

Korpusbau für das Variantengrammatik-Projekt

Klaus Rothenhäusler

Im Rahmen des Variantengrammatik Projekts wurde ein Korpus aus Texten von Internetauftritten lokaler Zeitungen erstellt, wobei lediglich redaktionelle Artikel aus den Lokalteilen im Korpus landen sollten. Tabelle 1 gibt Auskunft über die Verteilung der Artikel auf Regionen und Zeitungen im fertigen Korpus.

Tabelle 1: Verteilung der Artikel im Korpus

D-Südost	298477	D-Augsburger Allgemeine	87567
		D-Nürnberger Nachrichten	70497
		D-Oberbayerisches Volksblatt	40770
		D-Mittelbayerische Zeitung	32076
		D-Fränkischer Tag	28873
		D-Passauer Neue Presse	25040
		D-Frankenpost	13654
D-Mittelost	248384	D-Thüringer Allgemeine	159901
		D-Freie Presse	29916
		D-Mitteldeutsche Zeitung	29297
		D-Lausitzer Rundschau	14205
		D-Leipziger Volkszeitung	11347
		D-Dresdner Neue Nachrichten	3718
D-Nordost	218012	D-Volksstimme	61949
		D-Nordkurier	47548
		D-Märkische Allgemeine	41847
		D-Märkische Oderzeitung	33353
		D-Der Tagesspiegel	31391
		D-Schweriner Kurier	1065
		D-Ostsee-Zeitung	859
D-Mittelwest	209874	D-Rheinische Post	136871
		D-Hessische/Niedersächsische Allgemeine	25480
		D-Neue Westfälische	24810

		D-Kölner Stadt-Anzeiger	10461
		D-Rhein-Zeitung	6626
		D-Darmstädter Echo	5626
D-Südwest	203863	D-Schwäbische Zeitung	68945
		D-Schwarzwälder Bote	48623
		D-Südkurier	23737
		D-Stuttgarter Zeitung	22830
		D-Heilbronner Stimme	18159
		D-Stuttgarter Zeitung (Leonberger Kreiszeitung)	10565
		D-Badische Zeitung	7204
		D-Stuttgarter Zeitung (Marbacher Zeitung)	3800
D-Nordwest	194021	D-Osnabrücker Zeitung	87944
		D-Weser Kurier	44867
		D-Hannoversche Allgemeine	29052
		D-Hamburger Morgenpost	15768
		D-Kieler Nachrichten	10590
		D-Ostfriesen-Zeitung	5800
A-Süd	88634	A-Kleine Zeitung	49707
		A-Krone	38927
CH-West	53547	CH-Aargauer Zeitung	28429
		CH-Bernerzeitung.ch	13799
		CH-Basellandschaftliche Zeitung	11319
A-Mitte	41612	A-Der Standard	16681
		A-Oberösterreichische Nachrichten	15407
		A-Salzburger Nachrichten	6761
		A-Salzburger Fenster	2033
		A-Kurier (Oberösterreich)	724
		A-Wirtschaftsblatt	6
A-West	40850	A-Vorarlberg Online	34240
		A-Tiroler Tageszeitung	6610
CH-Ost	30696	CH-St. Galler Tagblatt	27247
		CH-Südostschweiz.ch	3449
STIR-Südtirol	15143	STIR-Der Vinschger	8986
		STIR-Südtirol Online	6157
CH-Süd	14391	CH-1815 - Das Oberwalliser Nachrichtenportal	7266
		CH-Neue Luzerner Zeitung	7125
LUX-Letzebuerg	13289	LUX-Luxemburger Wort	7293
		LUX-Tageblatt	5996
A-Ost	12626	A-Niederösterreichische Nachrichten	4855

		A-Wiener Zeitung	3206
		A-Kurier (Niederösterreich	1748
		A-Kurier (Wien)	1431
		A-Kurier (Burgenland)	1386
BELG-Eupen	11467	BELG-Grenz-Echo	11467
LIE-Vaduz	4229	LIE-Liechtensteiner Vaterland	4229

Die einzelnen Schritte bei zur Gewinnung und Verarbeitung der Daten werden im folgenden beschrieben.

