Fehlerklassen von Porters Deutsch-Stemmer und Lösungsvorschläge für die Behandlung von Komposita

Esther Neumann & Martin Natano

Universität Wien,
Institut für Germanistik,
Universitätsring 1, A-1010 Wien

Abstract Dieses Paper besteht aus zwei thematisch zusammenhängenden Teilen: Zum Ersten wurde eine eigenständige Analyse des Porter-Stemmers für Deutsch (2009) auf sein Fehlverhalten vorgenommen. Daraus ergaben sich eine Handvoll Fehlertypen, die im Genaueren beschrieben wurden. Zum Zweiten werden für einen der Typen, bei dem es sich um das falsche Stemming von Komposita handelt – also der Kombination von (freien) Wortstämmen oder (freien) lexikalischen Morphemen – verschiedene Lösungsansätze vorgestellt, die sich aus Überlegungen der aktuellen Forschungsliteratur zusammensetzen. Da im Bereich der Informationsrückgewinnung (*Information Retrieval*) – wo das Stemming hinzuzuzählen ist – linguistische Werkzeuge nicht zwingend in Verwendung sind, basieren die beiden Untersuchungen auf einer Methode, die linguistische Bezeichnungen zwar verwendet, sie aber nicht sprachwissenschaftlich instrumentalisiert.

1 Einleitung

Ein Stemmer hat die Aufgabe, Wörter von Einzelsprachen auf ihren Stamm zu reduzieren, um damit relevante Informationen in einer Sammlung von Texten finden zu können, z.B. in Abfragesystemen (*Query Systems*) für das Web. Je nach Komplexität der morphosyntaktischen Regeln der zu stemmenden Sprache sind die Anforderungen an den für den Stemmer geschriebenen Algorithmus unterschiedlich. Wir haben nach einem Stemmer für Deutsch gesucht, den wir nach seinem Fehlervorkommen analysieren können und sind bei der Google-basierten Suche auf die Arbeit von Martin Porter gestoßen. Er hat 1980 einen Suffix-Stemmer für Englisch entwickelt und bis zur Gegenwart noch rund ein Dutzend weitere für andere natürliche Sprachen. [1] Auf seiner Webseite `http://snowball.tartarus.org/` finden sich die Links zu den einzelnen Stemmern

[1] Das Porter-Verfahren wird in vielen Software-Produkten eingesetzt: z.B.: PostgreSQL (`http://www.postgresql.org/docs/9.1/static/textsearch-dictionaries.html`), NLTK (`http://nltk.org/api/nltk.stem.html`), Xapian (`http://xapian.org/docs/stemming.html`) und Solr (`http://wiki.apache.org/solr/LanguageAnalysis`).

mit der Beschreibung des Algorithmus und dem Algorithmus selbst. Mittels eines selbstgeschriebenen Programms, das die vorhandenen Wörter eines ausgewählten Textkorpus extrahiert und stemmt, haben wir den Stemmer für Deutsch getestet und die Fehlerquellen notiert. Diese wurden schließlich klassifiziert und genauer beschrieben. Wo bereits Forschungsliteratur vorhanden ist, haben wir eine Literaturangabe gemacht. Für den Porter-Stemmer für Deutsch ergaben sich Probleme im Bereich der deutschen Varietäten (Dialekte, Ideolekte, Soziolekte), der Fremdwörter, der Umlaute, der Abkürzungen und der Eigennamen. [2]
Mit dem falschen Stemming von Komposita, bei dem der Stemmer das zusammengesetzte Wort behandelt wie ein einzelnes lexikalisches Morphem, setzten wir uns genauer auseinander, indem wir hier nicht nur das Fehlverhalten im Detail beschreiben, sondern auch die aktuelle Forschungsliteratur nach Lösungsvorschlägen untersuchten.
Porters Stemming-Algorithmus für die deutsche Sprache basiert auf den Prinzipien des Algorithmus für die englische Sprache. Dieser Algorithmus besteht aus fünf Schritten mit eigenen Regelwerken zur Entfernung von Wortendungen, wobei die Schritte nacheinander auf das Resultat des jeweils vorherigen Schrittes angewandt werden. [Porter, 1980] Das Ergebnis des letzten Durchlaufs ist nach Normalisierung der Umlaute der Stem und führt durch die Anwendung an einem spezifischen Korpus zu einer Reduktion des Vokabulars um ein Drittel, so Porter. [Porter, 1980] Der Porter-Algorithmus für die deutsche Sprache beruht auf der Anwendung von drei Schritten, die jeweils aus mehreren Regeln bestehen und ebenfalls nacheinander am Wort angewendet werden. [Porter, 2009]
Im Anhang befindet sich eine detailierte Beschreibung dieses Algorithmus, der sinngemäß vom Englischen ins Deutsche übersetzt wurde.

