Werkzeuge zur Extraktion von signifikanten Wortpaaren als Web Service

Fabienne Fritzinger, Max Kisselew, Ulrich Heid, Andreas Madsack, Helmut Schmid

Universität Stuttgart
Institut für maschinelle Sprachverarbeitung
– Computerlinguistik –
Azenbergstr. 12
D 70174 Stuttgart
{fritzife|kisselmx|heid|madsacas|schmid}@ims.uni-stuttgart.de

1 Einleitung

Es gibt viele Techniken und Werkzeuge für die Extraktion von Daten zu signifikanten Wortkookkurrenzen aus Texten, d.h. zur Suche nach Wortverbindungen wie *eine Frage stellen* (Verb+Objektsnomen), *eine knifflige Frage* (Adjektiv+Nomen), *ein Fünkchen Hoffnung* (Nomen+Nomen). Ein Beispiel für ein kombiniertes morphosyntaktisch-statistisches Werkzeug ist die Sketch Engine (Kilgarriff et al. 2004), für ein statistisches Werkzeug ist das UCS toolkit (Evert 2005) beispielhaft. Diese Werkzeuge sind auch erhältlich (mit kommerzieller Lizenz bzw. via SourceForge[1]). Trotzdem ist es nicht trivial, diese und vergleichbare Werkzeuge zu beschaffen, zu installieren und anzuwenden, wenn ad hoc eine Fragestellung zu bearbeiten ist, bei der signifikante Wortverbindungen zu extrahieren sind.

Ziel des vorliegenden Papiers ist es, anhand eines Beispiels die Umrisse einer verteilten Ressourceninfrastruktur für die Extraktion signifikanter Wortpaare zu zeigen. Architekturen, Werkzeuge und Formate für eine solche verteilte Infrastruktur werden derzeit im D-SPIN-Projekt entwickelt[2]: die verteilte Infrastruktur soll Texte und Werkzeuge über Web Services zugänglich und kombinierbar machen. Damit entfällt für den Benutzer das Problem der Beschaffung, der Installation und z.T. der Einarbeitung in die Details der Nutzung der Werkzeuge. Wir simulieren ein solches durch Web Services unterstütztes Szenarium am Beispiel laufender Arbeiten zu Werkzeugen für die Extraktion signifikanter Wortpaare. Dabei ist einerseits die Bereitstellung der Werkzeuge in Web Services neu gegenüber ihrer bisherigen Nutzung, und zum anderen stellen die Extraktionsverfahren eine interessante Kombination von Dependenzparsing, (morpho)syntaktisch basierter Extraktion und den üblichen Signifikanzmaßen dar.

[1] http://www.sourceforge.net

[2] D-SPIN (Deutsche Sprachenressourcen-Infrastruktur) ist ein vom BMBF gefördertes Projekt (2008–2010), das Werkzeuge, Verfahrensweisen und rechtliche Rahmenbedingungen für eine nationale Ressourceninfrastruktur erarbeitet. Das Projekt wird von E. Hinrichs (Universität Tübingen) koordiniert. Die Autoren sind Werkvertragsnehmer der Universität Tübingen. D-SPIN ist das nationale deutsche Ergänzungsprojekt zum eurpäischen Projekt CLARIN, dessen Zielsetzung die Schaffung einer Sprachressourcen-Infrastruktur für ganz Europa ist. Details zu D-SPIN und CLARIN: http://www.sfs.uni-tuebingen.de/dspin/ und http://www.clarin.eu

2 Szenarium

Jemand analysiert deutsche Berichte über Medikamentenprüfungen und will sich über die wichtigsten lexikalischen Kollokationen (im Sinne von Hausmann 2004, Bartsch 2004, etc.), bzw. über die in der Textsorte ganz allgemein häufigsten Zweiwortkombinationen informieren, z.B. weil er solche Texte verfassen oder übersetzen will. Wir gehen hier beispielhaft von einem Übersetzer oder einem technischen Redakteur aus, der an den Daten interessiert ist, aber selbst keine Zeit, kein Interesse oder keine Kompetenz dafür hat, eigene computerlinguistische Werkzeuge zu entwickeln, oder auch nur die verteilt verfügbaren Werkzeuge zu modifizieren oder ihre Interaktion zu steuern.