Datenakquisition: Crawling

Ein webbasiertes Korpus vom geplanten Umfang kann manuell nicht erstellt werden, das Herunterladen von Webseiten muss automatisiert statt finden, was als Crawling bezeichnet wird. Die entsprechende Software heißt Crawler und das Ergebnis ist ein Crawl. Wir haben den vom Internet Archive entwickelten Heritrix Crawler (<http://crawler.archive.org/>) eingesetzt und so modifiziert, dass der Crawl direkt in eine PostgreSQL-Datenbank geschrieben werden kann. Ein Crawler muss mit Daten konfiguriert werden, die exakt definieren, welche Webseiten heruntergeladen werden sollen. Eine Beispielkonfiguration zeigt Abbildung 1. Sie setzt sich folgendermaßen zusammen:

1. **start-urls** geben an, welche Webseiten ein Crawler als erstes besuchen soll. URL (Uniform Resource Locator) ist die technische Bezeichnung für eine Internetadresse (oder präziser für die Adresse eines beliebigen Objekts im Internet, beispielsweise werden auch Bilder und andere

source	Weser Kurier
country	Deutschland
großregion	D-Nordwest
start-urls	http://www.weser-kurier.de/bremen.html
follow-urls	(region bremen sport/(region bremen))/[^\,"]+\.\html
take-urls	(region bremen sport/(region bremen))/[^\"]+arid[,0-9]+\.\html
non-takes	n.a.

Abbildung 1: Konfiguration des Crawlers für den *Weser Kurier*

Mediendaten mit URLs lokalisiert). Links auf einer Webseite werden genutzt, um auf URLs zu verweisen und ermöglichen so die Navigation zwischen Webseiten.

2. Der Crawler verfolgt alle Links auf einer besuchten Seite weiter, die dem **follow-urls** Muster entsprechen. Es wird in Form eines regulären Ausdrucks angegeben.
3. Tatsächlich gespeichert wird eine besuchte Seite nur, wenn sie einerseits dem **take-urls** Muster entspricht und
4. andererseits nicht auf des **non-takes** Muster passt.

Die Konfiguration muss individuell für jede Website ermittelt werden und basiert auf der Analyse von deren Struktur und Linkformat. Für Crawls, die über einen längeren Zeitraum erstellt werden, erweist sich als problematisch, dass Website-Betreiber immer wieder kleinere oder größere Strukturveränderungen vornehmen, die den Crawler aus dem Tritt bringen können. Um überhaupt auf solche Probleme aufmerksam zu werden, wurde die Anzahl heruntergeladener Dokumente pro Zeitung und Tag überwacht und bei starker Abweichung eine entsprechende Warnung ausgegeben, so dass meist zeitnah eine Anpassung möglich war. Mitunter kam es während des Crawl-Zeitraums aber auch vor, dass die Inhalte einer Zeitung nahezu komplett hinter einer Paywall verschwanden oder zumindest für Crawler gesperrt wurden. In dem Fall wurde kurzfristig die entsprechende Zeitung durch eine andere ersetzt.

Die heruntergeladenen Dokumente werden in einer relationalen Datenbank gespeichert. Neben den eigentlichen HTML-Daten werden HTTP-Header, Crawl-Datum, URL und übermitteltes Encoding gespeichert. Die Encoding-Angabe, gibt Auskunft darüber, wie die Bytefolgen der übermittelten Daten zu interpretieren sind, und ist deshalb entscheidend für eine saubere Weiterverarbeitung. Leider stimmt das übermittelte Encoding nicht immer mit dem tatsächlich verwendeten Encoding überein, weshalb die empfangenen Daten untersucht werden, um das Encoding zu prüfen. Aber auch dieses Verfahren funktioniert nicht vollkommen zuverlässig, weshalb im fertigen Korpus leider noch einige Encodingfehler zu finden sind.

Datenbereinigung: Scraping

Die gespeicherten HTML-Dokumente enthalten nicht nur den gewünschten Text sondern eine Menge unerwünschtes Material wie Navigationsmenüs,

Werbung, Kommentare usw. die mitsamt jeglichen HTML-Markups entfernt werden müssen. Diesen Verarbeitungsschritt bezeichnet man als Scraping.

Ebenso wie der Crawler muss auch ein Scraper individuell für jede Zeitung angepasst werden, wobei der Aufwand hier noch wesentlich höher liegt, weil meist unterschiedlicher HTML-Markup bei ein und derselben Zeitung zu finden ist beispielsweise für Sportberichte im Gegensatz zu Interviews. Zudem muss nicht nur der Text sondern auch jeweils die wichtigsten Metadaten möglichst sauber aus dem HTML-Code extrahiert werden.