2 Forschungsstand

Der Teil des vorliegenden Papers, der sich mit den Fehlerklassen des Porter-Stemmers auseinandersetzt, hat hauptsächlich Arbeiten von Martin Porter zur Grundlage. (Porter [1980, 2009] und seine Website) Aktuelle Lösungsansätze zum Decompounding finden sich bei Monz [2001], Braschler [2003], Tomlinson [2004], Savoy [2004].

3 Methode

Für die vorliegende Untersuchung wurde der Artikel *Der Trend zur Non-Fiction-Bewegung. Literatur als Suchbewegung* [3] aus der österreichischen Tageszeitung

[2] Dies soll keine vollständige Aufzählung der Fehlerquellen von Porters Stemmer darstellen. Es handelt sich dabei lediglich um die auffälligsten Klassen. Weniger auffällige sind das teilweise falsche Stemmen von Zeitformen, Probleme mit *ie* (z.B.: *blieb* müsste zu *bleib* werden, passiert aber nicht) und ungeregeltes Stemmen von Zirkumfixen.

[3] `http://www.wienerzeitung.at/themen_channel/literatur/buecher_aktuell/554464_Literatur-als-Suchbewegung.html`

Wiener Zeitung Online als zu stemmendes Korpus benutzt. Bei der Auswahl des Korpus waren Länge und Eloquenz ausschlaggebend. Dieses Korpus ist aber nur eine Ausgangsbasis und wird kombiniert mit dem Wissen über die gegenwärtige Sprache, die von einem/einer muttersprachlich Deutschsprechenden vorausgesetzt werden kann. Damit ist gemeint, dass der Text teilweise nur wenige Beispiele für die einzelnen Fehlerklassen hergibt, diese aber exemplarisch für eine stärkere Verwendung in anderen Texten stehen. So ist Umgangssprache im Korpus mit nur einem Beispiel vertreten, wird aber dennoch in diesem Paper behandelt, weil sie in modernen Kommunikationsmedien wie Chat, SMS, Blog, usw. üblich ist. Genauso finden sich für die Klasse *Abkürzungen* nur zwei Beispiele, betrachtet man aber das Medium SMS, so ist dort die Abkürzung omnipräsent. Um den Porter-Stemmer für Deutsch [Porter, 2009] benutzen zu können, wurde ein eigenes Programm geschrieben, das die vorhandenen Wörter des ausgewählten Textes extrahiert und stemmt. Die Library *NLTK* bietet dafür eine Implementierung an. Das Resultat wurde einer genauen Analyse unterzogen, wobei die Wörter, die Probleme bereiteten, aufgrund ihrer Eigentümlichkeiten in Klassen zusammengefasst und diese im Einzelnen beschrieben wurden. Als zweite Aufgabe stellte sich die Beschreibung eines Lösungsvorschlags für den Fehlertypus *Komposita*, bei der die vorhandene Forschungsliteratur zusammengefasst wird.
Schließlich muss noch angemerkt werden, dass ein Stemming-Algorithmus bisher prinzipiell nur eine Annäherung an eine Stammformreduktion ist. In diesem Sinne sind *Overstemming* (das fälschliche Entfernen eines Wortteils, der wie eine Endung aussieht) und *Understemming* (das Nicht-Entfernen eines Suffixes) bei allen Stemmern unvermeidbar und werden mittels eines gut geschriebenen Algorithmus allenfalls kleingehalten.

4 Fehlertypen

Die vorliegende Analyse entspricht der Intention Porters, nicht nach linguistischen Prinzipien zu analysieren, sondern den Stem nach seiner Relevanz für die Suchleistung in einem Information-Retrieval-System zu betrachten. [4] Wir haben daher die Stems des untersuchten Textes hinsichtlich ihrer praktischen und auch theoretischen Tauglichkeit bewertet. (Theorie und Praxis können, müssen aber nicht ident sein.) Außerdem ergaben sich bei der Untersuchung fünf Klassen, die das Wort nach seiner möglichen Fehleranfälligkeit beim Stemmen kennzeichnen. Ein gestemmtes Wort erhält also eine Markierung, wenn es sich dabei um einen Eigennamen, ein/en Fremd-/Dialektwort/umgangssprachlichen Ausdruck, eine Abkürzung, ein Kompositum handelt oder wenn darin ein Umlaut vorkommt. Entsprechen einem Wort mehrere Klassen, werden selbstverständlich alle zutreffenden eingetragen.
Der untersuchte Text besteht aus 715 Wörtern, die 419 Stems ergeben. Das mehrfache Vorkommen eines Wortes wird beim Stemming ignoriert. Von den 419 Stems sind 117 (theoretisch) falsch.

