Nutzung der Wortschatz-Datenbank werden für die Testmenge von 200 Wörtern 1.760 unterschiedliche Wortformen für die Lemmata gebildet. Im Durchschnitt werden somit 8,8 Wortformen pro Lemma erzeugt. Nur 10 der Wortformen sind als Fehler einzustufen, wobei nur zwei dieser Wörter in den Dokumenten vorkommen. Die generierten Wortformen haben eine Gesamthäufigkeit von 96.582 in der Kollektion. Dagegen haben die falschen Wortformen nur eine Kollektionshäufigkeit von 121. Somit wird eine Precision von 1 erreicht. Für 11 Types sind keine Wortformen enthalten. Für weitere 28 Wortformen sind die gesuchten Wortformen nicht zu finden. Es werden nur 161 Types (81 %) der Wortformen gefunden. Trotzdem wird, bezogen auf die Termhäufigkeit in der Kollektion, ein Recall von 0,91 erreicht. Somit halbiert sich die Anzahl der nicht gefundenen Wörter gegenüber dem Stammformansatz. Im Vergleich schneidet der Ansatz demzufolge auch beim F-Maß mit 0,95 im Vergleich zu 0,88 besser ab.

Levenshtein-Distanz

In Kombination mit dem Stammformansatz erzielt die Levenshtein-Distanz 1 mit einem Recall von 0,98 ein sehr gutes Ergebnis. Allerdings liegt die Precision nur bei 0,49, so dass jedes zweite Wort keine tatsächliche Variante ist. In Kombination mit der Wortschatz-Datenbank sind die Ergebnisse mit einem Recall von 0,40 und einer Precision von 0,84 genau umgekehrt. Wird die maximale Levenshtein-Distanz auf zwei gesetzt, erhöht sich in Kombination mit dem Stammformansatz der Recall leicht von 0,98 auf 0,99. In Kombination mit der Wortschatz-Datenbank steigert sich der Recall von 0,40 auf 0,75. Dies entspricht fast einer Verdoppelung. Durch die große Anzahl von generierten historischen Wortformen (26.112 für den Stammformansatz und 4.687 für die Wortschatz-Datenbank) sinkt die Precision dramatisch auf 0,06 bzw. 0,21.

Regelanwendung

Bei der Kombination des Stammformansatzes und der Regelanwendung wird ein Recall von 0,89 und eine Precision von 0,77 erreicht. Somit erzielt die Levenshtein-Distanz einen höheren Recall. Dagegen erreicht die Regelanwendung in Kombination mit dem Stammformansatz eine höhere Precision. Nach der Anwendung der Regeln auf die generierten Wortformen aus der Wortschatz-Datenbank wird die Zahl der nicht gefundenen Wortformen von 39 auf 17 reduziert. Auf diese Weise wird ein Recall von 0,99 erreicht, was ein sehr gutes Ergebnis ist. Somit lassen sich die häufigsten Wortformen aus der Kollektion generieren. Der einzige Nachteil ist die Precision, deren Wert von 1,0 auf 0,89 sinkt. Dies liegt vor allem an der geringen Anzahl von Belegen, die für die Regelgenerierung bei die Evaluierung verwendet werden. Insgesamt erreicht diese Kombination mit 0,94 den höchsten F-Wert der hier betrachteten kombinierten Methoden. Für beide Ansätze wird ungefähr dieselbe Recallsteigerung bei der Regelanwendung erzielt.

Anhand des F-Maßes zeigt sich die klare Überlegenheit des regelbasierten Ansatzes. Während mit der Levenshtein-Distanz der beste Wert nur bei 0,65 liegt, erreicht das regelbasierte Verfahren sowohl in Kombination mit dem Stammformansatz (0,83) als auch mit der Wortschatz-Datenbank (0,89) deutlich bessere Werte.

5.2.2. Evaluierung beschränkt auf historische Formen

Für diese Evaluierung werden aktuelle Wortformen und die zugehörigen historischen Schreibvarianten ausgewählt. 17 % der Token der Kollektion sind in historischer Schreibung. Um die Vergleichbarkeit zu erhöhen, findet die Testmenge aus Abschnitt 4.3.5 Verwendung. Die Testmenge enthält 239 unterschiedliche Wörter. Wiederum werden aus manuell gebildeten Lemmata Wortformen generiert und

analysiert. Anschließend werden die Regeln angewendet, um die historischen Varianten zu bilden.[21] Der Recall-Precision-Graph ist in Abbildung 5.8 dargestellt.

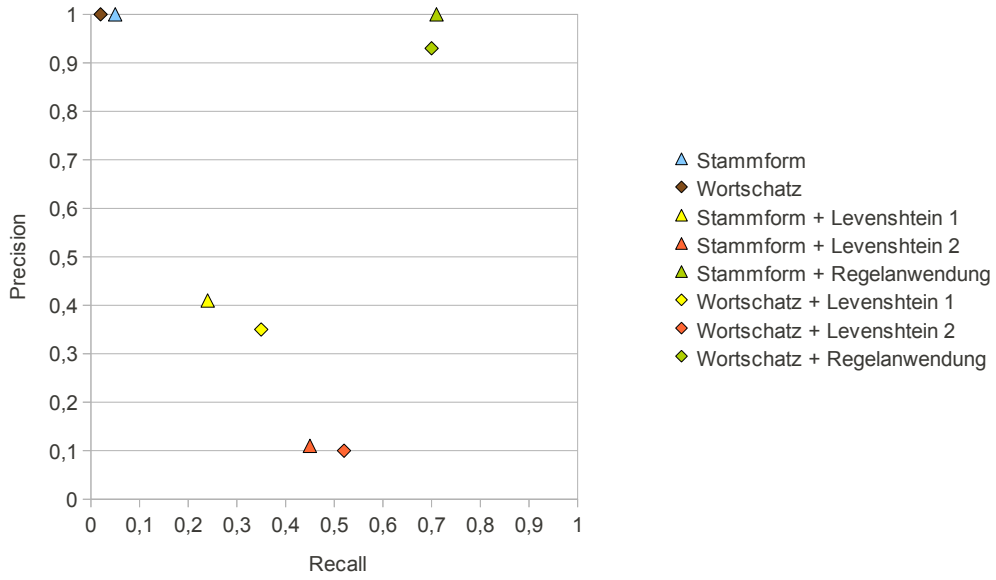


Abbildung 5.8.: Recall-Precision-Graph für die Suche nach historischen Types

Stammformbildung

Mit der Stammformbildung werden 202 Wortformen für die Suche nach Schreibvarianten gebildet. Im Durchschnitt sind das 0,85 Wortformen pro Lemma. Somit können nicht alle Lemmata mit dem Stammform-Algorithmus gefunden werden. Auf der anderen Seite erreicht der Algorithmus eine perfekte Precision, weil alle gefundenen Wortformen korrekt sind. Der Recall von 0,05 zeigt allerdings deutlich, dass die Stammformbildung für die historischen Texte nicht sinnvoll einsetzbar ist.

Wortschatz-Datenbank

Im Durchschnitt werden in der Wortschatz-Datenbank 7,7 Wörter pro Lemma generiert. Weil keine der 4 gebildeten falschen Wortformen in der Testkollektion vorkommt, wird eine perfekte Precision von 1 erreicht. Ähnlich wie bei der ersten Testmenge wird nur für 77 % der Wörter die gesuchte moderne Wortform gefunden.

[21] Für diesen Evaluierungsteil werden keine Retrieval-Werte für herkömmliche Suchmaschinen berechnet, weil diese offensichtlich nicht in der Lage sind, die historischen Wortformen zu den Lemmata zu finden.

Levenshtein-Distanz

Beschränkt auf historische Wortformen wird mit einer Levenshtein-Distanz von 1 lediglich ein Recall von 0,24 in Kombination mit dem Stammformansatz und ein Recall von 0,35 basierend auf der Wortschatz-Datenbank erzielt. Damit werden 3 von 4 (bzw. 2 von 3) Varianten nicht gefunden. Auch die Precision erreicht mit 0,41 für den Stammformansatz und 0,35 für die Wortschatz-Datenbank keinen guten Wert.

Bei einer maximalen Levenshtein-Distanz von 2 steigt der Recall von 0,25 auf 0,45 für die Suche mit dem Stammformindex. Für die Suche mit der Wortschatz-Datenbank steigt der Recall in ähnlichem Umfang (von 0,35 auf 0,52). Dabei wurden 2.389 Varianten für den Stammformansatz bzw. 2.167 Varianten für die Wortschatz-Datenbank gebildet. Auch in diesem Fall sinkt die Precision für die maximale Levenshtein-Distanz von 2 deutlich auf 0,11 bzw. 0,10.

Regelanwendung

Obwohl der Stammformansatz nicht viele Wortformen findet, liegt der Recall bei 0,71. Für die gefundenen Varianten mit einer Vorkommenshäufigkeit von 3906 erreicht diese Kombination eine perfekte Precision von 1,0.

Für die historischen Formen wird eine Precision von 0,93 und ein Recall von 0,7 erreicht. Der Recall sinkt im Vergleich zu Abschnitt 4.3.5, wo ein Recall von 0,88 erzielt wird. Dieser Unterschied liegt in der Berücksichtigung des Wörterbucheffektes, weil die vorangegangenen Experimente auf Paaren *aktuelle Wortform — historische Wortform* basieren, während nun Tripel aus *Lemma — generierte Wortform — historische Wortform* betrachtet werden. Insgesamt werden 23 % der aktuellen Wortformen aus der Testmenge nicht gefunden. Allerdings sinkt der Recall lediglich um 18 %. Wortformen, die nicht mit der gesuchten Wortform zu einem Lemma übereinstimmen, ermöglichen also trotzdem eine erfolgreiche Bildung der gesuchten historischen Variante.

Im Vergleich zur Kombination von Stammformbildung und Regelanwendung sinkt die Precision mit 0,93 nur leicht, während der Recall nahezu unverändert bleibt. Allerdings haben die gefundenen Varianten eine Vorkommenshäufigkeit von 13.198. Somit können mehr als dreimal so viele Schreibvarianten gefunden werden wie mit dem Stammformansatz. Folglich ist der Kombination des regelbasierten Ansatzes mit der Wortschatz-Datenbank trotz des mit 0,80 im Vergleich zu 0,83 etwas schlechteren F-Maßes der Vorzug zu geben. Bei der Nutzung der Levenshtein-Distanz liegt der beste Wert für das F-Maß bei 0,35. Beschränkt auf die historischen Formen zeigt sich somit besonders eindrucksvoll, dass die Nutzung der Levenshtein-Distanz keine echte Alternative darstellt.

5.2.3. Zusammenfassung

Die beiden Experimente mit Wörtern der ganzen Kollektion und den historischen Wörtern haben gezeigt, dass der entwickelte Ansatz erfolgreich beim Retrieval auf historischen Texten zum Einsatz kommen kann. Durch die starke Flexion der deutschen Sprache ist allerdings eine Methode erforderlich, die dieses Phänomen behandelt, um gute Retrievalergebnisse zu erhalten.

Werden die Schreibvarianten nicht in die Suche mit einbezogen, können 10 % der Wörter nicht gefunden werden. Die Transformationsregeln können den Verlust drastisch reduzieren. Allerdings ist der Preis dafür eine leicht sinkende Precision.

Des Weiteren zeigen die Experimente, dass der Stammform-Algorithmus bei historischen Wörtern nicht besonders gut funktioniert. Für den Levenshtein-Algorithmus werden die besten Ergebnisse in Kombination mit dem Stammform-Algorithmus für Wörter der gesamten Kollektion erreicht. Hier erzielt die Methode ähnliche Recall-Werte wie bei der Kombination mit der Wortschatz-Datenbank und der Regelanwendung. Allerdings hat die auf historische Wortformen beschränkte Evaluierung deutlich gezeigt, dass die Levenshtein-Distanz im Vergleich mit dem regelbasierten Ansatz nicht konkurrenzfähig ist.

Insgesamt ist die Qualität des regelbasierten Ansatzes relativ stabil. Die Levenshtein-Distanz erzielt nur mit Wörtern aus der gesamten Kollektion in Kombination mit dem Stammformansatz vergleichbare Ergebnisse. Dabei muss berücksichtigt werden, dass die gefundenen Wortformen des Stammformansatzes direkt den Recall erhöhen, weil diese Varianten im Text enthalten sind. Im Gegensatz dazu sind nur die wenigsten Wörter aus der Wortschatz-Datenbank auch im Text enthalten, so dass in jedem Fall noch eine Transformation nötig ist, um die historischen Formen zu generieren.

5.3. Google-Buchsuche für historische Dokumente

Eine große Anzahl von Benutzern sieht das Internet als wichtige oder wichtigste Quelle bei der Suche nach Information an [Höhn, 2008]. Deswegen ist eine Übertragbarkeit des vorgestellten Ansatzes auf Internetsuchmaschinen von großer Bedeutung. So wird z. B. von Buck und Kollegen mit WebMetric eine Erweiterung für eine unscharfe Suche nach Schreibvarianten im Mozilla Firefox Browser vorgestellt [Buck u. a., 2013]. Allerdings kann damit lediglich auf einer bereits bekannten Seite eine Suche nach historischen Wörtern durchgeführt werden. Eine Alternative ist die Erstellung eines eigenen Webdienstes, der als Frontend zu einer Internet-Suchmaschine fungiert. Dieser bietet gegenüber der Implementierung einer Firefox-Erweiterung den Vorteil der Browserunabhängigkeit. Außerdem muss keine Anwendung installiert werden. Dadurch kann die Suche sofort eingesetzt werden und das Nutzungshemmnis der Installation entfällt. Die modulare Struktur des

entwickelten Systems bietet die Möglichkeit, die einzelnen Module auch in andere Suchmaschinen einzubinden. Exemplarisch ist dies in der Bachelorarbeit [Nwabueze, 2011] am Beispiel der Google-Buchsuche gezeigt worden.

Technisch erfolgte die Realisierung auf Basis der Programmiersprache Java unter Einsatz von Apache Tomcat[22] und der integrierten Datenbank Derby[23]. Für die dynamische Erstellung der Webseiten kam JavaServer Pages[24] zum Einsatz.

Die Einbindung der Google-Buchsuche erfolgte über die Google Data API. Die Standardsuche enthält analog zur Google-Buchsuche nur ein einziges Eingabefeld (s. Abbildung 5.9). Darüber hinaus gibt es noch eine erweiterte Suche, bei der auch die Sprache und der Zeitraum angegeben werden können (s. Abbildung 5.10). Zusätzlich kann der Benutzer auch die Suche nach Flexionsformen deselektieren.

Abbildung 5.9.: Historische Google-Buchsuche

Die Rechercheergebnisse werden zunächst in einer Ergebnisliste angezeigt (s. Abbildung 5.11). Dabei werden die Suchbegriffe, wie bei Google, in einer frageabhängigen Zusammenfassung durch Fettdruck hervorgehoben. Auf diese Weise bekommt der Benutzer einen Überblick, in welchem Kontext der gefundene Begriff zum Einsatz kommt [Hearst, 1999].

Ausgehend von der Ergebnisliste hat der Benutzer die Möglichkeit, sich einzelne Ergebnisse in einer Detailansicht darstellen zu lassen (s. Abbildung 5.12). Durch die Einbindung der Ergebnisdarstellung der Google-Buchsuche mit der Google

[22] `http://tomcat.apache.org/` (letzter Aufruf 18.12.12)
[23] `http://db.apache.org/derby/` (letzter Aufruf 18.12.12)
[24] `http://www.oracle.com/technetwork/java/javaee/jsp/index.html` (letzter Aufruf 18.12.12)

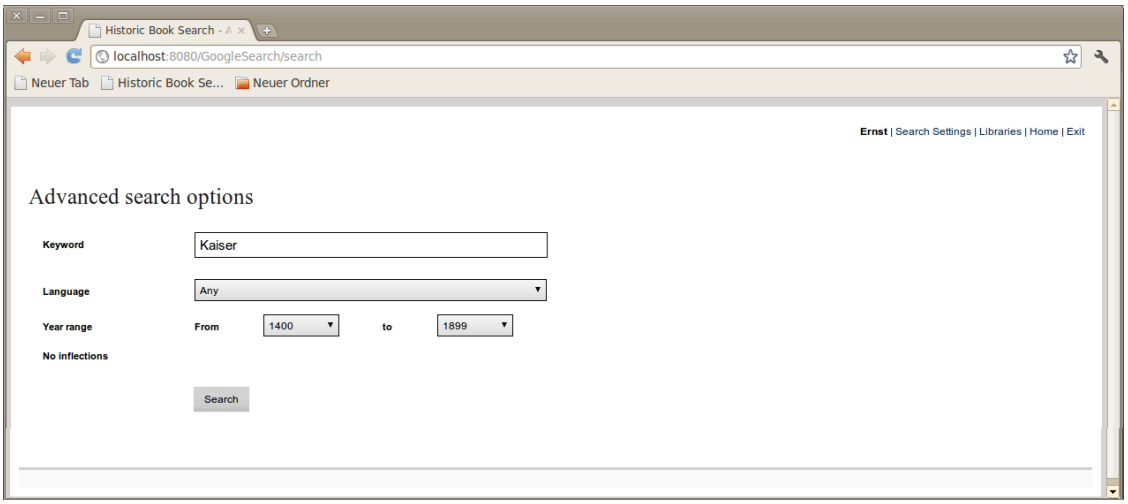


Abbildung 5.10.: Suchoptionen der erweiterten historischen Google-Buchsuche

Embedded Viewer API[25] wird das gefundene Buch dargestellt. Zusätzlich können die Werkzeuge Zoomen und Blättern mit genutzt werden.

In der frei zugänglichen Version der Buchsuche steht dem Benutzer ein Standardregelsatz zur Verfügung, der automatisch bei der Suche zum Einsatz kommt. Richtet sich der Benutzer einen Account ein, kann er auch eigene Regelsätze hochladen. Er kann zudem die vorhandenen Regelsätze herunterladen, an seine Bedürfnisse anpassen und wieder hochladen. Die abgespeicherten Regeln stehen dann bei allen nachfolgenden Suchen zur Verfügung. Die vorhandenen Regelsätze können, je nach Suchbedürfnis, aktiviert bzw. deaktiviert werden (s. Abbildung 5.13). Weil der Benutzer die Möglichkeit hat, Regeln auch selbst zu erstellen, kann ein sehr komplexer Regelsatz bei der Suche im Internet aufgrund des großen Indexes der Google-Buchsuche dazu führen, dass die Suche durch eine zu hohe Anzahl an zu suchenden Schreibvarianten zu lange dauert. Deswegen kann der Benutzer die Anzahl der Varianten beschränken, nach denen gesucht wird.

Gefundene Dokumente können in persönlichen Bibliotheken abgespeichert werden (s. Abbildung 5.14). Der Benutzer bekommt so die Möglichkeit, seine Ergebnisse zu strukturieren. Sie stehen ihm auch nach der Suche weiter zur Verfügung [Kriewel u. a., 2004]. Als Erweiterungsmöglichkeit könnten Regelsätze auch anderen Benutzern bzw. Benutzergruppen zur Verfügung gestellt werden. Dies würde den Erstellungsaufwand für Regelsätze für den Einzelnen weiter reduzieren.

[25] `http://code.google.com/intl/de-DE/apis/books/docs/viewer/developers_guide.html` (letzter Aufruf 18.12.12)

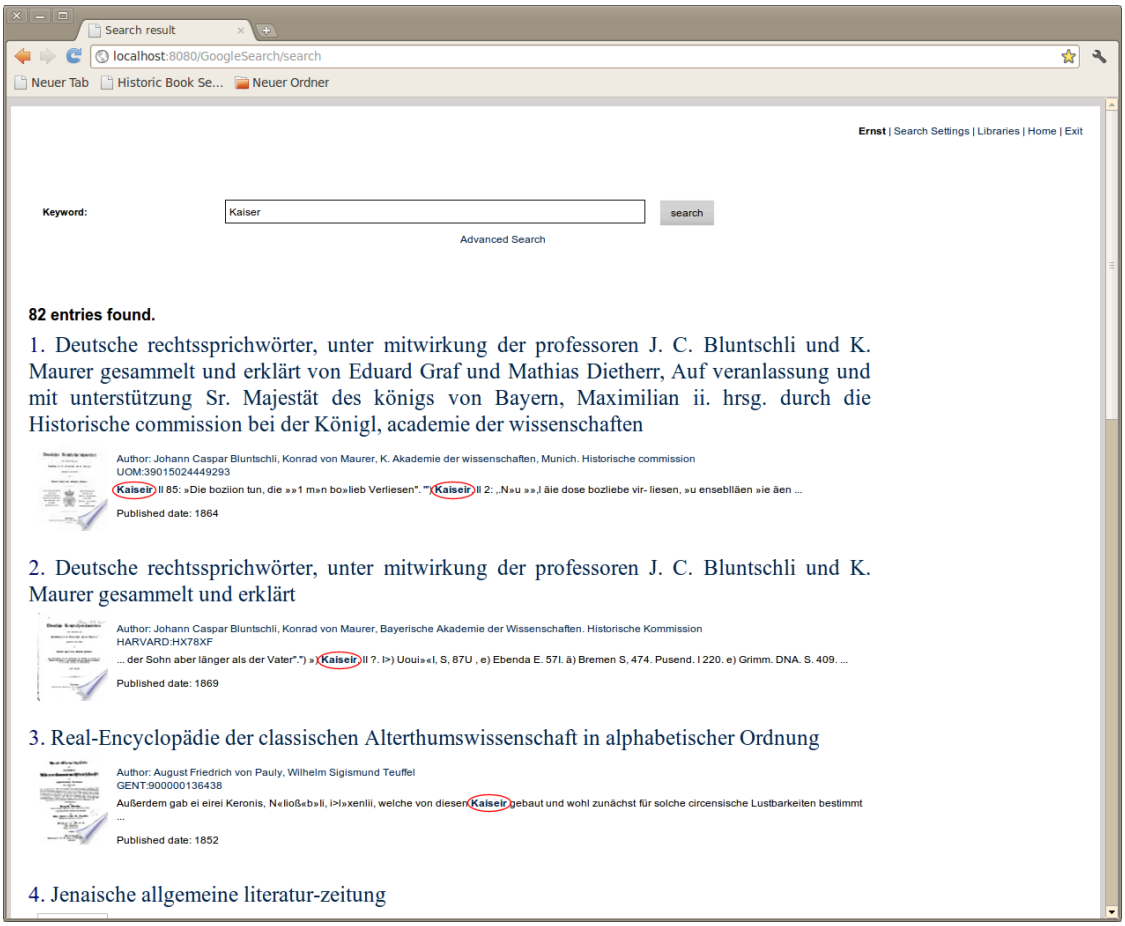


Abbildung 5.11.: Suchergebnis der historischen Google-Buchsuche

5.4. Zusammenfassung

In diesem Abschnitt wurde eine Suchmaschine für den regelbasierten Ansatz entwickelt. Diese beiden Komponenten ermöglichen dem Benutzer, eigenständig Regeln zu generieren und bei der Suche erfolgreich einzusetzen. Am Beispiel der Google-Buchsuche wurde die Einsatzfähigkeit des regelbasierten Ansatzes bei einer Suche im Internet gezeigt. Durch den modularen Aufbau der Suche könnte der regelbasierte Ansatz demzufolge auch bei weiteren Internetsuchmaschinen zum Einsatz kommen.

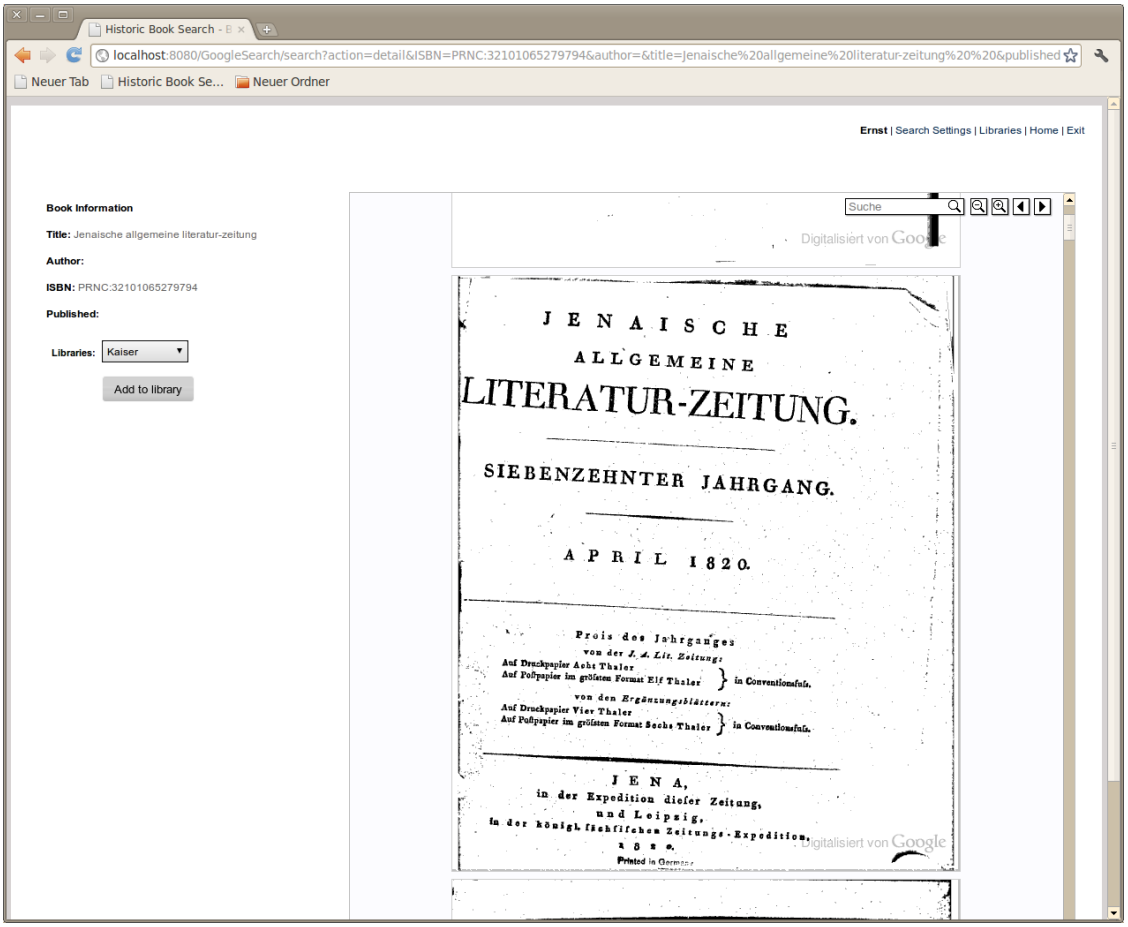


Abbildung 5.12.: Ergebnisdetails der historischen Google-Buchsuche

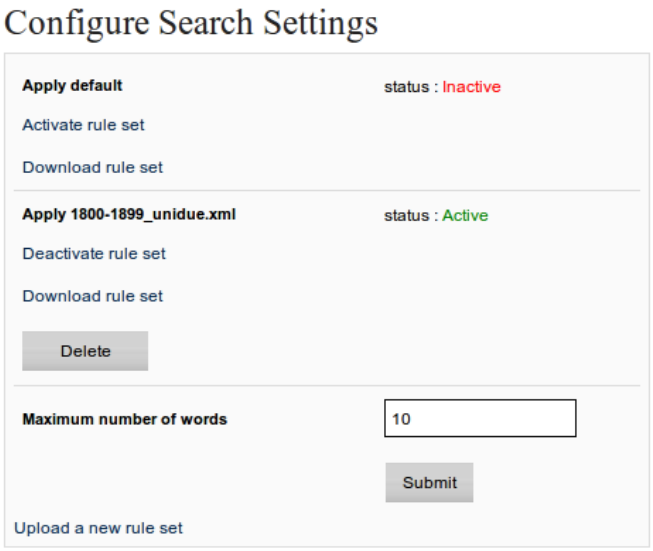


Abbildung 5.13.: Benutzereinstellungen der historischen Google-Buchsuche

Abbildung 5.14.: Persönliche Bibliothek der historischen Google-Buchsuche

Teil IV.

Benutzergesteuerte Regelerstellung

6. Methoden zur Erstellung von Belegen

Durch den Einsatz eines regelbasierten Verfahrens wird die Retrievalqualität bei historischen Dokumenten deutlich verbessert. Dies wurde im vorherigen Kapitel gezeigt. Ein Flaschenhals des Ansatzes ist allerdings die Erstellung der Trainingspaare für die Generierung der Regeln. Eine Möglichkeit zum Aufbau der Trainingsdaten ist die Nutzung von parallelen Korpora. Die Asymmetrien von Phrasen und Sätzen beim Alignment der Versionen lassen sich durch den Einsatz eines Satzaligners gefolgt von einem Wortaligner beheben [Bollmann u. a., 2011]. Das größte Problem stellt allerdings die Verfügbarkeit dar, weil zu einer Kollektion nur in den seltensten Fällen parallele Korpora existieren. Im Folgenden werden verbreitete Ansätze zur Sammlung von Belegen beschrieben.

6.1. LeXtractor

Gotscharek und Kollegen haben mit dem LeXtractor ein Werkzeug zur Konstruktion von historischen Lexika entwickelt [Gotscharek u. a., 2009a]. Die Lexikoneinträge können als Belege im Sinne der vorliegenden Arbeit aufgefasst werden. Der LeXtractor hat zwei Ansichten. Die erste bietet dem Benutzer einen Text an, in dem die Terme hervorgehoben sind, die der Rechtschreibprüfung unbekannt sind (s. Abbildung 6.1). In dieser Ansicht kann der Benutzer auf Basis des Textes Lexikoneinträge erzeugen. Die zweite Ansicht (s. Abbildung 6.2) zeigt eine Liste mit unbekannten Termen, die nach absteigender Termhäufigkeit geordnet ist. Durch die Arbeit mit dieser Liste kann der Prozentsatz der vom Lexikon abgedeckten Wörter schnell gesteigert werden. Wird ein unbekannter Term aus der Liste ausgewählt, werden auf Basis von Regeln (sog. Mustern) erzeugte Vorschläge für moderne Schreibungen angegeben (s. Abbildung 6.3). Dabei werden die benutzten Muster mit angezeigt. Bei der Aufnahme der Terme in das Lexikon können verschiedene Kategorien wie z. B. ”Wort ohne moderne Entsprechung” oder ”Eigenname” angegeben werden. Zur Unterstützung wird eine Liste mit Textstellen in denen der unbekannte Term vorkommt angeboten (s. Abbildung 6.4), wenn ein Wort für die Konstruktion eines Lexikoneintrags ausgewählt wird.

