Semiautomatische Konstruktion von Trainingsdaten für historische Dokumente

Andrea Ernst-Gerlach, Norbert Fuhr
Abteilung Informatik und Angewandte Kognitionswissenschaft
Fakultät für Ingenieurwissenschaften
Universität Duisburg-Essen
47048 Duisburg, Deutschland

Abstract

Für Retrieval in historischen Dokumenten wird eine Abbildung der Suchbegriffe auf die historischen Varianten in den Dokumenten benötigt. Für diese Abbildung wurde ein regelbasierter Ansatz entwickelt. Der Engpass dieses Ansatzes ist die Konstruktion der Trainingsdaten. Dabei muss ein Experte manuell den historischen Formen, die dem Spellchecker unbekannt sind, die aktuelle moderne Form zuordnen. Zur Verbesserung dieses Verfahrens werden nun die Vorschläge des Spellcheckers betrachtet. Aus jedem Vorschlag und dem zugehörigen unbekannten Wort wird ein Beleg gebildet. Aus diesen Belegen werden nun wie gewohnt Regeln generiert und die häufigsten Regeln akzeptiert. Experimentelle Ergebnisse basierend auf der bisherigen Belegkollektion zeigen, dass ein großer Teil der Regeln auf diese Weise generiert werden kann. Dadurch können die Trainingsdaten deutlich schneller und mit geringerem manuellem Aufwand erzeugt werden.

1 Einleitung

Die Anzahl der digitalen historischen Kollektionen steigt kontinuierlich. Aber bei verfügbarer Volltextsuche für die Kollektionen werden viele Dokumente nicht gefunden, weil sie eine nicht-standardisierte Rechtschreibung verwenden. In vielen Ländern war die Schreibweise über viele Jahrhunderte hinweg nicht festgelegt. So wurde z. B. die deutsche Sprache erst 1901/1902 standardisiert [Keller, 1986]. Davor galt das Prinzip "Schreibe wie Du sprichst" (phonologisches Prinzip der Rechtschreibung). Z. B. ist $akzeptieren$ die moderne Form der Schreibvariante $acceptieren$.

Die nicht-standardisierte Schreibweise führt zu Fehlern, wenn in historischen Teilen von digitalen Bibliotheken gesucht wird. Die meisten Benutzer geben den Suchbegriff in moderner Sprache ein, die sich von der historischen Sprache in den Dokumenten unterscheidet.

Auch populäre Digitalisierungsinitiativen wie Google Book Search[1] oder die europäische digitale Bibliothek[2] unterstützen bisher keine Suche nach Schreibvarianten. Um dieses Problem zu lösen, arbeitet unser Projekt an der Entwicklung einer Suchmaschine, bei der der Benutzer seine Anfragen in aktueller Schreibweise eingeben kann, wenn er in historischen Dokumenten suchen möchte (siehe [Ernst-Gerlach and Fuhr, 2007]).

Andere Ansätze benutzen dafür wörterbuchbasierte Methoden (z. B. [Hauser *et al.*, 2007]). Allerdings hat diese Vorgehensweise den entscheidenden Nachteil, dass nur Wörter gefunden werden, die im Wörterbuch enthalten sind. Außerdem ist der zeitliche Aufwand für den manuellen Aufbau der Wörterbücher relativ hoch.

Die entwickelte Suchmaschine überwindet diesen Nachteil mit einem regelbasierten Ansatz, um das gesamte Vokabular abzudecken und dadurch den Recall zu erhöhen. Dafür werden Transformationsregeln entwickelt, die aus einem Suchbegriff die historischen Varianten generieren.

Durch die Orts- und Zeitabhängigkeit der Regeln müssen die Regelsätze jeweils neu generiert werden, wenn ein neues Korpus verfügbar wird. Diese Arbeit muss ohne Hilfe von Informatikern z. B. von Linguisten und Historikern geleistet werden. Deswegen ist es notwendig, ein Werkzeug zu entwickeln, das den Benutzer auf leicht verständliche und schnelle Art bei der Regelgenerierung unterstützt und dessen Benutzung keine Informatikkenntnisse voraussetzt.