Als Beispielkorpus in unserem Experiment dienen die Texte über Medikamentenprüfungen der EMEA-Agentur[3], die in verschiedenen Sprachen (ca. 10-14 Millionen Wörter pro Sprache) auf Tiedemanns OPUS-Website[4] zur Verfügung stehen. Aus den deutschen Texten sollen Wortpaare verschiedener grammatischer Kategorien extrahiert werden: Adjektiv+Nomen (z.B. *antagonistische Wirkung, arzneiliche Bestandteile*), Verb+Objektsnomen (*Packungsbeilage beachten, Nebenwirkungen bemerken*) und Nomen+Genitivattribut (*Abfall des Blutdrucks, (wirksame) Bestandteile des Arzneimittels*). Die Wortpaare[5] sollen alphabetisch nach den Kollokationsbasen (Hausmann 2004) sortiert werden: Adjektiv+Nomen nach Nomen, Verb+Objektsnomen nach Nomen, Nomen+Genitivattribut nach Genitivattribut. Außerdem soll eine Sortierung nach der absoluten Frequenz und nach dem mit Hilfe des LogLikelihood Ratio Tests (Dunning 1993) bestimmten Signifikanzwert erfolgen (wie typisch sind die Kombinationen, bzw. wie fest zusammengehörig? Vgl. Evert 2005).

Der Benutzer sucht bei D-SPIN nach Vorschlägen zur Lösung des Problems. Wir nehmen für das vorliegende Experiment an, dass für die allgemeine Aufgabenstellung der Suche nach Kookkurrenzen der in Abschnitt 3 skizzierte Workflow vorgeschlagen werden kann. Auf dessen Bereitstellung als Webservice gehen wir in Abschnitt 4.1 ein. In Abschnitt 4.2 stellen wir ein einfaches Format dar, in dem die Ergebnisse an den Benutzer geliefert werden könnten.

3 Extraktion von Kollokationen

Für Deutsch (anders als z.B. für Englisch) erscheint es notwendig, die Extraktion von Daten zu signifikanten Wortpaaren auf syntaktisch analysiertem Text aufsetzen zu lassen: die drei Verbstellungsmuster des Deutschen und die relativ freie Konstituentenreihenfolge im Mittelfeld (Heid/Weller 2008), Kasussynkretismus (Evert 2004) und als Auswirkung von beidem strukturelle Ambiguitäten führen andernfalls zu schlechten Ergebnissen. Experimente mit Kilgarriffs Sketch Engine (Ivanova et al. 2008) haben gezeigt, dass eine Extraktion aus lediglich getaggtem und lemmatisiertem Text zu niedriger Precision führt, oder dass nur sehr eingeschränkte Kontexte benutzt werden können, um den Preis von sehr geringem Recall. Ein Vergleich zwischen rekursivem Chunking und Dependenzparsing (cf. Heid et al. 2008) hat darüber hinaus ergeben, dass eine Extraktion auf der Grundlage von *"chunked text"* zwar eine akzeptable Precision liefern kann, aber nur ca. 40% des Recalls, der auf geparsten Texten erzielt werden kann.

Für unsere Experimente benutzen wir den Dependenzparser FSPAR (Schiehlen 2003), der grammatische Funktionen markiert und lokale Ambiguitäten annotiert, bezüglich Etiketten (Subjekt vs. Objekt) und Attachment. Der Parser integriert die anderen Schritte der Korpusvorverarbeitung: Tokenizing, Tagging und

[3] Euorpean Medicines Agency.

[4] `http://urd.let.rug.nl/tiedeman/OPUS/EMEA.php`

[5] Eigentlich Paare von Lemmata. Da viele Kollokationen Präferenzen für einen bestimmten Numerus aufweisen, zitieren wir sie hier in der bevorzugten Form.

Lemmatisierung[6]. Im vorliegenden Fall wird der Ausgangstext dennoch zunächst tokenisiert um Satzgrenzen vorab zu identifizieren (zwecks Erhöhung der Verarbeitungsgeschwindigkeit des Parsers). Der Ablauf der Kollokationsextraktion ist in Abbildung 1 schematisch dargestellt.