Abbildung 6.1.: Textquelle im LeXtractor mit hervorgehobenen unbekannten Wörtern [Gotscharek u. a., 2009a]

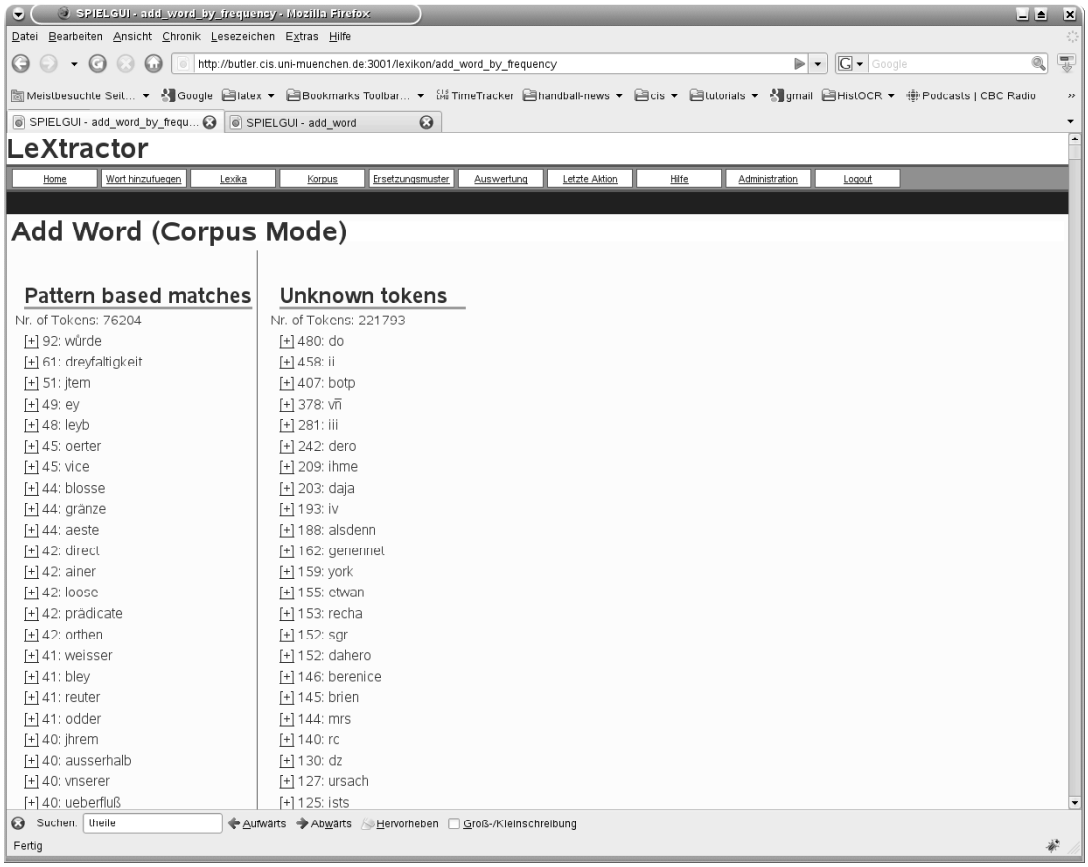


Abbildung 6.2.: Listenansicht des LeXtractors [Gotscharek u. a., 2009a]

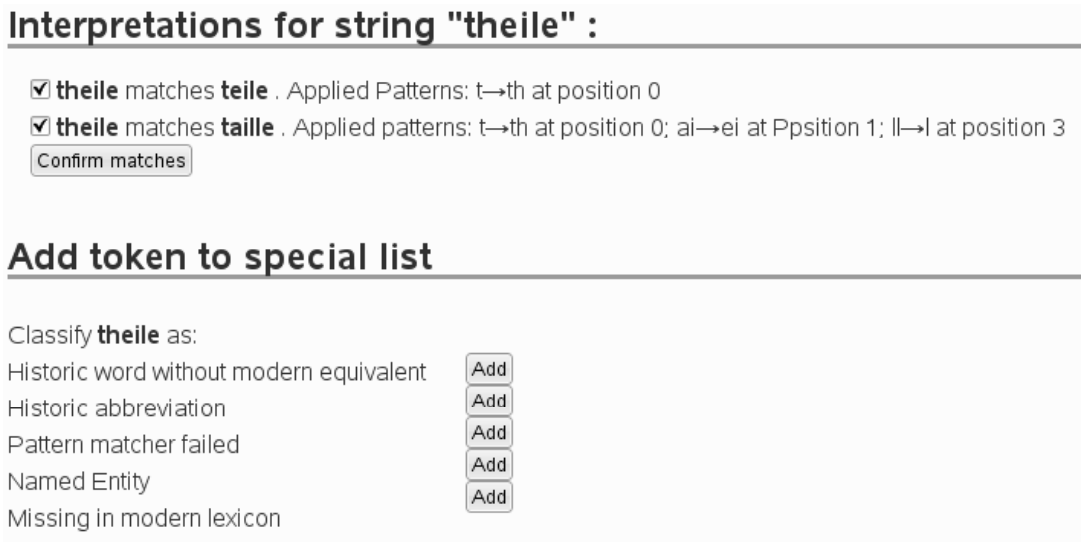


Abbildung 6.3.: Auswahl von Varianten beim LeXtractor [Gotscharek u. a., 2009a]

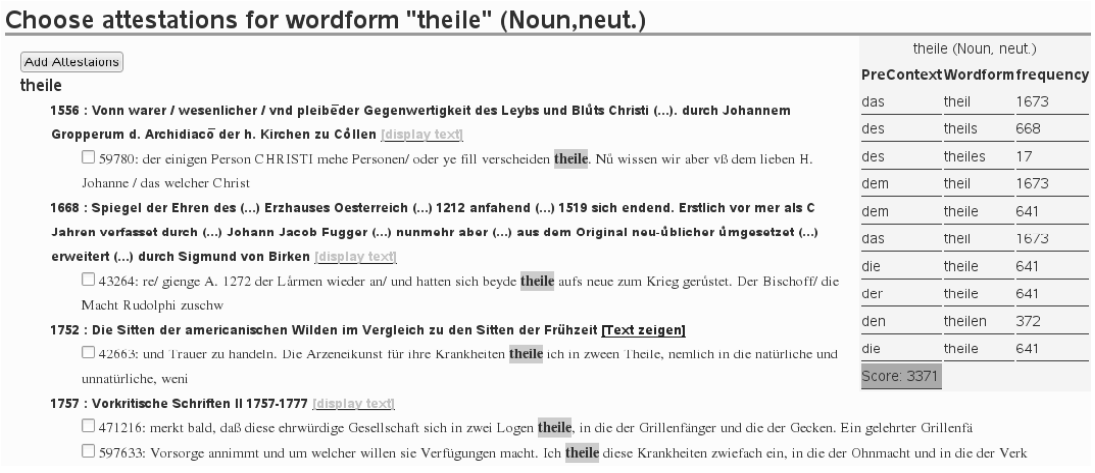


Abbildung 6.4.: Textstellen eines unbekannten Wortes im LeXtractor [Gotscharek u. a., 2009a]

Weil das Werkzeug zur Lexikonerstellung verwendet wird, müssen die Ergebnisse eine sehr hohe Precision aufweisen. Deswegen muss ein Experte alle unbekannten Wörter der Kollektion bearbeiten und jede Schreibweise beurteilen. Die Muster, die genutzt werden, um potenzielle moderne Formen in einem aktuellen Lexikon zu finden, werden manuell erstellt. Während der Konstruktion von Lexikoneinträgen ist es auch möglich, zusätzliche Muster vorzuschlagen, wenn eine neue Regel bemerkt wird.

6.2. Evidencer

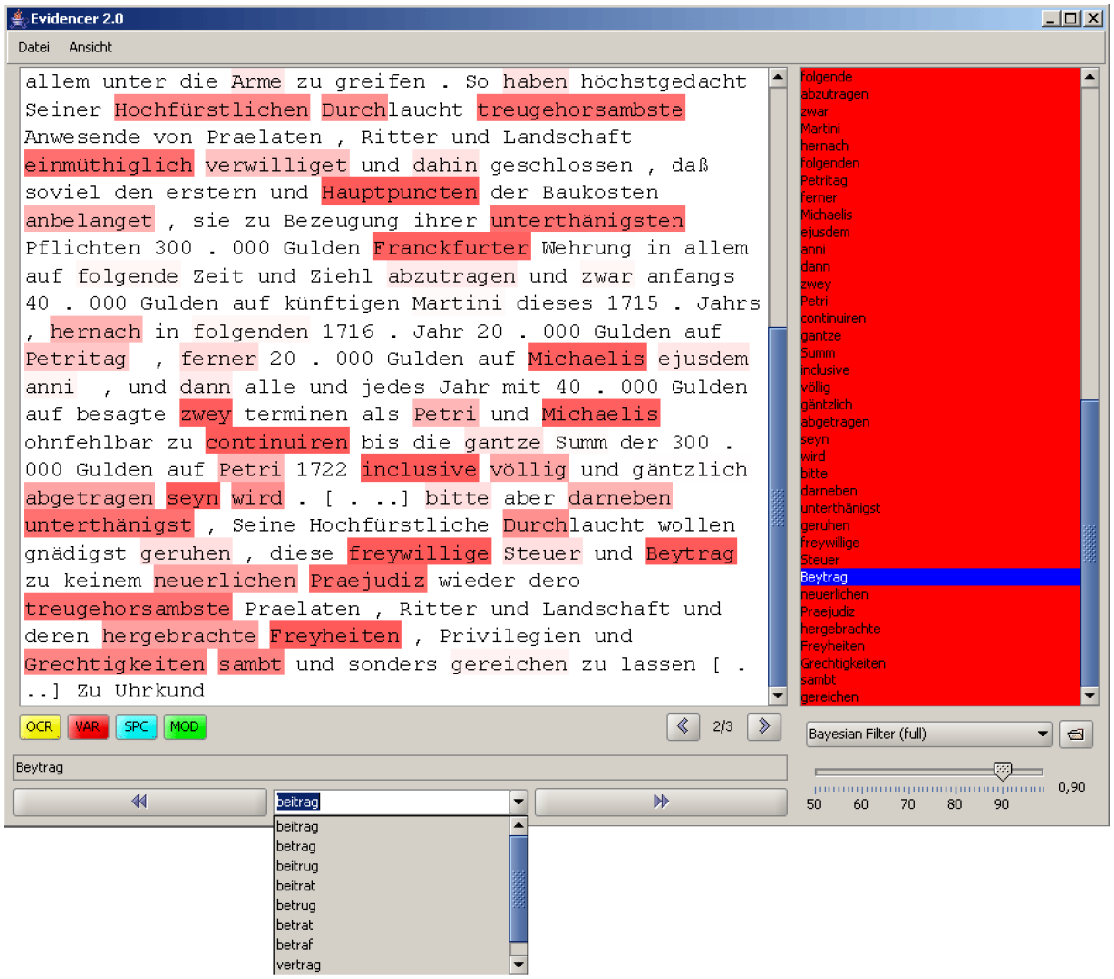


Abbildung 6.5.: Evidencer

Pilz und Luther haben eine Methode entwickelt, um die Sammlung von Belegen in ihrem Evidencer Werkzeug zu unterstützen und vor allem den Arbeitsaufwand für

diesen Schritt zu verringern (s. Abbildung 6.5) [Pilz und Luther, 2009]. Sie gehen von einem signifikanten Unterschied bei der Verteilung der N-Gramme zwischen Standard- und Nicht-Standardschreibungen aus. Zur Einteilung in moderne Schreibweisen und Schreibvarianten benutzen sie einen Bayes-Klassifizierer, der die Wahrscheinlichkeit schätzt, ob es sich um eine Schreibvariante handelt. Dazu werden Trainingsbeispiele benötigt. Nach der Trainingsphase wird eine Liste mit unbekannten Wörtern präsentiert. Diese sortiert die Wörter absteigend nach der Wahrscheinlichkeit für Schreibvarianten. Der Benutzer kann den Klassifizierer anpassen, indem er den Grenzwert zur Identifizierung als Variante verändert. Allerdings benötigt der Bayes-Klassifizierer eine große Menge an Trainingsdaten [Mitchell, 2001, S. 155].

6.3. VARD 2

VARD 2 ist ebenfalls in der Lage, moderne Formen für Schreibvarianten in historischen Dokumenten zu finden [Baron und Rayson, 2008]. Dabei ist das Ziel nicht die Suche nach Texten, sondern die Normalisierung von Dokumenten, um im Anschluss Techniken der Korpuslinguistik einsetzen zu können. Das Werkzeug markiert alle Wörter als potenzielle Varianten, die der Rechtschreibkorrektur unbekannt sind (s. Abbildung 6.6). Für jedes markierte Wort wird dem Benutzer eine Liste mit potenziellen zugehörigen modernen Schreibungen angeboten (s. Abbildung 6.7).

Um die Vorschläge zu generieren, werden die folgenden Methoden benutzt:

- Eine manuell erstellte Liste von 45.000 Beispielen für moderne Wörter und die zugehörigen Schreibvarianten.
- Eine modifizierte Version des Soundex-Algorithmus.
- Eine manuell erstellte Liste von Ersetzungsregeln. Jedes Wort, das durch die Regeln in ein Wort aus dem modernen Lexikon überführt werden kann, wird als mögliche historische Schreibvariante eines modernen Wortes betrachtet.

Basierend auf diesen drei Methoden wird ein Konfidenzwert für einen Vorschlag berechnet. Zusätzlich kommt ein Edit-Distanz-Algorithmus zur Gewichtung von Vorschlägen zum Einsatz. Der Konfidenzwert ist dabei kein fester Wert, sondern wird nach jedem Schritt automatisch angepasst.

Der Benutzer kann aus der Liste die passende moderne Form auswählen. Das historische Wort wird neben der modernisierten Schreibweise in einem XML-Tag abgespeichert. Die vom Benutzer eingegebenen Wortformen werden automatisch in das Wörterbuch der Anwendung integriert [Baron u. a., 2009]. Ein zweiter Modus bietet zudem die Möglichkeit, automatisch die Vorschläge mit dem höchsten Ranking zu akzeptieren, wenn der Wert über einem vom Benutzer festgelegten Mindestwert liegt (s. Abbildung 6.8). Es wird empfohlen, zunächst mit der

manuellen Bearbeitung zu beginnen. Im Anschluss können die noch verbliebenen unbekannten Wörter automatisch bearbeitet werden, nachdem der Konfidenzwert für ein neues Korpus ausreichend trainiert ist.

Rayson und Kollegen haben gezeigt, dass bereits der Vorgänger VARD deutlich bessere Ergebnisse bei der Normalisierung von Texten liefert, als herkömmliche Programme zur Rechtschreibkorrektur [Rayson u. a., 2005]. Der Nachteil von VARD 2 besteht in der Notwendigkeit von Trainingsdaten und Regeln als Eingabe. In beiden Fällen handelt es sich um manuell gesammelte Daten. Des Weiteren ist der Soundex-Algorithmus ein phonetischer Algorithmus, der für die englische Sprache entwickelt wurde. Er müsste durch die zeitliche und räumliche Veränderung der Aussprache nicht nur für jede Sprache, sondern auch für jedes Korpus angepasst werden. Dadurch sind phonetische Algorithmen nur bedingt für die Erstellung von Regeln für historische Schreibweisen geeignet.

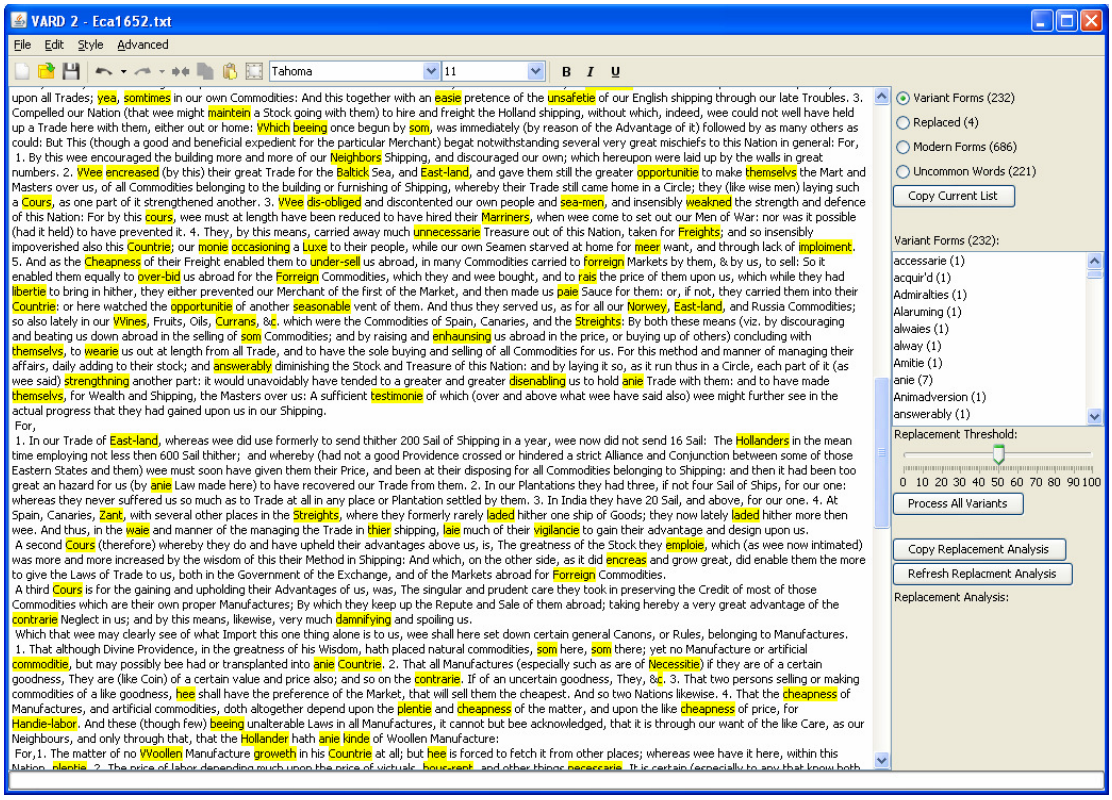


Abbildung 6.6.: Markierter Text in VARD 2 [Baron und Rayson, 2008]

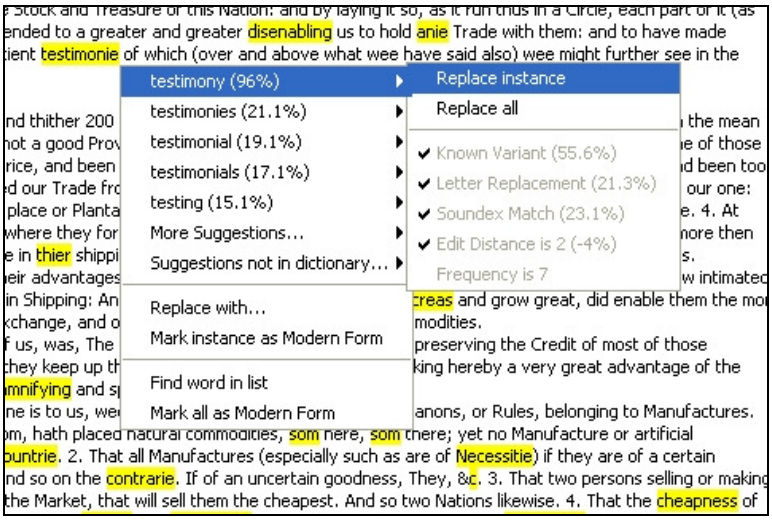


Abbildung 6.7.: Vorschläge von VARD 2 [Baron und Rayson, 2008]

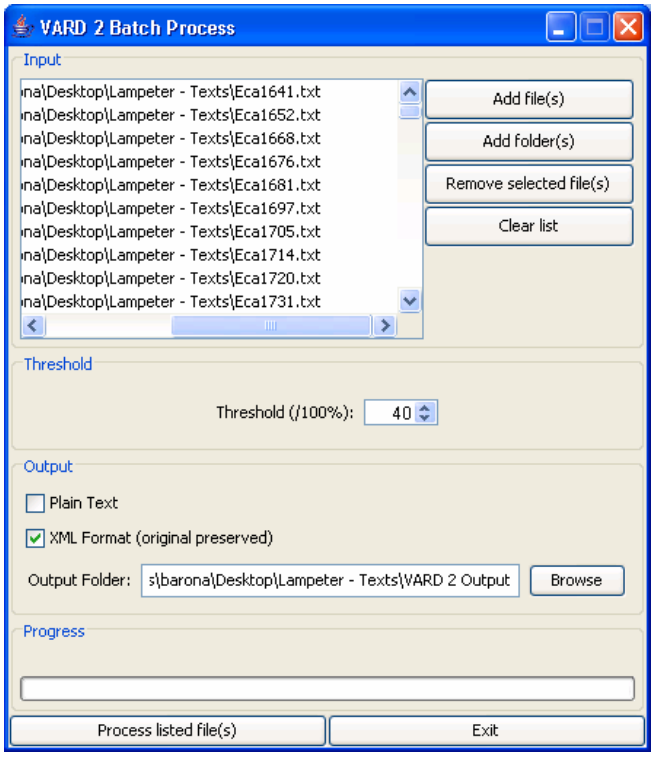


Abbildung 6.8.: Automatischer Modus von VARD 2 [Baron und Rayson, 2008]

Um den Nachteil der manuell erstellten Regeln zu beseitigen, wurde mit DICER (Discover and Investigation of Character Edit Rules) ein weiteres Werkzeug erstellt (s. Abbildung 6.9) [Baron und Rayson, 2009]. Es ermöglicht aus den mit VARD 2 gesammelten Varianten automatisch Regeln zu bilden, die eine historische Form auf eine moderne abbilden. Das Analyseergebnis wird in Form von Webseiten[26] dargestellt. Neben dem Regeltyp (einfügen, löschen, vertauschen) werden auch Angaben zur Häufigkeit der entsprechenden Ersetzung je nach Position gemacht. Für jede Regel stehen zusätzlich Informationen zur Häufigkeit der Buchstaben, vor und nach denen sie zum Einsatz kommt, zur Verfügung. Die mit Hilfe von DICER gebildeten Regeln können wieder in VARD 2 importiert werden. Allerdings bietet DICER keine Möglichkeit, die Regeln zu bearbeiten.

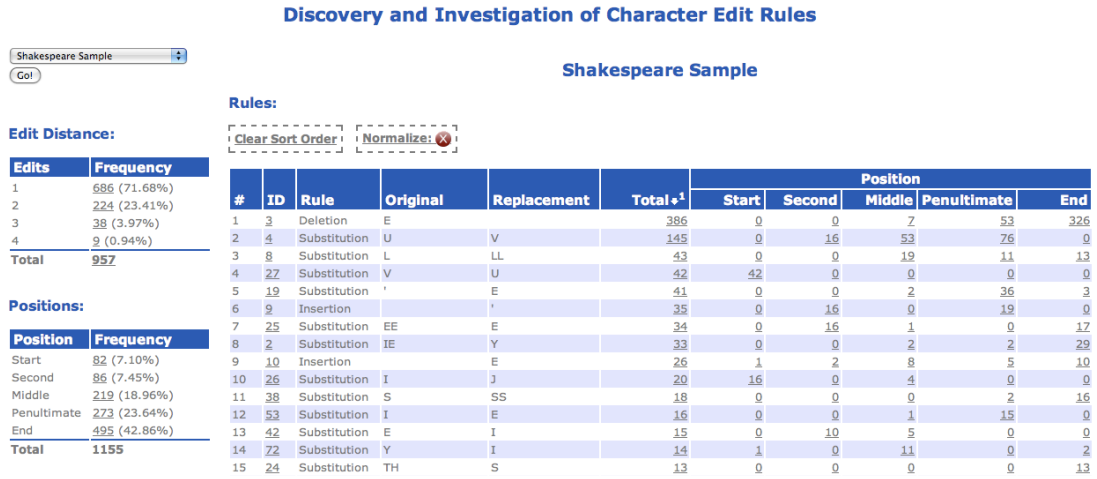


Abbildung 6.9.: DICER [Baron und Rayson, 2009]

6.4. Zusammenfassung

Der LexTractor benötigt (aufgrund seines Einsatzgebietes) ein großes Maß an manueller Interaktion, sowohl für die Bildung der Belege als auch für die Erstellung der Regeln. Der Evidencer sieht mit Blick auf die automatische Unterstützung für den Benutzer Erfolg versprechender aus. Der Benutzer hat hier die Möglichkeit, die Qualität der Ergebnisse durch einen Schwellenwert zu beeinflussen. Allerdings benötigt der Bayes-Klassifizierer eine große Anzahl an Trainingsdaten. Somit ist einiges an manueller Arbeit notwendig, bevor er eingesetzt werden kann. Außerdem kann der Benutzer nur dokumentweise vorgehen. Somit kann er nicht mehrere Vorkommen von möglichen Varianten in verschiedenen Texten gleichzeitig betrachten. VARD 2 sieht besonders wegen des ständig

[26] `http://corpora.lancs.ac.uk/dicer/` (letzter Aufruf 18.12.12)

angepassten Konfidenzwertes für die modernen Formen sehr vielversprechend aus. Der Konfidenzwert ist sonst vergleichbar mit dem Schwellenwert des Bayes-Klassifizierers aus dem Evidencer-Werkzeug.

Alle betrachteten Ansätze benötigen einen hohen manuellen Aufwand, bevor sie sich zur Generierung von Schreibvarianten einsetzen lassen. Deswegen würde ein automatisches Verfahren zur Erstellung der Belege den Zugang zu historischen Dokumenten für den Benutzer deutlich erleichtern. Außerdem werden die Regeln bisher lediglich mit einer Konsolenanwendung generiert. Weil es auch Benutzern ohne Informatikkenntnisse möglich sein soll, Regeln zu entwickeln, wird im folgenden Kapitel mit dem RuleGenerator ein entsprechendes Werkzeug vorgestellt.

7. Benutzergesteuerte Regelerstellung

Der letzte Abschnitt hat den hohen manuellen Aufwand für die Erstellung der Belege bei der Verwendung des bisherigen Ansatzes verdeutlicht. Nach Kempken werden für die manuelle Erstellung von 800 Belegen bereits mehrere Wochen benötigt [Kempken, 2005]. Mit Hilfe des LeXtractors dauert die Erstellung von 1.000 Lexikoneinträgen — und somit auch von Belegen — etwa einen Personenmonat [Gotscharek u. a., 2011]. Diese Prozedur wiederholt sich bei jeder neuen Kollektion, die aus einem anderen zeitlichen oder örtlichen Kontext stammt. Von Pilz wurde eine Untersuchung zur notwendigen Beleganzahl zur Erstellung einer Metrik für zeitliche Variation im Deutschen durchgeführt. Diese kommt zu dem Ergebnis, dass mindestens 4.000 Belege benötigt werden [Pilz, 2009, S. 182ff.]. Um diesen Aufwand drastisch zu reduzieren, ist es im Folgenden das Ziel, einen Algorithmus zu entwickeln und zu implementieren, der Belege automatisch generieren kann.

7.1. Automatische Belegerstellung

Die Grundlage für den entwickelten regelbasierten Ansatz bildet die Annahme, dass Schreibvarianten ein bestimmtes Maß an Regularität beinhalten. Basierend auf dieser Theorie soll im Folgenden auch die Generierung der Belege automatisch erfolgen. Zunächst werden Korrekturvorschläge der Rechtschreibprüfung Hunspell[27] für unbekannte Wörter untersucht. Hunspell wird u. a. bei Open Office, Mozilla Firefox und Google Chrome zur Prüfung der Rechtschreibung verwendet. Die Vorschläge für falsch geschriebene Wörter basieren auf N-Gramm-Vergleichen, Regeln und Aussprachedaten. Mit diesen Methoden werden dann auf Basis eines Wörterbuchs Vorschläge erstellt. Zur Zeit stehen Wörterbücher für 101 Sprachen zur Verfügung.

Die richtige moderne Form einer Schreibvariante befindet sich, wie eine Untersuchung der Vorschläge zur Rechtschreibprüfung ergab, häufig unter den Vorschlägen. Allerdings gibt es keine Möglichkeit, den passenden Vorschlag automatisch auszuwählen. Dies ergab auch eine Untersuchung von Koolen für die niederländische Sprache [Koolen, 2005]. Es wird im Folgenden davon ausgegangen,

[27] `http://hunspell.sourceforge.net/` (letzter Aufruf 18.12.12)

```
Bilde Trainingsmenge aus unbekannten Schreibweisen und Vorschlägen
Generiere Regelkandidaten (nur Regelkerne)
Solange Regelkandidaten r_j mit Regelhäufigkeit > min Regelhäufigkeit
existieren
  Sortiere Regelkandidaten nach Häufigkeit
  Akzeptiere den häufigsten nicht akzeptierten Regelkandidaten r_i
    Kennzeichne den Regelkandidaten r_i für alle zugehörigen Vorschläge s_i
    als markiert
    Akzeptiere alle s_i, bei denen alle Regelkandidaten akzeptiert wurden
    Wenn s_i akzeptiert ist, lösche alle konkurrierenden Vorschläge s_k
```

Abbildung 7.1.: Algorithmus zur automatischen Beleggenerierung

dass Regularitäten zwischen moderner Form und Schreibvariante deutlich häufiger als zwischen Schreibvarianten und falschen Vorschlägen zu finden sind. Deswegen konzentriert sich der Algorithmus (s. Abbildung 7.1) auf das Problem, den richtigen Vorschlag der Rechtschreibprüfung zu einer Variante zu bestimmen. Dabei wird der Vorschlag ausgewählt, der über die häufigeren Regelkandidaten verfügt.

Aus jeder unbekannten Schreibweise und den zugehörigen Vorschlägen wird dazu ein Beleg erstellt (s. Tabelle 7.1). Diese Belege bilden die Trainingsmenge für die Generierung möglicher Regelkandidaten. Weil in diesem Schritt noch keine endgültigen Regeln entstehen, sind hier die unterschiedlichen Regelkandidaten nicht relevant, und es werden lediglich die Regelkerne betrachtet. Auf diese Weise erhält man weniger Regeln, die aber eine größere statistische Signifikanz besitzen.

Je häufiger ein Regelkern in unterschiedlichen Belegen vorhanden ist, desto größer ist die Wahrscheinlichkeit, dass die Regel sinnvoll ist. Deswegen wird auch die Precision für Belege, die auf häufigeren Regelkernen basieren, höher sein. Demzufolge wird in jedem Durchlauf von den nicht akzeptierten Regelkandidaten derjenige mit der größten Häufigkeit akzeptiert. So kommt z. B. in Tabelle 7.1 die Regel $i \to y$ insgesamt siebenmal vor. Weil alle anderen Regeln nur einmal und somit seltener vorhanden sind, wird die Regel akzeptiert. Haben mehrere Regelkandidaten die gleiche Häufigkeit, werden zunächst Substitutionsregeln akzeptiert, weil diese meistens eine höhere Precision als Einfüge- und Löschregeln haben. Beispielsweise wird $i \to y$ gegenüber $s \to \varepsilon$ oder $\varepsilon \to h$ bevorzugt.

Wurde ein Regelkandidat akzeptiert, werden die zugehörigen Belege (und damit die Vorschläge der Rechtschreibprüfung) betrachtet. Basiert ein Beleg nur auf der bestätigten Regel, wird er direkt anerkannt. Benötigt er mehr als eine Regel, wird er akzeptiert, wenn alle anderen Regeln ebenfalls bestätigt sind (s. Tabelle 7.2). Sonst wird lediglich markiert, dass der Regelkandidat angenommen ist.