[4] "[T]he suffixes are being removed simply to improve IR perfomance, and not as a linguistic exercise." [Porter, 1980]

Legende:

N Eigenname
K Kompositum
U Umlaut
S Dialekt/Umgangssprache/Fremdwort
A Abkürzung

Wort	Stem	#	ok/falsch (Theorie)	ok/falsch (Praxis)	N	K	U	S	A
abs	abs	1	falsch	falsch					✓
anderen	and	1	falsch	ok					
anderes	and	1	falsch	ok					
anders	and	1	falsch	ok					
anhaftet	anhaftet	1	falsch	falsch					
ausgeklügeltem	ausgeklugelt	1	falsch	ok			✓		
becker	beck	1	falsch	falsch	✓				
beträchtlichen	betracht	1	falsch	falsch			✓		
bewegt	bewegt	1	falsch	falsch					
brigge	brigg	1	falsch	falsch	✓				
dafür	dafur	1	falsch	falsch			✓		
deren	der	1	falsch	falsch					
dezidiert	dezidiert	1	falsch	falsch					
dichtung	dichtung	1	falsch	falsch					
döblin	doblin	1	falsch	falsch	✓		✓		
einiges	einig	1	falsch	falsch					
einzuprügeln	einzuprugeln	1	falsch	ok			✓		
endgültig	endgult	1	falsch	ok			✓		
entwickelt	entwickelt	1	falsch	falsch					
epische	episch	1	falsch	ok					
erbaut	erbaut	1	falsch	falsch					
ergeht	ergeht	1	falsch	falsch					
erklären	erklar	1	falsch	ok			✓		
erklärten	erklart	1	falsch	falsch			✓		
erscheint	erscheint	1	falsch	falsch					
erschien	erschi	1	falsch	ok					
erschienen	erschi	1	falsch	ok					
erwartet	erwartet	1	falsch	falsch					
erzählen	erzahl	4	falsch	ok			✓		
erzählens	erzahl	2	falsch	ok			✓		
erzählte	erzahlt	2	falsch	falsch			✓		
erzählungen	erzahl	1	falsch	ok			✓		
etabliert	etabliert	1	falsch	falsch					
experimentellen	experimentell	1	falsch	falsch					
fiction	fiction	1	falsch	falsch				✓	
fiktion	fiktion	1	falsch	falsch					

Wort	Stem	#	ok/falsch (Theorie)	ok/falsch (Praxis)	N	K	U	S	A
fiktionale	fiktional	1	falsch	falsch					
fiktionales	fiktional	1	falsch	falsch					
fortwurstelt	fortwurstelt	1	falsch	falsch		✓			
führten	fuhrt	1	falsch	falsch			✓		
für	fur	4	falsch	ok			✓		
gehören	gehor	1	falsch	falsch			✓		
gehörige	gehor	1	falsch	falsch			✓		
gehört	gehort	3	falsch	falsch			✓		
geläutet	gelautet	1	falsch	falsch			✓		
gleichgültig	gleichgult	1	falsch	ok			✓		
grundregeln	grundregeln	1	falsch	falsch					
gründlichen	grundlich	1	falsch	ok			✓		
hatte	hatt	1	falsch	falsch					
hatten	hatt	2	falsch	falsch					
heißenbüttel	heissenbuttel	1	falsch	ok	✓		✓		
häufig	haufig	1	falsch	falsch			✓		
hören	hor	1	falsch	falsch			✓		
immer	imm	1	falsch	falsch					
immerhin	immerhin	1	falsch	falsch					
insofern	insof	1	falsch	falsch					
jonkes	jonk	1	falsch	falsch	✓				
jürgen	jurg	1	falsch	falsch	✓		✓		
konzentrationslager	konzentrationslag	1	falsch	ok		✓			
krimigenre	krimigenr	1	falsch	ok		✓			
krise	kris	3	falsch	ok					
kräftig	kraftig	1	falsch	falsch			✓		
kölner	koln	1	falsch	ok			✓		
königsdisziplin	konigsdisziplin	1	falsch	ok		✓	✓		
lehrt	lehrt	1	falsch	falsch					
leserbedürfnisse	leserbedurfnis	1	falsch	ok		✓	✓		
lexikon	lexikon	1	falsch	falsch					
liefern	lief	1	falsch	falsch					
literarische	literar	1	falsch	falsch					
literarischen	literar	2	falsch	falsch					
literatur	literatur	5	falsch	falsch					
lyrischen	lyrisch	1	falsch	falsch					
lässt	lasst	2	falsch	falsch			✓		
malte	malt	1	falsch	falsch					
manifest	manif	2	falsch	ok				✓	
möglichkeitswelten	moglichkeitswelt	1	falsch	ok		✓	✓		
mühe	muh	1	falsch	falsch			✓		
neben	neb	1	falsch	falsch					
okopenkos	okopenkos	1	falsch	falsch	✓				