Um die manuelle Arbeit gering zu halten, wurde das semiautomatische Scraping-Tool `sitescraper` (<https://code.google.com/p/sitescraper/>) eingesetzt, das an Hand einiger Beispiele Regeln lernt, um die gewünschten Daten zu extrahieren. Für alle Zeitungen wurden deshalb jeweils zehn Dokumente als Trainingsmaterial annotiert. Leider zeigte sich aber, dass die Generalisierungen, die `sitescraper` trifft völlig unzureichend waren. Es gelang auch mit größeren Mengen Trainingsmaterial nicht, befriedigende Ergebnisse zu erzielen, so dass letztlich doch der manuelle Weg beschritten werden musste. Für alle Zeitungen, unterschiedlichen HTML-Formate und unterschiedlichen Metadaten wurde deshalb manuell bestimmt, wie sie mit Hilfe eines XPath-Ausdrucks zu extrahieren sind. XPath ist eine Abfragesprache, mit der sich Pfade zu Knoten in XML-Dokumenten spezifizieren lassen. Die Extraktion erfolgt dann programmatisch mit Hilfe eines Python-Skripts. Zur Flankierung des Arbeitsablaufs wurde das Add-on `Firepath` (<https://addons.mozilla.org/de/firefox/addon/firepath/>) für den Firefox-Browser eingesetzt, das auf die Entwickler-Erweiterung `Firebug` (<https://addons.mozilla.org/de/firefox/addon/firebug/>) aufsetzt und es erlaubt, interaktiv XPath-Ausdrücke für eine Webseite zu entwickeln und grafisch auszuwerten. In Kombination mit dem kommandozeilenbasierten Programm `xmllint` (<http://xmlsoft.org/xmllint.html>), das umfangreichere Tests mit einem XPath Ausdruck ermöglicht, konnte so nach intensiver Einarbeitungszeit ein ausreichend schlanker Workflow erreicht werden.

Das Scraping operiert unmittelbar auf den in der Datenbank abgelegten HTML-Dokumenten und nutzt die zusätzlich gespeicherten Crawl-Metadaten, um daraus ein einfaches Pseudo-XML-Dokument zu erzeugen. Bevor es jedoch gespeichert wird, kommt noch ein erster Filter zum Einsatz: Für jedes Dokument wird an Hand des Textkörpers eine Prüfsumme (als SHA1-Hash-Wert) berechnet, mit Hilfe derer exakte Duplikate identifiziert werden können. Dokumente mit identischer Prüfsumme enthalten also den exakt gleichen Text, unterscheiden sich aber in der Herkunfts-URL und können auch in allen anderen Metadaten unterschiedlich sein. Findet sich

keine identische Prüfsumme unter den bereits berechneten, wird das XML-Dokument in der Datenbank abgelegt, andernfalls wird es verworfen und das HTML-Ausgangsdokument als Duplikat markiert. Dieser erste Filter ist sehr wichtig, denn er reduziert die Dokumentenanzahl um über 44% (von 3690579 auf 2059172).

Fast-Dubletten (Near Duplicate) Erkennung

Bevor die eigentliche linguistische Verarbeitung statt findet, werden nochmals Dokumente mit einer Fast-Dubletten-Erkennung ausgefiltert. Fast-Dubletten sind Dokumente, die sich lediglich durch wenige Wörter unterscheiden, dadurch aber für die exakte Duplikaterkennung unauffindbar bleiben. Zwar stellt das Information Retrieval eine Fülle an Verfahren bereit, die in einer Dokumentmenge ähnliche Dokumente zu einem gegebenen Ausgangsdokument finden, um jedoch alle untereinander ähnlichen Dokumente zu identifizieren, müsste jedes Dokument mit jedem anderen verglichen werden. Die Anzahl nötiger Dokument-Vergleiche wächst somit quadratisch mit der Anzahl an Dokumenten, wodurch sich der exakte Ansatz schon für ein noch mäßig großes Korpus wie das vorliegende verbietet, zumal ein Dokumentenvergleich rechnerisch keine ganz billige Operation ist.

Stattdessen setzt man bei der Fast-Dubletten-Suche auf Verfahren, die einen kleinen Fehler zulassen, sich dafür aber sehr effizient implementieren lassen. Insbesondere haben wir uns für den SpotSigs-Algorithmus [6] entschieden. Er erstellt platzeffiziente Repräsentationen der Dokumente und ordnet sie geschickt in einer Datenstruktur so an, dass ähnliche nahe beieinander liegen und dadurch die Anzahl Vergleiche dramatisch reduziert werden kann, um lediglich die zu finden, die oberhalb eines vorgegebenen Ähnlichkeitsschwellwerts liegen. Insgesamt wird auf diesem Weg nochmals eine Liste von 57968 Dokumenten (knapp 1,6% des Gesamtcrawls bzw. gut 2,8% der nach dem ersten Filter noch übrigen).