Vorschlag	Mögliche Varianten	Regelkandidaten
Geschicklichkei**t**	Geschicklichke**y**t	$i \rightarrow y$
Ungeschicklichkei**t**	Geschicklichke**y**t	$un \rightarrow \varepsilon$, $i \rightarrow y$
Unschicklichkei**t**	**Ge**schicklichke**y**t	$un \rightarrow ge$, $i \rightarrow y$
Schicklichkei**t**	**Ge**schicklichke**y**t	$\varepsilon \rightarrow ge$, $i \rightarrow y$
Gesch**wister**lichkei**t**	Geschi**ck**lichke**y**t	$w \rightarrow \varepsilon$, $ster \rightarrow ck$, $i \rightarrow y$
jederzei**t**	jederze**y**t	$i \rightarrow y$
jeder**ar**t	jeder**zey**t	$ar \rightarrow zey$
jeder**l**ei	jeder**z**eyt	$l \rightarrow z$, $i \rightarrow yt$
jeder**mann**	jeder**zeyt**	$mann \rightarrow zeyt$
derzei**tig**	**je**derze**y**t	$\varepsilon \rightarrow je$, $i \rightarrow y$, $ig \rightarrow \varepsilon$

Tabelle 7.1.: Beispiel für Trainingsdaten der automatischen Belegerstellung und generierte Regelkerne

Vorschlag	Mögliche Variante	Entscheidung
Geschicklichkei**t**	Geschicklichke**y**t	akzeptieren
jederzei**t**	jederze**y**t	akzeptieren
obglei**ch**	obgle**y**ch	akzeptieren
Sonderhei**t**	**In**sonderhe**y**t	markiere $i \rightarrow y$ als akzeptiert

Tabelle 7.2.: Beispiel zur Akzeptanz des Regelkerns $i \rightarrow y$ bei automatischer Belegerstellung

Nach der Akzeptanz eines Vorschlages können im nächsten Schritt falsche Vorschläge aussortiert werden. Es wird davon ausgegangen, dass zu jeder Schreibvariante nur eine moderne Schreibung existiert. Dadurch können die weiteren Vorschläge für historische Schreibweisen entfernt werden. Diese Annahme stellt eine Vereinfachung dar. Wie Pilz gezeigt hat, verfügt die Schreibvariante *Hunngern* über die beiden modernen Formen *Ungarn* und *Hungern* [Pilz, 2009, S. 140]. Die Vereinfachung ist an dieser Stelle allerdings notwendig, um die automatische Beleggenerierung überhaupt zu ermöglichen und somit den manuellen Aufwand für die Konstruktion der Trainingsdaten deutlich zu reduzieren. Bei der späteren manuellen Bearbeitung von Belegen existiert diese Einschränkung nicht. Außerdem ist davon auszugehen, dass Regeln, die durch diese Annahme verloren gehen, durch andere Belege mit generiert werden. Während des Löschungsprozesses werden die falschen Belege auch von weiteren zugehörigen Regelkernen entfernt. Anschließend startet der Prozess wieder mit dem häufigsten unbehandelten Regelkandidaten.

Neben einer geeigneten Auswahl der Testmenge hängt der Erfolg eines Lernverfahrens oftmals auch von einer geeigneten Parameterwahl ab [Witten und Frank, 2000, S. 249]. In einigen Systemen zum Lernen von Regeln ist es möglich, Grenzwerte für die Komplexität von Regeln einzugeben [Fürnkranz, 1999].

Analog dazu kann der Benutzer den Prozess der automatischen Beleggenerierung durch folgende Parameter beeinflussen:

- *Minimale Wortlänge*: Rechtschreibprogramme generieren für kurze Wörter meistens mehr Vorschläge als für lange Wörter. Zusätzlich ist die Wahrscheinlichkeit, eine falsche historische Form zu generieren, deutlich höher, weil sich kurze Wörter mit einer größeren Wahrscheinlichkeit ähneln als lange Wörter. Zudem ist nach Zipf die Wahrscheinlichkeit für ein Wort umgekehrt proportional zur Länge [Zipf, 1935]. Daraus schließt Reynaert, dass es bei der Rechtschreibkorrektur für kurze Wörter deutlich mehr Vorschläge gibt als für lange Wörter [Reynaert, 2004a]. Deswegen kann die Precision für die automatischen Belege durch eine minimale Wortlänge erhöht werden.
- *Minimale Anzahl an Regelvorkommen*: Der Kern des Ansatzes besteht darin, dass die Anzahl der Belege, in denen eine Regel vorkommt, einen Indikator für die Precision darstellt. Dabei ist die Regelhäufigkeit ein offensichtlicher Parameter. Als untere Grenze muss darüber hinaus eine Regel mindestens zweimal vorkommen.
- *Maximale Anzahl der Regelanwendungen pro Wort*: Je mehr Regelanwendungen benötigt werden, um ein modernes Wort auf eine potenzielle Variante abzubilden, desto unwahrscheinlicher ist es, dass es sich um eine Schreibvariante handelt. Insbesondere kurze Wörter können sehr leicht auf komplett andere Wörter abgebildet werden. Beispielsweise kann durch drei Regelanwendungen auf *derzeitig* das Wort *jederzeyt* generiert werden (s. Tabelle 7.1).

Bei den Parametern bevorzugt ein Historiker möglicherweise eine kürzere Wortlänge, eine geringere Anzahl an Regelvorkommen sowie eine höhere maximale Anzahl an Regelanwendungen, um einen hohen Recall zu erreichen. Auf diese Weise kann er direkt mit der Suche auf der Kollektion beginnen. Im Gegensatz dazu wird ein Linguist eventuell genau die umgekehrte Parameterauswahl treffen.

7.2. Evaluierung der automatischen Belegerstellung

Die automatische Beleggenerierung wird im Folgenden mit dem manuellen Aufbau der Belegdatenbank verglichen. Darüber hinaus werden die Parametereinstellungen evaluiert, wobei auch die temporären Unterschiede untersucht werden.

7.2.1. Evaluierung des Aufbaus der Belegdatenbank

Als Testkollektion dient die aufgebaute Belegdatenbank (s. Abschnitt 4.3.4). An diesem Beispiel soll nachvollzogen werden, wie viel manueller Aufwand sich bei der Erstellung einer Trainingskollektion einsparen lässt [Ernst-Gerlach und

Fuhr, 2010b]. Um den sukzessiven Aufbau der Belegdatenbank nachzustellen, wird die Anzahl der als Testdaten verwendeten Belege von 1.000 schrittweise um jeweils 1.000 bis auf 10.000 Belege erhöht. Für jede Testmenge kommt das Verfahren zum Einsatz. Die Nullhypothese für die Precision nimmt an, dass die Precision nicht monoton fällt. Die Alternativhypothese für die Precision geht hingegen von einer monoton fallenden Precision bei steigender Anzahl der Belege aus. Die Nullhypothese für den Recall besagt, dass der Recall nicht monoton steigt. Die zugehörige Alternativhypothese für den Recall geht indessen von einem monoton steigenden Recall aus. Die Hypothesen werden im Folgenden mit einem Cochran-Armitage-Test überprüft. Als Parametereinstellung wird mit einer Mindestwortlänge von fünf Buchstaben, mindestens zwei Regelvorkommen und maximal zwei Regelanwendungen pro Wort eine Recall-orientierte Auswahl getroffen. Dies wird insbesondere durch den geringen Wert für die minimale Vorkommenshäufigkeit der Regeln deutlich.

Es werden zunächst die Recall- und Precision-Werte für die Regelkerne berechnet (s. Abbildungen 7.2, 7.3 und Tabelle 10.3). Die Precision sinkt signifikant von 0,28 bei 1.000 Belegen bis 0,17 bei 10.000 Belegen. Der p-Wert liegt unter 2.2e-16. Dies lässt sich durch die deutliche Parameterwahl zugunsten des Recall erklären. Eine Regel muss nur zweimal vorkommen, um akzeptiert zu werden. Deswegen steigt mit zunehmender Anzahl an Testdaten auch die Wahrscheinlichkeit, dass ein falscher Regelkern in einem weiteren potenziellen Belegpaar vorkommt. Beim Recall sind dagegen keine Auswirkungen der steigenden Anzahl der Trainingsdaten zu bemerken. Er hat eine Spannbreite von 0,47 bis 0,55. Somit lässt sich ungefähr die Hälfte der Regelkerne automatisch generieren.

Weil sich die einzelnen Regeln sehr stark in ihrer Anwendungshäufigkeit unterscheiden, wird zusätzlich auch der Recall basierend auf der Vorkommenshäufigkeit der einzelnen Regeln berechnet. Hier zeigte sich deutlich, dass vor allem die besonders häufigen Regeln generiert werden, da immer mindestens 90 % der Regelkerne zu finden sind. Der p-Wert von 5.103e-16 verdeutlicht den hoch signifikanten Anstieg bei einer Betrachtung des Recalls der Regelvorkommen.

Es wurden noch die Recall-Werte der Regelkerne bezogen auf die Gesamtmenge der Belege berechnet (s. Abbildung 7.4 und Tabelle 10.4), um die Entwicklung der Regelabdeckung einschätzen zu können. Als Vergleichsbasis für das vorgestellte Verfahren dient dabei der Recall der Regelkerne bei manueller Erstellung der Belegdatenbank. Der Benutzer muss 2.000 Belege manuell betrachten, um denselben Recall zu erreichen wie mit dem automatischen Verfahren. Betrachtet man den manuellen Aufbau der Testkollektion bezogen auf die Regelhäufigkeit, so zeigt sich, dass der Benutzer über 9.000 Belege manuell erzeugen muss, um den Recall so zu erhöhen, wie es mit den automatisch erzeugten Belegen möglich ist. Ein hoch signifikanter Anstieg ist mit einem p-Wert $< 2.2e\text{-}16$ für die Regelkerne auf Basis der Gesamtdaten in allen Fällen klar erkennbar.

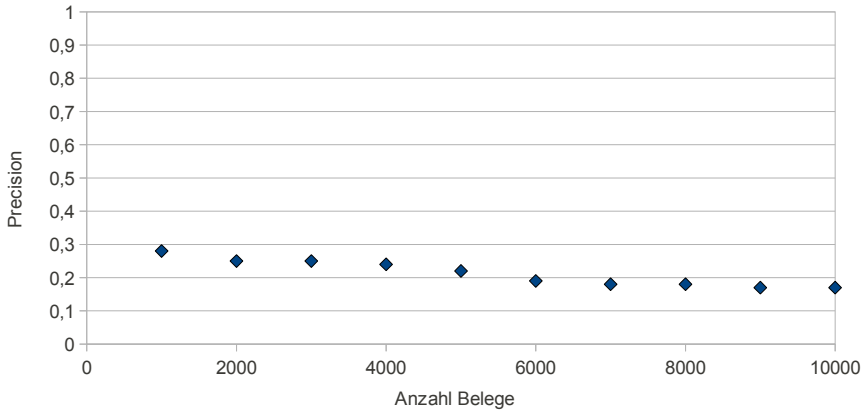


Abbildung 7.2.: Precision für Regelkerne beim Aufbau der Belegdatenbank

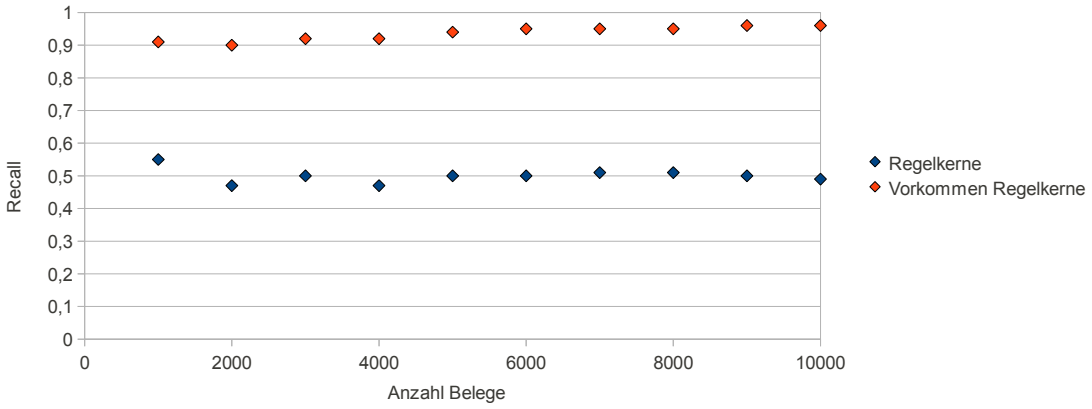


Abbildung 7.3.: Recall für Regelkerne beim Aufbau der Belegdatenbank

Um festzustellen, wie viele Belege mit den generierten Regelkernen wiederzufinden sind, wird auch der Recall für die Belegabdeckung berechnet (s. Abbildung 7.5 und Tabelle 10.5). Dies geschieht sowohl für die automatische als auch für die manuelle Vorgehensweise. In beiden Fällen werden mit den aus 1.000 Belegen generierten Regeln bereits fast 90 % der Belege abgedeckt. Dadurch wird nochmals die Regelhaftigkeit von Schreibvarianten verdeutlicht. Der Anstieg der Belegabdeckung ist mit einem p-Wert $<$ 2.2e-16, sowohl bei der manuellen als auch bei der automatischen Erstellung der Belege, hoch signifikant. Allerdings muss der Benutzer etwa 2.000 Belege manuell bewerten, um denselben Recall wie beim automatischen Ansatz zu erzielen.

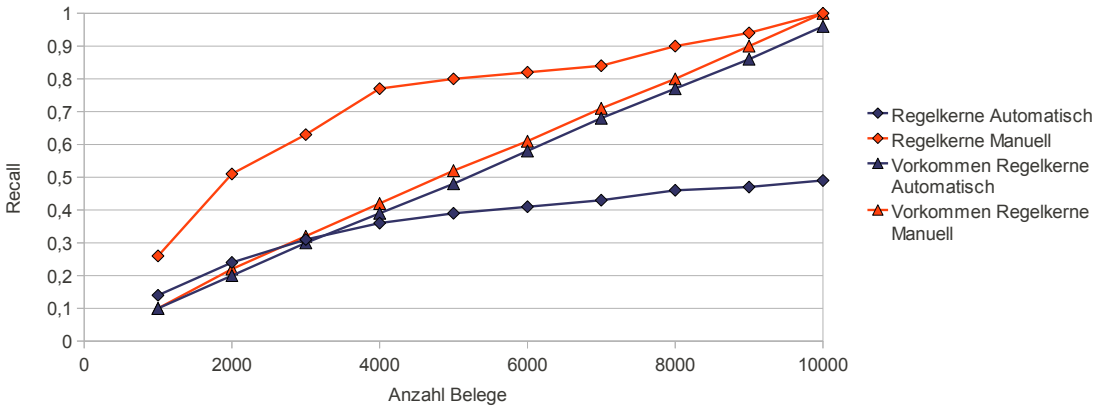


Abbildung 7.4.: Recall für Regelkerne auf Basis der Gesamtdaten beim Aufbau der Belegdatenbank

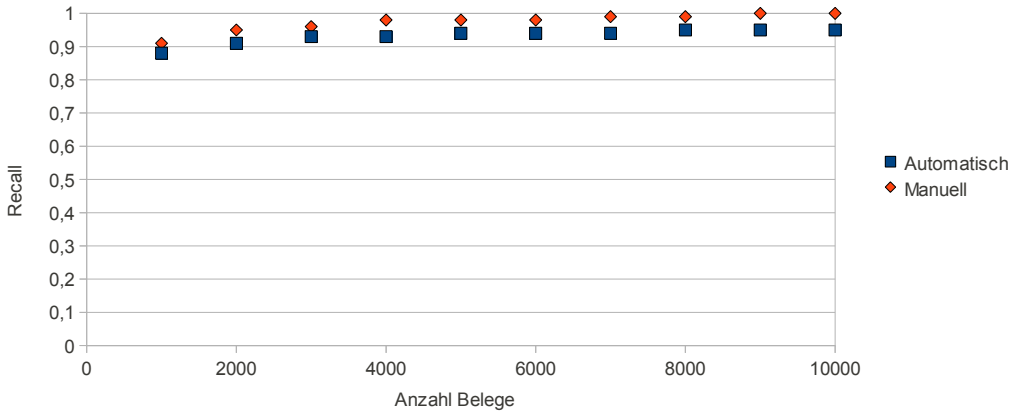


Abbildung 7.5.: Recall Belegabdeckung beim Aufbau der Belegdatenbank

Zusammenfassung

Wie die Evaluierung deutlich demonstriert hat, werden die häufigen Regelkerne ausgewählt. Wenn der Benutzer diese direkt akzeptiert, muss er sich nur noch die Wörter anschauen, bei denen die automatische Zuordnung eines Vorschlags nicht möglich ist. Somit konnte gezeigt werden, dass die automatische Belegerstellung eine enorme Zeitersparnis ermöglicht. Werden die bereits genannten Werte mit etwa einem Personenmonat für die Erstellung von 1.000 Belegen zugrunde gelegt, so lassen sich mit dem vorgestellten Algorithmus je nach angestrebter Regelqualität pro Kollektion mehrere Personenmonate an Arbeitszeit einsparen.

7.2.2. Evaluierung der Parameter

Für die Evaluierung der Parameter werden aus der Kollektion zufällig jeweils 10 Dokumente aus dem 16. bis 19. Jahrhundert ausgewählt [Ernst-Gerlach und Fuhr, 2010a]. Es ist davon auszugehen, dass die Korrektheit der Vorschläge der Rechtschreibprüfung abnimmt, je älter die Dokumente sind. Deshalb werden verschiedene Läufe für jedes Jahrhundert durchgeführt. Die Nullhypothese besagt dementsprechend, dass Recall und Precision nicht steigen. Demgegenüber geht die Alternativhypothese von steigenden Werten aus. Um die Sprachunabhängigkeit des Ansatzes zu zeigen, werden die Experimente ebenfalls für zehn zufällig ausgewählte Dokumente der Shakespeare-Kollektion[28] durchgeführt.

Für jede Teilkollektion wird zunächst die automatische Belegerstellung angewendet. Anschließend werden die Ergebnisse anhand von 200 zufällig ausgewählten unbekannten Wörtern überprüft. Dabei werden jeweils unterschiedliche Läufe mit maximal einer, zwei bzw. drei Regelanwendungen durchgeführt. Die Recall- und Precision-Werte werden basierend auf einer minimalen Vorkommenshäufigkeit der Regeln von 2, 5 und 10 berechnet.

Bei allen Durchläufen werden nur Wörter berücksichtigt, die mindestens fünf Buchstaben haben. Die Ergebnisse sind in den Tabellen 10.6, 10.7 und 10.8 enthalten. Es werden zwei verschiedene Precision-Werte berechnet. Der erste basiert auf allen akzeptierten Belegen, während der zweite nur Belege berücksichtigt, die tatsächlich Schreibvarianten enthalten. Als Vergleichsbasis für diese Evaluierung wird jeweils der erste Vorschlag der Rechtschreibprüfung für jedes unbekannte Wort ausgewählt und die entsprechenden Recall- und Precision-Werte werden berechnet. Weil sich der Recall-Wert nicht verändert, wenn auch Belege Berücksichtigung finden, die keine Schreibvarianten enthalten, wird der Recall lediglich einmal berechnet.

Zeitliche Evaluierung

Die Rechtschreibprüfung bietet im Durchschnitt 5,4 Vorschläge pro unbekanntem Wort für das 16. Jahrhundert, 4,9 für das 17. und 18. Jahrhundert und 4,6 Wörter für Dokumente aus dem 19. Jahrhundert an. Somit sinkt die Anzahl der Vorschläge für die moderneren Wörter leicht. Die Ergebnisse werden anhand eines Cochran-Armitage-Tests überprüft.

Die Precision-Werte (s. Abbildung 7.6 und Tabelle 10.6) für die verschiedenen Jahrhunderte steigen, wie erwartet, mit der Zeit. Eine Ausnahme bildet das 19. Jahrhundert. Der Prozentsatz der unbekannten Wörter sinkt mit der Zeit (s. Tabelle 7.3). Zusätzlich ist die Anzahl der Types im 18. Jahrhundert größer als im

[28] `http://www.perseus.tufts.edu/hopper/collection?collection=Perseus:collection:Renaissance` (letzter Aufruf 18.12.12)

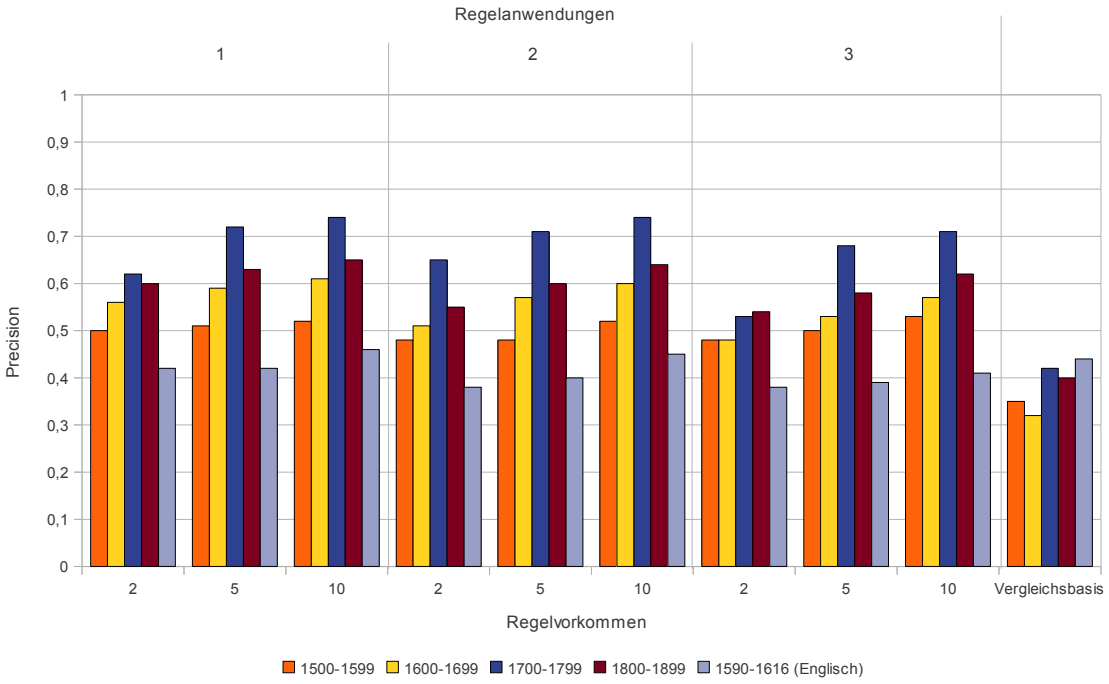


Abbildung 7.6.: Precision für unterschiedliche Parameter auf Basis aller unbekannten Terme

Deutsch				**Englisch**
1500-1599	**1600-1699**	**1700-1799**	**1800-1899**	**1590-1616**
0,36	0,33	0,10	0,07	0,06

Tabelle 7.3.: Anteil unbekannter Wörter bei der Evaluierung der Parameter

19. Jahrhundert. Somit sind im 19. Jahrhundert nur halb so viele Wörter unbekannt wie im 18. Jahrhundert. Die daraus resultierende kleinere Trainingsmenge hat zu der sinkenden Precision geführt. Trotzdem ist bei der Entwicklung der Precision-Werte in 7 von 9 Fällen jeweils ein signifikanter Anstieg zu erkennen. Die p-Werte liegen zwischen 0,9 % und 3,93 % basierend auf allen Termen bzw. zwischen 0,0604 % und 4,588 % bei einer Betrachtung der Schreibvarianten. Eine Ausnahme bilden hier 2 bzw. 3 Regelanwendungen, wenn die Regeln nur zweimal vorkommen müssen. Hier ergibt sich ein maximaler p-Wert von 26,04 %. Der Recall steigt ebenfalls mit der Zeit (s. Abbildung 7.7 und Tabelle 10.8). Ausnahmen bilden drei Regelanwendungen im 19. Jahrhundert und drei Regelanwendungen mit zwei Regeln für das 16. Jahrhundert. Der Recall steigt jedoch in allen Fällen signifikant. Der p-Wert liegt dabei zwischen 0,1138 % und 1,456 %. Insgesamt zeigt sich eine mit der Zeit steigende Qualität des Ansatzes, vorausgesetzt, man hat ungefähr gleich große Trainingsmengen.

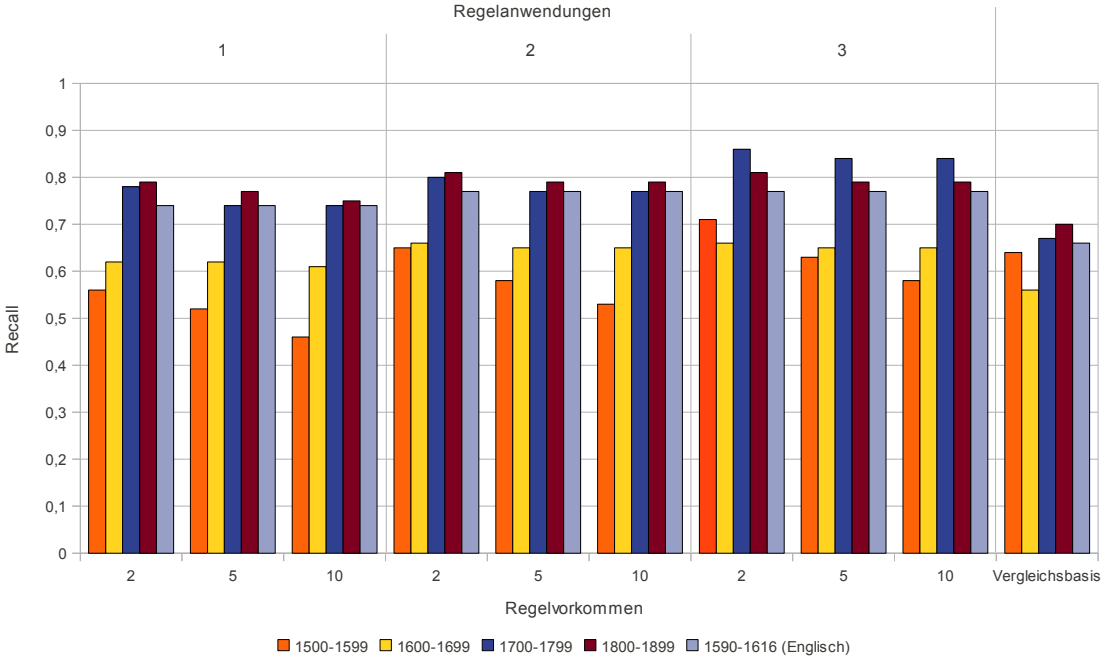


Abbildung 7.7.: Recall für unterschiedliche Parameter

Beschränkung der Anzahl der Regelanwendungen

Die Precision sinkt in drei von vier Fällen, wenn die Anzahl der Regelwendungen steigt. Dagegen steigen alle Recall-Werte. Trotz der geringen Anzahl an Durchläufen wird deutlich, dass die Beschränkung der Anzahl der Regelanwendungen ein sinnvoller Parameter für die Qualität der automatischen Belege ist. Allerdings ist die Steigerung der Precision nur sehr gering, wenn die Anzahl der Regelanwendungen zu stark beschränkt wird.

Minimale Anzahl an Regelvorkommen

Die minimale Anzahl der Regelvorkommen erreicht bessere Ergebnisse als die Beschränkung der Anzahl der Regelanwendungen. Für das Deutsche finden sich lediglich zwei Durchläufe (beide für das 16. Jahrhundert) bei denen die Precision zwischendurch sinkt.

Die Verbesserungen sind auch für neuere Texte größer. Die Unterschiede für die verschiedenen Regelvorkommen sind nur gering. In diesem Fall müsste der Schwellenwert höher sein, um einen merklichen Effekt für die Precision-Werte zu erzielen. Wie erwartet, sinkt der Recall-Wert in allen Fällen für das Deutsche. Auffällig ist der Recall-Wert für das Englische. Hier ist lediglich ein Unterschied festzustellen, wenn nur eine Regel angewendet wird. Die minimale Anzahl der

Regelvorkommen hat keine Auswirkungen auf den Recall. Weil ungefähr drei von vier Wörtern bereits gefunden werden, macht es den Eindruck, dass die Parametereinstellungen für das Englische gut gewählt sind.

Unterschiedliche Precision-Werte

Die Precision für alle unbekannten Terme zeigt, wie groß die Anzahl der fehlerhaften Belege ist. Wenn Regeln aufgrund von fehlerhaften Belegen generiert werden, gehen keine Belege verloren. Deswegen ist ebenfalls interessant, die Precision nur für die historischen Varianten zu betrachten (s. Abbildung 7.8 und Tabelle 10.7). Wie erwartet, ist die Precision für diese deutlich höher. Die höchste Precision liegt bei 0,82 für das 18. Jahrhundert bei nur einer Regelanwendung, wenn die Regel mindestens fünfmal vorkommt. Keiner der beschränkten Precision-Werte ist für das Deutsche niedriger als 0,57. Somit ist mehr als jeder zweite ausgewählte Vorschlag eine Variante. Selbst bei der Betrachtung des niedrigsten Precision-Wertes für alle unbekannten Wörter (0,48) ist fast jeder zweite Beleg richtig.

Die Precision-Werte für das Englische sind niedriger als die für das Deutsche. Bei der beschränkten Precision ist aber ebenfalls fast jeder zweite Vorschlag korrekt.

Abbildung 7.8.: Precision für unterschiedliche Parameter beschränkt auf Schreibvarianten

Vergleichsbasis

Für die deutschen Beispiele ist die Precision der automatisch ausgewählten Belege im Vergleich mit dem ersten Vorschlag der Rechtschreibprüfung immer deutlich besser. Mit Ausnahme des 16. Jahrhunderts gilt dies auch für den Recall. Für das 16. Jahrhundert wird der Recall sogar von den meisten Durchläufen mit weniger restriktiven Parametereinstellungen von der Vergleichsbasis übertroffen. Im Gegensatz zu der automatischen Belegerstellung weisen weder die Precision-Werte ($p = 7{,}699\,\%$ bzw. $p = 13\,\%$) noch der Recall-Wert ($p = 12{,}54\,\%$) der Vergleichsbasis einen signifikanten Anstieg der Werte auf.

Bei den Ergebnissen für die Durchläufe, die auf den Shakespeare-Dokumenten basieren, ist der Recall immer besser als die Vergleichsbasis. Für die Precision, die auf allen Wörtern basiert, sind die Ergebnisse unterschiedlich. Werden nur die Precision-Werte für die historischen Varianten betrachtet, ist die Precision für die Vergleichsbasis (0,58) besser als die beste Precision für die automatische Belegauswahl (0,50). Bei einer Erhöhung der minimalen Anzahl der Regelvorkommen auf 100 ist die Precision identisch mit der Vergleichsbasis, aber der Recall ist immer noch höher als bei der Vergleichsbasis (0,71 zu 0,66). Demzufolge sind die Parametereinstellungen für die englischen Testdaten entgegen der ursprünglichen Annahme nicht optimal gewählt.

Zusammenfassung

Die Ergebnisse für das 18. Jahrhundert sind bemerkenswert, wenn der Schwellenwert für die minimale Anzahl der Regelhäufigkeit von zwei auf fünf erhöht wird. Die Precision steigt dann bei einer Regelanwendung von 0,62 auf 0,72 für unbekannte Wörter und von 0,70 auf 0,82 speziell für Schreibvarianten. Eine minimale Anzahl von fünf Regelanwendungen ist, wie genauere Untersuchungen ergeben haben, in diesem Fall ein sehr guter Schwellenwert. Auf diese Weise wird eine große Anzahl falscher Belege nicht automatisch akzeptiert. Dies verdeutlicht die Wirksamkeit des Parameters.

Besonders die Dokumente aus dem 16. und 17. Jahrhundert enthalten viele Lehnwörter. In den meisten Fällen sind diese in lateinischer Sprache. Einige Dokumente beinhalten sogar ganze Sätze in lateinischer Sprache. Um diese Terme nicht als unbekannte Wörter einzustufen, könnten Werkzeuge zur Sprachidentifizierung zum Einsatz kommen. Auf diese Weise wäre es auch möglich, für die betreffenden Passagen eine Rechtschreibprüfung für die entsprechende Sprache einzusetzen und passende Regeln zu generieren.

Für Einfüge- und Löschregeln könnte die Precision durch Einbeziehung von minimalem Kontext gesteigert werden. Beispielsweise könnte die Einfügeregel $\varepsilon \to h$ durch Regeln der Form $t \to th$ ersetzt werden.

Pilz und Luther haben ihren Bayes-Klassifizierer mit verschiedenen Parametereinstellungen evaluiert [Pilz und Luther, 2009]. Für ihre 8.000 Belege zum Training haben sie Belege aus derselben Kollektion verwendet, auf der auch die vorliegende Evaluierung basiert. Das beste Ergebnis war eine Precision von 73,08 % für die Klassifizierung bei Schreibvarianten. Dieser Wert wurde mit Quadrigrammen in Kombination mit einem gegebenen Wörterbuch erzielt. Der Evaluierung liegen teilweise andere Testdaten aus derselben Kollektion zugrunde. Trotzdem wird deutlich, dass die Precision-Werte des automatischen Ansatzes (zwischen 0,48 und 0,74) relativ gut sind, da sie im Gegensatz zum Bayes-Klassifizierer ohne Trainingsdaten auskommen.