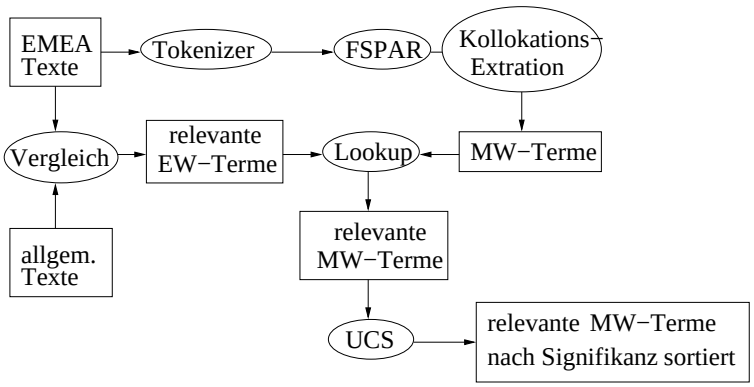


Abbildung 1: Schema der Prozessierungspipeline für die Extraktion von Mehrworttermen.

3.1 Einzelwörter

Es sollen terminologisch relevante Mehrwortausdrücke extrahiert werden, d.h. solche, die terminologisch relevante Einzelwörter enthalten. Derartige Einzelwörter werden vorab anhand des POS-getaggten und lemmatisierten Textes mit Hilfe eines Vergleichs der relativen Häufigkeit in Fach- und Nicht-Fachtexten ermittelt (vgl. Ahmad et al. 1992). Als allgemeinsprachliche Referenz für den Vergleich der relativen Frequenzen verwenden wir Texte aus Zeitungskorpora (bspw. aus der *Frankfurter Rundschau*, 1992/93, FR).

Zwei verschiedene Arten von Listen sind das Ergebnis dieses Verfahrens: eine Liste enthält diejenigen Einzelworttterme, die im Referenztext nicht vorkommen (diese Liste ist nach der absoluten Häufigkeit der Terme sortiert, Vgl. Tabelle 1(a)); die andere Liste besteht aus Termen, die im Referenztext vorkommen und für jeden Termkandidaten aus einem Quotienten, der besagt, wieviel häufiger der Term im Fachtext auftritt als im allgemeinsprachlichen Text. Letztere Liste ist nach diesem Quotienten absteigend sortiert (Tabelle 1 (b)).

(a) Nur EMEA, Nicht FR

Termkandidat	Frequenz
Durchstechflasche	5638
Injektionsstelle	3489
Pharmakokinetik	3426
Hämoglobinwert	3395
Fertigspritze	3271
Ribavirin	3234
Gebrauchsinformation	2801
Dosisanpassung	2580
Epoetin	2302
Hydrochlorothiazid	2128

(b) EMEA und FR

Termkandidat	Quotient	Frequenz
Filmtablette	25522	6389
Injektionslösung	19854	4970
Packungsbeilage	14710	7365
Niereninsuffizienz	14233	3563
Verkehrstüchtigkeit	13558	3394
Leberfunktion	8385	2099
Hypoglykämie	8353	2091
Toxizität	7957	1992
Einnehmen	7035	7045
Hypotonie	6823	1708

Tabelle 1: Die ersten 10 Nomina der beiden Einzelwort-Termlisten.

[6]In einer anderen Werkzeugumgebung müßten u.U. separate Tools für diese Funktionen vorgeschlagen werden.

Für jede Wortart werden für die beiden Listen-Typen Schwellen gesetzt, um das Fachvokabular möglichst präzise zu erfassen. Die daraus resultierenden Ergebnisse werden dann im nächsten Schritt mit ihren Kollokationspartnern versehen.

3.2 Signifikante Wortpaare

Die Mehrwortausdrücke werden aus dem geparsten Text mit Hilfe von in Perl formulierten regulären Suchanfragen extrahiert. Für Verb+Objekt-Paare wird dabei z.B. nach Verben und ihren Akkusativ-Objekten in Aktivsätzen, bzw. nach Subjekten und Verben von Passivsätzen gesucht. Wir untersuchen drei Arten von Mehrwortausdrücken: Adjektiv+Nomen-Paare, Verb+Objekt-Paare sowie Nomen+Genitiv-Nomen-Paare. Die Suche findet alle Lemmapaare, für die der Parser eine dieser drei syntaktischen Kombinationen annotiert hat. Um die häufigsten, bzw. die signifikantesten Paare zu ermitteln, wird für jedes Paar mit dem UCS-Toolkit[7] (Evert 2005) der LogLikelihood-Wert ermittelt. Die Daten werden u.a. nach absteigendem LogLikelihood-Wert sortiert. Ergebnisbeispiele sind in Tabelle 2 angegeben.