Die Fast-Dubletten Analyse muss vor der Hauptverarbeitung durchgeführt werden, weil sie auf globale Statistiken zugreifen muss, die erst nach einem Komplettdurchlauf vorliegen.

Datenannotation: Verarbeitungspipeline

Die automatische linguistische Annotation erfolgt in einer sogenannten Verarbeitungspipeline. Das bedeutet lediglich, dass die Dokumente nacheinander unterschiedliche Verarbeitungsmodule durchlaufen, die jeweils Annotationen

hinzufügen und dabei eventuell auf Ergebnisse zuvor durchlaufener Module zugreifen.

Technische Grundlage der Pipelinearchitektur ist das UIMA Framework (<https://uima.apache.org/>). UIMA liefert ein Gerüst, das den modularen Aufbau von Anwendungen zur Verarbeitung von unstrukturierten Daten insbesondere Textdaten erleichtert.

Analysekomponenten der Verarbeitungspipeline

In der Pipeline durchlaufen die Textdokumente nacheinander die folgenden Module.

Eingabe

Die Verarbeitungspipeline erhält als Eingabe einen Datenbank-Export mit den Pseudo-XML-Dokumente, die der Scraper produziert hat. Eine direkte Anbindung an die Datenbank wurde bewusst vermieden, so dass Datenbank und Verarbeitungspipeline problemlos getrennt voneinander umgezogen werden können.

Da der XML-Export im Metadatenfeld für das Datum (`<date>`) das nicht weiter verarbeitete Resultat des Scrapers enthält, finden sich dort ganz unterschiedliche Formate von Datumsangaben, z.B.

29.01.12
vom 29.01.2012
Mi. 25. Juli 2012
vor 5 Stunden

All diese Angaben werden in ein einheitliches Datumsformat gebracht, insbesondere wird auch die relative Angabe im letzten Beispiel so aufgelöst, dass sie als absolutes Datum zur Verfügung steht. Als Referenzzeitpunkt kann der Crawl-Zeitpunkt verwendet werden in der Annahme, dass die relativen Datumsangaben dynamisch erzeugt werden beim Besuch der Internetseite. Zur Auflösung der Zeitangaben kommt HeidelbergTime (<https://code.google.com/p/heideltime/>) zum Einsatz, das um einige Regeln erweitert werden musste, so dass letztlich nur noch für 0.29% der Dokumente die Original-Datumsangaben nicht verarbeitet werden konnte, was oft an Fehlern liegt, die schon in der Quelle vorhanden sind.

Beim Einlesen der Dokumente wird auch noch ein letztes Mal gefiltert: Da sich bei einigen Internetauftritten der beobachteten Zeitungen die gewünschten Texte über die Site-Struktur nicht sauber von Agenturmeldungen trennen

ließen, hat das VG-Projekt Filter beigesteuert, die Texte aussortieren, in denen gängige Agenturkürzel an verdächtigen Stellen auftauchen. Außerdem fallen Texte heraus, die offensichtlich keine standardsprachlichen Äußerungen darstellen, was an einem erhöhten Ziffern-zu-Buchstaben Verhältnisses auffällt.

Während in der Eingabe noch das ganze XML-Dokument betrachtet wird, kümmern sich die folgenden Modulen zur automatischen Sprachverarbeitung lediglich um den Inhalt des Textkörpers (`<body>`) und des Titel (`<title>`) Elements aus dem Metadaten-Header.

Tokenizer

Aufgabe des Tokenisierers ist die Bestimmung von Wortgrenzen im Eingabetext. Um korrekt zu funktionieren, muss insbesondere der Tagger mit Token gefüttert werden, die denen entsprechen, mit denen das Modell erstellt wurde, dass er nutzt um Wortarten zuzuweisen. Deshalb folgt unser Tokenisierer eng dem beim TreeTagger mitgelieferten, der sprachabhängige Regeln definiert, um beispielsweise korrekt mit Klitisierungen umgehen zu können.

TreeTagger (<http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/>)

Der TreeTagger [4] gehört wohl zu den am weitesten verbreiteten automatischen Sprachverarbeitungstools überhaupt. Er läuft äußerst robust, schnell und liefert sowohl für Wortarten als auch Lemma von Haus aus sehr gute Ergebnisse. Seine Genauigkeit bezüglich der vergebenen Wortarten-Tags auf Testdaten liegt bei 97.53%. Verwendet wird das STTS Tagset mit insgesamt 54 Tags.