Basierend auf der Evaluierung muss eine Annahme korrigiert werden: Es bestehen auch Regularitäten zwischen Schreibvarianten und falschen Vorschlägen. Weil diese Regularitäten teilweise mit denen zwischen Schreibvariante und korrektem Vorschlag übereinstimmen, können die falschen Vorschläge sogar die korrekten Vorschläge bestätigen. Beispielsweise kommt die Regel $i \rightarrow y$ in der Tabelle 7.1 siebenmal vor, obwohl nur zwei korrekte Vorschläge in der Tabelle enthalten sind.

7.2.3. Zusammenfassung

Die experimentellen Ergebnisse haben gezeigt, dass die automatischen Belege den Prozess der Beleggenerierung sehr gut unterstützen können. Mit den generierten Belegen lassen sich die häufigsten Regeln bilden. Daher kann sofort ein Regelsatz generiert werden, der die Retrievalqualität deutlich verbessert, ohne dass der Benutzer die Regeln zwingend bearbeiten muss. Durch eine geeignete Parameterauswahl je nach Kollektion und Sprache kann der Prozess vom Benutzer gesteuert werden. Die automatische Beleggenerierung hat somit den Engpass des entwickelten Verfahrens beseitigt. Im folgenden Abschnitt wird das darauf aufbauende Werkzeug zur semi-automatischen Regelgenerierung vorgestellt.

7.3. RuleGenerator

Die meisten Humanisten haben eine recht hohe Hemmschwelle bei der Nutzung neuer Technologien [Rimmer u. a., 2006]. Dies gilt auch bei speziell für diesen Bereich entwickelten Applikationen [Juola, 2006]. Das kann sowohl an dem fehlenden Komfort der vorhanden Anwendungen als auch an dem hohen Vertrauen in die herkömmlichen Verfahren liegen. Dies lässt sich durch die fehlende Deckung der Vorstellungen der Entwickler mit den Erwartungen der Wissenschaftler an entsprechende Werkzeuge erklären. Dieses Problem muss bei der Entwicklung von Werkzeugen für die digitalen Geisteswissenschaften berücksichtigt werden, damit sie auch zur Unterstützung bei der Bearbeitung von Forschungsfragen genutzt werden [Garces, 2006]. Zudem stellen ungewohnte

Betriebssysteme eine zusätzliche Hürde dar [Crane, 2007]. Der RuleGenerator soll deswegen plattformunabhängig implementiert werden und es Benutzern ohne Informatikkenntnisse ermöglichen, Regeln für ein Korpus zu erstellen. Dazu muss er die Bedürfnisse der Zielgruppe nach einer geringen Nutzungshemmschwelle und einer intuitiven Bedienung berücksichtigen [Johnson, 2006]. Die entsprechenden Anwendungen müssen deswegen extrem benutzerfreundlich sein.

Bisher können Regeln nur mit einer Konsolenanwendung erstellt und mit einem Editor weiterbearbeitet werden. Mit dem RuleGenerator [Ernst-Gerlach u. a., 2011] (s. Abbildung 7.9) wird ein Werkzeug entwickelt, das dem Benutzer die automatische Generierung von Belegen und Regeln mit einer graphischen Benutzerschnittstelle ermöglicht. Eine Visualisierung unterstützt den Benutzer im Bereich des Data Mining bei der Bildung von neuen Regeln. Dies wurde von Nguyen und Kollegen gezeigt [Nguyen u. a., 2006]. Demzufolge sollen insbesondere Übersichten über die Belege sowie die Regelmengen und -kandidaten visualisiert werden. Ausgehend von seinen Bedürfnissen wird sich der Benutzer mehr auf den Recall oder die Precision seiner Suche konzentrieren. Das Werkzeug soll an dieser Stelle die notwendige Flexibilität bieten. Dabei sollen Änderungen in einer Komponente automatisch in die anderen Komponenten übernommen werden, ohne dass eine Aktualisierung der Ansicht erforderlich ist [Hearst, 1999].

Vor der Erstellung des RuleGenerators wurde in der Masterarbeit [Awakian, 2010] zunächst eine Anforderungsanalyse durchgeführt. Darauf aufbauend entstand ein Design-Konzept. Hierfür wurden zunächst mit einer hierarchischen Aufgaben-Analyse die Teilaufgaben festgelegt. Anschließend wurde das konzeptionelle Modell der Benutzeroberfläche mit Hilfe von Mockups umgesetzt und mit kognitiven Durchgängen getestet, bevor das Konzept implementiert wurde.

Bei der Analyse ergaben sich die Schwerpunkte Bildung und Bearbeitung von Belegen sowie Generierung und Bearbeitung von Regeln. Deswegen wird zunächst eine horizontale Zweiteilung in die Komponenten SmartEvidencer (s. Abbildung 7.9 oben) für die Belege und RuleModification (s. Abbildung 7.9 unten) für die Regeln vorgenommen. Der SmartEvidencer gliedert sich wiederum in die Komponenten Evidences (rechts oben) zur Belegsammlung und -bearbeitung und HistoricText (links oben) mit den Textquellen. Die RuleModification ist in die Komponenten RuleSelector- und die RuleVisualization aufgeteilt. Die RuleSelector-Komponente (links unten) erlaubt es dem Benutzer, durch die Regelmenge zu navigieren und bestimmte Regeln zu finden. Weil der Benutzer mit graphischen Darstellungen schneller und effektiver arbeiten kann [Hearst, 1999], werden in der RuleModification die Details ausgewählter Regeln rechts unten in der Visualisierungs-Komponente dargestellt. Sowohl beim SmartEvidencer als auch bei der RuleModification wird jeweils in der linken Teilkomponente der Überblick gegeben und entsprechend der Lese- und Interaktionsrichtung werden rechts weitere Details zur Verfügung gestellt. Bei der Erstellung bzw. Auswahl eines neuen Belegs in der Evidencer-Komponente erfolgt die Anzeige der daraus erstellten Regeln in der darunter liegenden

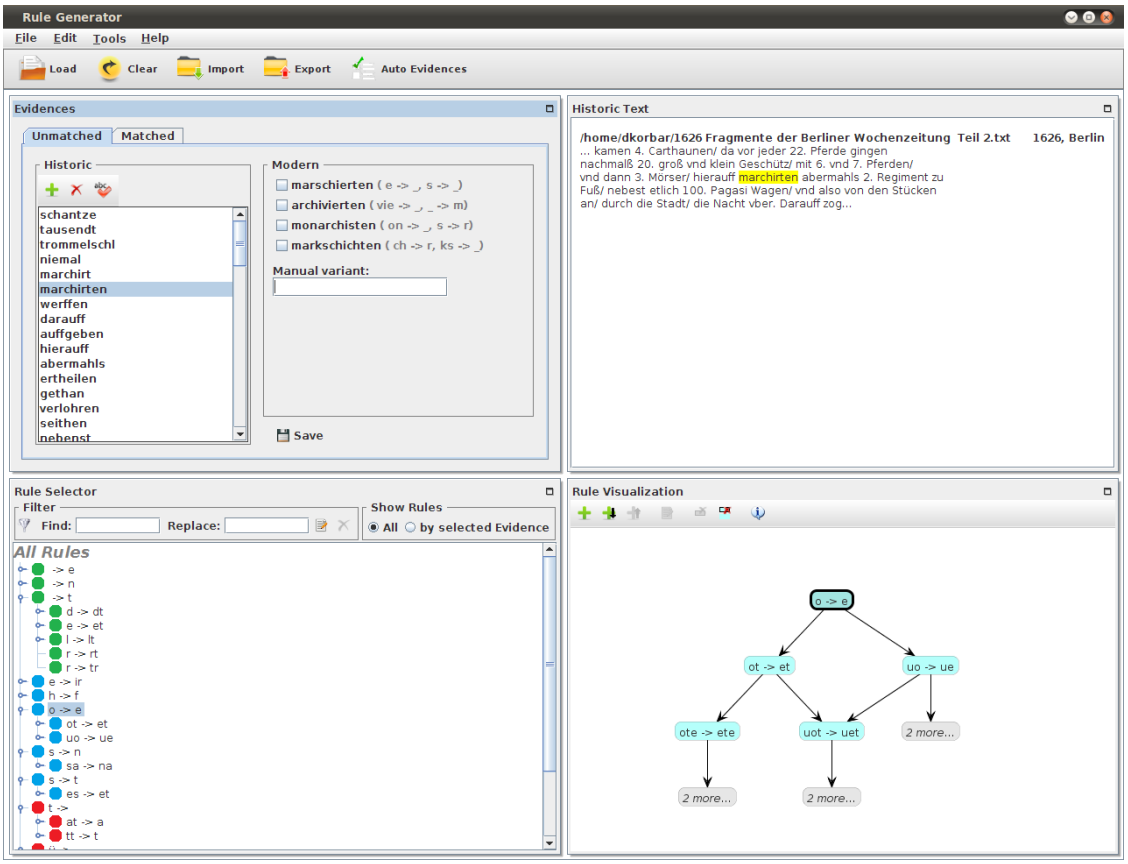


Abbildung 7.9.: RuleGenerator

Komponente RuleSelector. Durch diese Gestaltung der Oberfläche soll sich die Anzahl der Interaktionen mit diagonalen Sichtbeziehungen reduzieren. Im Menü des RuleGenerators bietet sich die Gelegenheit zum Im- und Export der Regeln. Um mögliche Bedienungsfehler zu minimieren, werden Funktionen ausgegraut, deren Einsatz in der jeweiligen Situation des Benutzers nicht sinnvoll ist. Die einzelnen Komponenten des RuleGenerators werden im Folgenden näher erläutert.

7.3.1. SmartEvidencer

Der SmartEvidencer dient zur Sammlung der Belege. Für jeden erstellten Beleg werden sofort Regeln generiert. Auf diese Weise wird eine iterative Erstellung der Regeln umgesetzt, und die Bereitstellung eines ersten Regelsatzes kann jederzeit erfolgen. In den Prozess des Belegesammelns ist auch die automatische Beleggenerierung integriert. Die automatischen Belege ermöglichen es, die Trainingsdaten deutlich schneller und mit geringerem manuellen Aufwand zu erzeugen. Der Benutzer kann diesen Prozess beeinflussen, indem er beim Laden

der Texte die minimale Wortlänge, die maximale Anzahl der Regelanwendungen pro Wort sowie die minimale Anzahl an Regelvorkommen als Parameter einstellt (s. Abbildung 7.10). Darüber hinaus können Zeit und Ort für den zu erstellenden Regelsatz eingegeben und durch die Auswahl der Sprache das Wörterbuch für die Rechtschreibprüfung festgelegt werden. Dabei stehen automatisch die Sprachen zur Verfügung, für die das entsprechende Wörterbuch von Hunspell hinterlegt ist.

Die Ergebnisse dieses Prozesses werden dem Benutzer in einer Liste präsentiert (s. Abbildung 7.11). Diese zeigt die modernen und historischen Formen sowie den Regelkern an. Die Anzeige der Regelkerne unterstützt den Benutzer bei der Einordnung der vorgeschlagenen Belege. So sind z. B. sowohl die phonetische Ähnlichkeit der Regelkerne als auch die Anzahl der benötigten Regelkerne Indizien für korrekte Belege. Der Benutzer kann anschließend einzelne Belege oder alle Belege gleichzeitig bearbeiten. Dabei kann er die Belege entweder direkt oder provisorisch akzeptieren. Vorläufig akzeptierte Belege werden wie akzeptierte Belege behandelt. Allerdings werden sie in der Belegliste mit einem Fragezeichen markiert. Diese Belege können zu einem späteren Zeitpunkt noch endgültig anerkannt werden. Auf diese Weise kann der Benutzer aus den erzeugten Belegen Regeln generieren und mit diesen nach Dokumenten suchen. Zu einem späteren Zeitpunkt kann er die provisorischen Belege überprüfen und den Regelsatz überarbeiten. Diese Vorgehensweise setzt die von Mangu und Brill vorgeschlagene Kombination von manuellen und automatischen Methoden zum Regellernen um, bei der der Benutzer nur noch die unsicheren Regeln bzw. in diesem Fall auch die unsicheren Belege bearbeiten muss [Mangu und Brill, 1997].

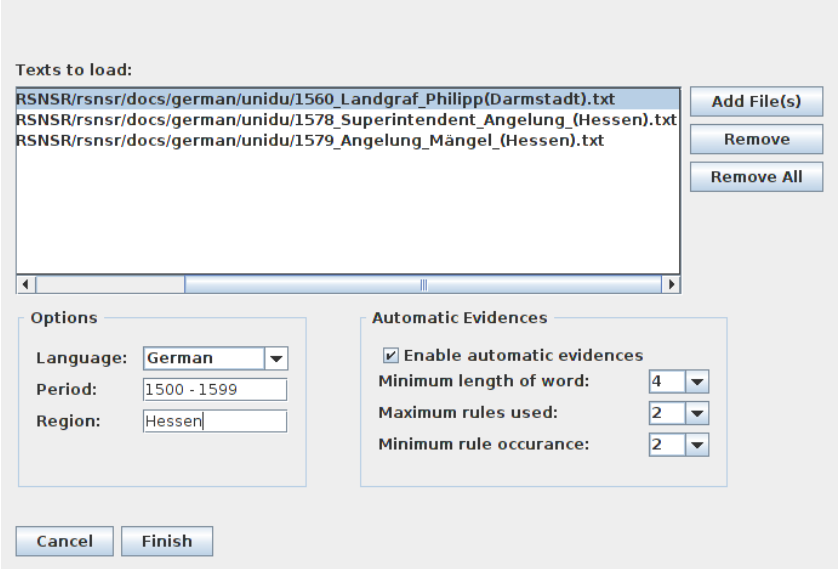


Abbildung 7.10.: Text laden

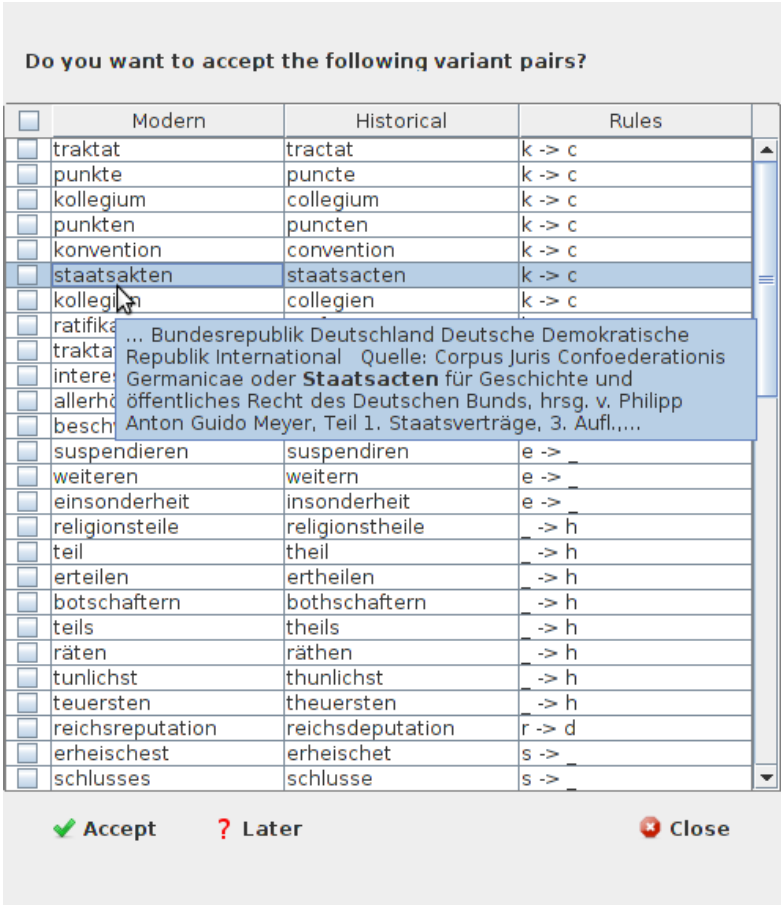


Abbildung 7.11.: Benutzeroberfläche für automatische Belege

Evidencer

Im Anschluss an die Bearbeitung der automatischen Belege kann der Benutzer in dem Unmatched-Reiter (s. Abbildung 7.9 oben links) aus den noch nicht zugeordneten Wörtern weitere Belege bilden. Dazu werden dem Benutzer die unbekannten Wörter in einer Liste angezeigt. Zu einem ausgewählten Wort kann er entweder einen Vorschlag der Rechtschreibprüfung akzeptieren oder selbst einen Vorschlag eingeben. Zu jedem Vorschlag der Rechtschreibprüfung werden ebenfalls die entsprechenden Regelkerne für eine Abbildung auf den unbekannten Term mit angezeigt. Es besteht zudem die Möglichkeit, der Liste neue Schreibvarianten hinzuzufügen. Außerdem können Wörter, bei denen es sich nicht um eine Schreibvariante handelt, aus der Liste gelöscht bzw. einem Wörterbuch hinzugefügt werden. Der Matched-Reiter zeigt die akzeptierten Belege mit den Regeln an. Zudem besteht in dieser Ansicht auch die Möglichkeit, bereits bestätigte Regeln noch weiter zu bearbeiten. Beispielsweise lassen sich provisorisch anerkannte Belege endgültig akzeptieren oder es kann zu einer historischen Form eine weitere moderne

Form angegeben werden. Fehlerhafte Belege können wieder gelöscht werden. Eine Filterung der Belege kann anhand des Status (provisorisch bzw. endgültig akzeptiert) erfolgen.

HistoricText

Die HistoricText-Komponente stellt die möglichen Schreibvarianten in Textauszügen von 40 Wörtern Länge hervorgehoben dar. Auf diese Weise kann der Benutzer bei Bedarf aus den Kontextinformationen die genaue Bedeutung des Wortes erschließen. Falls der Textauszug nicht ausreichend ist, kann auch eine Anzeige des gesamten Textes erfolgen.

7.3.2. RuleModification

Die Bearbeitung der aus den Belegen generierten Regeln erfolgt in der RuleModification-Komponente (s. auch [Korbar, 2010]). Diese Komponente soll sowohl die Erstellung von Regeln als auch das Löschen und Bearbeiten von Regeln unterstützen.

Visualisierungen werden häufig eingesetzt, um gelerntes Wissen in einer für den Benutzer verständlichen Form darzustellen [Couturier u. a., 2007]. Sie ermöglichen es, große Mengen an Informationen effizienter und effektiver zu verstehen [Foo und Hendry, 2007]. Weil der Benutzer nicht nur Regeln generieren, sondern auch die Möglichkeit bekommen soll, Regeln selbst zu bearbeiten, muss eine geeignete Form der Regelvisualisierung und -bearbeitung entwickelt werden. Auf diesem Gebiet sind bereits einige verwandte Arbeiten entstanden. So wurde z. B. von Pilz und Kollegen ein Werkzeug zur Evaluierung von Abstandsmaßen (s. Abbildung 7.12) vorgestellt, das neben der Visualisierung auch die Möglichkeit bietet, die Abstandsmaße in einer Tabelle zu bearbeiten [Pilz u. a., 2007].

Weil die meisten Regeln für Schreibvarianten eine Anwendung in beliebiger Reihenfolge erlauben, ergeben sich viele Kombinationsmöglichkeiten und Permutationen. Deswegen wurde von Kempken und Kollegen eine Visualisierung von Regeln vorgestellt, mit der sich z. B. relevante Regeln oder auch Redundanzen finden lassen, um Regelsätze zu optimieren [Kempken u. a., 2007]. Abbildung 7.13 führt z. B. alle Varianten zu *genannt* auf. Die farbig dargestellten Rechtecke stellen die vom Benutzer ausgewählten Varianten dar. Die grünen Rechtecke stehen für Varianten, die auch in Dokumenten vorkommen. In der TreeMap-Visualisierung kann für ausgewählte Varianten überprüft werden, mit welchen Regeln die Bildung der Variante möglich ist.

Das neu zu entwickelnde Werkzeug soll übersichtlich aufgebaut sein, damit der Benutzer es möglichst intuitiv bedienen kann. Insbesondere ist eine geeignete

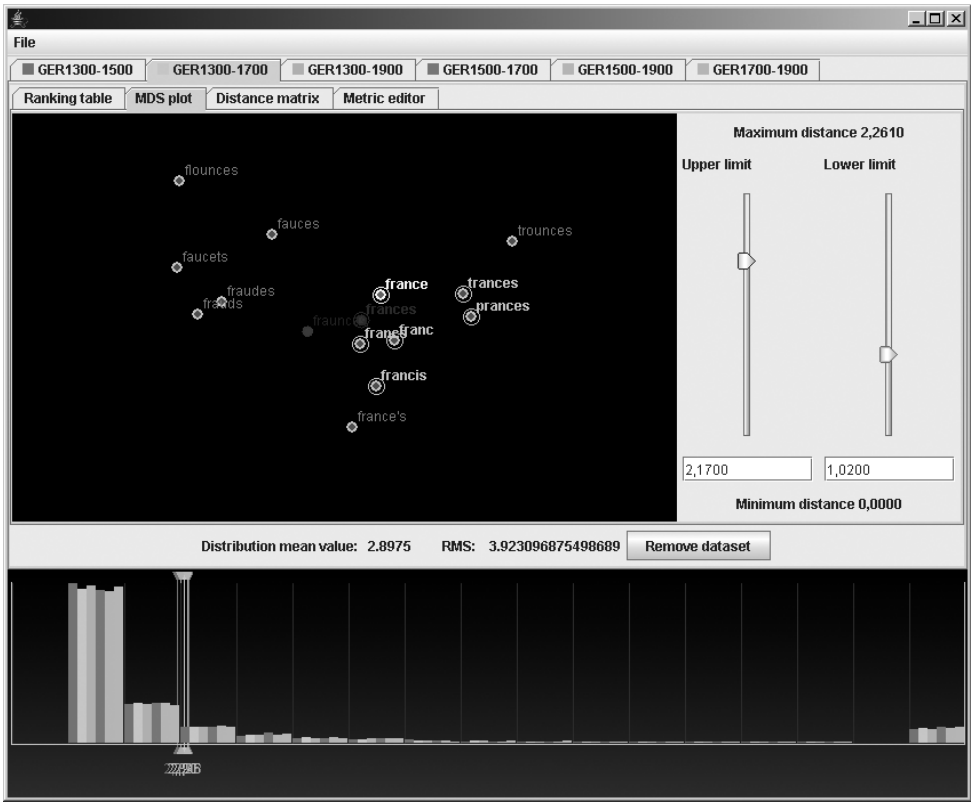


Abbildung 7.12.: MetricEvaluationTool [Pilz u. a., 2007]

Visualisierung der Regeln erforderlich. Dazu wurden in der Diplomarbeit [Korbar, 2010] einige Verfahren untersucht.

RuleSelector

Der RuleSelector soll dem Benutzer einen Überblick über die gesamte Regelmenge geben und ihn bei der Suche nach einer bestimmten Regel unterstützen. Zu diesem Zweck wird ein Filter implementiert, der es ermöglicht, die vorhandene Regelmenge anhand der folgenden Regelkriterien zu filtern (s. Abbildung 7.14):

- Es können Vorgaben zum Find- und Replace-Teil der Regeln gemacht werden. Dabei erfolgt die Filterung direkt bei der Eingabe durch den Benutzer, so dass er sofort einen Überblick über das Filterergebnis erhält. Der gefundene Kontext einer Regel bekommt bei der Darstellung eine rote Markierung. Oberregeln, die Filterkriterien selbst nicht erfüllen, aber entsprechende Unterregeln haben, werden zur besseren Einordnung der gefilterten Regeln in ausgegrauter Form ebenfalls angezeigt.

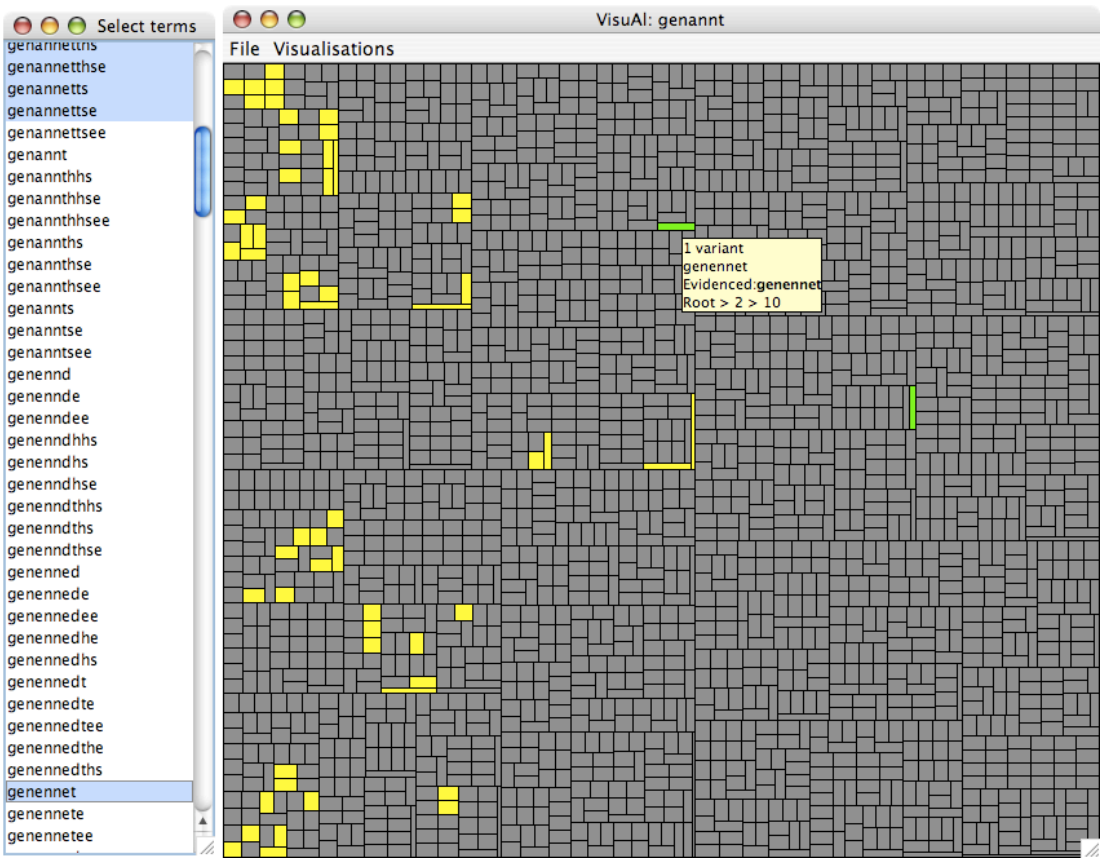


Abbildung 7.13.: TreeMap Visualisierung von Regeln [Kempken u. a., 2007]

- Außerdem lässt sich die Suche auf den Regeltyp (insert, delete oder substitute) beschränken.
- Darüber hinaus kann sich der Benutzer sowohl alle Regeln anzeigen lassen als auch nur die Regeln für einen konkreten Beleg.

Die Visualisierung der Regeln erfolgt in einer Baum-basierten Darstellung. Dabei werden zwar Regeln mit mehreren Oberregeln zweifach im Baum abgebildet, aber im Gegensatz zur Listendarstellung ist die Regelstruktur für den Benutzer deutlich besser erkennbar. Außerdem ist der Anteil von Regeln mit mehreren Oberregeln relativ gering. Die Analyse eines Regelsatzes hat ergeben, dass nur etwa 5 % der untersuchten Regeln mehr als eine Oberregel besaßen (Details s. [Korbar, 2010]).

RuleVisualization

Die RuleVisualization-Komponente ermöglicht es dem Benutzer, die im RuleSelector ausgewählte Regel im Detail zu betrachten. Außerdem erlaubt sie es, die selektierte Regel zu modifizieren, zu löschen oder neue Regeln in Abhängigkeit

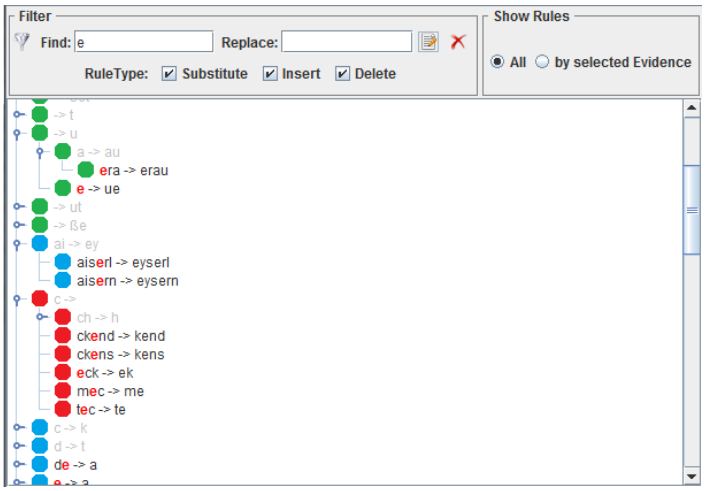


Abbildung 7.14.: RuleSelector

der ausgewählten Regel zu erstellen. Eine Regel kann sowohl mehrere Ober- als auch Unterregeln haben. Deswegen lassen sich Regeln nicht in einem Baum, sondern nur in einem gerichteten azyklischen Graphen darstellen. Bei Unterregeln wird der von der Oberregel geerbte Kontext um zusätzliche Kontextinformationen erweitert. Dadurch verschlechtert sich gegebenenfalls der Recall. Da angenommen wird, dass kürzere Regeln eine größere Anzahl an positiven Beispielen abdecken [Fürnkranz, 1999], kann sich aber die Precision der Regel verbessern. Demzufolge sollte auch das Gewicht der Unterregeln höher als das der Oberregeln sein. Für die Visualisierung wurden zunächst die im JUNG-Framework[29] vorhandenen Layout-Algorithmen Directed Acyclic Graph, Fruchterman-Reingold, Sugiyama und Tree Layout für Graphen evaluiert (Details s. [Korbar, 2010]). Neben der Darstellung eines gerichteten azyklischen Graphen sollte dabei die Hierarchie der Regeln visualisiert werden. Knoten mit der gleichen Oberregel sollen auch nebeneinander gruppiert werden. Dabei ist die Anzahl der sich überschneidenden Kanten zu minimieren. Weil die untersuchten Algorithmen nicht direkt für die gegebenen Daten geeignet sind, entstand ein Konzept für ein Layout (s. Abbildung 7.9 unten rechts), welches Ansätze aus den bestehenden Verfahren übernimmt (Details s. [Korbar, 2010]). Zur optischen Platzierung der Knoten des Graphen wird die hierarchische Einordnung von Knoten des Sugiyama-Layouts mit der rekursiven Zeichenweise von üblichen Tree-Layouts kombiniert. Dabei wird der Wurzelknoten auf der obersten Ebene angeordnet, während alle anderen Knoten in Abhängigkeit von der maximalen Anzahl passierter Kanten bis zum Wurzelknoten angeordnet werden. Daraus ergibt sich ein Layout, bei dem alle Kanten nach unten gerichtet sind. Um die Übersichtlichkeit der Visualisierung nach dem Fokus-und-Kontext-Prinzip zu verbessern, werden nur die Regeln angezeigt, die für die ausgewählte Regel relevant sind. In expandierbaren Knoten werden weitere Regeln zusammengefasst und bei

[29] `http://jung.sourceforge.net/` (letzter Aufruf 18.12.12)

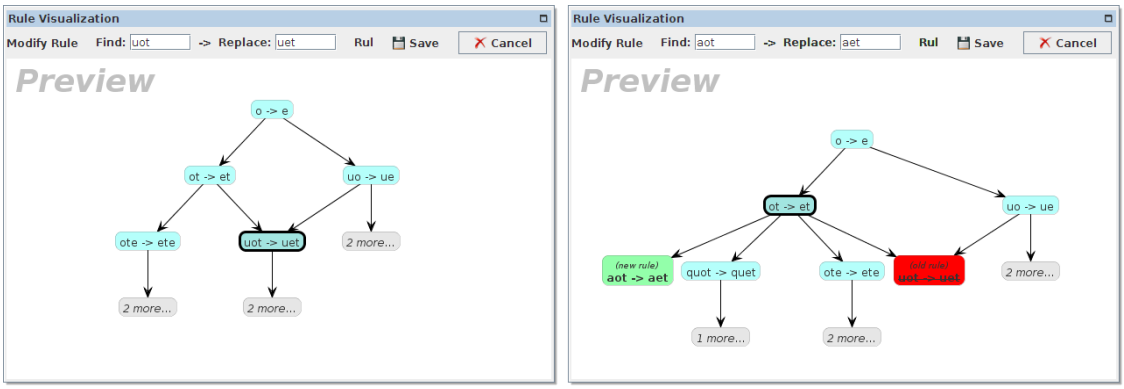


Abbildung 7.15.: Preview Modus

Bedarf zusätzlich angezeigt. Weil das JUNG-Framework bereits Verschiebe- und Zoom-Operationen unterstützt, stehen diese automatisch bei der Visualisierung von Regeln zur Verfügung.