(a) Adjektiv+Nomen

Adjektiv	Nomen	Frqz	LogL
allgemeiner	Risikofaktor	64	655
starke	Nierenfunktion	108	511
potentielle	Reaktion	60	508
arzneilicher	Bestandteil	39	458
kardiotoxische	Substanz	43	457
späte	Bewegungsstörung	33	410

(b) Verb+Objektsnomen

Nomen	Verb	Frqz	LogL
Nebenwirkung	bemerken	1665	11381
Packungsbeilage	beachten	1303	9241
Apotheker	fragen	1051	8533
Maschine	bedienen	543	6973
Einnahme	vergessen	677	5584
Anwendung	empfehlen	884	4396

Tabelle 2: Wortpaare mit Angabe der Absolutfrequenz und des LogLikelihood-Wertes.

4 Einbettung in Web Services

Ziel unserer Arbeit ist es, eine verteilte Ressourcenlandschaft zu simulieren, in der die EMEA-Texte, der Parser samt Vorverarbeitung, die Extraktionsroutinen für Mehrwortkandidaten und das UCS-Toolkit jeweils an verschiedenen Orten vorliegen und durch Web Services verbunden werden müssen. Im Gegensatz zu existierenden Web Services, die sich mit linguistischen Anfragen auf Wortebene befassen (wie z.B. im Projekt "Deutscher Wortschatz" entstanden[8]), legen wir unser Hauptaugenmerk auf die Verarbeitung von umfangreichen Textsammlungen in Größenordnungen von etwa 50MB - 200MB.

Für die Realisierung dieser Web Services nehmen wir ein dreistufiges Schichtenmodell an, das eine klare Trennung der verschiedenen Anforderungen und Aufgaben darstellt (Vgl. Abbildung 2). Die äußerste Schicht, auf die wir hier nicht gesondert eingehen werden, befasst sich mit Fragen von Zugriffsberechtigungen, Authentifizierung, Abrechnung, etc. In unserer Implementierung setzen wir voraus, dass es eine solche Schicht bereits gibt[9]. Die innerste Schicht (oder auch der Kern) des Modells enthält (computer-) linguistische Werkzeuge, die bereits vorhanden sind, und die ohne prinzipielle Änderungen in die Web Service Infrastruktur eingebunden werden können. Wir konzentrieren uns hier auf das Bindeglied des Schichtenmodells, auf die mittlere Schicht. Diese befasst sich mit den Schnittstellen zwischen Werkzeugen und dem Benutzer. Es gilt dabei u.a. Fragen zu klären, die die Definition von Ein- bzw. Ausgabeformaten betreffen,

[7] Erhältlich unter `http://www.collocations.de`

[8] Siehe `http://wortschatz.uni-leipzig.de/Webservices/`

[9] Innerhalb des D-SPIN/CLARIN-Projektes gibt es Spezialisten, die sich intensiv mit den genannten Problemstellungen befassen.

welche einerseits mit den Web Services vollständig kompatibel sein sollen, andererseits aber auch mächtig genug sein müssen, die linguistisch annotierten Zwischenergebnisse adäquat zu beschreiben.

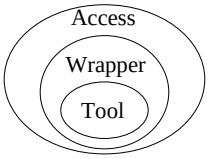


Abbildung 2: Schichtenmodell der Aufgabenteilung.

4.1 Stand der Arbeiten

Bei der Implementierung des vorgestellten Beispielszenarios handelt es sich um laufende Arbeiten, die zum Zeitpunkt der Abfassung des vorliegenden Papiers noch nicht vollständig abgeschlossen sind. Dieser Abschnitt enthält daher eine Momentaufnahme, in der wir auf Gegebenheiten und Ergebnisse auf dem Stand von Januar 2009 eingehen.

4.1.1 Technische Aspekte

Auf Grundlage der REST-Architektur (Richardson/Ruby 2007) entwickeln wir einen Web Service, der die Funktionalität einer Pipeline aus den oben genannten Werkzeugen realisiert. Wir gehen von einem übergeordneten Web Service aus, der zur Laufzeit mehrere Komponenten-Web Services (möglichst an die Module der Pipeline angepasst) aufruft. Diese Teil-Services werden in die Pipeline von Abbildung 1 eingebaut, und in Abbildung 3 (mithilfe gestrichelter Linien) skizziert.