Wort	Stem	#	ok/falsch (Theorie)	ok/falsch (Praxis)	N	K	U	S	A
paradigmatisch	paradigmat	1	falsch	falsch					
paradoxerweise	paradoxerweis	1	falsch	ok					
proklamierte	proklamiert	1	falsch	falsch					
rainer	rain	1	falsch	falsch	✓				
reality	reality	1	falsch	falsch				✓	
rilkes	rilk	2	falsch	falsch	✓				
scheint	scheint	1	falsch	falsch					
schreibt	schreibt	1	falsch	falsch					
seattle	seattl	1	falsch	ok	✓				
shields	shield	1	falsch	falsch	✓				
sorgte	sorgt	1	falsch	falsch					
später	spat	1	falsch	falsch			✓		
stammt	stammt	1	falsch	falsch					
stellte	stellt	1	falsch	falsch					
suchten	sucht	1	falsch	falsch					
szenerie	szeneri	1	falsch	ok					
tageszeitung	tageszeit	1	falsch	falsch		✓			
thematisierte	thematisiert	1	falsch	falsch					
totenglöcklein	totenglocklein	1	falsch	ok		✓	✓		
trägt	tragt	1	falsch	falsch			✓		
verarbeitete	verarbeitet	1	falsch	falsch					
vinylschallplatte	vinylschallplatt	1	falsch	ok		✓			
wahlweise	wahlweis	1	falsch	ok					
watte	watt	1	falsch	falsch					
weise	weis	1	falsch	ok					
weiterschwelte	weiterschwelt	1	falsch	falsch		✓			
widmeten	widmet	1	falsch	falsch					
wieder	wied	2	falsch	ok					
wirklichkeitsaffirmation	wirklichkeitsaffirmation	1	falsch	falsch		✓			
wirklichkeitsliteratur	wirklichkeitsliteratur	1	falsch	falsch		✓			
zeitgemäß	zeitgemass	1	falsch	ok		✓	✓		
zeugt	zeugt	1	falsch	falsch					
zumindest	zumind	1	falsch	ok					
österreich	osterreich	1	falsch	ok	✓		✓		
überholt	uberholt	1	falsch	falsch		✓	✓		
übernimmt	ubernimmt	1	falsch	falsch		✓	✓		
übernommen	ubernomm	1	falsch	falsch		✓	✓		
üppig	uppig	1	falsch	ok			✓		

4.1 Umlaute

Manches Stemming eines Wortes mit einem Umlaut würde eine Entfernung des Umlauts (also der Umlaut-Striche) verlangen, manches nicht. Der Porter-Stemmer entfernt jedoch alle Umlaute. Es ließe sich argumentieren, dass das für

die Suchleistung nicht relevant sei, im gegebenen Fall dass es in der deutschen Sprache keine Wörter gibt, die abzüglich des Umlautes den gleichen Stamm haben. Dem ist aber nicht so. Die Suche nach *Spaten* liefert beispielsweise auch Dokumente mit dem Wort *später* zurück, was offensichtlich falsch ist, denn beide Wörter haben nach dem Snowball-Verfahren den Stamm *spat*. Verzichten lässt sich auf die Entfernung von Umlauten aber auch nicht, da sonst die Zusammenführung verwandter Wörter effektiv verhindert werden würde (z.B.: *natürlich* und *Natur*).