Innerhalb der RuleVisualization-Komponente stellt eine Werkzeugleiste die verschiedenen Möglichkeiten zur Modifikation der Regelmenge bereit. Zur Erstellung von neuen Regeln gibt es Dialoge mit verschiedenen Unterstützungsmöglichkeiten. Der Benutzer kann eine Regel frei eingeben, ohne dass dafür ein Beleg existieren muss. Er kann aber auch einen Dialog verwenden, der ihn bei der Generalisierung bzw. Spezialisierung von Regeln unterstützt. Funktionen, die für eine ausgewählte Regel nicht zur Verfügung stehen, werden ausgegraut. Bei der Durchführung von Modifikationen wird der Benutzer von einem Vorschau-Modus unterstützt. Dieser wird aktiviert, sobald der Benutzer einen Modifizierungs-Dialog öffnet. Der Vorschau-Modus visualisiert den Graph in dem Zustand, in dem er nach Durchführung der eingegebenen Änderung wäre. Jede Eingabe innerhalb eines Modifizierungs-Dialogs löst eine animierte Aktualisierung des Vorschau-Modus aus. So sieht der Benutzer sofort die Konsequenzen seiner Änderung und ihm wird die Entscheidungsfindung erleichtert. Dies ist in Abbildung 7.15 zu sehen. Links wird die Regel $uot \rightarrow uet$ ausgewählt und die Schaltfläche zur Modifizierung betätigt. Daraufhin wird sowohl im Find- als auch im Replace-Teil der Regel das u durch ein a ersetzt. Nach dieser Änderung wird die Visualisierung des Preview-Modus aktualisiert. Dieser zeigt nun die ursprüngliche Form der Regel an (*old rule*) sowie die Form der Regel, die vorliegt, sofern diese Änderung bestätigt wird (*new rule*). Zusätzlich werden die Änderungen auch farblich markiert.

7.4. Benutzerevaluierung des RuleGenerators

Im Rahmen einer Benutzerstudie wurde für den RuleGenerator die Verständlichkeit und Bedienungsfreundlichkeit der Anwendung untersucht. Beides sollte sowohl für

die Erstellung von Belegen und Regeln als auch für die Interaktion der beiden Anwendungsteile überprüft werden.

7.4.1. Vorstudie

In einem ersten Schritt wurde der Prototyp des RuleGenerators auf seine Funktionsfähigkeit geprüft. Ziel der nachfolgend dargestellten Evaluierung ist der generelle Test der Abläufe. Dazu wird die Benutzerfreundlichkeit im Rahmen einer eyetrackergestützten Benutzerstudie evaluiert (Details s. [Awakian, 2010; Korbar, 2010]). Der Eyetracker zeichnet dabei die Augenbewegungen der Probanden auf. Diese können anschließend ausgewertet werden. Dabei kann festgestellt werden, welche Bereiche die Aufmerksamkeit des Benutzers auf sich gezogen haben und an welchen Stellen er bestimmte Funktionen sucht.

An der Evaluierung nahmen zehn Studenten der Universität Duisburg-Essen teil. Unter den Testpersonen befanden sich sieben Informatiker, zwei Sozialwissenschaftler und ein Erziehungswissenschaftler. Mit einem Computer arbeiteten die Testpersonen seit acht bis zwanzig Jahren und alle Testpersonen geben an, den Computer täglich zu verwenden.

Vor der Evaluierung bekamen die Benutzer zunächst eine kurze Anleitung (s. Abbildung 11.1, 11.2 und 11.3) sowie eine mündliche Einführung in die Applikation. Es war den Testpersonen gestattet, während der Evaluierung Fragen zu stellen. Die Fragen sowie die Maus- und Augenbewegungen der Personen wurden beobachtet und protokolliert bzw. vom Eyetracker aufgezeichnet. Die Evaluierung bestand aus einer Reihe von Aufgaben (s. Abbildung 11.4), die von den Testpersonen innerhalb der RuleGenerator Applikation durchzuführen waren. Dadurch sollte überprüft werden, ob die Anwendung sich intuitiv bedienen lässt. Daran schloss sich ein kurzes Interview an, und die Probanden füllten einen Fragebogen aus, mit dem einzelne Aspekte der Benutzerfreundlichkeit untersucht wurden. Auf diesem gab es verschiedene Aussagen zum RuleGenerator, die die Probanden anhand einer fünfstufigen Likert-Skala mit 1 (= trifft nicht zu) bis 5 (= trifft voll zu) bewerten sollten. Die Ergebnisse sind in Boxplot-Diagrammen dargestellt. Innerhalb der Box liegen dabei 50 % der Antworten. Der senkrechte Strich in der Box stellt den Median dar. Im Bereich der an die Box angrenzenden Linien liegen die restlichen Antworten, mit Ausnahme einzelner Ausreißer, die als Punkte dargestellt sind.

Der Fragebogen umfasste Aussagen zu den folgenden Bereichen:

- Allgemeine Bedienung
- Evidencer
- RuleSelector
- RuleVisualization

Die Aussagen werden im Folgenden in einer verkürzten Form dargestellt. Der Fragebogen ist im Anhang enthalten (s. Abbildungen 11.5, 11.6 und 11.7).

Allgemeine Bedienung Die Aussagen zur allgemeinen Bedienung beziehen sich auf die Applikation insgesamt. Wie in Abbildung 7.16 zu sehen ist, ist die Bewertung der Applikation im Allgemeinen positiv. Der Median liegt immer mindestens bei vier und auch die weiteren Werte liegen bis auf die Ausreißer im positiven Bereich. Lediglich die Icons der Applikation werden in der Aussage "Icons eindeutig" mittelmäßig bewertet. Im Interview mit den Probanden bestätigten sich diese Ergebnisse. Die Probanden haben vor allem Probleme mit den Icons in der RuleModification.

Evidencer Die Probanden bewerteten die Aussagen zu den Belegen (s. Abbildung 7.17) fast alle im Median mit 4. Die Aussage "Beleg editieren einfach" wird etwas schlechter bewertet. Im Interview zeigte sich, dass einige Personen Probleme hatten, den Zugang zu dieser Funktion zu finden. Sie erwarteten ebenfalls eine Erreichbarkeit der Operation über die Werkzeugleiste, diese lässt sich jedoch lediglich über einen Doppelklick aktivieren.

RuleSelector Die Aussagen zum RuleSelector (s. Abbildung 7.18) werden wiederum überwiegend mit einem Median von 4 bewertet. Eine Ausnahme bildet hier die Aussage "Regel finden einfach". Sie wird von den Benutzern mit einem Median von drei bewertet. Die Benutzer hatten beim Finden von Regeln vor allem Probleme damit, dass der Regelbaum bei der Benutzung des Filters nicht automatisch expandiert wird. Dies ergibt die Auswertung der Interviews und der Eyetracker-Daten.

RuleVisualization Die Aussagen zur Regelvisualisierung (s. Abbildung 7.19) werden im Median in den meisten Fällen mindestens mit 4 bewertet. Etwas schlechter schneidet hier die Aussage "Buttons bei RuleVisualization eindeutig" mit einem Median von 2,5 ab. Diese Bewertung deckt sich mit der schlechten Bewertung der Icons bei den allgemeinen Aussagen zu der Anwendung. Die Aussage "Animation in RuleVisualization verwirrend" wird mit einen Median von 2 nicht als verwirrend empfunden.

Beobachtungen während der Evaluierung

Durch die Verwendung des Eyetrackers wird deutlich, dass viele Teilnehmer den Bestätigungsdialog bei Modifikationen erst sehr spät bemerken. Nachdem der Benutzer den Button zum Löschen der ausgewählten Regel betätigt hat, schaut er auf den Belegteil oben, ohne den noch zu bestätigenden Dialog zu

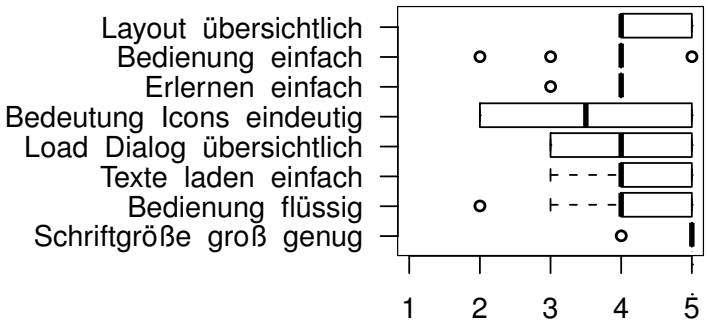


Abbildung 7.16.: Bewertung allgemeiner Aussagen zum RuleGenerator

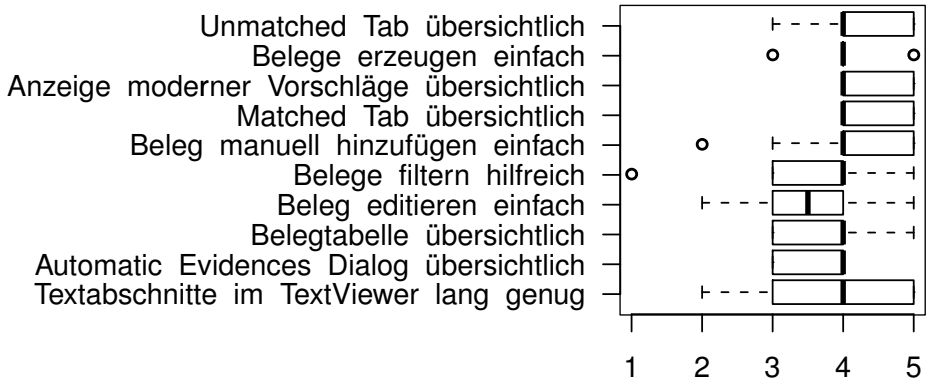


Abbildung 7.17.: Aussagenbewertung zu Belegen

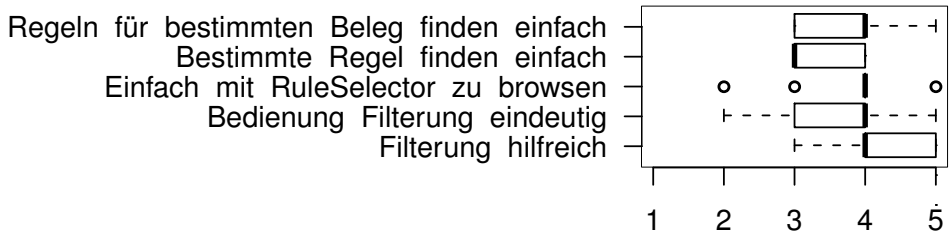


Abbildung 7.18.: Aussagenbewertung zum RuleSelector

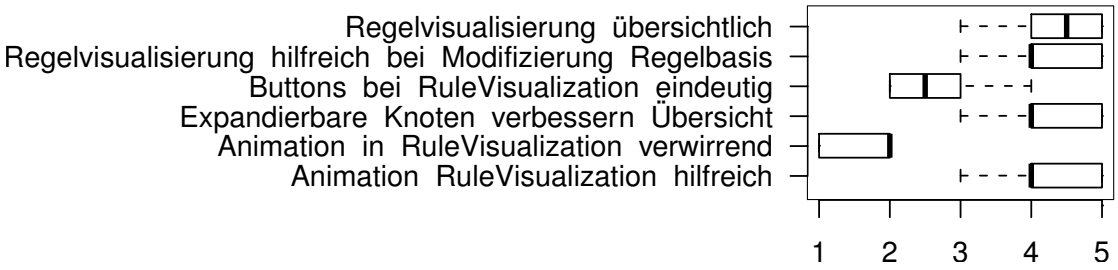


Abbildung 7.19.: Aussagenbewertung zur RuleVisualization

bemerken (s. Abbildung 7.20). Weitere Mängel, wie der fehlende Edit-Button im Belegteil, können anhand der gesammelten Eyetracker-Daten untersucht und bestätigt werden. Dadurch ist z. B. ersichtlich, dass Probanden bei der Bearbeitung von Aufgaben, in denen Belege zu editieren sind, den Belegteil sequentiell nach einer Möglichkeit zum Editieren absuchen, anstatt den Beleg nach einem Doppelklick zu editieren (s. Abbildung 7.21). Weil es bei der Bearbeitung von Regeln in der Werkzeugleiste einen entsprechenden Button gibt, sind die Benutzer durch ein an dieser Stelle nicht durchgängiges Bedienkonzept irritiert.

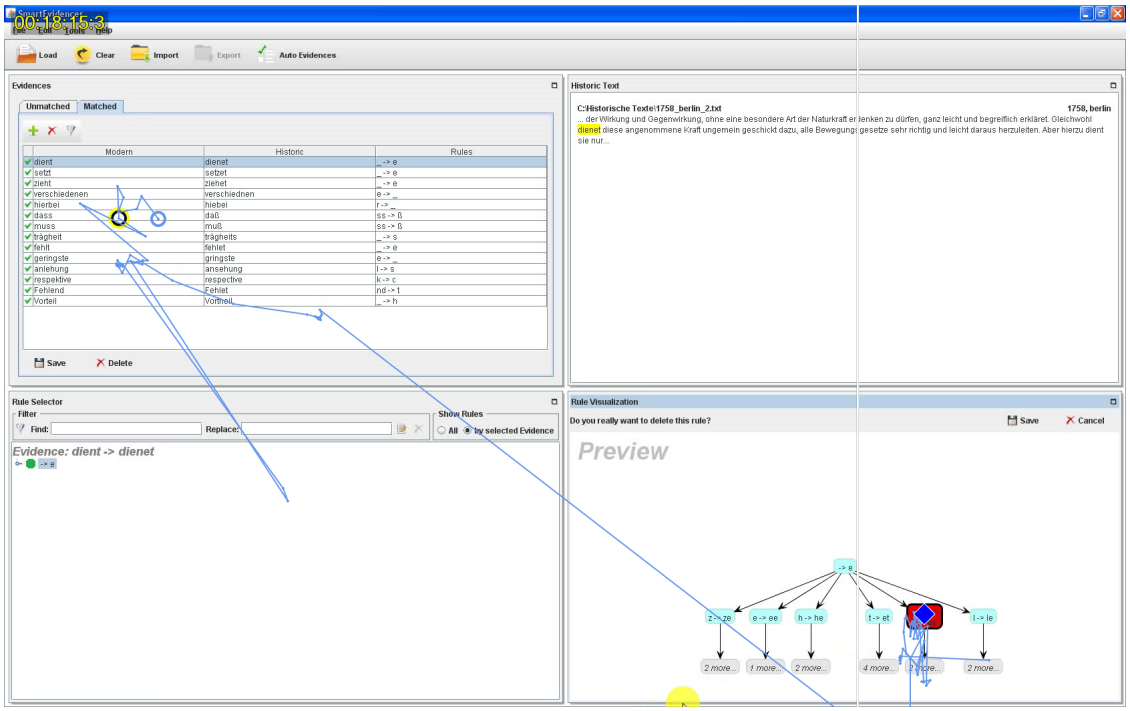


Abbildung 7.20.: Blickverlauf ”Regel modifizieren”

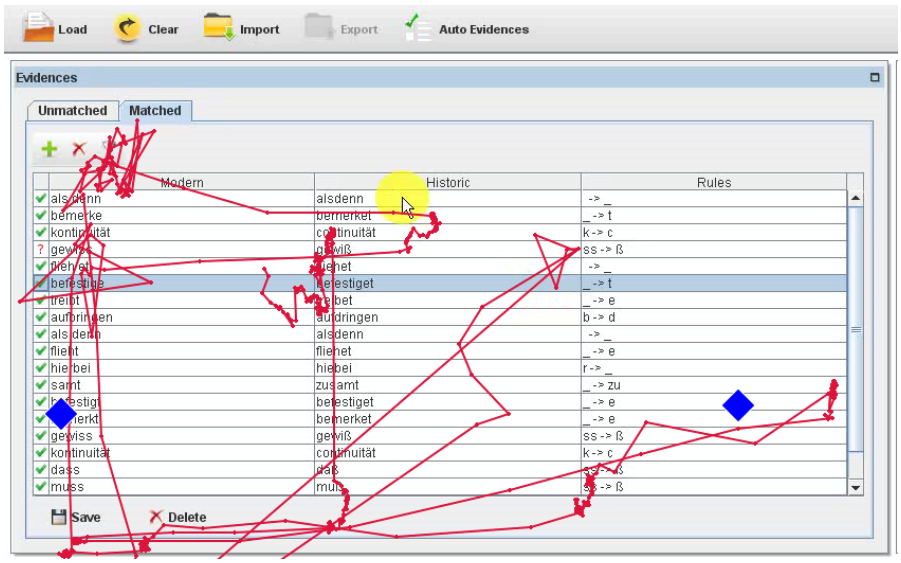


Abbildung 7.21.: Blickverlauf ”Beleg editieren”

Bei der Bearbeitung von Belegen zeigen sich die folgenden weiteren Probleme:

- Die Benutzer können den Button zum Wörterbuch nicht finden.
- Einigen Benutzern ist der Kontext des Textauszugs zu kurz. Dabei ist ihnen nicht klar, dass bei Bedarf auch der gesamte Text zur Verfügung steht.

Folgende Beobachtungen werden bei der Bearbeitung der Regeln gemacht:

- Es wird Feedback bei einem erfolgreichen Import von Regeln vermisst.
- Ein Benutzer nutzt den Filter zum Finden von Regeln nicht.
- Zwei Personen finden den Button zur Anzeige aller Regeln nicht.
- Über die Hälfte der Testpersonen bemängelt, dass die Elternknoten der gefundenen Regeln nicht automatisch expandiert werden.
- Zwei Personen verwechseln die Funktion Löschen einer Regel mit dem Löschen von Regeln mit all ihren Unterregeln.
- Knapp die Hälfte der Probanden bemerkt den Bestätigungsdialog zum Löschen von Regeln zunächst nicht.
- Bei der Erstellung von Regeln auf Basis einer vorhandenen Regel löschen alle Benutzer den bereits zur Verfügung gestellten Text.
- Die Testpersonen erwarten anstatt der Filterfunktion unter ”find” und ”replace” analog zu anderen Anwendungen die Funktionen ”Suchen” und ”Ersetzen”.

Auswertung der Interviews

Durch die Interviews am Ende der Evaluierung konnte genauer überprüft werden, welche Programmteile besonders benutzerfreundlich sind, wo genau Probleme bei der Benutzung liegen und wie das Werkzeug weiterentwickelt werden kann. Sieben Probanden betonen dabei die einfache Benutzung der Anwendung. Das gut gestaltete Layout wird von sechs Personen als positive Eigenschaft des Programms genannt. Für die Hälfte der Befragten ist der Button für das Wörterbuch nicht eindeutig genug.

Anpassung der Implementierung

Auf Basis der Evaluierung wurde der Prototyp des RuleGenerators weiterentwickelt (s. auch [Korbar, 2010]). Der Benutzer erhält nun eine Meldung über einen erfolgreichen Regelimport. Bei einer Verwendung des Regelfilters wird der Baum ausgehend von den Filterergebnissen automatisch expandiert. Der Dialog zur Bestätigung der Modifizierung von Regeln hebt nun zunächst den Hintergrund rot hervor, bevor die Farbe langsam wieder auf die anfängliche Farbe zurückgesetzt wird, um die Aufmerksamkeit des Benutzers zu erregen. Zusätzlich wird der Benutzer mit einer Fehlermeldung darauf aufmerksam gemacht, dass er die Löschung der Regel noch bestätigen muss. Keine der Testpersonen benutzt bei der Regelerstellung den vorgegebenen Text. Deswegen wird diese Option aus dem Menü entfernt. Da die Auswertung der Daten des Eyetrackers ergab, dass viele Benutzer bei der Bearbeitung der Regeln ein Kontextmenü suchen, wurde dieses noch implementiert. Weil die Benutzer die Möglichkeiten zur Anpassung der Visualisierung bisher nur selten benutzten, wurde noch eine Legende mit den Möglichkeiten zur Anpassung der Visualisierung erstellt. Des Weiteren wurden eindeutigere Icons für einige Funktionen gewählt und die mittelmäßig bewerteten Filterfunktionen überarbeitet. Die Auswertung der aufgezeichneten Eyetracker-Daten zeigten weitere Probleme auf, wie etwa fehlende alternative Möglichkeiten, um bestimmte Funktionen auszuführen. Diese wurden durch eine Implementierung von zusätzlichen Möglichkeiten – wie etwa weiteren Kontextmenüs – behoben.

7.4.2. Evaluierung bezogen auf spätere Benutzer

Im zweiten Teil der Evaluierung wurde der verbesserte RuleGenerator im Rahmen einer qualitativen Evaluierung mit klassischen Benutzern aus den Humanwissenschaften getestet. Um einen Vergleich zur ersten Benutzerevaluierung zu ermöglichen, wurden dieselben Aufgaben verwendet. Im Gegensatz zur ersten Evaluierung kam der Eyetracker dabei allerdings nicht zum Einsatz, weil so eine Evaluierung bei den Probanden vor Ort möglich war. Zum Ausgleich wurden

die Testpersonen gebeten, die Bearbeitung der Aufgaben laut zu kommentieren. Darüber hinaus wurden die Probanden bei der Bearbeitung der Aufgaben beobachtet.

An der Evaluierung nahmen vier Probanden teil. Zwei haben einen sprachwissenschaftlichen Hochschulabschluss (Germanistik bzw. Computerlinguistik) und die anderen beiden einen geisteswissenschaftlichen Abschluss (Religionswissenschaft bzw. Journalismus). Somit sind beide anvisierten Benutzergruppen bei der Evaluierung vertreten. Die Probanden sind zwischen 27 und 41 Jahre alt und alle weiblich. Einen Computer benutzen sie seit 15 bis 20 Jahren. Eine Probandin verwendet den Computer mehrmals pro Woche, alle anderen Probanden nutzen ihn täglich.

Bearbeitung der Aufgaben

Im Folgenden sind die während der Aufgabenbearbeitung gemachten Beobachtungen dargestellt. Dabei sind die Aufgaben bzw. Aufgabenteile nach Themenbereichen zusammengefasst worden.

Laden von Texten (1a, 2a): Alle Probanden haben zunächst Zeit und Ort nicht angegeben. Nach einem Hinweis des Programms wird dies nachgeholt. Es wird einmal angemerkt, dass die Schaltfläche zum Laden besser mit "OK" statt mit "Finish" beschriftet sein sollte.

Erstellen von Belegen (1b, 1c, 1d, 2b, 2d, 2j): Sowohl das Erstellen von automatischen Belegen als auch das Akzeptieren sowie das manuelle Einfügen von Vorschlägen und Belegen bereitet keine Probleme. Allerdings ist in drei von vier Fällen die Vorgehensweise unklar, wenn es zu einem Wort keine moderne Schreibung gibt. So wird z. B. der Beleg *Astronomie - Sternenwissenschaft* erstellt, obwohl der Probandin klar ist, dass mit diesem Beleg vermutlich keine Regeln gebildet werden, die weitere Varianten bilden können. Nach der Bearbeitung von weiteren Belegen wird der Löschbutton gefunden und im Folgenden eingesetzt, um Wörter, zu denen keine moderne Form existiert, aus der Liste zu entfernen.

Löschen von Belegen (1e): Alle Probanden sind in der Lage, einen Beleg zu löschen. Einmal wird angemerkt, dass es besser wäre, nachzufragen, ob der Beleg wirklich gelöscht werden soll.

Änderung von Belegen (2k): Die Änderung von Belegen bereitet keine Schwierigkeiten.

Finden von Belegen (1e, 1g, 1h): Den Testpersonen gelingt es, die Belege in der Liste zu finden. Zweimal wird erwähnt, dass eine automatische Sortierung der Belegliste hilfreich wäre. Eine Testperson findet die Funktion im Folgenden noch. Eine weitere Probandin setzt sie ganz selbstverständlich ein.

Import von Regeln (1f, 2e): Beim Import von Regeln haben die Probanden keine Probleme. Somit hat die neu implementierte Meldung über den erfolgten Regelimport die Schwierigkeiten bei dieser Aufgabe beseitigt.

Finden von Regeln (1g, 1h, 1i, 2g, 2h, 2i): Die Nutzung des RuleSelectors erfordert eine gewisse Eingewöhnung. Einmal wird zunächst im Find-Teil ein Pfeil mit eingegeben. Ein anderes Mal wird ein Suchbutton vermisst, bevor die unten bereits erfolgte Anzeige der entsprechenden Regeln bemerkt wird. Bei der Suche nach Regeln, die nicht mit dem Regelkern beginnen, kommt es bei drei von vier Probanden zu Problemen. So wird erst nach $e \to \varepsilon$, dann nach $\varepsilon \to e$ und erst anschließend nach $r \to \varepsilon$ gesucht. Die neu implementierte Expansion des Regelbaums hat die Benutzer dabei unterstützt, das Konzept des Regelkerns zu verstehen. Letztendlich können alle Probanden die gesuchten Regeln finden. Allerdings wäre es bei der Suche nach Regeln hilfreich, wenn man alle Regeln auf einmal wieder einklappen könnte.

Löschen von Regeln (1g, 1h, 1i, 2g): Die Löschung von Regeln gelingt allen Testpersonen. In einem Fall werden allerdings die Regel und alle ihre Unterregeln einzeln gelöscht. Dabei wird angemerkt, dass eine gleichzeitige Löschung einer Regel inklusive aller Unterregeln vorteilhaft wäre. Eine andere Probandin versucht zunächst, die Regeln im RuleSelector zu löschen.

Erstellen von Regeln (1j, 2i): Die Möglichkeit, Regeln zu erstellen, suchen drei Probanden zunächst im Filter des RuleSelectors, bevor sie das entsprechende Menü in der RuleVisualization Komponente finden. Bei der zweiten Aufgabe zur Erstellung von Regeln können dann alle direkt eine neue Regel mit dem entsprechenden Menü erstellen. Dabei kommt zum Teil auch das neu implementierte Kontextmenü zum Einsatz.

Modifizieren von Regeln (2h): Alle Testpersonen sind in der Lage, eine Regel zu ändern.

Regelanzeige (2f): Die Anzeige der Regeln gelingt in jedem Fall problemlos.

Hinzufügen zum Wörterbuch (2c): Die eine Hälfte der Probanden kann das Wörterbuch nicht finden. Der zweiten Hälfte gelingt es hingegen ohne Probleme, die Aufgabe zu bearbeiten.

Fragebogenauswertung

Im folgenden Abschnitt wird der im Anschluss an die Evaluierung von den Probanden ausgefüllte Fragebogen ausgewertet. Der Fragebogen stimmt mit dem aus der ersten Evaluierung überein.

Allgemeine Bedienung Die Aussagen zur allgemeinen Bedienung werden sehr häufig von 2 Personen mit 4 bewertet. Meistens werden die Behauptungen von jeweils einer weiteren Testperson mit 3 und einer mit 5 bewertet. Ausnahmen bilden hier die Äußerungen "Bedienung einfach" und "Schriftgrößen groß genug". Die Bedienung beurteilt eine Testperson mit 2. Dies war in der ersten Evaluierung auch der Fall. Die Schriftgröße bewerten sogar drei Testpersonen mit 2. Das sind genau die Probanden aus beiden Evaluierungen, die bei der Evaluierung nur mit dem Laptop ohne weiteren externen Monitor arbeiten. Um für diese Fälle vorzusorgen, könnte der Benutzer die Gelegenheit bekommen, die Schriftgröße an die eigenen Bedürfnisse anzupassen. Die Aussage "Bedienung flüssig" bewerten sogar 3 von 4 Probanden mit 5. Im Vergleich zur ersten Evaluierung hat sich das Ergebnis, mit Ausnahme der Schriftgröße, noch etwas verbessert. Dies gilt insbesondere für die Eindeutigkeit der überarbeiteten Icons.

Evidencer Die Probanden bewerten die Behauptungen zu den Belegen fast alle mindestens mit 4. Ausnahmen bilden hier die Äußerungen "Beleg editieren einfach" und "Autoevidence Dialog übersichtlich". Diese beurteilt jeweils eine Testperson mit 3. Die Aussagen "Beleg manuell hinzufügen einfach", "Belege filtern hilfreich", "Beleg editieren einfach" und "Textauszug lang genug" bekommen im Vergleich zur ersten Evaluierung eine deutlich bessere Bewertung.

RuleSelector Die Behauptungen zum RuleSelector werden mit 4 oder besser beurteilt. Eine Ausnahme bildet die Äußerung zur Eindeutigkeit der Belegfilterung. Sie wird von einer Probandin lediglich mit 3 bewertet. 2 weitere Probanden beurteilen diese Aussage hingegen sogar mit 5. Eine weitere Abweichung ist die Behauptung zum Browsen im RuleSelector, die von 2 Probanden lediglich mit drei bewertet wird. Dies ist auf die notwendige Eingewöhnung bei der Erkennung des Regelkerns zurückzuführen. Erst im Anschluss ist es sinnvoll, den RuleSelector zum Browsen zu verwenden. Die entsprechende Äußerung wird etwas schlechter als in der ersten Evaluierung bewertet. Alle anderen Aussagen haben ein besseres Resultat als bei der ersten Evaluierung.

RuleVisualization Bei den Behauptungen zur Regelvisualisierung verbessern sich die Bewertungen ebenfalls in fast allen Fällen. Die ersten 3 Äußerungen werden mindestens mit vier beurteilt. Die Aussage, dass die expandierbaren Knoten übersichtlich sind, hat allerdings eine Probandin lediglich mit 3 bewertet. Der negativen Aussage, dass die RuleVisualization verwirrend ist, stimmen die Probanden nicht zu. Allerdings wird die Behauptung, dass die RuleVisualization hilfreich ist, etwas schlechter beurteilt als bei der ersten Evaluierung. Dies kann daran liegen, dass die zweite Gruppe der Probanden stärker mit dem RuleSelector arbeitet und deswegen die RuleVisualization-Komponente als weniger hilfreich eingeschätzt.

Auswertung der Interviews

In den Interviews werden die folgenden Äußerungen zum RuleGenerator gemacht:

Positive Äußerungen:

- Zwei Testpersonen betonen die Nützlichkeit des Werkzeugs für die Arbeit mit historischen Texten.
- Zwei Probanden heben die Übersichtlichkeit der Anwendung hervor. Dies gilt insbesondere für die Visualisierung der Regeln. Außerdem sind die Änderungen leicht nachzuvollziehen.
- Eine Testperson findet positiv, dass sowohl Belege als auch Regeln geändert werden können.
- Einmal wird betont, dass der Kontext der Belege anhand der Textauszüge sofort deutlich wird.