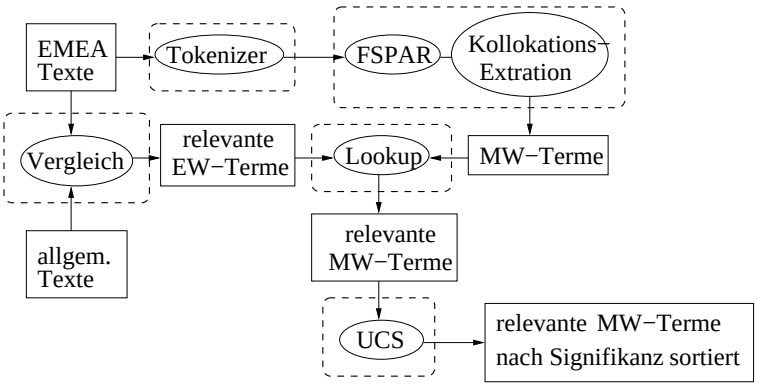


Abbildung 3: Prozessierungspipeline mit angedeuteten Webservice-Komponenten.

Als Eingabe soll der übergeordnete Service einen fachsprachlichen Text (im aktuellen Fall aus der EMEA-Domäne) und einen allgemeinsprachlichen Referenztext entgegennehmen, dann intern die Teil-Services anstoßen und schließlich als Ausgabe eine Liste mit terminologisch relevanten Wortkookkurrenzpaaren produzieren, die nach Kollokationstyp getrennt und jeweils nach Signifikanz (LogLikelihood) absteigend sortiert sind.

Problematisch sind hierbei vor allem die beiden Teil-Services “Vergleich” und “Lookup”, da diese jeweils zwei verschiedene Eingabedateien erwarten. Der Service “Vergleich” ist vor allem insofern schwierig, als

dass er vom Benutzer zwei Dateien übergeben bekommt. Eine mögliche Alternative wäre hier, den Benutzer zwischen hinterlegten allgemeinsprachlichen Texten wählen zu lassen. Beim Service “Lookup” ergibt sich zudem die Schwierigkeit, daß die beiden Eingabedateien erst produziert werden müssen (mit potentiell unterschiedlich langen Produktionszeiten) und dieser Service erst gestartet werden kann, wenn beide Dateien zur Verarbeitung vorliegen.

Unser Webservice kann in seiner momentanen Implementierung nur jeweils eine Eingabedatei bearbeiten; daher wurde zunächst nur ein Teil der Pipeline (tatsächlich) realisiert. Es handelt sich dabei um denjenigen Teil, in dem es um die Extraktion von signifikanten Mehrwortausdrücken aus einem gegebenen Text geht (Vgl. Abbildung 4 unten).

Das in Abschnitt 2 gegebene Szenarium können wir dennoch aufrecht erhalten, da die statistischen Signifikanzmaße meist ohnehin domänenspezifische Kollokationen hoch bewerten, auch ohne dass der Eingabetext vorher mit einem allgemeinsprachlichen Referenztext abgeglichen wurde (siehe hierzu auch die Beispiele aus Tabelle 2, die sich ohne Abgleich genauso verändert hätten.).

4.1.2 Implementierung

Zu Testzwecken wurde zunächst ein auf Python basierender Webserver implementiert. Dieser ermöglicht die Realisierung der Grundfunktionalitäten der Pipeline ohne auf HTTP-Anforderungen eingehen zu müssen. Der aktuelle Stand der Implementierung ist in Abbildung 4 angedeutet. Dort sind die realisierten Teile in schwarzen Linien dargestellt, während die noch nicht implementierten Teile der Architektur aus Abbildung 3 grau gedruckt sind.

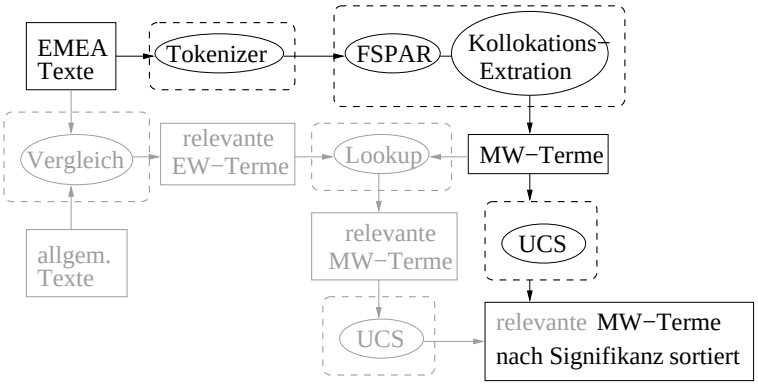


Abbildung 4: Prozessierungspipeline mit bisher realisierten Webservice-Komponenten.