Um die Ergebnisse des Taggers weiter zu verbessern, wurden seitens des VG-Projekts zusätzliche Vollformen Lexika zur Verfügung gestellt, die insbesondere die Abdeckung von Eigennamen verbessern sollen. Zusätzlich zu diesen Listen haben wir alle Wörter im Korpus identifiziert, die dem Standard TreeTagger für das Deutsche unbekannt waren. Diese Wörter wurde mit SMOR (<https://code.google.com/p/cistern/wiki/SMOR>) morphologisch analysiert, um mit einiger Sicherheit ihre möglichen Wortarten und das Lemma zu bestimmen. Mit der Vereinigung der beiden Listen als Zusatzlexika wurde dann ein neues Tagging-Modell erstellt.

RFTagger (<http://www.cis.uni-muenchen.de/~schmid/tools/RFTagger/>)

Der RFTagger [5] ist ebenfalls ein Wortarten-Tagger, die vergebenen Tags sind jedoch sehr viel feinkörniger und enthalten wo anwendbar Information

über Kasus, Numerus, Genus, Definitheit, Tempus und Modus. Der Tagger nutzt Entscheidungsbäume, um Kontexte zu klassifizieren und ein passendes Tag zu vergeben. Für das VG-Korpus wurde das für Forschungszwecke frei zur Verfügung stehende Modell für das Deutsche verwendet, das auf Daten des TIGER Korpus [1] beruht. Die publizierte Genauigkeit liegt bei 92,2% auf einem Testset des TIGER-Korpus.

Morphisto (<http://www1.ids-mannheim.de/lexik/home/lexikprojekte/lexiktextgrid/morphisto.html>)

Zur morphologischen Analyse wird morphisto genutzt, das in Form umfangreicher handgeschriebener Regeln eine Morphologie für das Deutsche definiert, die in einen endlichen Automaten übersetzt zur Analyse von Wortformen genutzt werden kann.

Da allerdings keine Fugenelemente markiert werden, haben wir sie in den Analysen rekonstruiert und eine zusätzliche Morphkategorie <FUG> eingeführt. Morphisto liefert alle möglichen Analysen für ein Eingabewort, eine Disambiguierung mit Hilfe der Taggerausgaben wird nicht mehr vorgenommen. Bei Analysen, die die maximale Länge (4095 Zeichen) für Attribute in der CWB überschreiten, werden die komplexesten Einzelanalysen entfernt, d.h. diejenigen, die aus den meisten Morphen bestehen.

Named Entity Recognition, NER (<http://nlp.stanford.edu/software/CRF-NER.shtml>)

Ein Eigennamenerkennungsmarkierer markiert in einem Text auftretende Namen und ordnet sie einer kleinen Menge von Kategorien zu. Der verwendete Eigennamenerkennungsmarkierer aus Stanford arbeitet statistisch, muss also mit zuvor bezüglich Eigennamen annotierten Daten trainiert werden. Für das Deutsche werden fertig trainierte Modelle zur Verfügung gestellt, die vier Kategorien von Eigennamen unterscheiden: Personen (PER), Orte (LOC), Organisationen (ORG) und Sonstige (MISC), die vor allem für Ableitungen von geographischen Namen vergeben werden, z.B. Einwohnerbezeichnungen wie "Österreicheröder SSchweizer". Das verwendete Modell [2] nutzt die Trainingsdaten eines CoNLL Shared Task aus dem Jahr 2003 [7] und bindet als semantisches Generalisierungsmerkmal zusätzlich distributionelle Information ein, die aus einem großen nicht annotierten deutschen Zeitungskorpus (HGC, <http://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/hgc.html>) gewonnen wird und so helfen kann, die geringe Größe des Trainingskorpus auszugleichen. Die publizierten Ergebnisse liegen bei

einer Präzision von 86.6% und einer Abdeckung (Recall) von 71,2%.

Sentence Splitter

Zur Bestimmung von Satzgrenzen werden lediglich die STTS-Tags ausgewertet und Satzgrenzen gesetzt, wo satzbeendende Interpunktionszeichen ("\$.") gefunden werden.