Negative Äußerungen:

- Die fehlende Möglichkeit, alle geöffneten Regeln in der RuleVisualization auf einmal einzuklappen, wird zweimal bemängelt.
- Einmal wird vorgeschlagen, eine Sortierung der Regeln nach dem Alphabet und nicht nach der Reihenfolge der Erstellung vorzunehmen.
- Eine Probandin findet die Beschriftung der Schaltflächen nicht eindeutig genug.
- Eine weitere Testperson findet die Schaltknöpfe nicht groß genug.
- Drei Testpersonen ist die Schriftgröße zu gering.

Es werden die folgenden Verbesserungsvorschläge gemacht:

- Eine Probandin schlägt eine Vorführung des Programms zur Erleichterung der Arbeit mit der Anwendung vor.

- Zwei Testpersonen wünschen sich eine Option, mit der Regeln auch im RuleSelector geändert werden können. Auf diese Weise könnte bei häufiger Nutzung schneller gearbeitet werden.
- Eine Schaltfläche, mit der es möglich ist, alle Regeln auf einmal einzuklappen, wird von einer weiteren Probandin vorgeschlagen.
- Eine Testperson regt an, zu verdeutlichen, warum die Regeln im RuleSelector unterschiedliche Farben haben.

Bei den eigenen Beobachtungen ist die Lernkurve bei der Bearbeitung der Aufgaben deutlich erkennbar. Dieser Eindruck deckt sich auch mit den Aussagen der Probanden. Die Lösung von Aufgaben, die in ähnlicher Form zum zweiten Mal vorkommen, erfolgt deutlich zügiger. Wenn es bei der Bearbeitung von Aufgaben Probleme gibt, werden sie oftmals im Nachhinein noch gelöst, sobald die entsprechende Funktion im weiteren Verlauf der Arbeit mit dem Werkzeug gefunden wird. Durch die Erstellung einer Einführung in das Programm ließen sich die Anfangsprobleme reduzieren. So werden z. B. bei den negativen Aspekten zum Teil Funktionen vermisst, die aber vorhanden sind, wie die Möglichkeit, die Belege alphabetisch zu sortieren oder das Löschen einer Regel inklusive ihrer Unterregeln. Außerdem wird im Filter nach dem Suchbutton gesucht, bevor realisiert wird, dass das Filterergebnis unten bereits dargestellt ist.

Einige Programmfunktionen, die auch aus anderen Programmen bekannt sind, wie Anpassung der Fenster, Verschieben und Zoomen finden Verwendung. Des Weiteren sind durch die Evaluierungen noch einige Optimierungsmöglichkeiten, wie z. B. das Zusammenklappen des RuleSelectors und eine eindeutigere Beschriftung einiger Schaltflächen, deutlich geworden.

7.4.3. Zusammenfassung

Die Evaluierung zeigt, dass auch Benutzer ohne spezielle Computerkenntnisse das Programm nach einer kurzen Einführung nutzen können. Sowohl die Aufgaben zur Erstellung von Belegen als auch die zur Regelgenerierung können von den Probanden gelöst werden. Mit steigender Nutzungsdauer nimmt die Effizienz bei der Bearbeitung deutlich zu. Bei der Evaluierung fällt auch die Interaktion der einzelnen Komponenten positiv auf. Die im ersten Teil der Evaluierung aufgedeckten Probleme konnten durch die Anpassung der Implementierung behoben werden.

7.5. Zusammenfassung

In diesem Abschnitt wurde ein Ansatz zur automatischen Konstruktion der Trainingspaare für die automatische Regelgenerierung entwickelt. Nachdem die Evaluierung die Wirksamkeit des Verfahrens gezeigt hat, wurde eine

Benutzeroberfläche zur Generierung von Regeln und Belegen vorgestellt. Es wurde ein System mit einem hohen Automatisierungsgrad erstellt (s. Abschnitt 4.1). Allerdings kann der Benutzer den Grad der Automatisierung seinen Bedürfnissen entsprechend anpassen. Dabei wird dem Benutzer Unterstützung für den gesamten Prozess der Regelgenerierung angeboten. Allerdings bleibt es ihm selbst überlassen, wie viele der vorgeschlagenen Belege (und Regeln) er akzeptiert. Im Rahmen einer Benutzerevaluierung wurde gezeigt, dass neuen Benutzern die Bedienung der Applikation leicht fällt und die grundlegenden Konzepte der Applikation verstanden werden.

Der im Rahmen dieser Arbeit entwickelte Ansatz macht keine Vorgaben zu den Belegen. Für das Vokabelproblem können demzufolge ebenfalls Regeln generiert werden. Auf diese Weise hat der Benutzer auch ohne Zugriff auf ein historisches Lexikon die Möglichkeit, für seine Kollektion das Vokabelproblem zu lösen.

Teil V.

Zusammenfassung und Ausblick

8. Zusammenfassung

Die vorliegende Arbeit hat das Problem der Suche in historischen Dokumenten mit nicht-standardisierter Schreibweise bearbeitet. Durch die große Zahl an Variationen in der Rechtschreibung können klassische Ansätze der Grund- bzw. Stammformreduktion nicht zum Einsatz kommen. Stattdessen ist die Erzeugung von historischen Varianten des Suchbegriffs erforderlich. Mit Hilfe eines aktuellen Wörterbuchs, das die Flexionsformen enthält, wird der Suchbegriff zunächst auf die zugehörigen Vollformen abgebildet. Daraus werden anschließend die entsprechenden historischen Varianten erzeugt. Auf diese Weise kann nach einer Anfrageerweiterung eine Standardsuchmaschine zum Einsatz kommen. Zu diesem Zweck wurde im Rahmen dieser Arbeit ein maschinelles Lernverfahren entwickelt, mit dem sich geeignete Transformationsregeln generieren lassen.

Die Transformationsregeln haben in Kombination mit den Vollformen aus der Wortschatz-Datenbank in einer Evaluierung sehr gute Ergebnisse erzielt. Wie bei allen Ansätzen, die auf einem Wörterbuch basieren, ist es nicht möglich, alle Wortformen zu finden. Allerdings sind fast alle erzeugten Wortformen korrekt.

Ein Engpass des Ansatzes waren die manuell gesammelten Belege aus moderner und historischer Schreibung, die als Eingabe bei der Regelerstellung notwendig sind. Deswegen wurde im weiteren Verlauf der Arbeit eine Methode entwickelt, mit der sich die Wortpaare automatisch generieren lassen. Das Verfahren ist sehr anpassungsfähig, weil der Benutzer die Gelegenheit hat, die Parameter entsprechend seiner Prioritäten zu wählen. Auf diese Weise kann er den Prozess der Beleggenerierung kontrollieren. Die Evaluierung hat ergeben, dass der Ansatz sowohl zur automatischen Generierung von Belegen als auch zur Entwicklung eines ersten Regelsatzes erfolgreich eingesetzt werden kann.

Die automatische Beleggenerierung wurde in den RuleGenerator integriert. Dieser bietet eine Benutzeroberfläche zur automatischen Konstruktion von Belegen und Regeln. Diese versetzt den Benutzer in die Lage, ohne Programmierkenntnisse eigenständig Regeln für historische Korpora zu erzeugen und ermöglicht somit Retrieval auf Texten mit nicht-standardisierter Rechtschreibung. Dem Benutzer wird dabei die Gelegenheit gegeben, die automatischen Vorschläge für Belege und Regeln zu löschen, zu editieren sowie neue zu erstellen. Der Ansatz ist sehr flexibel, weil der Benutzer den Prozess entsprechend seiner Erwartungen an Recall und Precision der Suchmaschine beeinflussen kann. Auf diese Weise wird das Werkzeug auch unterschiedlichen Benutzergruppen wie Historikern und Linguisten gerecht.

Neuen Benutzern fällt die Bedienung der Applikation leicht und sie können die grundlegenden Konzepte der Anwendung verstehen. Dies konnte mit der Benutzerevaluierung gezeigt werden. Die dabei gefundenen Probleme wurden in einer Anpassung der Implementierung behoben. Dies konnte im Rahmen einer weiteren Evaluierung nachgewiesen werden.

Durch einen Austausch der linguistischen Ressourcen, wie z. B. des Vollformen-Wörterbuchs, kann der vorgestellte Ansatz auch bei weiteren Sprachen zum Einsatz kommen. Die entwickelte Methode ist daher sprachunabhängig.

Die Suchmaschine ermöglicht in Kombination mit dem RuleGenerator die Suche nach Texten mit Schreibvarianten. Dabei können aus dem Suchbegriff die Schreibvarianten gebildet werden. Somit ist ein Korrektur-System entwickelt worden, das die Störquelle aus dem Noisy Channel Model ausschalten kann. Es ist gelungen, ein leistungsfähiges System zu erstellen, das zugleich von den Benutzern trotz der hohen Komplexität nur geringe Computerkenntnisse erfordert. Die verbesserten Suchmöglichkeiten auf Dokumenten mit historischer Schreibung erleichtern den Zugang zu den entsprechenden Dokumenten deutlich. Dies wurde durch eine erfolgreiche Evaluierung von unterschiedlichen Aspekten des Systems gezeigt.

Der regelbasierte Ansatz ermöglicht es demzufolge, in Kombination mit der automatischen Erstellung von Belegen bereits mit geringem Aufwand die Retrievalqualität deutlich zu erhöhen, während andere Ansätze wesentlich mehr Aufwand benötigen.

Durch die automatische Erstellung konnte die arbeitsintensive Phase der Belegerstellung drastisch reduziert werden. Anstelle von mehreren Wochen größtenteils manueller Arbeit kann nun ein Großteil der Belege automatisch erzeugt werden. Eine Überarbeitung muss nur erfolgen, wenn die Retrievalqualität den eigenen Bedürfnissen entsprechend optimiert werden soll. Somit wurde ein Verfahren zur effizienten Regelerstellung entwickelt.

9. Weitere Anwendungsmöglichkeiten

Dieses Kapitel behandelt zunächst die Erweiterungsmöglichkeiten des entwickelten Werkzeugs für die Behandlung von Schreibvarianten. Anschließend wird erläutert, bei welchen Problemen die im Rahmen der vorliegenden Arbeit entwickelte Lösung ebenfalls zum Einsatz kommen könnte.

9.1. Erweiterungsmöglichkeiten

Bei den Regeln unterscheidet der RuleGenerator zwischen Einfüge-, Lösch- und Substitutionsregeln. Einen weiteren Regeltyp könnten abgeleitete Regeln darstellen. Diese Regeln könnten automatisch aus den Substitutionsregeln abgeleitet werden, wenn es im Regelgraphen eine Verbindung zwischen den Regeln gibt. Existieren z. B. die Regeln $ss \rightarrow ß$ und $s \rightarrow ß$, ließen sich daraus die Regeln $ss \rightarrow s$, $s \rightarrow ss$, $ß \rightarrow ss$ und $ß \rightarrow s$ ableiten (s. Abbildung 9.1). In Tabelle 9.1 ist ein Beispiel für die aus Trainingsdaten generierten Klassen dargestellt. Als Erweiterung des RuleGenerators könnte ein Werkzeug zur Visualisierung und Modifikation von abgeleiteten Regeln mit Hilfe des JUNG-Frameworks implementiert werden. Die Darstellung könnte z. B. in Anlehnung an [Tarassenko, 2008] als Netzwerkvisualisierung erfolgen, bei

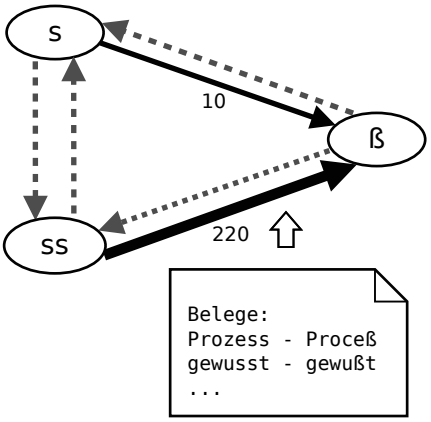


Abbildung 9.1.: Visualisierung abgeleiteter Regeln

s ß ss ass	→	1
k c z t d sc	→	2
ä a i e y ü ue ae ai el	→	3
g ch	→	4
we ib	→	5
üg ieh	→	6
so da	→	7
ö oe	→	8
r lisch	→	9

Tabelle 9.1.: Abgeleitete Regeln

der die Stärke der Verbindungslinien die Häufigkeit der Regeln verdeutlicht. Mit Hilfe der Visualisierung sollte auch eine Bearbeitung der Regeln möglich sein. So könnte dargestellt werden, welche Äquivalenzklassen sich je nach eingestellter Mindesthäufigkeit ergeben. Durch die abgeleiteten Regeln ließen sich z. B. auch phonetische Klassen analog zu denen von Soundex automatisch generieren.

Anhand von Logfiles ist es möglich, dass sich die Gewichte der Regeln dynamisch entsprechend der ausgewählten Varianten optimieren. Basierend auf der Annahme, dass Varianten in ausgewählten Texten auch relevant sind, könnten indirekte Relevanzinformationen [Hearst, 1999] genutzt werden. Damit würde sich das System ähnlich wie VARD 2 mit steigender Nutzungsdauer selbst verbessern. Hierbei erweist es sich als Vorteil, dass die Regeln nicht bereits bei der Erstellung des Indexes, sondern erst bei der konkreten Suche zum Einsatz kommen. So können nach jeder Nutzung der Suchmaschine die Gewichte der gebildeten Varianten angepasst werden, ohne dass eine Neuindexierung der Texte erforderlich ist.

Der RuleGenerator könnte es ermöglichen, in einem Text bzw. Textauszug ein Wort zu markieren und dazu direkt einen Beleg zu erzeugen. Eine weitere Option wäre, die Größe des Textauszugs editierbar zu machen, damit der Benutzer sie an seine Bedürfnisse anpassen kann.

Zudem sollte untersucht werden, ob sich wie bei der Namensidentifikation (s. Abschnitt 2.4.2) weitere Verbesserungen bei den Retrievalergebnissen durch eine Verknüpfung mit anderen Verfahren, wie z. B. der FlexMetric, erzielen lassen. Eine Kombination des RuleGenerators mit DICER (s. Abschnitt 6.3) könnte zudem den Benutzer bei der Regelerstellung zusätzlich unterstützen. Auf diese Weise wäre es möglich, die Qualität der Regeln weiter zu verbessern.

9.2. Weitere Einsatzmöglichkeiten

Das vorgestellte Werkzeug kann auch in anderen Bereichen eingesetzt werden. Dazu zählen die Patentsuche, die Texterkennung, die Suche nach Zitaten

sowie die Stammformreduktion. Im Folgenden werden die weiteren möglichen Verwendungsgebiete näher erläutert.

9.2.1. Patentsuche

Mit zunehmendem technologischen Fortschritt werden immer mehr Patente angemeldet [Li u. a., 2007]. Die Suche nach Patenten spielt während des gesamten Entwicklungszyklusses von Produkten eine bedeutende Rolle. Dies gilt sowohl für die Recherche nach dem Stand der Technik als auch für den Schutz der eigenen Entwicklungen. Um den Anwendungsbereich von Patenten zu vergrößern, aber auch zur Erschwerung der Suche, sind die Formulierungen in den Patenten häufig sehr vage [Graf und Azzopardi, 2008]. Bei der Eintragung von Patenten kommt es außerdem zu Schreibfehlern, die zum Teil auch beabsichtigt sind, um die Suche zu behindern. Die meisten großen Unternehmen haben Patente unter 10 bis 100 unterschiedlichen Namen angemeldet [Breitzman, 2005]. Beispielsweise sind für das Unternehmen *AKZO Nobel* verschiedene Kombinationen aus den Namensteilen und ihren jeweiligen Variationen (z. B. *AKCO*, *AKSO*) zu finden.

Sehr häufig kennen erfahrene Patentrechercheure die Namen von Firmen, die in relevanten Bereichen arbeiten, und geben sie bei ihrer Suche direkt mit an. Durch die Fehler wird trotzdem ein Großteil der Patente nicht gefunden. Deswegen muss das entsprechende Produkt neu entwickelt werden. Dies verursacht direkte Kosten [Graf und Azzopardi, 2008]. Falls durch die Entwicklung ein vorhandenes Patent, das aufgrund der Fehler nicht gefunden wurde, verletzt wird, fallen hierfür zusätzliche Kosten an. Auf diese Weise wird mit Hilfe der Schreibfehler die Konkurrenz zu Mehrausgaben verleitet und darüber hinaus sogar ein finanzieller Vorteil für das eigene Unternehmen erzielt. Bei potentiellen Rechtsverletzungen entstehen Kosten von bis zu mehreren Millionen Euro [Graf und Azzopardi, 2008]. Deswegen wird bei der Patentsuche ein Recall von 1 angestrebt, weil bereits ein nicht gefundenes Dokument zu einem Verstoß führen kann. In Patentdokumenten kommen häufig auch Zahlen vor, die eine große Bedeutung haben, z. B. in Bezeichnungen von chemischen Elementen [Becks u. a., 2011]. Auch diese können unterschiedliche Schreibungen aufweisen, je nachdem, ob sie in Ziffernschreibweise (z. B. *Cobalt(II)-chlorid*) oder ausgeschrieben (z. B. *Cobaltdichlorid*) in den Patenten vorkommen.

Es könnte untersucht werden, inwieweit der regelbasierte Ansatz zur Lösung des Problems einsetzbar ist und welche Modifikationen gegebenenfalls notwendig sind. Allerdings müssten die Belege für die Patentsuche manuell erstellt werden, weil eine Rechtschreibprüfung bei Firmennamen nicht zum Einsatz kommen kann. Jedoch sollte sich dieser Aufwand bei einem so rechercheintensiven Bereich lohnen. Sofern Zugriff auf den Index besteht, könnte überprüft werden, ob bei der automatischen Beleggenerierung die Belege auch mit Hilfe von Ähnlichkeitsmaßen gebildet werden können.

Die Namensvariationen lassen sich in die folgenden Gruppen einteilen, die allerdings auch in kombinierter Form vorkommen:

- Leichte Variation im Firmennamen
- Verschiedene Detailbezeichnungen zu den Firmennamen wie z. B. Branche oder die entsprechende Abteilung.
- Unterschiede in der Gesellschaftsform der Firma. Diese variieren auch je nach Land.
- Oftmals verfügen Firmennamen über geographische Zusätze wie z. B. *Nestle* (*Italiana*) oder *Nestle* (*Deutschland*). Die geographischen Bezeichnungen sind dabei z. T. auch in der Landessprache.

Ingwersen und Kollegen führen folgende Beispiele für unterschiedliche Bezeichnungen von Firmen in Datenbanken an [Ingwersen und Christensen, 1997]:

- *Nat. Phys. Lab., Teddington, UK*
- *Nat. Phys. Labs., Teddington, UK*
- *National Phys. Lab., Teddington, UK*
- *NPL, Teddington, UK*

Darüber hinaus können auch die Landesbezeichnungen variieren (z. B. *UK*, *United Kingdom* oder *Great Britain*) oder ganz entfallen. In [Lu u. a., 1998] sind Listen mit Indikatorworten für Firmennamen und Namen von Organisationen in unterschiedlichen Schreibweisen wie z. B. *BROS*, *BROS.* und *BROTHERS* enthalten. Der regelbasierte Ansatz könnte in Verbindung mit solchen Listen bei der Patentsuche zum Einsatz kommen. Dabei sollte auch eine Liste mit geographischen Bezeichnungen verwendet werden.

Ein Teil der Probleme ist auch auf OCR-Fehler zurückzuführen. Dies führt z. B. bei MicroPatent PatSearch zu Varianten bei den Alkyl-Gruppen[30]:

- *l → i: methyi, ethyi, propyi, butyi*
- *l →1: methy1, ethy1, propy1, buty1*

Da sich chemische Namen in Teilen oft wiederholen, könnte hier ein regelbasiertes Verfahren sehr hilfreich sein. Das Gleiche gilt für Tippfehler in chemischen Namen wie z. B. *Esther* anstatt *Ester*.

[30] `http://www.ir-facility.org/c/document_library/get_file?uuid=d14db67f-e896-44f8-a51a-6539201ecfc9&groupId=10156` (letzter Aufruf 18.12.12)

9.2.2. Texterkennung

Pilz und Kollegen haben einen Vergleich von manuell gebildeten Regeln und automatisch generierten Regeln des entwickelten Ansatzes für Texterkennung vorgenommen [Pilz u. a., 2008a]. Von den 75 generierten Regeln waren 39 Regeln mit den manuell gebildeten Regeln identisch. Weitere 9 Regeln stimmten bis auf den Kontext überein. 5 manuelle Regeln ließen sich durch Kombination von automatischen Regeln bilden. Trotz einer beschränkten Trainingsmenge ist eine große Übereinstimmung zwischen den manuellen und den automatischen Regeln festzustellen. So konnten die 10 häufigsten manuell gebildeten Regeln alle mit dem automatischen Verfahren generiert werden.

Die automatische Regelgenerierung kann demzufolge direkt bei der Suche in Dokumenten, die OCR-Fehler enthalten, Verwendung finden. Darüber hinaus ist auch ein Einsatz bei der Korrektur von OCR-Fehlern denkbar. Wegen der hohen graphischen Ähnlichkeit sind Kontextinformationen in diesem Fall nicht sehr hilfreich. Falls keine speziellen Werkzeuge für die graphische Ähnlichkeit zur Verfügung stehen, sind allgemeine Regeln in diesem Fall besser geeignet. Die automatische Regelgenerierung könnte somit in die vorhandenen Verfahren zur Texterkennung integriert werden. Auf diese Weise ist eine Optimierung der bereits vorhandenen Programme möglich.

9.2.3. Suche nach Zitaten

Die Identifizierung von Zitaten aus Nachschlagewerken in Primärliteratur ist ein wichtiger Aspekt von digitalen Bibliotheken [Cortez u. a., 2007]. Häufig enthält Sekundärliteratur allerdings keine Angabe über das Zitat bzw. die Information ist für Werkzeuge zur Zitatsuche nicht identifizierbar [Ernst-Gerlach und Crane, 2008]. Die Verknüpfung der Zitate mit dem Originaltext ist jedoch entscheidend, um Beiträge aus der Sekundärliteratur einzuordnen, weil Zitate nur einen Ausschnitt des Originaldokumentes darstellen. Deswegen ist es wichtig, Zugang zum Kontext des Zitates zu bekommen, um es korrekt klassifizieren zu können [Romanello u. a., 2009].

Dokumente mit Zitaten folgen oftmals unterschiedlichen Zitationsschemen [Crane u. a., 2007]. Dies gilt besonders, wenn sie in einer Zeit entstanden sind, zu der analog zu der Standardisierung der Rechtschreibung noch kein Standard für Zitate existierte. Romanello und Kollegen haben sich mit der automatischen Identifizierung von Zitaten mit standardisierten Zitationsangaben auf Basis von Trainingsdaten befasst [Romanello u. a., 2009]. Fehlen diese Zitationsangaben, muss anhand der Suche von Textpassagen aus der Sekundärliteratur in der Primärliteratur überprüft werden, ob Zitate enthalten sind. Neben direkten wörtlichen Zitaten sind auch andere Zitierweisen zu beachten, wie z. B. eine geänderte Wortreihenfolge, eine Auslassung von Termen und Termunterschiede wie beispielsweise Schreibvarianten

[Crane, 2002]. In einigen Fällen basieren Zitate auf anderen Editionen als den vorliegenden digitalen Quellen. Dadurch finden sich mitunter Variationen, die die unterschiedlichen Versionen des Textes widerspiegeln. Es lassen sich auch Zitate finden, die den zitierten Text modifizieren. Dies gilt insbesondere für Nachschlagewerke wie Lexika und Grammatiken. Das Ziel ist in diesem Fall nicht, den Originaltext wiederzugeben, sondern einen anderen Aspekt z. B. der Grammatik oder der Lexikographie zu illustrieren.

Eine manuelle Analyse von Zitaten aus *A Latin Dictionary*[31] von Lewis und Short, dem *Overview of Latin Syntax*[32] von Anne Mahoney und der *Latin Grammar*[33] von Greenough wurde durchgeführt, um die speziellen Probleme bei der Identifizierung von Zitaten aufzudecken [Ernst-Gerlach und Crane, 2008]. Die Analyse ergab, dass bei Zitaten oftmals auch einzelne Wörter sowie die Wortreihenfolge verändert werden. Deswegen wird bei dem entwickelten Verfahren zur Zitatsuche für alle Wörter aus dem Sekundärtext, die auch im Primärtext vorkommen, überprüft, ob in einem Suchfenster von fünf Wörtern vor oder nach dem gefundenen Wort weitere Übereinstimmungen zu finden sind. Bei zwei Übereinstimmungen schließt sich eine unscharfe Suche nach Ähnlichkeiten zwischen den Wörtern im Suchfenster an.

Das Verfahren könnte durch den vorgestellten Ansatz zur automatischen Generierung von Regeln weiter verbessert werden: Werden mehrere Belege für veränderte Wörter gefunden, lassen sich anhand der Belege automatisch Regeln bilden. Diese könnten bei darauf folgenden Suchen nach Zitaten verwendet werden, um aus der Menge der potentiellen Zitate die tatsächlichen Zitate zu identifizieren.

Häufig unterscheiden sich auch verschiedene Editionen eines Textes, wie z. B. bei Don Quixote, dessen Originaldokument nicht mehr existiert [Furuta u. a., 2001]. Mit einem Vergleich von verschiedenen Editionen könnten Belege für Schreibvarianten automatisch gebildet und durch die gelernten Regeln die Suche nach Zitaten erleichtert werden. Diese Vorgehensweise ließe sich auch zur Unterstützung beim Alignment von parallelen Korpora einsetzen.

9.2.4. Stammformreduktion

In Abschnitt 5.2 wurde bereits gezeigt, dass herkömmliche Verfahren zur Stammformreduktion für historische Dokumente in deutscher Sprache nicht geeignet sind. Für modernes Englisch funktionieren Verfahren zur Bildung von Stammformen allerdings gut. Becker und König haben Ansätze zur regelbasierten Bildung von Stammformen vorgestellt [Becker und König, 2002]. Deswegen sollte überprüft

[31] `http://www.perseus.tufts.edu/hopper/text.jsp?doc=Perseus:text:1999.04.0059` (letzter Aufruf 18.12.12)

[32] `http://www.perseus.tufts.edu/hopper/text?doc=Perseus:text:1999.04.0022` (letzter Aufruf 18.12.12)

[33] `http://www.perseus.tufts.edu/hopper/text?doc=Perseus:text:1999.04.0001` (letzter Aufruf 18.12.12)

werden, ob durch eine Rückführung der historischen Varianten auf aktuelle Wortformen Verfahren zur Stammformreduktion auch bei historischen Texten in englischer Sprache zum Einsatz kommen können. Eine weitere Möglichkeit wäre, die Regeln für die Bildung der Stammform mit den Regeln für Schreibvarianten zu kombinieren.

9.3. Zusammenfassung

In diesem Kapitel wurden einige Erweiterungsmöglichkeiten für den vorgestellten regelbasierten Ansatz aufgezeigt. Diese könnten die Leistungsfähigkeit des Verfahrens weiter steigern. Darüber hinaus wurden weitere Einsatzfelder aufgezeigt. Besonders viel versprechend erscheint die Patentsuche, da in diesem rechercheintensiven Gebiet eine Unterstützung der Suche besonders wichtig ist. Insgesamt wurde damit das Potential des Ansatzes verdeutlicht.

Teil VI.

Anhang

10. Tabellen

Ansatz	Recall	Precision
Manuelle Regeln	0,09	0,53
Variantgraph	0,69	0,48
Automatische Regeln	0,88	0,45

Tabelle 10.1.: Recall- und Precision-Werte für automatische und manuelle Regeln sowie den Variantgraph

Ansatz	Terme der gesamten Kollektion			Begrenzt auf historische Wortformen		
	Recall	**Precision**	**F_1-Maß**	**Recall**	**Precision**	**F_1-Maß**
Herkömmliche Suchmaschine	0,36	1,00	0,53	-	-	-
Stammform	0,82	0,94	0,88	0,05	1,00	0,10
Wortschatz-Datenbank	0,91	1,00	0,95	0,02	1,00	0,04
Stammform + Levenshtein 1	0,98	0,49	0,65	0,24	0,41	0,30
Stammform + Levenshtein 2	0,99	0,06	0,11	0,45	0,11	0,18
Stammform + Regelanwendung	0,89	0,77	0,83	0,71	1,00	0,83
Wortschatz-Datenbank + Levenshtein 1	0,40	0,85	0,54	0,35	0,35	0,35
Wortschatz-Datenbank + Levenshtein 2	0,75	0,21	0,33	0,52	0,10	0,17
Wortschatz-Datenbank + Regelanwendung	0,99	0,89	0,94	0,70	0,93	0,80

Tabelle 10.2.: Retrievalergebnisse für Wortformen für die Bildung von historischen Wortformen der Lemmata

	Anzahl Belege									
	1000	**2000**	**3000**	**4000**	**5000**	**6000**	**7000**	**8000**	**9000**	**10000**
Precision	0,28	0,25	0,25	0,24	0,22	0,19	0,18	0,18	0,17	0,17
Recall Regelkerne	0,55	0,47	0,50	0,47	0,50	0,50	0,51	0,51	0,50	0,49
Recall Vorkommen der Regelkerne	0,91	0,90	0,92	0,92	0,94	0,95	0,95	0,95	0,96	0,96

Tabelle 10.3.: Recall und Precision für Regelkerne auf Basis der Testdaten des jeweiligen Durchlaufs

		Anzahl Belege									
		1000	**2000**	**3000**	**4000**	**5000**	**6000**	**7000**	**8000**	**9000**	**10000**
Regel	**Automatisch**	0,14	0,24	0,31	0,36	0,39	0,41	0,43	0,46	0,47	0,49
	Manuell	0,26	0,51	0,63	0,77	0,80	0,82	0,84	0,90	0,94	1
Regelvorkommen	**Automatisch**	0,10	0,20	0,30	0,39	0,48	0,58	0,68	0,77	0,86	0,96
	Manuell	0,10	0,22	0,32	0,42	0,52	0,61	0,71	0,80	0,90	1

Tabelle 10.4.: Recall für Regelkerne auf Basis der Gesamttestdaten

	Anzahl Belege									
	1000	**2000**	**3000**	**4000**	**5000**	**6000**	**7000**	**8000**	**9000**	**10000**
Automatische Belegabdeckung	0,88	0,91	0,93	0,93	0,94	0,94	0,94	0,95	0,95	0,95
Manuelle Belegabdeckung	0,91	0,95	0,96	0,98	0,98	0,98	0,99	0,99	1	1

Tabelle 10.5.: Recall Belegabdeckung

Schwellenwerte		Deutsch				Englisch
Regelanwendungen	**Regelvorkommen**	**1500 - 1599**	**1600 - 1699**	**1700 - 1799**	**1800 - 1899**	**1590 - 1616**
1	**2**	0,50	0,56	0,62	0,60	0,42
	5	0,51	0,59	0,72	0,63	0,42
	10	0,52	0,61	0,74	0,65	0,46
2	**2**	0,48	0,51	0,65	0,55	0,38
	5	0,48	0,57	0,71	0,60	0,40
	10	0,52	0,60	0,74	0,64	0,45
3	**2**	0,48	0,48	0,53	0,54	0,38
	5	0,50	0,53	0,68	0,58	0,39
	10	0,53	0,57	0,71	0,62	0,41
Vergleichsbasis		0,35	0,32	0,42	0,40	0,44

Tabelle 10.6.: Precision basierend auf allen unbekannten Termen

Schwellenwerte		Deutsch				Englisch
Regelanwendungen	**Regelvorkommen**	**1500 - 1599**	**1600 - 1699**	**1700 - 1799**	**1800 - 1899**	**1590 - 1616**
1	**2**	0,58	0,63	0,70	0,71	0,48
	5	0,59	0,66	0,82	0,75	0,45
	10	0,57	0,66	0,82	0,76	0,50
2	**2**	0,60	0,61	0,75	0,65	0,46
	5	0,58	0,66	0,78	0,71	0,46
	10	0,62	0,68	0,79	0,74	0,50
3	**2**	0,58	0,57	0,62	0,64	0,46
	5	0,61	0,61	0,77	0,68	0,45
	10	0,61	0,64	0,77	0,71	0,46
Vergleichsbasis		0,39	0,36	0,48	0,43	0,58

Tabelle 10.7.: Precision-Werte beschränkt auf Schreibvarianten

Schwellenwerte		Deutsch				Englisch
Regelanwendungen	**Regelvorkommen**	**1500 - 1599**	**1600 - 1699**	**1700 - 1799**	**1800 - 1899**	**1590 - 1616**
1	**2**	0,56	0,62	0,78	0,79	0,74
	5	0,52	0,62	0,74	0,77	0,74
	10	0,46	0,61	0,74	0,75	0,74
2	**2**	0,65	0,66	0,80	0,81	0,77
	5	0,58	0,65	0,77	0,79	0,77
	10	0,53	0,65	0,77	0,79	0,77
3	**2**	0,71	0,66	0,86	0,81	0,77
	5	0,63	0,65	0,84	0,79	0,77
	10	0,58	0,65	0,84	0,79	0,77
Vergleichsbasis		0,64	0,56	0,67	0,70	0,66

Tabelle 10.8.: Recall-Werte für unterschiedliche Parameter

11. Evaluierungsunterlagen

Einführung

Die deutsche Rechtschreibung wurde erst im Jahre 1901 vereinheitlicht. Durch die Abweichung von der aktuellen Standardschreibung sind historische Dokumente über herkömmliche Suchmaschinen oft nur schwer zu finden. Im Rahmen des Projektes RSNSR wurde ein regelbasierter Ansatz zur Suche in historischen Texten entwickelt. Für historische Texte werden zeit- und ortsabhängige Regeln generiert. Mit Hilfe dieser Regeln werden dann eingegebene Suchterme in die jeweiligen historischen Formen umgewandelt.