In Kürze soll die Pipeline jedoch auf eine Kombination von Python und einem Apache-Webserver umgestellt werden, damit der Web Service (aus technischer Sicht[10]) auch extern zur Verfügung gestellt werden kann. Diese Umstellung ist für zwei Komponenten der Pipeline (“Tokenizer” und “UCS”) bereits realisiert.

Die Modifizierbarkeit der Pipeline durch den Benutzer ist im momentanen Aufbau noch eingeschränkt: im vorliegenden Szenario (Vgl. Abschnitt 2) gehen wir von einem computerlinguistischen Laien aus, der über die Zwischenschritte und auch die Zwischenergebnisse der Pipeline-Komponenten keine Auskunft erhält. In Zukunft soll dem Benutzer jedoch die Möglichkeit gegeben werden, die Zusammensetzung der Pipeline und auch die Präsentation der Ergebnisse beeinflussen zu können (Vgl. hierzu Abschnitt 4.2.1).

[10] Auf Fragen bezüglich Authentifizierung und Zugriffsberechtigungen gehen wir hier nicht gesondert ein.

Momentan ist es neben einer Anfrage an den übergeordneten Service möglich, auch Anfragen an die drei implementierten Teil-Services (nämlich “Tokenizer”, “FSPAR+Kollokationsextraktion” und “UCS”) zu stellen, jedoch ohne die Möglichkeit, Komponenten durch gleichwertige Werkzeuge (die sich in einer verteilten Ressourcenlandschaft u.U. an verschiedenen Orten befinden) auszutauschen. Hierfür müssen in Zukunft vor allem noch Fragen des Eingabe- und Ausgabeformats sowie der Konvertierung in derartige Formate geklärt werden. Ein erster möglicher Ansatz in dieser Richtung wird in Abschnit 4.2.2 beschrieben.

Da der auf einer Dependenz-Grammatik basierende Parser einen für den computerlinguistischen Laien eher unverständlichen Output liefert, kann dieser Service im Moment nur in Kombination mit anschließender Kollokationsextraktion aufgerufen werden. Aus rein technischer Sicht ist es jedoch auch problemlos möglich, dem Benutzer die Parser-Ausgabe direkt abzuliefern.

4.1.3 Ergebnisse

Die Pipeline greift unter anderem auf statistische Assoziationsmaße (in UCS: LogLikelihood) zurück, daher ist es notwendig, eine ausreichend große Textmenge (mindestens 5-10 Millionen Wörter) in die Prozessierungspipeline einzuspeisen. Das EMEA-Korpus, welches in der Beispiel-Implementierung als Testkorpus verwendet wurde, besteht aus etwa 10 Millionen Wortformen (das entspricht im aktuellen Fall 67MB in ASCII-Format) und lässt sich von sämtlichen Komponenten problemlos verarbeiten. Insgesamt dauert ein vollständiger Durchlauf durch die Pipeline (auf einem Dual AMD Opteron Rechner mit 2 x 2,2 GHz und 16GB RAM) etwa 35 Minuten, wovon die meiste Zeit (etwa 29 Minuten) auf das Parsing entfällt.

Bevor unser Web Service nach außen hin bereitgestellt wird, müsste noch untersucht werden, ob es eine Obergrenze an Daten gibt, die die einzelnen Komponenten maximal verarbeiten können (in Bezug auf Speicherbedarf und Rechenzeitkapazität). Außerdem bleibt noch zu klären, wie mit simultanen Anfragen umgegangen wird.

Die Ausgaberoutine liegt momentan in zwei verschiedenen Varianten vor. Zum einen hat der Benutzer die Möglichkeit, die Ergebnisse seiner Anfrage auf dem Bildschirm angezeigt zu bekommen (Standard Output), zum anderen kann der Web Service den Benutzer über einen eindeutigen Link zum Download der Ergebnisdatei führen, sobald diese fertig gestellt ist. Obwohl untypisch für einen Webservice, wäre in Anbetracht der vom Umfang der Anfrage abhängigen Laufzeiten das Zusenden einer E-Mail mit entsprechendem Download-Link durchaus eine denkbare Alternative.