Beispiele

Erzählungen → *erzahl*
häufig → *haufig*
Mühe → *muh*
später → *spat*

4.2 Dialekt/Umgangssprache

Prinzipiell sind die Porter-Stemmer für die Schriftsprache der jeweiligen Zielsprache konzipiert und nicht für die geschriebene Mundart oder Dialektformen. Es ergibt sich dadurch allerdings eine Schwachstelle in der Suchleistung, da heute ein immer größerer Anstieg einer Vermischung von Schrift- und mündlicher Sprache im geschriebenen Deutsch verzeichnet werden kann. [5] So wäre zum Beispiel die Suche in den Chatprotokollen eines Dialektschreibenden mithilfe dieses Verfahrens wenig zielführend. Der große Vorteil der Schriftsprache ist ihre relative Normiertheit. Auch der (traditionelle) Dialekt bzw. die verschiedenen Dialektformen der deutschen Sprache würden genügend grammatische und orthographische Regeln aufweisen, um dafür individuelle Stemming-Regeln aufbauen zu können. [6] Viel schwieriger würde es sich aber gestalten, einen Stemming-Algorithmus für die Umgangssprache (≈ Zwischenstellung zwischen Dialekt und Standardsprache) zu finden, da diese einem permanenten Wandel und unzähligen Varietäten unterworfen ist. Dies zeigt sich in SMS, E-Mail, Speech-to-Text-Systemen, moderner Literatur und eigentlich jeglicher Form von geschriebenem Sozio-, Medio- und Idiolekt.

[5] Vgl. neue Medien wie SMS, Chat, Blog, E-Mail, usw.

[6] z.B.: Wiesinger, Peter und Raffin, Elisabeth. 1982. Bibliographie zur Grammatik der deutschen Dialekte. Laut-, Formen-, Wortbildungs- und Satzlehre 1800-1980. Europäische Hochschulschriften. Reihe 1, Dt. Sprache u. Literatur, Bd. 509.
Žirmunskij, Viktor M. 1962. Deutsche Mundartkunde. Vergleichende Laut- und Formenlehre der deutschen Mundarten. Veröffentlichungen des Instituts für Deutsche Sprache und Literatur.
Hausenblas, Adolf. 1914. Grammatik der nordwestböhmischen Mundart. Laut- und Formenlehre mit Textproben. Verl. des Vereines für Geschichte der Deutschen in Böhmen. Beiträge zur Kenntnis deutsch-böhmischer Mundarten.
Jakob, Julius. 1980. Wörterbuch des Wiener Dialektes. Mit einer kurzgefaßten Grammatik. Nachdr. d. Ausg. von 1929. Die bibliophilen Taschenbücher.

Beispiele

fortwurstelt → *fortwurstelt*

4.3 Abkürzungen

Ebenso wie beim Fehlertypus *Dialekt/Umgangssprache* handelt es sich bei Abkürzungen um Wörter, die im Porter-Algorithmus nicht eigens behandelt werden und daher einen Stem liefern, der möglicherweise nichts mit dem abgekürzten Wort zu tun hat, z.B.: *abs.* für *absolut*. Abkürzungen kommen in den meisten schriftsprachlich geschriebenen deutschen Texten regelmäßig aber nicht besonders häufig vor. (Ausnahmen bilden moderne Medien wie SMS, Blog, E-Mail, usw.)[7]

Beispiele

abs. → *abs*
USA → *usa*

4.4 Sprachidentifizierung

Immer häufiger trifft man eine Vermischung verschiedener Einzelsprachen innerhalb eines Textes an [8] und auch der Einbezug von Umgangssprache in Texten ist – wie bereits schon erwähnt – ein gegenwärtiges Phänomen. Die Stemmer, die Porter auf seiner Website anbietet – z.B. romanische, germanische und skandinavische Sprachen – sind jedoch alle auf eine Einzelsprache spezialisiert und daher immer nur auf einen einzelsprachlichen Textkorpus sinnvoll anwendbar. Bei einem gemischtsprachigen Text wäre aber eine automatische Erkennung und in Folge die Auswahl eines passenden Stemmers erforderlich. Das Gleiche gilt für Fachvokabular und Eigennamen. Das lässt sich zwar nicht über die Arbeit am Stemmer-Algorithmus bewerkstelligen – ist vielmehr die Aufgabe im Bereich Language Identification [9] – gehört aber zum Themenkreis des Stemming.

[7] In diesem Sinne wäre das Ignorieren dieser Wortgruppe innerhalb eines Algorithmus aufgrund seiner geringen Auswirkung auf die Abfrageleistung akzeptabel. Liegt eine auffällig hohe Anzahl an Abkürzungen in einem Korpus vor, könnten diese über eine vorgefertigte Liste expandiert werden. Anbieten würde sich diese Vorgehensweise beispielsweise für das Stemming technischer Dokumentation.