Clause Annotator

Beim Gliedsatzerkenner handelt es sich um eine eigene Entwicklung die auf den Daten der TüBa/DZ (<http://www.sfs.uni-tuebingen.de/ascl/ressourcen/corpora/tuepp-dz.html>) beruht. Er verwendet lexikalische, Wortart- und orthographische Merkmale, um zum Teil regelbasiert zum Teil statistisch Entscheidungen über Gliedsatzgrenzen zu treffen. Wo möglich wird auch markiert, ob ein Gliedsatz koordinierend oder subordinierend angebunden wird und von welchem semantischen Typ er ist (*kausal*, *modal*, etc.). Zusätzlich werden innerhalb der Gliedsätze topologische Felder annotiert.

Clause Tempus Annotator

Die im vorherigen Modul erkannten Gliedsätze werden zusätzlich auf Inflektionsmerkmale untersucht, um die Annotation mit einer Tempusangabe zu erweitern. Es handelt sich hierbei ebenfalls um eine Eigenentwicklung, die ausschließlich auf handgeschriebenen Regeln basiert.

Gazetteer Annotator

Dem "Handbuch deutscher Kommunikationsverben"[3] folgend wurde eine Liste von Kommunikationsverben erstellt und entsprechend lemmabasiert annotiert. Die Klassifikation des Handbuchs wurde dabei als Attribut der Annotation beigefügt.

Dependenzparsing (Don Tuggener)

Das Korpus wurde mit dem Dependenzparser *ParZu*¹ geparst. Der Parser verfügt über eine handgeschriebene Grammatik und ein statistisches Modul, das beim Auflösen von Ambiguitäten konsultiert wird. Da in der CWB

¹<https://github.com/rsennrich/ParZu>

jedoch keine hierarchische Strukturen (Dependenzrelationen) abgebildet werden können, konnten aus dem Parse nur flache Merkmale extrahiert und in die Korpusannotation integriert werden (grammatikalische Funktion, verbesserte Lemmatisierung).

CWB-Indizierung

Die Verarbeitungspipeline erzeugt als Ausgabe Dateien im vertikalisierten Eingabeformat der Corpus Workbench (CWB, <http://cwb.sourceforge.net/>) in Kombination mit einem Skript, das die nötigen Befehle zur Indizierung des Korpus für die Verwendung in der CWB enthält. Das Skript kann mit minimalen Anpassungen für ein anderes System angepasst werden, so dass die Korpusinstallation auf einem beliebigen Rechner eine Sache von Minuten ist.

Datenpublizierung: CQPWeb

Die Publikierung des Korpus erfolgt händisch auf einem Server mit einer lauffähigen Instanz von CQPWeb, dem webbasierten Frontend für die CWB.

Literatur

- [1] Sabine Brants u. a. „The TIGER treebank“. In: *Proceedings of the workshop on treebanks and linguistic theories*. 2002, S. 24–41.
- [2] Manaal Faruqi und Sebastian Padó. „Training and Evaluating a German Named Entity Recognizer with Semantic Generalization“. In: *Proceedings of KONVENS 2010*. Saarbrücken, Germany, 2010.
- [3] Gisela Harras, Kristel Proost und Edeltraud Winkler. *Handbuch deutscher Kommunikationsverben (Handbook of German Communication Verbs)*. Schriften des Instituts für Deutsche Sprache 10.2. Berlin: De Gruyter Mouton, 2007. URL: get-book.cfm?BookID=31919.
- [4] Helmut Schmid. „Improvements In Part-of-Speech Tagging With an Application To German“. In: *Proceedings of the EACL SIGDAT-Workshop*. Dublin, 1995, S. 47–50.

- [5] Helmut Schmid und Florian Laws. „Estimation of Conditional Probabilities with Decision Trees and an Application to Fine-grained POS Tagging“. In: *Proceedings of the 22Nd International Conference on Computational Linguistics - Volume 1*. COLING '08. Manchester, United Kingdom: Association for Computational Linguistics, 2008, S. 777–784. ISBN: 978-1-905593-44-6. URL: <http://dl.acm.org/citation.cfm?id=1599081.1599179>.
- [6] Martin Theobald, Jonathan Siddharth und Andreas Paepcke. „Spotsigs: robust and efficient near duplicate detection in large web collections“. In: *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2008, S. 563–570.
- [7] Erik F. Tjong Kim Sang und Fien De Meulder. „Introduction to the CoNLL-2003 Shared Task: Language-independent Named Entity Recognition“. In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*. CONLL '03. Edmonton, Canada: Association for Computational Linguistics, 2003, S. 142–147. DOI: 10.3115/1119176.1119195. URL: <http://dx.doi.org/10.3115/1119176.1119195>.