Im Folgenden gehen wir davon aus, dass der Benutzer für eine neue Kollektion eine Volltextsuche ermöglichen möchte. Weil für Ort und Zeit der Kollektion kein Regelsatz vorhanden ist, muss der Benutzer zunächst Belege sammeln. Ein Beleg besteht aus der Flexionsform des Lemmas (im Folgenden Wortform genannt) und der zugehörigen historischen Variante. Erst im zweiten Schritt können die Regeln generiert werden.

Um die verschiedenen Benutzerinteressen darzustellen, wurden zwei Szenarios von Benutzertypen entworfen. Die Benutzertypen sollen dabei in erster Linie die Spannweite des notwendigen Bedarfs an Unterstützung bei der Erstellung von Belegen und Regeln darstellen. Z. B. könnte für einen Linguisten bereits die Bildung der Belege ein interessantes Forschungsthema sein. Er möchte 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. Demzufolge wird er einen automatischen Ansatz bevorzugen. Auch wenn er deswegen zunächst einige Dokumente nicht findet, reicht es ihm aus, falls er die Möglichkeit hat, den Regelsatz später zu überarbeiten.

Ausgehend von seinen Bedürfnissen wird sich der Benutzer mehr auf den Recall oder die Precision seiner Suche konzentrieren. Das Tool soll an dieser Stelle die not-

[1] http://books.google.com/ g. a. 27.08.2010
[2] http://www.europeana.eu/portal/ g. a. 27.08.2010

wendige Flexibilität bieten. 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.

Der vorliegende Artikel hat die folgende Struktur: Zunächst wird ein kurzer Überblick über die verwandten Arbeiten im Bereich der Erstellung von Trainingsdaten gegeben. Anschließend wird in Abschnitt 3 die Regelgenerierung erläutert. Abschnitt 4 zeigt, wie der Algorithmus zur Regelgenerierung auch zur automatischen Generierung von Belegen und Regeln verwendet werden kann. Der Ansatz wird in Abschnitt 5 evaluiert. Der letzte Abschnitt fasst den Artikel zusammen und gibt einen Ausblick auf zukünftige Arbeiten.

2 Verwandte Arbeiten

Gotscharek et. al. [Gotscharek *et al.*, 2009] haben mit dem LeXtractor ein Werkzeug zur Konstruktion von historischen Lexika entwickelt. Die Lexikoneinträge können auch als Belege für unseren Ansatz aufgefasst werden. Das Werkzeug hat zwei Ansichten. Die erste bietet dem Benutzer einen Text mit hervorgehobenen unbekannten Termen. In dieser Ansicht kann der Benutzer auf Basis des Textes Lexikoneinträge erzeugen. Die zweite Ansicht 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.

Da das Werkzeug zur Lexikonerstellung verwendet wird, müssen die Ergebnisse eine hohe Präzision aufweisen. Deswegen muss ein Experte alle unbekannten Wörter der ganzen Kollektion bearbeiten und jede Schreibweise beurteilen. Zur Unterstützung wird eine Liste mit Textstellen angeboten, wenn ein Wort für die Konstruktion eines Lexikoneintrags ausgewählt wird. Der LeXtractor verwendet manuell erstellte Regeln (sog. Patterns), um potenzielle moderne Formen in einem aktuellen Lexikon zu finden. Während der Konstruktion von Lexikoneinträgen ist es auch möglich, zusätzliche Patterns vorzuschlagen, wenn eine neue Regel bemerkt wird.

Pilz und Luther [Pilz and Luther, 2009] haben eine Methode entwickelt, um die Sammlung von Belegen in ihrem Evidencer Werkzeug zu unterstützen. Die Methode soll vor allem den Arbeitsaufwand für diesen Schritt verringern. Der Evidencer benutzt dazu einen Bayes'scher Klassifizierer. Dabei nehmen sie an, dass sich die Verteilung der N-gramme signifikant zwischen den Standard- und den Nicht-Standardschreibungen unterscheidet. Zur Einteilung in richtige Schreibweisen und Schreibvarianten schätzt der Klassifizierer die Wahrscheinlichkeit, ob es sich um eine Schreibvariante handelt. Um diese Wahrscheinlichkeit abzuschätzen, 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 für mögliche Varianten verändert.