[8] Beispielsweise Anglizismen – z.B.: Carstensen, Broder und Busse, Ulrich. 2001. Anglizismen-Wörterbuch. Der Einfluss des Englischen auf den deutschen Wortschatz nach 1945.
Schlobinski, Peter. 2000. Anglizismen im Internet. Networx 14. `http://www.mediensprache.net/de/networx/docs/networx-14.aspx`

[9] Mehr zur Identifizierung von Sprachen z.B. in Řehůřek, Radim un Kolkus, Milan. 2009. Language Identification on the Web. Extending the Dictionary Method. In: Computational Linguistics and Intelligent Text Processing. 10th International Conference, CICLing 2009, Mexico City, Mexico. Proceedings Hg. von Alexander Gelbukh. 2009.

Die Erforderlichkeit einer Sprachidentifizierung ist kein Spezifikum des Porter-Stemmers, sondern wäre ein notwendiger Vorbereitungsschritt für jedes erfolgreiche Stemming. [10]

Beispiele

Manifest → *manif*
reality → *reality*

4.5 Komposita

Die spontane Neubildung und die Verwendung von bereits bestehenden Komposita sind in der deutschen Sprache üblich und stellen den häufigsten Worttyp dar. [11]. Es handelt sich dabei um ein zusammengesetztes Wort, das aus der Verbindung von mindestens zwei bereits vorhandenen Wörtern hervorgeht. Die deutschen Komposita werden – gemäß der geltenden deutschen Rechtschreibung – zusammengeschrieben. [12] Es werden mehrere Formen von Komposita unterschieden: z.B.: Nomen-Nomen, Verb-Nomen, Verb-Verb, Nomen-Adjektiv, Adjektiv-Verb, u.a. Da das Snowball-Verfahren des Porter-Stemmers ursprünglich für die englische Sprache konzipiert wurde und englische Komposita mit einem Leerzeichen geschrieben werden (z.B. *apple juice concentrate*) stellt es nicht den Versuch an, Komposita für den Zweck des Suffix-Stripping aufzutrennen. So entspricht – wenn man das Verfahren für eine deutsche Textsuche benutzt – beispielsweise das Suchwort *Buch* nicht den Worten *Buchmarkt* und *Sachbuch*. Dieses Verhalten kann erwünscht sein, steht aber im Gegensatz zu der Funktionsweise des originalen Porter-Stemmers für die englische Sprache, bei dem die Einzelteile des Kompositums getrennt betrachtet werden. Für eine optimale IR Performance der deutschen Sprache wird daher ein spezielles Verfahren benötigt, das in der Lage ist, alle Wörter – bzw. im linguistischen Sinne Morpheme – eines Kompositums auf ihre Stammform zu reduzieren.

Beispiele

Buchmarkt → *buchmarkt*
Tageszeitung → *tageszeit*
Vinylschallplatte → *vinylschallplatt*

---

[10] Beschäftigt haben sich mit diesem Problem bereits Macdonald *et al.*. In ihrem Paper [Macdonald, 2006] entwickeln sie eine sprachspezifische Technik, um eine korrekte Stemmingmethode anzuwenden.

[11] Vgl. dazu z.B.: Naumann, Bernd. 2000. Einführung in die Wortbildungslehre des Deutschen.

[12] Vgl.: Regeln und Wörterverzeichnis. Überarbeitete Fassung des amtlichen Regelwerks 2004. Rat für deutsche Rechtschreibung. München und Mannheim, Februar 2006, `http://rechtschreibrat.ids-mannheim.de/`