Die Erstellung dieser Regeln beginnt mit der Erstellung von Belegen *(evidences)*. Belege sind Wortpaare. die sich aus einer *historischen (z.B. Morgenröthe)* und einer *modernen (z.B. Morgenröte)* Wortform zusammensetzen. In einer Sammlung von historischen Texten müssen also *historische* Wortformen identifiziert, und die dafür passenden *modernen* Wortformen gefunden werden. Zu einer historischen Wortform kann es auch mehrere gültige moderne Wortformen geben. Dann würden dadurch mit dieser historischen Wortform mehrere Belege existieren. Der eben genannte Beleg wäre der folgende:

Morgenröte → Morgenröthe

Mit Hilfe des im Projekt RSNSR entwickelten Verfahrens können nun aus diesen Belegen Regeln generiert werden. Regeln bestehen immer aus einem *find*- und *replace*-Teil. Bei eingegebenen Wörtern wird dann die moderne Wortform nach dem *find*-Teil durchsucht, und sofern gefunden, durch den *replace*-Teil ersetzt, um so die historische Wortform zu erzeugen. Regeln für den obigen Beleg wären:

_ → h t → th te → the

Im Rahmen des Projektes RSNSR wurde das SmartEvidencer Tool entwickelt, um den Prozess der Belegsammlung und Regelgenerierung zu unterstützen.

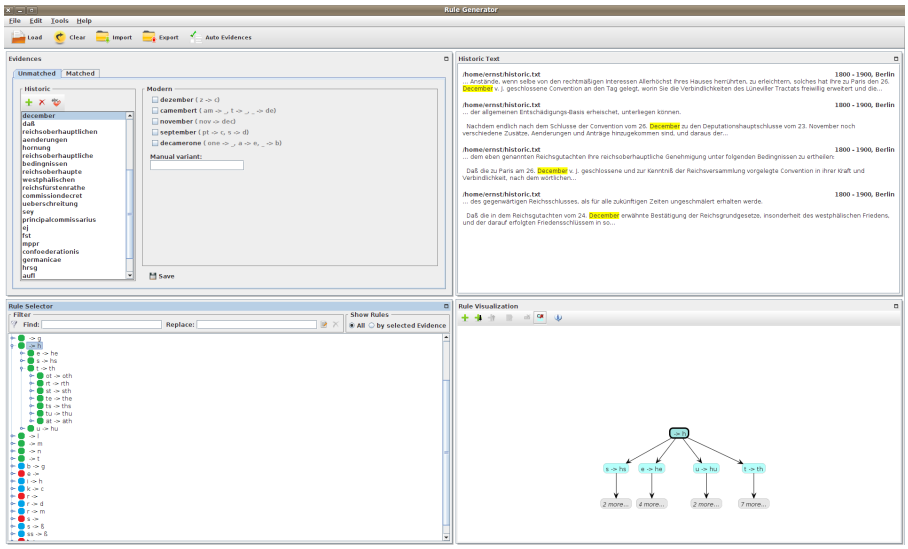
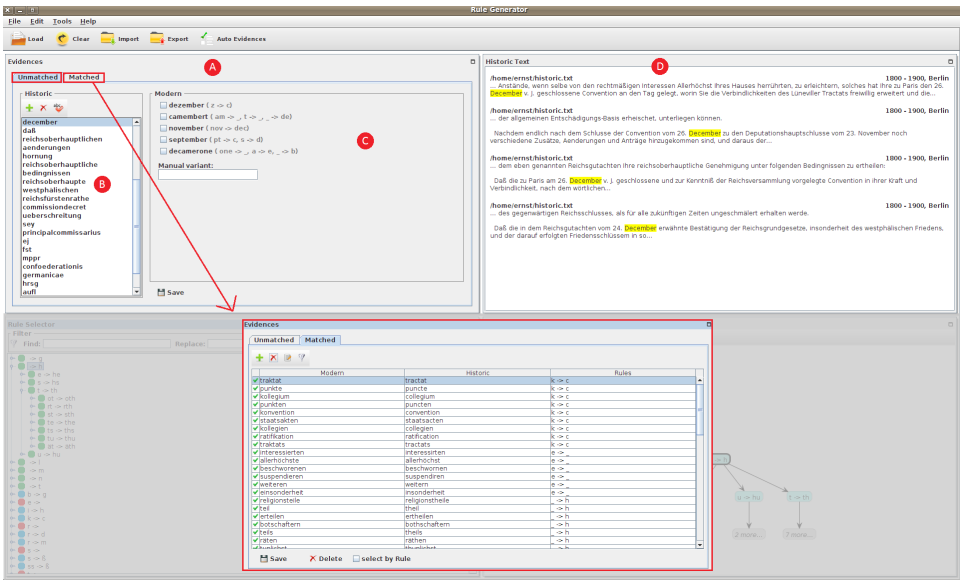
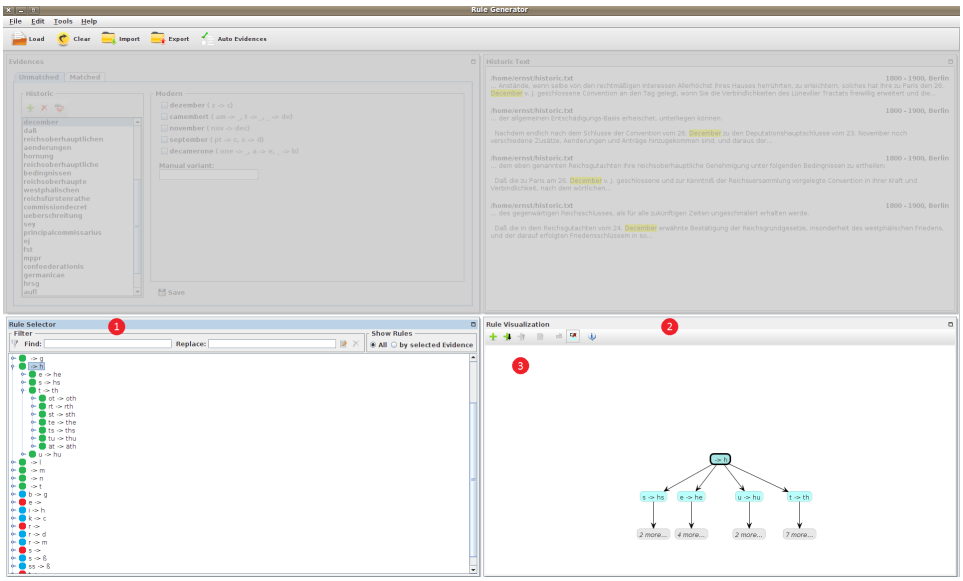


Abbildung 11.1.: Einführungstext der Evaluierung



A) **Evidence**: Der Evidencer dient zur Erstellung (im Tab **„Unmatched“**) bzw. Bearbeitung (im Tab **„Matched“**) von Belegen.
B) **Historic**: Hier werden die vom Tool erkannten historische Wortformen angezeigt.
C) **Modern**: Hier werden die vom Tool generierten Vorschläge angezeigt
D) **Historic Text**: Hier werden Abschnitte des geladenen Textes angezeigt.

Abbildung 11.2.: Einführung SmartEvidencer



1) **Rule Selector**: Der Rule Selector dient der Auswahl einer Regel und ermöglicht das Browsen der Regelbasis. Die selektierte Regel wird in der Rule Visualization angezeigt.
2) **Rule Visualization**: Die Rule Visualization zeigt die für die selektierte Regel relevanten Ober- und Unterregeln an. Sie unterstützt Standardoperationen wie *Panning* und *Zooming*.
3) **Regelmodifikation**: Modifikationen in der Regelbasis werden über die Toolbar durchgeführt.

Abbildung 11.3.: Einführung RuleModification

Aufgabe 1

a) Laden Sie die Texte **„1758_berlin_1.txt“** und **„1758_berlin_2.txt”,** die sich in dem Ordner **„ernst/eval”** befinden (*Region* und *Period* finden Sie im Dateinamen) und **deaktivieren** Sie dabei die Funktion der automatischen Beleggenerierung.
b) Wählen Sie für alle Belege im **„unmatched“**-Tab die korrekte moderne Wortform aus und speichern Sie diese. Falls historische Wortformen ohne korrekte Vorschläge für die moderne Wortform vorhanden sind, geben Sie bitte einen eigenen Vorschlag ein.
c) In der **„matched“** Tabelle fügen Sie die historische Wortform **„vortheil“** ein**.**
d) Erstellen Sie einen neuen Beleg für die historische Wortform **„Fehlet“.**
e) Löschen Sie den Beleg der historischen Wortform **„Erkläret “** und der modernen Wortform **„erklärt“**.
f) Importieren Sie die Regeldatei **„Berlin_Regeln_1.xml“** aus dem Ordner **„ernst/eval“**.
g) Für den Beleg der historischen Wortform **„Dienet“** und der modernen Wortform **„dient“**, löschen Sie die Regel **„n → ne“**.
h) Für den Beleg der historischen Wortform **„Hiebei“** und der modernen Wortform **„hierbei“**, löschen Sie die Regel **„er → e“** und alle ihre Unterregeln.
i) Selektieren Sie im *Rule Selector* die Regel **„e → “,** und finden Sie in der *Rule Visualization* die Regel **„ren → rn“**. Löschen Sie diese.
j) Erstellen Sie eine neue Regel **„te → the“.**

Aufgabe 2

a) Löschen Sie die aktuelle Session und laden Sie die Texte **„1758_berlin_3.txt“** und **„1758_berlin_4.txt”,** die sich in dem Ordner **„ernst/eval”** befinden und **aktivieren** Sie dabei die Funktion der automatischen Beleggenerierung.
b) Speichern Sie **zehn** korrekte automatisch generierte Belege.
c) Fügen Sie die historische Wortform **„Fixsternenhimmels“** dem Wörterbuch hinzu.
d) Wählen Sie für **zehn** Belege im **„unmatched“**-Tab die korrekte moderne Wortform aus und speichern Sie diese. Falls historische Wortformen ohne korrekte Vorschläge für die moderne Wortform vorhanden sind, geben Sie bitte einen eigenen Vorschlag ein.
e) Importieren Sie die Regeldatei **„Berlin_Regeln_2.xml“** aus dem Ordern **„ernst/eval“**.
f) Lassen Sie sich im Rule Selector alle Regeln anzeigen.
g) Finden Sie die Regel **„kon → con“** und löschen Sie diese und alle ihre Unterregeln.
h) Finden Sie die Regel **„er → e“** und modifizieren Sie diese zur Regel **„ar → a“**.
i) Finden Sie die Regel **„üss → üß“** und erstellen Sie eine neue Regel **„küss → küß“** auf deren Basis.
j) Aktivieren Sie die automatische Beleggenerierung und speichern Sie die drei neuen Belege.
k) Ändern Sie die erste moderne Wortform der historischen Wortform **„Befestiget“** in **„Befestigen“**.

Abbildung 11.4.: Evaluierungsaufgaben

Fragebogen Versuchsperson

Versuchsperson:

F 1.1 Alter: _____

F 1.2 Geschlecht: _____

F 1.3 Muttersprache: _____

F 1.4 Studiengang: _____

F 1.5 Seit wie vielen Jahren benutzen Sie bereits Computer? _____ Jahre

		Täglich	Mehrmals pro Woche	Einmal pro Woche	Mehrmals pro Monat	Einmal pro Monat
F 1.6	Wie häufig verwenden Sie den Computer?	○	○	○	○	○

Abbildung 11.5.: Fragebogen Seite 1

Abschlussfragebogen

		Gar nicht		etwas		sehr
Allgemeine Fragen:		1	2	3	4	5
F 2.1	Das Layout des Tools war übersichtlich.	○	○	○	○	○
F 2.2	Die Bedienung des Tools war einfach.	○	○	○	○	○
F 2.3	Die Erlernung des Tools war einfach.	○	○	○	○	○
F 2.4	Die Bedeutung der Icons im Tool war eindeutig.	○	○	○	○	○
F 2.5	Der Load Dialog war übersichtlich.	○	○	○	○	○
F 2.6	Es war einfach Texte zu laden.	○	○	○	○	○
F 2.7	Die Bedienung des Tools war flüssig.	○	○	○	○	○
F 2.8	Die Schriftgröße des Tools war groß genug.	○	○	○	○	○
Fragen zu Belegen (Evidences)		1	2	3	4	5
F 2.9	Der **unmatched-tab** war übersichtlich.	○	○	○	○	○
F 2.10	Die Erzeugung der Belege war einfach.	○	○	○	○	○
F 2.11	Die Anzeige der modernen Vorschläge war übersichtlich.	○	○	○	○	○
F 2.12	Der **matched-tab** war übersichtlich.	○	○	○	○	○
F 2.13	Das manuelle hinzufügen eines Belegs war einfach.	○	○	○	○	○
F 2.14	Das Filtern der Belege war hilfreich.	○	○	○	○	○
F 2.15	Das Editieren eines Beleges war einfach.	○	○	○	○	○
F 2.16	Die Tabelle der Belege war übersichtlich.	○	○	○	○	○
F 2.17	Der **Automatic Evidences** Dialog war übersichtlich.	○	○	○	○	○
F 2.18	Die Text Abschnitte im **HistoricText** waren lang genug.	○	○	○	○	○

Abbildung 11.6.: Fragebogen Seite 2

Fragen zum RuleSelector (Rules)		1	2	3	4	5
F 2.19	Es war einfach die Regeln für einen **bestimmten Beleg** zu finden.	○	○	○	○	○
F 2.20	Es war einfach eine **bestimmte** Regel zu finden.	○	○	○	○	○
F 2.21	Es war einfach den *Rule Selector* zum **Browsen** der Regeln zu verwenden.	○	○	○	○	○
F 2.22	Die Bedienung der Filterung war eindeutig.	○	○	○	○	○
F 2.23	Die Filterung war hilfreich.	○	○	○	○	○

Fragen zur RuleVizualisation (Rules)		1	2	3	4	5
F 2.24	Die Visualisierung der Regeln war übersichtlich.	○	○	○	○	○
F 2.25	Die Visualisierung der Regeln war bei der Modifizierung der Regelbasis hilfreich.	○	○	○	○	○
F 2.26	Die Funktion der Buttons in der *Rule Visualization* war eindeutig.	○	○	○	○	○
F 2.27	Die expandierbaren Knoten *(in der Visualisierung rechts)* haben die Übersicht verbessert.	○	○	○	○	○
F 2.28	Die Animationen in der Rule Visualization waren verwirrend.	○	○	○	○	○
F 2.29	Die Animationen in der Rule Visualization waren hilfreich.	○	○	○	○	○

Abbildung 11.7.: Fragebogen Seite 3

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

[Abels und Hahn 2005] Abels, Sven ; Hahn, Axel: Pre-processing text for web information retrieval purposes by splitting compounds into their morphemes. In: Beigbeder, Michel (Hrsg.) ; Yee, Wai Gen (Hrsg.): *Open Source Web Information Retrieval, Compiègne, France, September 19, 2005*. Compiègne, Frankreich, September 2005, S. 7–10. – ISBN 2-913923-19-4

[Adriaans 2005] Adriaans, Frans: *Historic Document Retrieval: Exploring Strategies for 17th Century Dutch*, University of Amsterdam, Masterarbeit, Juni 2005

[Agbaria 2009] Agbaria, Evelyn: *Die deutsche Rechtschreibung*. Pons GmbH, 2009 (Einfach richtig). – URL `http://books.google.de/books?id=oPE2lZrFFT8C`. – ISBN 9783125170858

[Archer u. a. 2003] Archer, Dawn ; McEnery, Tony ; Rayson, Paul ; Hardie, Andrew: Developing an automated semantic analysis system for Early Modern English. In: Archer, Dawn (Hrsg.) ; Rayson, Paul (Hrsg.) ; Wilson, Andrew (Hrsg.) ; McEnery, Tony (Hrsg.): *Proceedings of the Corpus Linguistics 2003 conference* Bd. 16, Centre for Computer Corpus Research on Language: Lancaster University, März 2003, S. 22–31. – ISBN 1 86220 131 5

[Audenaert u. a. 2006] Audenaert, Neal ; Furuta, Richard ; Urbina, Eduardo: A General Framework for Feature Identification. In: [Sun u. a., 2006], S. 5–8

[Awakian 2010] Awakian, Ara: *Development of a User Interface for Interactive Rule Generation*, Universität Duisburg-Essen, Masterarbeit, Juli 2010

[Baker 2007] Baker, Peter S.: *Introduction to Old English*. Blackwell Publishing, 2007. – ISBN 1405152729

[Baron und Rayson 2009] Baron, A. ; Rayson, P.: Automatic standardisation of texts containing spelling variation: How much training data do you need? In: *Corpus Linguistics 2009*. Liverpool, UK, 2009

[Baron 2011] Baron, Alistair: *Dealing with Spelling Variation in Early Modern English Texts*, Lancaster University, Dissertation, Februar 2011

[Baron und Rayson 2008] Baron, Alistair ; Rayson, Paul: VARD 2: A tool for dealing with spelling variation in historical corpora. In: *Proceedings of the Postgraduate Conference in Corpus Linguistics*, 2008. – Aston University, Birmingham

[Baron u. a. 2009] Baron, Alistair ; Rayson, Paul ; Archer, Dawn: Automatic Standardization of Spelling for Historical Text Mining. In: *Proceedings of Digital Humanities 2009*. University of Maryland, USA, Juni 2009

[Bates 1990] Bates, Marcia J.: Where Should the Person Stop and the Information Search Interface Start? In: *Information Processing and Management* 26 (1990), Oktober, Nr. 5, S. 575–591. – ISSN 0306-4573

[Bates 2002] Bates, Marcia J.: The Cascade of Interactions in the Digital Library Interface. In: *Information Processing and Management* 38 (2002), Mai, S. 381–400. – ISSN 0306-4573

[Becker und König 2002] Becker, Tanja ; König, Esther: Lexikonfreie Lemmatisierung für Substantive des Deutschen. In: *KONVENS 2002*. Saarbrücken, September 2002

[Becks u. a. 2011] Becks, Daniela ; Mandl, Thomas ; Womser-Hacker, Christa: Spezielle Anforderungen bei der Evaluierung von Patent-Retrieval-Systemen. In: Griesbaum, Joachhim (Hrsg.) ; Mandl, Thomas (Hrsg.) ; Womser-Hacker, Christa (Hrsg.): *Information und Wissen: global, sozial und frei? - Proceedings des 12. Internationalen Symposiums für Informationswissenschaft (ISI 2011)* Bd. 58. Boizenburg : Verlag Werner Hülsbusch, 2011, S. 197–208. – ISSN 0938-8710

[Biella u. a. 2003] Biella, Daniel ; Dyllong, Eva ; Kaiser, Herbert ; Luther, Wolfram ; Mittmann, Thomas: Edition électronique de la réception de Nietzsche des années 1865 à 1945. In: *Proceedings International Cultural Heritage Informatics Meeting (ICHIM03)*. Toronto, Kanada : Archimuse, September 2003

[Biella u. a. 2005] Biella, Daniel ; Dyllong, Eva ; Luther, Wolfram ; Pilz, Thomas: An On-line Literature Research System with Rule-Based Search. In: *Proceedings of the 4th European Conference on e-Learning (ECEL2005)*. Amsterdam, Niederlande : ACL, Reading, November 2005, S. 67–76. – ISBN 1-905305-12-5

[Bollmann u. a. 2011] Bollmann, Marcel ; Petran, Florian ; Dipper, Stefanie: Rule-Based Normalization of Historical Texts. In: *Proceedings of the RANLP 2011 Workshop on Language Technologies for Digital Humanities and Cultural Heritage*. Hissar, Bulgaria, September 2011, S. 34–42

[Braun 2002] Braun, Loes: *Information Retrieval from Dutch Historic Corpora*, Universiteit Maastricht, Masterarbeit, November 2002

[Breitzman 2005] Breitzman, Anthony: Automated identification of technologically similar organizations. In: *Journal of the American Society for Information Science and Technology* 56 (2005), Nr. 10, S. 1015–1023. – ISSN 1532-2890

[Brill und Moore 2000] Brill, Eric ; Moore, Robert C.: An improved error model for noisy channel spelling correction. In: *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. Stroudsburg, PA, USA : Association for Computational Linguistics, 2000 (ACL '00), S. 286–293

[Buck u. a. 2013] Buck, Christoph ; Pilz, Thomas ; Dubovizky, Katharina ; Luther, Wolfram: Personalized fuzzy search in historical texts with non standard spelling. In: Rata, Gerogeta (Hrsg.): *Linguistic Studies of Human Language*, 2013, S. 325–336. – ISBN 978-960-9549-45-5

[Camps und Daudé 2003] Camps, Rafael ; Daudé, Jordi: Improving the efficacy of approximate personal name matching. In: Düsterhöft, Antje (Hrsg.) ; Thalheim, Bernhard (Hrsg.): *Natural Language Processing and Information Systems, 8th International Conference on Applications of Natural Language to Information Systems*, Bonner Köllen Verlag, 2003 (GI-Edition - Lecture Notes in Informatics (LNI)), S. 70–76. – ISBN 3-88579-358-X

[Carstensen u. a. 2009] Carstensen, Kai-Uwe ; Ebert, Christian ; Jekat, Susanne ; Ebert, Cornelia ; Langer, Hagen ; Klabunde, Ralf ; Carstensen, Kai-Uwe (Hrsg.) ; Ebert, Christian (Hrsg.) ; Jekat, Susanne (Hrsg.) ; Ebert, Cornelia (Hrsg.) ; Langer, Hagen (Hrsg.) ; Klabunde, Ralf (Hrsg.): *Computerlinguistik und Sprachtechnologie: Eine Einführung*. Spektrum Akademischer Verlag, 2009. – 3. Auflage. – ISBN 3827420237

[Cendrowska 1987] Cendrowska, Jadzia: PRISM: An Algorithm for Inducing Modular Rules. In: *International Journal of Man-Machine Studies* 27 (1987), Nr. 4, S. 349–370. – ISSN 0020-7373

[Chavez-Demoulin u. a. 2000] Chavez-Demoulin, V. C. ; Roehrl, R. A. ; Roehrl, R. A. ; Weinberg, A.: *The WEB archives: A time-machine in your pocket!* März 2000. – URL `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.31.7173&rep=rep1&type=pdf`

[Cohen und Singer 1999] Cohen, William W. ; Singer, Yoram: Context-Sensitive Learning Methods for Text Categorization. In: *ACM Trans. Inf. Syst.* 17 (1999), April, Nr. 2, S. 141–173. – ISSN 1046-8188

[Cortez u. a. 2007] Cortez, Eli ; Silva, Altigran S. da ; Gonçalves, Marcos A. ; Mesquita, Filipe ; Moura, Edleno S. de: FLUX-CiM: Flexible Unsupervised Extraction of Citation Metatdata. In: [Rasmussen u. a., 2007], S. 215–224. – ISBN 978-1-59593-644-8

[Couturier u. a. 2007] Couturier, Olivier ; Hamrouni, Tarek ; Yahia, Sadok B. ; Nguifo, Engelbert M.: A scalable association rule visualization towards displaying large amounts of knowledge. In: *Proceedings of the 11th International Conference Information Visualization*. Washington, DC, USA : IEEE Computer Society, 2007 (IV '07), S. 657–663. – ISBN 0-7695-2900-3

[Crane 2002] Crane, Gregory: *Cultural heritage digital libraries: Needs and components*. Bd. 2458. S. 626–637. In: Agosti, Maristella (Hrsg.) ; Thanos, Costantino (Hrsg.): *Proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries* Bd. 2458. London, UK : Springer, 2002. – ISBN 3-540-44178-6

[Crane 2007] Crane, Gregory: *Classics and the Computer: An End of the History*. S. 46–55. In: Schreibman, Susan (Hrsg.) ; Siemens, Ray (Hrsg.) ; Unsworth, John (Hrsg.): *A Companion to Digital Humanities*, Blackwell Publishing, 2007. – ISBN 9780470999875

[Crane u. a. 2007] Crane, Gregory ; Bamman, David ; Babeu, Alison: *Philology in an Electronic Age*. 2007. – URL `http://hdl.handle.net/10427/42688`

[Dollinger 2005] Dollinger, Rainer: *Optimierung von Web-Suchmaschinen am Beispiel von Apache Lucene*, FH Oberösterreich, Diplomarbeit, September 2005

[Downie u. a. 2006] Downie, J. Stephen ; Jones, M. Cameron ; Hu, Xiao: Voice Mining: A Promising New Application of Data Mining Techniques in the Humanities Domain. In: [Sun u. a., 2006], S. 299–301

[Ernst-Gerlach und Crane 2008] Ernst-Gerlach, Andrea ; Crane, Gregory: Identifying Quotations in Reference Works and Primary Materials. In: *Proceedings of the 12th European conference on Research and Advanced Technology for Digital Libraries*. Berlin, Heidelberg : Springer-Verlag, 2008 (ECDL '08), S. 78–87. – ISBN 978-3-540-87598-7

[Ernst-Gerlach und Fuhr 2006] Ernst-Gerlach, Andrea ; Fuhr, Norbert: Generating Search Term Variants for Text Collections with Historic Spellings. In: [Lalmas u. a., 2006], S. 49–60. – ISBN 3540333479

[Ernst-Gerlach und Fuhr 2007] Ernst-Gerlach, Andrea ; Fuhr, Norbert: Retrieval in Text Collections with Historic Spelling using Linguistic and Spelling Variants. In: [Rasmussen u. a., 2007], S. 333–341. – ISBN 978-1-59593-644-8

[Ernst-Gerlach und Fuhr 2010a] Ernst-Gerlach, Andrea ; Fuhr, Norbert: Advanced Training Set Construction for Retrieval in Historic Documents. In: Cheng, Pu-Jen (Hrsg.) ; Kan, Min-Yen (Hrsg.) ; Lam, Wai (Hrsg.) ; Nakov, Preslav (Hrsg.): *Information Retrieval Technology: 6th Asia Information Retrieval Societies Conference, AIRS 2010 Taipei, Taiwan, December 1-3, 2010*

Proceedings Bd. 6458. Berlin/Heidelberg : Springer, 2010, S. 131–140. – ISBN 978-3-642-17186-4

[Ernst-Gerlach und Fuhr 2010b] Ernst-Gerlach, Andrea ; Fuhr, Norbert: Semiautomatische Konstruktion von Trainingsdaten für historische Dokumente. In: Atzmueller, Martin (Hrsg.) ; Benz, Dominik (Hrsg.) ; Hotho, Andreas (Hrsg.) ; Stumme, Gerd (Hrsg.): *LWA 2010 Lernen, Wissen & Adaptivität Workshop Proceedings*, URL `http://www.kde.cs.uni-kassel.de/conf/lwa10/proceedings/proceedings.pdf`, 2010, S. 193–198

[Ernst-Gerlach u. a. 2011] Ernst-Gerlach, Andrea ; Korbar, Dennis ; Awakian, Ara: Entwicklung einer Benutzeroberfläche zur interaktiven Regelgenerierung für die Suche in historischen Dokumenten. In: Griesbaum, Joachhim (Hrsg.) ; Mandl, Thomas (Hrsg.) ; Womser-Hacker, Christa (Hrsg.): *Information und Wissen: global, sozial und frei? - Proceedings des 12. Internationalen Symposiums für Informationswissenschaft (ISI 2011)* Bd. 58. Boizenburg : Verlag Werner Hülsbusch, 2011, S. 209–220. – ISSN 0938-8710

[Esser 2004] Esser, Wolfram M.: Fault-Tolerant Fulltext Information Retrieval in Digital Multilingual Encyclopedias with Weighted Pattern Morphing. In: McDonald, Sharon (Hrsg.) ; Tait, John (Hrsg.): *Advances in Information Retrieval* Bd. 2997, Springer, 2004, S. 338–352. – ISBN 978-3-540-21382-6

[Ferber 2003] Ferber, Reginald: *Information Retrieval. Suchmodelle und Data-Mining-Verfahren für Textsammlungen und das Web.* Heidelberg : dpunkt Verlag, 2003. – ISBN 978-3-89864-213-2

[Fliedner 2001] Fliedner, Gerhard: *Computerlinguistik und Sprachtechnologie: Eine Einführung.* Kap. Korrekturprogramme, S. 411–417, Spektrum Akademischer Verlag, 2001. – ISBN 3827410274

[Foo und Hendry 2007] Foo, Schubert ; Hendry, Douglas: *Evaluation of Visual Aid Suite for Desktop Searching.* Bd. 4675. S. 333–344. In: Kovács, László (Hrsg.) ; Fuhr, Norbert (Hrsg.) ; Meghini, Carlo (Hrsg.): *Research and Advanced Technology for Digital Libraries* Bd. 4675, Springer, 2007. – ISBN 978-3-540-74850-2

[Fuhr 1999] Fuhr, Norbert: Information Retrieval in Digitalen Bibliotheken. In: *21. DGI-Online-Tagung – Aufbruch ins Wissensmanagement.* Frankfurt : DGI, 1999

[Fuhr 2005] Fuhr, Norbert.: Information Retrieval — From Information Access to Contextual Retrieval. In: Eibl, Maximilian (Hrsg.) ; Wolff, Christian (Hrsg.) ; Womser-Hacker, Christa (Hrsg.): *Designing Information Systems. Festschrift für Jürgen Krause.* Konstanz : UVK Verlagsgesellschaft, 2005, S. 47–57. – ISBN 3-89669-564-9.