VARD 2 [Baron and Rayson, 2008] ist ebenfalls in der Lage, moderne Formen für Schreibvarianten in historischen Dokumenten zu finden. Das Werkzeug markiert alle potenziellen Varianten, die nicht in einem modernen Lexikon zu finden sind. Für jedes markierte Wort wird dem Benutzer eine Liste mit potenziellen zugehörigen modernen Schreibungen angeboten. Der Benutzer kann aus der Liste dann die passende moderne Form auswählen. 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. Um diese Vorschläge zu generieren, werden die folgenden drei Methoden benutzt:

- manuelle Liste von Beispielen für moderne Wörter und die zugehörigen Schreibvarianten,
- modifizierte Version des SoundEx-Algorithmus,
- manuell erstellte Liste von Ersetzungsregeln.

Basierend auf diesen drei Methoden wird ein Konfidenzwert für einen Vorschlag berechnet. Der Konfidenzwert ist dabei kein fester Wert, sondern wird nach jedem Schritt automatisch angepasst.

Der erste Ansatz benötigt (aufgrund seines Einsatzgebietes) ein großes Maß an manueller Interaktion sowohl für die Bildung der Belege als auch für die Bildung der Regeln. Der zweite Ansatz sieht mit Blick auf die automatische Unterstützung für den Benutzer vielversprechender aus. Der Benutzer hat hier die Möglichkeit, die Qualität der Ergebnisse durch einen Schwellwert zu beeinflussen. Allerdings benötigt der Klassifizierer eine große Anzahl an Trainingsdaten. Somit ist einiges an manueller Arbeit notwendig bevor der Klassifizierer eingesetzt werden kann. Außerdem kann der Benutzer nur dokumentweise vorgehen. Somit kann er nicht mehrere Vorkommen von möglichen Varianten in verschiedenen Texten gleichzeitig betrachten. Der dritte Ansatz sieht besonders wegen des ständig angepassten Konfidenzwertes für die modernen Formen sehr vielversprechend aus. Der Konfidenzwert ist ansonsten vergleichbar mit dem Bayes'scher Klassifizierer aus dem zweiten Ansatz. 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. Für die deutsche Sprache gibt es mit der Kölner Phonetik [Postel, 1969] ebenfalls einen phonetischen Algorithmus. Allerdings sind phonetische Algorithmen nur bedingt für die Erstellung von Regeln für historische Schreibweisen geeignet, weil sie durch die zeitliche und räumliche Veränderung der Aussprache ebenfalls angepasst werden müssten.

Zusammenfassend lässt sich feststellen, dass keiner der betrachteten Ansätze Belege automatisch generieren kann. Dadurch benötigen alle Werkzeuge einen hohen manuellen Aufwand, bevor sie einsatzbereit sind. Deswegen würde ein Werkzeug, das automatisch Belege für Trainingsmengen erzeugen kann, den Zugang zu historischen Dokumenten für den Benutzer deutlich erleichtern.

3 Generierung von Transformationsregeln

Im Folgenden werden unsere bisherigen Methoden zum Sammeln von Belegen und zur Regelgenerierung [Ernst-Gerlach and Fuhr, 2006] kurz erläutert. Zunächst werden Trainingsdaten benötigt, die moderne Wortformen auf zugehörige historische Varianten abbilden. Mit Hilfe einer Rechtschreibprüfung bekommen wir eine Liste von potenziellen historischen Schreibweisen. Zur Rechtschreibprüfung benutzen wir Hunspell[3], der zur Zeit Wörterbücher für 98 Sprachen zur Verfügung stellt. Die Vorschläge

[3] http://hunspell.sourceforge.net/ g. a. 27.08.2010

für falsch geschriebene Wörter basieren auf N-gramm-Vergleichen, Regeln und Aussprachedaten. Mit diesen Methoden werden dann auf Basis eines Wörterbuchs Vorschläge erstellt.