5 Lösungsvorschläge für die Behandlung von Komposita

Dass die deutschen Komposita für das IR ein eigenes Verfahren brauchen, dessen sind sich Monz [2001], Braschler [2003], Tomlinson [2004] und Savoy [2004] einig. Man spricht diesbezüglich von *Decompounding* oder *Compound-Splitting*. Dabei wird das Kompositum auf seine einzelnen Morpheme aufgeteilt. So schreiben etwa Monz *et al.*: "Decompounding seems to be an essential component of any information retrieval system for [...] German." [Monz, 2001] Und Braschler/Ripplinger: "An important finding of this study is that decompounding contributes more to performance improvements than stemming." [Braschler, 2003] Und auch das Ergebnis von Tomlinsons Untersuchung lässt darauf schließen, dass das Decompounding einen wesentlichen Teil zur Retrieval Performance beiträgt: "Overall for German [...] the *SearchServer* [13] scored significantly higher on average, presumably because of compound-splitting." [Tomlinson, 2004]
Wie sieht nun die Behandlung der Komposita in den einzelnen Forschungsarbeiten aus?
Der Decompounder, den Monz *et al.* verwenden, arbeitet durch rekursives Analysieren des Wortes und findet dabei bekannte Nomen. Die daraus resultierenden Morpheme werden mit dem Lemmatizer des Tree-Tagger-Projektes für Deutsch auf ihre Stammform reduziert. [14] Die beiden Autoren merken an, dass sie den Porter-Stemmer nicht benutzt haben, weil er dazu tendiere, aggressiver zu sein – also mehr wegzuschneiden – als ein Lemmatizer. [Monz, 2001]
Braschler/Ripplinger entwickeln keine eigene Methode, sondern sie vergleichen bereits vorhandene Decompounder. Am besten schneiden dabei *Spider_FS* und *MPRO_LS_d* ab. Sie erfüllen die Kriterien, die Anzahl der Treffer zu erhöhen und inkorrekte Autrennungen zu vermeiden. *Spider_FS* trennt aggressiv, *MPRO_LS_d* vermeidet Auftrennungen, die linguistisch nicht korrekt sind. Beide Decompounder haben zwar unterschiedliche Schwerpunkte, zeigen aber ähnliche Ergebnisse, so das Conclusio von Braschler/Ripplinger.
Tomlinson vergleicht eine experimentelle Version des *SearchServer*, der einen lexikalischen Stemmer beinhaltet und die Zerlegung von Komposita inkludiert, mit dem Porter-Stemmer. Er kommt dabei zum Ergebnis, dass der *SearchServer* eine bessere Retrieval Performance bietet als der Porter-Stemmer. Dieser Umstand ist der Tatsache zuzurechnen, dass der *SearchServer* Decompounding durchführt und der Porter-Stemmer nicht.
Savoy benutzt folgende Methode für das Decompounding: Das zu untersuchende Kompositum wird mit Hilfe einer Wortliste in bekannte Wörter aufgeteilt. Gibt es dabei mehrere Möglichkeiten, wählt der Algorithmus die wahrscheinlichste (nach Anzahl der Vorkommen der einzelnen Bestandteile) aus. Für den Erfolg dieses Decompoundings gibt Savoy allerdings keine Angaben.

[13] Anm.: Es handelt sich beim *SearchServer* um einen lexikalischen Stemmer, d.h. Stemming basierend auf einem Wörterbuch oder einem Lexikon der Zielsprache.

[14] Ein Lemmatizer reduziert die Flexionsformen eines Wortes auf eine Grundform. Der *Tree-Tagger* ist ein Werkzeug, um Texte mit Wortart und Lemmainformationen zu versehen.

6 Conclusio

Die Untersuchung des Porter-Stemmers für Deutsch ergab eine Reihe von Fehlerklassen. Diese zu definieren und zu beschreiben haben wir uns zur Aufgabe gemacht. Da es in der für uns zugänglichen Forschungsliteratur bisher noch keinen Aufsatz über die gesammelten Fehlerklassen gibt, basiert unsere Untersuchung auf eigenen Beobachtungen. Die Fehler, die bei der Verwendung des Porter-Stemmers für Deutsch auftreten, waren aufgrund ihrer Auffälligkeit gut zu finden und zu kategorisieren. Wir vermuten, dass die gefundenen Fehlerklassen in allen deutschsprachigen Texten zu finden sind, sind uns aber bewusst, dass die Häufigkeitsverteilung in dem von uns untersuchten Text nicht repräsentativ sein kann, da er von einem einzigen Autor geschrieben und vergleichsweise kurz ist. Der zweite Teil der Arbeit beschäftigt sich mit dem Auftrennen von Komposita – eine der von uns definierten Fehlerklassen – und basiert gegengleich nur auf bereits vorhandener Forschungsliteratur zu diesem Thema. Die Auswahl dieser Aufgabenstellung ergab sich daraus, dass die Komposita-Auftrennung ein typisches Problem für das Stemming der deutschen Sprache ist. Wir wollten herausfinden, welche Lösungsansätze bereits bestehen und wie diese aussehen. Es zeigte sich, dass es zu Decompounding bereits einige Arbeiten gibt und alle Ergebnisse dafür sprechen, dass die Durchführung von Decompounding die Retrieval Performance steigert.