[Furuta u. a. 2001] Furuta, Richard ; Kalasapur, Siddarth S. ; Kochumman, Rajiv ; Urbina, Eduardo ; Vivancos-Pérez, Ricardo: The Cervantes Project: Steps to a Customizable and Interlinked On-line Electronic Variorum Edition Supporting Scholarship. In: Constantopoulos, Panos (Hrsg.) ; Sølvberg, Ingeborg (Hrsg.): *Research and Advanced Technology for Digital Libraries: 5th European Conference, ECDL 2001* Bd. 2163, Springer, 2001, S. 71–82. – ISBN 3-540-42537-3

[Fürnkranz 1999] Fürnkranz, Johannes: Separate-and-Conquer Rule Learning. In: *Artificial Intelligence Review* 13 (1999), Januar, Nr. 1, S. 3–54. – ISSN 0269-2821

[Gadd 1990] Gadd, T. N.: PHOENIX: the algorithm. In: *Program* 24 (1990), September, Nr. 4, S. 363–369. – ISSN 0033-0337

[Garces 2006] Garces, Juan: The Septuagint and the Possibilities of Humanities Computing: from Assistant to Collaborator. In: [Sun u. a., 2006], S. 72–73

[Geyken und Hanneforth 2006] Geyken, Alexander ; Hanneforth, Thomas: TAGH: A Complete Morphology for German based on Weighted Finite State Automata. In: Yli-Jyrä, Anssi (Hrsg.) ; Karttunen, Lauri (Hrsg.) ; Karhumäki, Juhani (Hrsg.): *Finite State Methods and Natural Language Processing. 5th International Workshop, FSMNLP 2005, Helsinki, Finland, September 1-2, 2005. Revised Papers* Bd. 4002. Springer, 2006, S. 55–66. – ISBN 978-3-540-35469-7

[Giusti u. a. 2007] Giusti, Rafael ; Candido, Arnaldo ; Muniz, Marcelo ; Cucatto, Lívia ; Aluísio, Sandra M.: Automatic Detection of Spelling Variation in Historical Corpus: An Application to Build a Brazilian Portuguese Spelling Variants Dictionary. In: Davies, Matthew (Hrsg.) ; Rayson, Paul (Hrsg.) ; Hunston, Susan (Hrsg.) ; Danielsson, Pernilla (Hrsg.): *Proceedings of the Corpus Linguistics Conference CL2007* University of Birmingham (Veranst.), Juli 2007, S. 1–20

[Gotscharek u. a. 2009a] Gotscharek, Annette ; Neumann, Andreas ; Reffle, Ulrich ; Ringlstetter, Christoph ; Schulz, Klaus U.: Enabling information retrieval on historical document collections: the role of matching procedures and special lexica. In: *Proceedings of The Third Workshop on Analytics for Noisy Unstructured Text Data*. New York, NY, USA : ACM, 2009 (AND '09), S. 69–76. – ISBN 978-1-60558-496-6

[Gotscharek u. a. 2009b] Gotscharek, Annette ; Reffle, Ulrich ; Ringlstetter, Christoph ; Schulz, Klaus U.: On Lexical Resources for Digitization of Historical Documents. In: *Proceedings of the 9th ACM symposium on Document engineering (DOCENG 2009)*. New York, NY, USA, 2009 (DocEng '09), S. 193–200. – ISBN 978-1-60558-575-8

[Gotscharek u. a. 2011] Gotscharek, Annette ; Reffle, Ulrich ; Ringlstetter, Christoph ; Schulz, Klaus U. ; Neumann, Andreas: Towards information retrieval on historical document collections: the role of matching procedures and special lexica. In: *International Journal on Document Analysis and Recognition* 14 (2011), Juni, S. 159–171. – ISSN 1433-2833

[Graf und Azzopardi 2008] Graf, Erik ; Azzopardi, Leif: *A methodology for building a patent test collection for prior art search*. S. 60–71. In: *Proceedings of The Second International Workshop on Evaluating Information Access*, 2008

[Hauser 2007] Hauser, Andreas: *OCR Postcorrection of Historical Texts*, Ludwig-Maximilians-Universität München, Magisterarbeit, Oktober 2007

[Hauser u. a. 2007] Hauser, Andreas ; Heller, Markus ; Leiss, Elisabeth ; Schulz, Klaus U. ; Wanzeck, Christiane: Information Access to Historical Documents from the Early New High German Period. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-2007) Workshop on Analytics for Noisy Unstructured Text Data*, Januar 2007

[Hausser 2000] Hausser, Roland: *Grundlagen der Computerlinguistik*. Berlin; Heidelberg [u.a.] : Springer, 2000. – ISBN 3-540-67187-0

[Hearst 1999] Hearst, Marti A.: *Modern Information Retrieval*. Kap. User Interfaces and Visualization, S. 257–323. Boston, MA, USA : Addison-Wesley, 1999. – ISBN 020139829X

[Hedstrom und Ross 2003] Hedstrom, Margaret (Hrsg.) ; Ross, Seamus (Hrsg.): *Invest to Save: Report and Recommendations of the NSF-DELOS Working Group on Digital Archiving and Preservation*. 2003. – Prepared for the National Science Foundation's (NSF) Digital Library Initiative & the European Union

[Heller 2004] Heller, Markus: Ein Indexierungssystem für CEI-kodierte Urkunden. Suchmaschinentechnologie für die historischen Wissenschaften. In: *Berichte der Jahrestagung 2004 der AGE*, 2004

[Heller 2006] Heller, Markus: Approximative Indexierungstechnik für historische deutsche Textvarianten. In: *Abhandlungen der Arbeitsgemeinschaft Geschichte und EDV* Bd. 31. Center for Historical Social Research, 2006, S. 288–307

[Hockey 2004] Hockey, Susan: The history of humanities computing. Malden : Blackwell, 2004 (Blackwell companions to literature and culture), Kap. 1

[Holley 2009] Holley, Rose: How Good Can It Get? Analysing and Improving OCR Accuracy in Large Scale Historic Newspaper Digitisation Programs. In: *D-Lib Magazine* 15 (2009), März/April, Nr. 3/4. – URL `http://www.dlib.org/dlib/march09/holley/03holley.html`. – ISSN 1082-9873

[Höhn 2008] Höhn, Winfried: Heuristiken zum Postprocessing von OCR-Ergebnissen. In: Baumeister, Joachim (Hrsg.) ; Atzmüller, Martin (Hrsg.): *Proc. Lernen, Wissen & Adptivität 2008, Würzburg, Germany* Bd. 448, Department of Computer Science, University of Würzburg, Germany, Oktober 2008, S. 78–82

[Ingwersen und Christensen 1997] Ingwersen, Peter ; Christensen, Finn H.: Data set isolation for bibliometric online analyses of research publications: Fundamental methodological issues. In: *Journal of the American Society for Information Science* 48 (1997), Nr. 3, S. 205–217. – ISSN 1097-4571

[Jessop 2006] Jessop, Martyn: The Inhibition of Geographical Information in Digital Humanities Scholarship. In: [Sun u. a., 2006], S. 100–103

[Johnson 2006] Johnson, Ian: The SHSSERI Collaborative KnowledgeSpace: a New Approach to Resource Fragmentation and Information Overload. In: [Sun u. a., 2006], S. 103–105

[Jordan 2005] Jordan, Matthias: *DAFFODIL: Proaktive Vorlagefunktionen*, Universität Dortmund, Diplomarbeit, 2005

[Juola 2006] Juola, Patrick: Killer Applications in Digital Humanities. In: [Sun u. a., 2006], S. 105–106

[Jussen u. a. 2007] Jussen, Bernhard ; Mehler, Alexander ; Ernst, Alexandra: A Corpus Management System for Historical Semantics. In: *Sprache und Datenverarbeitung. International Journal for Language Data Processing* 31 (2007), Nr. 1-2, S. 81–89

[Järvelin u. a. 2006] Järvelin, Anni ; Kumpulainen, Sanna ; Pirkola, Ari ; Sormunen, Eero: Dictionary-independent translation in CLIR between closely related languages. In: *Proceedings of the 6th Dutch-Belgian Infromation Retrieval Workshop (DIR 2006)*. Delft, Niederlande, 2006, S. 25–32

[Kempken 2005] Kempken, Sebastian: *Bewertung historischer und regionaler Schreibvarianten mit Hilfe von Abstandsmaßen*, Universität Duisburg-Essen, Diplomarbeit, Dezember 2005

[Kempken u. a. 2006] Kempken, Sebastian ; Luther, Wolfram ; Pilz, Thomas: Comparison of distance measures for historical spelling variants. In: Bramer, Max (Hrsg.): *Artificial Intelligence in Theory and Practice* Bd. 217. Springer Boston, 2006, S. 295–304. – ISBN 978-0-387-34654-0

[Kempken u. a. 2007] Kempken, Sebastian ; Pilz, Thomas ; Luther, Wolfram: Visualization of rule productivity in deriving nonstandard spellings. In: *Proc. SPIE 6495, Visualization and Data Analysis 2007* Bd. 6495, International Society for Optical Engineering, Januar 2007, S. 64950M. – ISSN 0277-786X

[Kendall und French 2006] Kendall, Tyler ; French, Amanda: Digital Audio Archives, Computer-Enhanced Transcripts, and New Methods in Sociolinguistic Analysis. In: [Sun u. a., 2006], S. 110–112

[Kernighan u. a. 1990] Kernighan, Mark D. ; Church, Kenneth W. ; Gale, William A.: A spelling correction program based on a noisy channel model. In: Karlgren, Hans (Hrsg.): *Proceedings of the 13th conference on Computational linguistics - Volume 2*. Stroudsburg, PA, USA : Association for Computational Linguistics, 1990 (COLING '90), S. 205–210. – ISBN 952-90-2028-7

[Khaltarkhuu und Maeda 2008] Khaltarkhuu, Garmaabazar ; Maeda, Akira: Developing a Traditional Mongolian Script Digital Library. In: Buchanan, George (Hrsg.) ; Masoodian, Masood (Hrsg.) ; Cunningham, Sally (Hrsg.): *Proceedings of the 11th International Conference on Asian Digital Libraries: Universal and Ubiquitous Access to Information (ICADL 08)* Bd. 5362. Berlin, Heidelberg : Springer, 2008, S. 41–50. – ISBN 978-3-540-89532-9

[Klas 2007] Klas, Claus-Peter: *DAFFODIL Strategische Unterstützung bei der Informationssuche in Digitalen Bibliotheken*, Universität Duisburg-Essen, Dissertation, Juni 2007

[Koolen 2005] Koolen, Marijn: *Constructing Language Resources for Historic Document Retrieval*, University of Amsterdam, Masterarbeit, Juni 2005

[Koolen u. a. 2006] Koolen, Marijn ; Adriaans, Frans ; Kamps, Jaap ; Rijke, Maarten de: A Cross-Language Approach to Historic Document Retrieval. In: [Lalmas u. a., 2006], S. 407–419. – ISBN 3540333479

[Korbar 2010] Korbar, Dennis: *Visualisierung von Regelstrukturen und -Modifikationsmöglichkeiten für die Suche in Texten mit nicht-standardisierter Rechtschreibung*, Universität Duisburg-Essen, Diplomarbeit, Juli 2010

[Koster 2006] Koster, Elwin: How to Annotate Historical Townplans? In: [Sun u. a., 2006], S. 113–114

[Kranz 1998] Kranz, Florian: *Eine Schifffahrt Mit Drei F: Positives Zur Rechtschreibreform*. Vandenhoeck & Ruprecht, 1998 (Kleine Reihe V&R). – ISBN 9783525340059

[Kriewel u. a. 2004] Kriewel, Sascha ; Klas, Claus-Peter ; Schaefer, André ; Fuhr, Norbert: Daffodil - Strategic Support for User-Oriented Access to Heterogeneous Digital Libraries. In: *D-Lib Magazine* 10 (2004), Juni, Nr. 6. – URL `http://www.dlib.org/dlib/june04/kriewel/06kriewel.html`. – ISSN 1082-9873

[Lalmas u. a. 2006] Lalmas, M. (Hrsg.) ; MacFarlane, A. (Hrsg.) ; Rueger, S. (Hrsg.) ; Trombos, A. (Hrsg.) ; Tsikrika, T. (Hrsg.) ; Yavlinsky, A. (Hrsg.): *Advances in Information Retrieval - 28th European Conference on IR Research, ECIR 2006, London, UK, April 10-12, 2006*. Bd. *3936*. Heidelberg : Springer Verlag, 2006. (Lecture Notes in Computer Science). – ISBN 3540333479

[Lemnitzer und Zinsmeister 2006] Lemnitzer, Lothar ; Zinsmeister, Heike: *Korpuslinguistik: Eine Einführung*. Tübingen : Narr Francke Attemto Verlag, 2006 (narr studienbücher)

[Levenshtein 1966] Levenshtein, Vladimir I.: Binary codes capable of correcting deletions, insertions, and reversals. 1966 (8). – Forschungsbericht. – 707–710 S

[Lezius u. a. 1998] Lezius, Wolfgang ; Rapp, Reinhard ; Wettler, Manfred: A freely available morphological analyzer, disambiguator and context sensitive lemmatizer for German. In: *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 2*. Stroudsburg, PA, USA : Association for Computational Linguistics, 1998 (ACL '98), S. 743–747

[Li u. a. 2007] Li, Xin ; Chen, Hsimchun ; Zhang, Zhu ; Li, Jiexun: Automatic Petent Classification using Citation Network Information: An Experimental Study in Nanotechnology. In: [Rasmussen u. a., 2007], S. 419–427. – ISBN 978-1-59593-644-8

[Lischka 2011] Lischka, K.: Der Staat spart Google digitalisiert. In: *Spiegel Online* (2011), 26. März. – URL `http://www.spiegel.de/netzwelt/netzpolitik/digitale-bibliotheken-der-staat-spart-google-digitalisiert-a-753229.html`

[Liu u. a. 1991] Liu, Lon-Mu ; Babad, Yair M. ; Sun, Wei ; Chan, Ki-Kan: Adaptive post-processing of OCR text via knowledge acquisition. In: *Proceedings of the 19th annual conference on Computer Science*. New York, NY, USA : ACM, 1991 (CSC '91), S. 558–569. – ISBN 0-89791-382-5

[Lu u. a. 1998] Lu, Xin A. ; Miller, David J. ; Richard, Wassum J.: *Phrase recognition method and apparatus*. Oktober 1998. – US Patent 5819260

[Lüdeling und Walter 2010] Lüdeling, Anke ; Walter, Maik: *Handbuch Deutsch als Fremd- und Zweitsprache (Neubearbeitung)*. Kap. Korpuslinguistik, S. 315–322. Berlin : Mouton de Gruyter, 2010 (Handbücher zur Sprach- und Kommunikationswissenschaft 35)

[Mangu und Brill 1997] Mangu, Lidia ; Brill, Eric: Automatic Rule Acquisition for Spelling Correction. In: *Proceedings of the 14th International Conference on*

Machine Learning. San Francisco, CA, USA : Morgan Kaufmann, 1997 (ICML '97), S. 187–194. – ISBN 1-55860-486-3

[Mehler u. a. 2009] Mehler, Alexander ; Gleim, Rüdiger ; Waltinger, Ulli ; Ernst, Alexandra ; Esch, Dietmar ; Feith, Tobias: eHumanities Desktop — eine webbasierte Arbeitsumgebung für die geisteswissenschaftliche Fachinformatik. In: *Proceedings of the Symposium Sprachtechnologie und eHumanities, 26. and 27. February, Duisburg-Essen University* Universität Duisburg-Essen (Veranst.), 2009

[Mehler u. a. 2011] Mehler, Alexander ; Schwandt, Silke ; Gleim, Rüdiger ; Jussen, Bernhard: Der eHumanities Desktop als Werkzeug in der historischen Semantik: Funktionsspektrum und Einsatzszenarien. In: *Journal for Language Technology and Computational Linguistics (JLCL)* 26 (2011), Nr. 1, S. 97–117

[Michael 1999] Michael, Jörg: Doppelgänger gesucht Ein Programm für kontextsensitive phonetische Textumwandlung. In: *c't – Magazin für Computertechnik* (1999), Nr. 25, S. 252–271

[Mihov u. a. 2005] Mihov, Stoyan ; Schulz, Klaus U. ; Ringlstetter, Christoph ; Dojchinova, Veselka ; Nakova, Vanja: A Corpus for Comparative Evaluation of OCR Software and Postcorrection Techniques. In: *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on* Bd. 1. Washington, DC, USA : IEEE Computer Society, 2005, S. 162–166. – ISSN 1520-5263

[Mischke und Luther 2005] Mischke, Lothar ; Luther, Wolfram: Document Image De-warping Based on Detection of Distorted Text Lines. In: Roli, Fabio (Hrsg.) ; Vitulano, Sergio (Hrsg.): *Proceedings of the 13th international conference on Image Analysis and Processing – ICIAP 2005* Bd. 3617. Berlin, Heidelberg : Springer-Verlag, 2005, S. 1068–1075. – ISBN 978-3-540-28869-5

[Mitchell 2001] Mitchell, Tom M.: *Machine learning*. New York [u.a.] : McGraw-Hill, 2001 (McGraw Hill series in computer science). – ISBN 0-07-115467-1

[Modes 2012] Modes, Sarah: *Dialekt und Orthographie: Eine Untersuchung des Einflusses von Dialekt auf die Orthographie von Grundschulkindern*. GRIN Verlag, 2012. – ISBN 9783656193913

[Moreau u. a. 2007] Moreau, Fabienne ; Claveau, Vincent ; Sébillot, Pascale: Automatic morphological query expansion using analogy-based machine learning. In: Amati, Giambattista (Hrsg.) ; Carpineto, Claudio (Hrsg.) ; Romano, Giovanni (Hrsg.): *Proceedings of the 29th European conference on IR research* Bd. 4425. Berlin, Heidelberg : Springer-Verlag, 2007, S. 222–233. – ISBN 978-3-540-71494-1

[Mostern 2006] Mostern, Ruth: Digital Gazetteers and Temporal Directories for Digital Atlases. In: [Sun u. a., 2006], S. 149–150

[Nguyen u. a. 2006] Nguyen, DucDung ; Ho, TuBao ; Kawasaki, Saori: Knowledge visualization in hepatitis study. In: Misue, Kazuo (Hrsg.) ; Sugiyama, Kozo (Hrsg.) ; Tanaka, Jiro (Hrsg.): *Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation, APVIS 2006, Tokyo, Japan, February 1-3, 2006* Bd. 60. Darlinghurst, Australia : Australian Computer Society, 2006, S. 59–62. – ISBN 1-920682-41-4

[Nwabueze 2011] Nwabueze, Emmanuel: *Rule-Based Google Search For Historic Documents*, Universität Duisburg-Essen, Bachelorarbeit, Juli 2011

[Ogden u. a. 1999] Ogden, William ; Cowie, James ; Davis, Mark ; Ludovik, Eugene ; Molina-salgado, Hugo ; Shin, Hyopil: Getting Information from Documents You Cannot Read: An Interactive Cross-Language Text Retrieval and Summarization System. In: *Joint ACM DL/SIGIR Workshop on Multilingual Information Discovery and Access*, 1999

[O'Rourke u. a. 1997] O'Rourke, Alan J. ; Robertson, Alexander M. ; Willett, Peter ; Eley, Penny ; Simons, Penny: Word Variant Identification in Old French. In: *Information Research* 2 (1997), Nr. 4

[Peters 2001] Peters, Carol (Hrsg.): *Cross-Language Information Retrieval and Evaluation*. Bd. *2069*. Berlin : Springer, 2001. (Lecture Notes in Computer Science)

[Petersohn 2005] Petersohn, Helge: *Data Mining: Verfahren, Prozesse, Anwendungsarchitektur*. München : Oldenbourg, 2005. – ISBN 978-3-486-57715-0

[Pfeifer u. a. 1996] Pfeifer, Ulrich ; Poersch, Thomas ; Fuhr, Norbert: Retrieval Effectiveness of Proper Name Search Methods. In: *Information Processing and Management* 32 (1996), November, Nr. 6, S. 667–679. – ISSN 0306-4573

[Pilz 2003] Pilz, Thomas: *Unscharfe Suche in Textdatenbanken mit nichtstandardisierter Rechtschreibung am Beispiel von Frakturtexten zur Nietzsche-Rezeption*, Universität Duisburg-Essen, Staatsexamensarbeit, 2003

[Pilz 2009] Pilz, Thomas: *Nichtstandardisierte Rechtschreibung - Variationsmodellierung und rechnergestützte Variationsverarbeitung.*, Universität Duisburg-Essen, Dissertation, September 2009

[Pilz u. a. 2008a] Pilz, Thomas ; Ernst-Gerlach, Andrea ; Kempken, Sebastian ; Rayson, Paul ; Archer, Dawn: The Identification of Spelling Variants in English and German Historical Texts: Manual or Automatic? In: *Literary and Linguistic Computing* 23 (2008), Nr. 1, S. 65–72

[Pilz und Luther 2009] Pilz, Thomas ; Luther, Wolfram: Automated support for evidence retrieval in documents with nonstandard orthography. In: Featherston, Sam (Hrsg.) ; Winkler, Susanne (Hrsg.): *The Fruits of Empirical Linguistics* Bd. 1, Mouton de Gruyter, Mai 2009, S. 211–228

[Pilz u. a. 2008b] Pilz, Thomas ; Luther, Wolfram ; Ammon, Ulrich: Retrieval of Spelling Variants in Nonstandard Texts – Automated Support and Visualization. In: *SKY Journal of Linguistics* 21 (2008), S. 155–200

[Pilz u. a. 2005] Pilz, Thomas ; Luther, Wolfram ; Ammon, Ulrich ; Fuhr, Norbert: Rule-based search in text databases with non standard orthography. In: *Proceedings ACH/ALLC 2005, Victoria, 15 - 18 Jun 2005*, 2005

[Pilz u. a. 2007] Pilz, Thomas ; Philipsenburg, Axel ; Luther, Wolfram: Visualizing the Evaluation of Distance Measures. In: *Proceedings of Ninth Meeting of the ACL Special Interest Group in Computational Morphology and Phonology*. Stroudsburg, PA, USA : Association for Computational Linguistics, Juni 2007 (SigMorPhon '07), S. 84–92

[Pirkola u. a. 2003] Pirkola, Ari ; Toivonen, Jarmo ; Keskustalo, Heikki ; Visala, Kari ; Järvelin, Kalervo: Fuzzy translation of cross-lingual spelling variants. In: *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. New York, NY, USA : ACM, 2003 (SIGIR '03), S. 345–352. – ISBN 1-58113-646-3

[Postel 1969] Postel, Hans J.: Die Kölner Phonetik. Ein Verfahren zur Identifizierung von Personennamen auf der Grundlage der Gestaltanalyse. In: *IBM-Nachrichten* (1969), Nr. 19, S. 925–931

[Quasthoff 1998] Quasthoff, Uwe: Tools for Automatic Lexicon Maintenance: Acquisition, Error Correction, and the Generation of Missing Values. In: *Proceedings of the first International Conference on Language Resources & Evaluation* European Language Resources Association (Veranst.), Universitätsverlag, 1998, S. 853–856

[Quasthoff und Wolff 1999] Quasthoff, Uwe ; Wolff, Christian: Korpuslinguistik und große einsprachige Wörterbücher. In: *Linguistik Online* 3(2) (1999). – URL `http://epub.uni-regensburg.de/6848/`

[Quinlan 1986] Quinlan, J. Ross: Induction of Decision Trees. In: *Machine Learning* 1 (1986), März, S. 81–106. – ISSN 0885-6125

[Rasmussen u. a. 2007] Rasmussen, Edie M. (Hrsg.) ; Larson, Ray R. (Hrsg.) ; Toms, Elaine G. (Hrsg.) ; Sugimoto, Shigeo (Hrsg.): *ACM/IEEE Joint Conference on Digital Libraries, JCDL 2007, Vancouver, BC, Canada, June 18-23, 2007, Proceedings*. New York, NY, USA : ACM, 2007. – ISBN 978-1-59593-644-8

[Rauber u. a. 2002] Rauber, Andreas ; Aschenbrenner, Andreas ; Witvoet, Oliver ; Bruckner, Robert M. ; Kaiser, Max: Uncovering information hidden in Web archives. In: *D-Lib Magazine* 8 (2002), Dezember, Nr. 12. – URL `http://www.dlib.org/dlib/december02/rauber/12rauber.html`. – ISSN 082-9873

[Rayson u. a. 2005] Rayson, Paul ; Archer, Dawn ; Smith, Nick: VARD versus Word. A comparison of the UCREL variant detector and modern spell checkers on English historical corpora. In: *Proceedings of the Corpus Linguistics 2005 conference* Bd. 1. Birmingham, UK, 2005, S. 14–17

[Reffle 2011] Reffle, Ulrich: Efficiently generating correction suggestions for garbled tokens of historical language. In: *Natural Language Engineering* 17 (2011), April, S. 265–282. – ISSN 1351-3249

[Reynaert 2004a] Reynaert, Martin: Multilingual text induced spelling correction. In: *Proceedings of the COLING 2004 Workshop on Multilingual Linguistic Resources*. Stroudsburg, PA, USA : Association for Computational Linguistics, 2004 (MLR '04), S. 110–117

[Reynaert 2004b] Reynaert, Martin: Text induced spelling correction. In: *Proceedings of the 20th international conference on Computational Linguistics*. Stroudsburg, PA, USA : Association for Computational Linguistics, 2004 (COLING '04)

[Reynaert 2005] Reynaert, Martin: *Text-induced spelling correction*, University of Tilburg, Dissertation, Dezember 2005

[Rimmer u. a. 2006] Rimmer, Jon ; Warwick, Claire ; Blandford, Ann ; Gow, Jeremy ; Buchanan, George: User Requirements for Humanities Digital Libraries. In: [Sun u. a., 2006], S. 336–339

[Robertson und Willett 1998] Robertson, Alexander M. ; Willett, Peter: Applications of n-grams in textual information systems. In: *Journal of Documentation* 54 (1998), Nr. 1, S. 48–69

[Romanello u. a. 2009] Romanello, Matteo ; Boschetti, Federico ; Crane, Gregory: Citations in the digital library of classics: extracting canonical references by using conditional random fields. In: *Proceedings of the 2009 Workshop on Text and Citation Analysis for Scholarly Digital Libraries*. Stroudsburg, PA, USA : Association for Computational Linguistics, 2009 (NLPIR4DL '09), S. 80–87. – ISBN 978-1-932432-58-9

[Russel und Odell 1918] Russel, Robert ; Odell, Margaret: *1,261,167*. US Patent. 1918

[Selbach 2008] Selbach, Stefan: Verbesserte N-Gramm Volltextsuche in Kombination mit wortbasierter Suche. In: Baumeister, Joachim (Hrsg.) ; Atzmüller, Martin (Hrsg.): *Proc. Lernen, Wissen & Adptivität 2008, Würzburg, Germany* Bd. 448, Department of Computer Science, Universität Würzburg, Oktober 2008, S. 87–90

[Shannon 2001] Shannon, Claude E.: A mathematical theory of communication. In: *SIGMOBILE Mob. Comput. Commun. Rev.* 5 (2001), Januar, S. 3–55. – ISSN 1559-1662

[Shneiderman und Maes 1997] Shneiderman, Ben ; Maes, Pattie: Direct manipulation vs. interface agents. In: *interactions* 4 (1997), November, S. 42–61. – ISSN 1072-5520

[Stewart u. a. 2007] Stewart, Gordon ; Crane, Gregory ; Babeu, Alison: A New Generation of Textual Corpora. In: [Rasmussen u. a., 2007], S. 356–365. – ISBN 978-1-59593-644-8

[Strohmaier u. a. 2003] Strohmaier, Christian ; Ringstetter, Christoph ; Schulz, Klaus U. ; Mihov, Stoyan: A visual and interactive tool for optimizing lexical postcorrection of OCR results. In: *Proceedings of the IEEE Workshop on Document Image Analysis and Recognition, DIAR'03*, 2003

[Strunk 2003] Strunk, Jan: *Information Retrieval for Languages that lack a fixed orthography.* 2003. – URL `http://www.linguistics.ruhr-uni-bochum.de/~strunk/LSreport.pdf`

[Sun u. a. 2006] Sun, Chengan (Hrsg.) ; Menasri, Sabrina (Hrsg.) ; Ventura, Jérémy (Hrsg.): *Digital Humanities 2006: The First International Conference of the Alliance of Digital Humanities Organisations (ADHO).* Paris, Frankreich, 5 – 9 Juli 2006

[Taghva und Stofsky 2001] Taghva, Kazem ; Stofsky, Eric: OCRSpell: an interactive spelling correction system for OCR errors in text. In: *International Journal of Document Analysis and Recognition* 3 (2001), S. 125–137

[Tarassenko 2008] Tarassenko, Sergey: *Visualisierung mehrwertiger Attriute in Digitalen Bibliotheken*, Universität Duisburg-Essen, Diplomarbeit, Juni 2008

[TextGrid 2004] TextGrid: *TextGrid: Modulare Plattform für verteilte und kooperative wissenschaftliche Textdatenverarbeitung. Erstellung eines Community-Grid für die Geisteswissenschaften.* 2004. – URL `http://www.textgrid.de/fileadmin/TextGrid/TextGrid-Antrag_oeffentlich_060315.pdf`

[Toivonen u. a. 2005] Toivonen, Jarmo ; Pirkola, Ari ; Keskustalo, Heikki ; Visala, Kari ; Järvelin, Kalervo: Translating cross-lingual spelling variants using transformation rules. In: *Information Processing and Management* 41 (2005), Juli, S. 859–872. – ISSN 0306-4573

[Trotsche 1866] Trotsche, Carl Heinrich C.: *Der mecklenburgische Civil-Proceß.* Bd. 1: Allgemeiner Teil. 1866. – Signatur: Dt 19 Ak 55 [1]

[Wegener 2005] Wegener, Heide: Das Hühnerei vor der Hundehütte : von der Notwendigkeit historischen Wissens in der Grammatikographie des Deutschen. In: Berner, Elisabeth (Hrsg.) ; Böhm, Manueal (Hrsg.) ; Voeste, Anja (Hrsg.): *Ein gross und narhafft haffen : Festschrift für Joachim Gessinger.* Potsdam : Univ.-Verl., 2005, S. 248–260. – ISBN 3-937786-35-X

[Wermke u. a. 2009] Wermke, Matthias (Hrsg.) ; Kunkel-Razum, Kathrin (Hrsg.) ; Stolze-Stubenrecht, Werner (Hrsg.): *Duden - Die deutsche Rechtschreibung: auf der Grundlage der aktuellen amtlichen Rechtschreibregeln.* 25. Auflage. Mannheim : Dudenverlag, 2009

[Witten und Frank 2000] Witten, Ian H. ; Frank, Eibe: *Data Mining: Praktische Werkzeuge und Techniken für das maschinelle Lernen.* San Francisco, CA, USA : Morgan Kaufmann Publishers, 2000

[Zaslavsky u. a. 2001] Zaslavsky, Arkady B. ; Bia, Alejandro ; Monostori, Krisztián: Using Copy-Detection and Text Comparison Algorithms for Cross-Referencing Multiple Editions of Literary Works. In: Constantopoulos, Panos (Hrsg.) ; Sølvberg, Ingeborg (Hrsg.): *Proceedings of the 5th European Conference on Research and Advanced Technology for Digital Libraries* Bd. 2163. Berlin/Heidelberg : Springer, 2001, S. 103–114. – ISBN 3-540-42537-3

[Zipf 1935] Zipf, George K.: *The Psycho-Biology of Language: An Introduction to Dynamic Philology.* Cambridge, MA, USA : The M.I.T Press, 1935. – 2. Auflage

[Zobel und Dart 1996] Zobel, Justin ; Dart, Philip: Phonetic string matching: lessons from information retrieval. In: Frei, Hans-Peter (Hrsg.) ; Harman, Donna (Hrsg.) ; Schäuble, Peter (Hrsg.) ; Wilkinson, Ross (Hrsg.): *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* New York, NY, USA : ACM, 1996, S. 166–172. – ISBN 0-89791-792-8