Durch eine manuelle Überprüfung wird festgestellt, ob es sich bei den unbekannten Wörtern wirklich um Schreibvarianten handelt. Ist dies der Fall, so werden die zugehörigen modernen Formen bestimmt. Zusätzlich benötigen wir noch die Termhäufigkeiten der historischen Formen. Anschließend können wir uns auf den nächsten Schritt konzentrieren — die Regelgenerierung.

Die automatische Regelgenerierung startet mit den erstellten Trainingsdaten. Dabei verwenden wir ein Tripel bestehend aus der modernen Wortform, der zugehörigen Schreibvariante sowie der Kollektionshäufigkeit der Schreibvariante.

Zunächst vergleichen wir jeweils die beiden Wörter und bestimmen sogenannte "Regelkerne". Diese beinhalten die notwendigen Transformationen und identifizieren den zugehörigen Kontext. Z. B. ergibt sich für die moderne Wortform *unnütz* und die historische Form *unnuts* die folgende Menge, die aus zwei Regelkernen besteht: (unn(ü→u)t), (t(z→s)).

In zweiten Schritt werden für jeden Regelkern die zugehörigen Regelkandidaten bestimmt. Diese berücksichtigen auch die Kontextinformationen (z. B. Konsonant (C) oder Wortende ($)) der modernen Schreibweise. Für das oben gezeigte Beispiel werden unter anderem die folgenden Regelkandidaten generiert: ü→ u, nü→ nu, üt→ ut, nüt→ nut, Cü→ Cu, z$→s$.

Im letzten Schritt werden die nützlichen Regeln durch Pruning der Regelmenge (wobei auch die Kollektionshäufigkeit berücksichtigt wird) durch eine modifizierte Version des PRISM Algorithmus (siehe [Cendrowska, 1987]) bestimmt. Dafür werden zunächst automatisch negative Belege erstellt und Precisionwerte für die Regeln berechnet. Anschließend werden die Regeln ausgewählt, die eine festgelegte Mindestprecision sowie eine Mindestvorkommenshäufigkeit aufweisen.

4 Automatisch akzeptierte Belege

Der letzte Abschnitt hat verdeutlicht, dass der bisherige Ansatz am Anfang einen hohen manuellen Aufwand benötigt. Deswegen ist es unser Hauptziel einen Algorithmus zu entwickeln, der Belege automatisch generieren kann, um diesen Aufwand zu reduzieren.

Die Annahme, dass Schreibvarianten ein bestimmtes Maß an Regularität beinhalten, bildet die Basis für den regelbasierten Ansatz. Basierend auf dieser Annahme sollen im Folgenden auch die Belege automatisch generiert werden. Die richtige moderne Form einer Schreibvariante befindet sich häufig unter den Vorschlägen des Spellcheckers. Wir nehmen an, dass diese Regularitäten zwischen moderner Form und Schreibvariante deutlich seltener auch zwischen Schreibvarianten und falschen Vorschlägen zu finden sind. Deswegen konzentriert sich unser Algorithmus (siehe Abbildung 1) auf das Problem, den richtigen Vorschlag des Spellcheckers 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 ein Beleg generiert (siehe Tabelle 1). Diese Belege bilden die Trainingsmenge aus der mögliche Regelkandidaten generiert werden. Da wir in diesem Schritt noch nicht die endgültigen Regeln generieren, sind hier die unterschiedlichen Regelkandidaten nicht relevant und es werden nur die Regelkerne betrachtet. Auf diese Weise erhalten wir eine eindeutigere Verteilung der Regeln.