Literaturverzeichnis

Braschler, Martin und Ripplinger, Bärbel. 2003. Stemming and Decompounding for German Text Retrieval. *Pages 177–192 of:* Sebastiani, Fabrizio (ed), *Advances in Information Retrieval.* Lecture Notes in Computer Science. Springer.

Macdonald, Craig et al. 2006. University of Glasgow at WebCLEF 2005: Experiments in Per-Field Normalisation and Language Specific Stemming. *Pages 898–907 of: CLEF'05 Proceedings of the 6th international conference on Cross-Language Evalution Forum: accessing Multilingual Information Repositories.* Springer.

Monz, Christof und Rijke, Maarten De. 2001. Shallow Morphological Analysis in Monolingual Information Retrieval for Dutch, German and Italian. *Pages 262–277 of: Evaluation of Cross-Language Information Retrieval Systems, CLEF 2001.* Lecture Notes in Computer Science. Springer.

Porter, Martin F. 1980. An Algorithm for Suffix Stripping. *Programm 14, Nr. 3*, 130–137.

Porter, Martin F. 2009. *German Stemming Algorithm.* `http://snowball.tartarus.org/algorithms/german/stemmer.html`.

Savoy, Jaques. 2004. Report on CLEF-2003 Monolingual Tracks: Fusion of Probabilistic Models for Effective Monolingual Retrieval. *Pages 322–336 of:* Peters, Carol, Gonzalo, Julio, Braschler, Martin, & Kluck, Michael (eds), *Comparative Evaluation of Multilingual Information Access Systems.* Lecture Notes in Computer Science. Springer.

Tomlinson, Stephen. 2004. Lexical and Algorithmic Stemming Compared for 9 European Languages with Hummingbird SearchServer TM at CLEF 2003. *In: In Carol Peters, editor, Working Notes for the CLEF 2003 Workshop. http://clef.iei.pi.cnr.it/2003/WN web/19.pdf.* Springer.

A Funktionsweise des Porter-Stemmers für Deutsch

Den Erläuterungen der drei Schritte werden einige Definitionen vorangestellt. Für den Algorithmus gelten folgende Buchstaben als Vokale: *a*, *e*, *i*, *o*, *u*, *y*, *ä*, *ö*, *ü*. Alle anderen Buchstaben werden als Konsonanten behandelt, sowie *u* und *y*, wenn sie zwischen zwei Vokalen stehen. *ß* wird durch *ss* ersetzt.
Desweiteren werden zwei Variablen benötigt: R1 und R2. R1 bezeichnet die Region nach dem ersten Nicht-Vokal der auf einen Vokal folgt, kann aber auch leer sein wenn dieses Muster im Wort nicht vorkommt. R2 lässt sich auf dieselbe Weise finden wie R1, sucht das Muster aber in R1 und nicht im Wort. Diese beiden Variablen werden in den Regeln zur bedingten Entfernung von Endungen verwendet. Dergestalt können R1 und R2 aussehen (Porter gibt ein englischsprachiges Beispiel an, hier wird ein deutsches benutzt): *diversifizieren* → R1: *ersifizieren*, R2: *sifizieren* oder *Information* → R1: *formation*, R2: *mation*.
Die drei Schritte stellen sich folgendermaßen dar:

Schritt 1 Es wird die längste Endung aus folgenden Gruppen entfernt, wenn diese in R1 liegt:
- a) em, ern, er
- b) e, en, es
- c) s (wenn nach *b*, *d*, *f*, *g*, *h*, *k*, *l*, *m*, *n*, *r* oder *t*)

Wird eine Endung aus Gruppe b) entfernt, und steht davor *niss* wird das letzte *s* entfernt.

Schritt 2 Es wird die längste Endung aus folgenden Gruppen entfernt, wenn diese in R1 liegt:
- a) en, er, est
- b) st (wenn nach *b*, *d*, *f*, *g*, *h*, *k*, *l*, *m*, *n* oder *t*)

Schritt 3
- a) end, ung (löschen wenn in R2. Wenn *ig* davor, löschen wenn in R2 und nicht nach *e*)
- b) ig, ik, isch (löschen wenn in R2 und nicht nach *e*)
- c) lich, heit (löschen wenn in R2. Wenn *er* oder *en* davor, löschen wenn in R1)
- d) keit (löschen wenn in R1. Wenn *lich* oder *ig* davor, löschen wenn in R2)

Nach dem letzten Schritt werden die Umlaut-Striche entfernt.