Je häufiger ein Beleg in unterschiedlichen Belegen auftaucht, desto höher 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 (siehe Tabelle 2). Haben mehrere Regelkandidaten die gleiche Häufigkeit werden zunächst Substitutionsregeln akzeptiert, da diese meistens eine höhere Precision als Einfüge- und Löschregeln haben. Z. B. wird $i \rightarrow y$ gegenüber $s \rightarrow \emptyset$, $\emptyset \rightarrow h$ bevorzugt.

Regelkern	**Regelhäufigkeit**	**Entscheidung**
$i \rightarrow y$	7	akzeptieren
$un \rightarrow \emptyset$	1	
$un \rightarrow ge$	1	
$\emptyset \rightarrow ge$	1	
$w \rightarrow \emptyset$	1	
$ster \rightarrow ck$	1	
$ar \rightarrow zey$	1	
$l \rightarrow z$	1	
$i \rightarrow yt$	1	
$mann \rightarrow zeyt$	1	
$\emptyset \rightarrow je$	1	
$ig \rightarrow \emptyset$	1	

Tabelle 2: Beispiel für sortierte Regelkerne

Vorschlag	**Mögliche Variante**	**Entscheidung**
Geschicklichkei**t**	Geschicklichkey**t**	akzeptieren
jederzei**t**	jederzey**t**	akzeptieren
obgle**i**ch	obgley**c**h	akzeptieren
Sonderhe**i**t	**In**sonderhey**t**	markiere $i \rightarrow y$ als akzeptiert

Tabelle 3: Beispiel zur Akzeptanz des Regelkerns $i \rightarrow y$

Nachdem wir einen Regelkandidaten akzeptiert haben, werden die zugehörigen Belege (und damit die Vorschläge der Rechtschreibprüfung) betrachtet. Basiert ein Beleg nur auf der akzeptierten Regel, wird er direkt akzeptiert. Basiert er auf mehreren Regeln, wird er akzeptiert, wenn alle anderen Regeln ebenfalls akzeptiert sind. Ansonsten wird lediglich markiert, dass der Regelkandidat akzeptiert ist (siehe Tabelle 3).

Nachdem wir nun akzeptierte Vorschläge haben, können wir im nächsten Schritt falsche Vorschläge aussortieren. Dafür nehmen wir an, dass zu jeder Schreibvariante nur eine moderne Schreibung existiert. Dadurch können wir die weiteren Vorschläge für historische Schreibweisen entfernen. Diese Annahme stellt eine Vereinfachung dar. Wie Pilz [Pilz, 2009] gezeigt hat, verfügt die Schreibvariante $Hunngern$ über die modernen Formen $Ungarn$ und $Hungern$. Die Vereinfachung ist an dieser Stelle notwendig, um die automatische Beleggenerierung überhaupt zu ermöglichen und somit den manuellen Aufwand für die Konstruktion der Trainingsdaten deutlich zu reduzieren. Diese Einschränkung gibt es bei der späteren ma-

```
Bilde Trainingsmenge aus 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
    Markiere 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 1: Algorithmus zur automatischen Beleggenerierung

Vorschlag	**Mögliche Varianten**	**Regelkandidaten**
Geschicklichkeit	Geschicklichkeyt	$i \rightarrow y$
Ungeschicklichkeit	Geschicklichkeyt	$un \rightarrow \emptyset, i \rightarrow y$
Unschicklichkeit	**Ge**schicklichkeyt	$un \rightarrow ge, i \rightarrow y$
Schicklichkeit	**Ge**schicklichkeyt	$\emptyset \rightarrow ge, i \rightarrow y$
Geschwisterlichkeit	Geschicklichkeyt	$w \rightarrow \emptyset, ster \rightarrow ck, i \rightarrow y$
jederzeit	jederzeyt	$i \rightarrow y$
jederart	jederzeyt	$ar \rightarrow zey$
jederlei	jederzeyt	$l \rightarrow z, i \rightarrow yt$
jedermann	jederzeyt	$mann \rightarrow zeyt$
derzeitig	jederzeyt	$\emptyset \rightarrow je, i \rightarrow y, ig \rightarrow \emptyset$

Tabelle 1: Beispiel für Trainingsdaten und generierte Regeln

nuellen Bearbeitung von Belegen nicht. Außerdem gehen wir davon aus, dass die Regeln, die durch diese Annahme verloren gehen, durch andere Belege mitgeneriert werden. Während des Löschungsprozesses werden die falschen Belege auch von weiteren zugehörigen Regelkernen entfernt. Anschließend startet der Prozess wieder mit dem häufigsten unbehandelten Regelkandidaten.

Der Benutzer kann diesen Prozess 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. Deswegen kann die Precision für die automatischen Belege durch eine minimale Wortlänge erhöht werden.
- Minimale Anzahl an Regelvorkommen: Der Kern unseres Ansatzes besteht darin, dass die Regelhäufigkeit ein Indikator für die Precision ist. Deswegen 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. Z. B. kann durch drei Regelanwendungen auf *derzeitig* das Wort *jederzeyt* generiert werden (siehe Tabelle 1).

Aus dieser Parameterauswahl bevorzugt der 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, sofern er möchte, direkt mit der Suche auf der Kollektion beginnen. Im Gegensatz dazu wird der Linguist eventuell genau die umgekehrte Parameterauswahl treffen.

Der vorgestellte Ansatz wurde bereits in den RuleGenerator integriert. Der RuleGenerator ist ein interaktives Werkzeug und bietet eine graphische Benutzeroberfläche mit der der Benutzer Belege sammeln (siehe [Awakian, 2010]) und Regeln generieren kann (siehe [Korbar, 2010]). Die Ergebnisse des Prozesses zur automatischen Regelgenerierung werden dem Benutzer in einer graphischen Benutzeroberfläche in Form einer Liste angezeigt (siehe Abbildung 1). Darin werden zur Zeit Tripel aus moderner Wortform, Schreibvariante und den zugehörigen Regeln dargestellt. Der Benutzer kann anschließend einzelne Belege oder alle Belege komplett akzeptieren. Zusätzlich bietet die Liste Zugang zu den Textstellen in denen die Varianten vorkommen. Dadurch kann der Benutzer die Wörter auch in dem Kontext betrachten, wenn der Bedarf besteht. Im Anschluss an die Bearbeitung der automatischen Belege kann der Benutzer aus den noch nicht zugeordneten unbekannten Wörtern weitere Belege bilden.

5 Evaluierung

Als Testkollektion wurde unsere Belegedatenbank gewählt. Sie wurde basierend auf Texten der Nietzsche Rezeption[4] sowie weiteren kleineren Kollektionen[5] aufgebaut. Die

[4] `http://www2.inf.uni-due.de/Studienprojekte/Nietzsche/pp2001/die_cd/die_cd.htm` g. a. 27.08.2010

[5] Digitales Archiv Hessen-Darmstadt `http://www.digada.de/index.html` g. a. 27.08.2010, Bibliotheca Augustana. FH Augsburg. `http://www.hs-augsburg.de/~harsch/augustana.html`, g. a. 27.08.2010, documentArchiv.de `http://www.documentarchiv.de` g.a. 27.08.2010

	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 Regel	0,55	0,47	0,50	0,47	0,50	0,50	0,51	0,51	0,50	0,49
Recall Regelvorkommen	0,91	0,90	0,92	0,92	0,94	0,95	0,95	0,95	0,96	0,96

Tabelle 4: Recall und Precision für Regelkerne auf Basis 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 5: Recall für Regelkerne auf Basis Gesamttestdaten

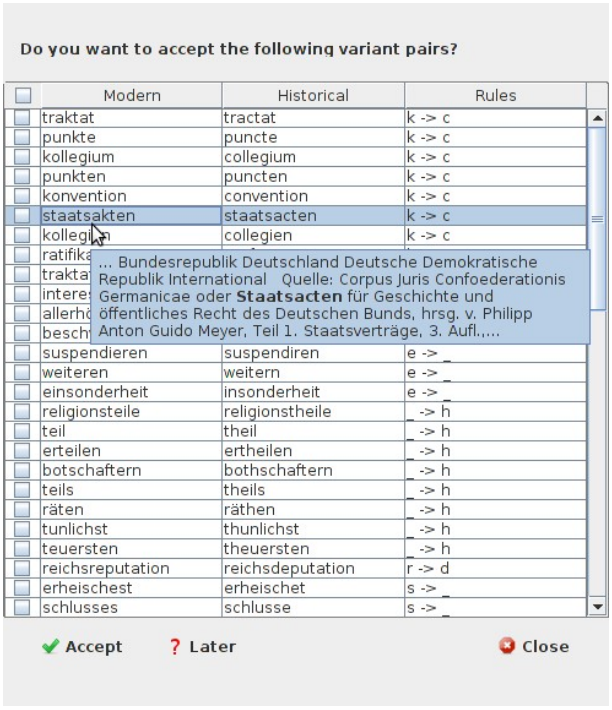

Abbildung 2: Benutzeroberfläche für automatische Belege

Texte stammen überwiegend aus dem 16. bis 19. Jahrhundert. An diesem Beispiel soll nachvollzogen werden, wie viel manueller Aufwand bei der Erstellung einer Trainingskollektion gespart werden kann. Um den sukzessiven Aufbau der Kollektion nachzustellen, wurde die Anzahl der als Testdaten verwendeten Belege von 1000 schrittweise um jeweils 1000 bis auf 10000 Belege erhöht. Für jede Testmenge wurde unser Verfahren angewendet. Als Parametereinstellung wurde mit einer Mindestwortlänge von fünf, mindestens zwei Regelvorkommen und maximal zwei Regelanwendungen pro Wort eine recallorientierte Auswahl getroffen. Dies wird insbesondere durch den geringen Wert für die minimale Vorkommenshäufigkeit der Regeln deutlich.

Es wurden zunächst Recall und Precisionwerte für die Regelkerne berechnet (siehe Tabelle 4). Die Precision sinkt von 0,28 Punkten bei 1000 Belegen bis 0,17 bei 10000 Belegen. Dies lässt sich durch die deutliche Parameterwahl zu Gunsten des Recall erklären. Da eine Regel nur zweimal vorkommen musste, um akzeptiert zu werden, steigt mit steigender 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.

Da sich die einzelnen Regeln sehr stark in ihrer Anwendungshäufigkeit unterscheiden, wurde zusätzlich auch noch der Recall basierend auf der Vorkommenshäufigkeit der einzelnen Regeln berechnet. Hier zeigt sich deutlich, dass vor allem die besonders häufigen Regeln generiert werden, da immer mindestens 90 % der Regelkerne gefunden werden.

Um die Entwicklung der Regelabdeckung einschätzen zu können, wurde noch die Entwicklung der Recallwerte der Regelkerne bezogen auf die Gesamtmenge der Belege berechnet (siehe Tabelle 5). Als Vergleichbasis für unser Verfahren diente dabei der Recall der Regelkerne bei manueller Erstellung der Testkollektion. Der Benutzer muss 2000 Belege manuell betrachten, um denselben Recall zu erreichen wie mit dem automatischen Verfahren. Betrachtet man den manuellen Aufbau der Testkollektion bezogen auf die Regelhäufigkeit zeigt sich, dass der Benutzer sogar über 9000 Belege manuell erzeugen muss, um den Recall so zu erhöhen, wie er es mit den automatisch erzeugten Be-

	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 6: Recall Belegabdeckung

legen könnte.

Um festzustellen, wie viele Belege man mit den generierten Regelkernen wiederfinden könnte, wurde auch noch der Recall für die Belegabdeckung berechnet (siehe Tabelle 6). Dies geschah sowohl für die automatische als auch für die manuelle Vorgehensweise. Dabei zeigt sich, dass in beiden Fällen bereits mit den aus 1000 Belegen generierten Regeln fast 90% der Belege abgedeckt werden. Dadurch wird nochmals die Regelhaftigkeit von Schreibvarianten gezeigt. Der Benutzer muss in etwa 2000 Belege manuell bewerten, um denselben Recall wie beim automatischen Ansatz zu erzielen.

6 Fazit

In diesem Artikel wurde ein Ansatz zur automatischen Konstruktion von Belegen vorgestellt. Die Belege werden als Eingabe für den Prozess der Regelgenerierung benötigt, der Retrieval in Texten mit nicht-standardisierter Rechtschreibung ermöglicht. Der vorgestellte Ansatz bietet dem Benutzer die Möglichkeit mehrere Parameter einzustellen. Dadurch ist der Ansatz sehr flexibel, weil der Benutzer den Prozess der Beleggenerierung entsprechend seiner Erwartungen an Recall und Precision beeinflussen kann.

Die Evaluierung hat deutlich gezeigt, dass die häufigen Regelkerne ausgewählt werden. Wenn der Benutzer diese direkt akzeptiert, muss er sich nur noch die Wörter anschauen, bei denen es nicht möglich war, einen Vorschlag automatisch zuzuordnen. Im weiteren Verlauf des Projektes soll daran gearbeitet werden, die Liste der verblieben unbekannten Wörter nach sinkender Irregularität zu sortieren.

Da die automatische Belege bereits nach sinkender Häufigkeit sortiert sind, wird diese Liste demnächst anhand der Regeln sortiert. Dadurch bekommt der Benutzer schneller einen Überblick über mögliche Regeln. Außerdem lässt sich so neben veränderten Parametereinstellungen auch die relativ geringe Precison steigern, weil der Benutzer die automatischen Belege schneller bearbeiten kann.

Literatur

[Awakian, 2010] A. Awakian. Development of a user-interface for an interactive rule development. Master's thesis, University of Duisburg-Essen, 2010.

[Baron and Rayson, 2008] A. Baron and P. Rayson. 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.

[Cendrowska, 1987] J. Cendrowska. PRISM: An algorithm for inducing modular rules. *International Journal on Man-Machine Studies*, 27(4):349–370, 1987.

[Ernst-Gerlach and Fuhr, 2006] Andrea Ernst-Gerlach and Norbert Fuhr. Generating search term variants for text collections with historic spellings. volume 3936 of *Lecture Notes in Computer Science*, Heidelberg, 2006. Springer Verlag. ISBN 3540333479.

[Ernst-Gerlach and Fuhr, 2007] Andrea Ernst-Gerlach and Norbert Fuhr. Retrieval in text collections with historic spelling using linguistic and spelling variants. In *JCDL '07: Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, pages 333–341, New York, NY, USA, 2007. ACM.

[Gotscharek *et al.*, 2009] A. Gotscharek, A. Neumann, U. Reffle, Ch. Ringlstetter, and K. U. Schulz. Enabling information retrieval on historical document collections: the role of matching procedures and special lexica. In *AND '09: Proceedings of The Third Workshop on Analytics for Noisy Unstructured Text Data*, pages 69–76, New York, NY, USA, 2009. ACM.

[Hauser *et al.*, 2007] Andreas Hauser, Markus Heller, Elisabeth Leiss, Klaus U. Schulz, and Christiane Wanzeck. 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*, 2007. Hyderabad, India, January.

[Keller, 1986] R. Keller. *Die Deutsche Sprache und ihre historische Entwicklung.* Helmut Buske Verlage, Hamburg, Germany, 1986. ISBN 3875481046.

[Korbar, 2010] D. Korbar. Visualisation of rule structures and rule modification possibilities for texts with non-standard spelling. Master's thesis, University of Duisburg-Essen, 2010.

[Pilz and Luther, 2009] T. Pilz and W. Luther. Automated support for evidence retrieval in documents with non-standard orthography. In Susanne Winkler Sam Featherston, editor, *The Fruits of Empirical Linguistics Process*, volume 1, pages 211–228, 2009.

[Pilz, 2009] T. Pilz. *Nichtstandardisierte Rechtschreibung - Variationsmodellierung und rechnergestützte Variationsverarbeitung.* PhD thesis, University of Duisburg-Essen, 2009.

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