4.2 Geplante Arbeiten

Die aktuelle Implementierung stellt einen im Vorhinein festgelegten Durchgang durch die Pipeline der Kollokationsextraktion dar. Der Benutzer kann nur zu Beginn in die Prozedur eingreifen: durch Auswahl des zu bearbeitenden Textes. In Anbetracht verschiedener Nutzergruppen (bspw. Linguisten, Fachleute der Historiker) erscheint es jedoch sinnvoll, die Pipeline weiter zu modularisieren, damit der Benutzer sie optimal an seine Vorkenntnisse und Bedürfnisse anpassen kann. Dazu müssen zum einen Interaktionspunkte zum Service definiert werden, an welchen der Benutzer die weitere Verarbeitung (u.U. basierend auf Zwischenergebnissen, die er bis dahin erhalten hat) beeinflussen kann, zum anderen müssen Formate gefunden werden, die ohne Informationsverlust die Kombination verschiedener Werkzeuge ermöglichen.

4.2.1 Interaktivität

In Zukunft soll an verschiedenen Stellen in die Prozedur eingegriffen werden können. Die Definition eines geeigneten Web Interfaces für die Interaktion des Benutzers mit den Webservices ist dabei unumgänglich. Die Interaktion selbst könnte dann über eine übergeordnete Web Application stattfinden, die Eingaben des Benutzers entgegennimmt und daraufhin Web Services aufruft.

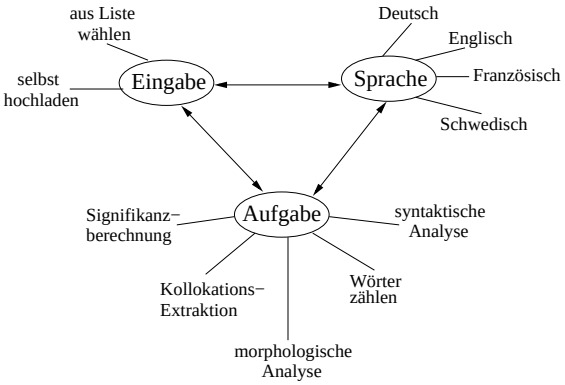


Abbildung 5: Auswahlparameter vor Beginn des eigentlichen Web Services.

Bevor eine Verarbeitungspipeline gestartet wird, müssen im Dialog mit dem Benutzer zunächst grundlegende Fragen, etwa nach der Sprache des zu verarbeitenden Textes, nach der Art der Übergabe und auch nach der Verarbeitung der Eingabe geklärt werden. Diese drei Parameter sind stark voneinander abhängig (Vgl. hierzu auch Abbildung 5), da bspw. nicht für jede Sprache die selbe Anzahl an Werkzeugen (und damit Bearbeitungsmöglichkeiten) gegeben sind. Weiterhin ist von der Wahl der Aufgabe, die der Web Service bearbeiten soll, das Eingabeformat abhängig (z.B. Tokenisieren, Parsing: Text, Signifikanzberechnung: Wortpaare).

Einem Benutzer mit Vorkenntnissen in der linguistischen Datenverarbeitung soll ermöglicht werden, für die einzelnen Verarbeitungsschritte der von ihm ausgewählten Pipeline selbst zu bestimmen, welche Werkzeuge verwendet werden sollen (z.B. Wahl des Formalimus bzw. der Technologie, die für das Parsing benutzt wird). Außerdem soll der Benutzer (auf Wunsch) Zwischenergebnisse der einzelnen Komponenten erhalten können. Hierfür müssen für jedes Modul der Pipeline Interaktionspunkte gesetzt werden, an welchen der Benutzer in den laufenden Prozess eingreifen kann (Vgl. Graphik in Abbildung 6). Um eine derartige Modularität der Pipeline-Komponenten zu realisieren, ist es unumgänglich, für alle (alternativen) Verarbeitungskomponenten nutzbare Ein- und Ausgabeformate zu entwickeln (Vgl. hierzu auch Abschnitt 4.2.2).

Neben einem Einfluss auf die Zusammensetzung der Prozessierungspipeline sollte der Benutzer zudem auswählen können, in welchem Format ihm die Ergebnisse präsentiert werden sollen: z.B. eine Sortierung der Kollokationskandidaten nicht nur absteigend nach Signifikanz, sondern u.U. auch alphabetisch nach Substantiven oder nach den Kollokatoren. Manche Benutzer möchten vielleicht auch Beispielsätze zu den Kollokationen sehen, oder eine Verteilung über Singular und Plural (d.h. eventuelle morphosyntaktische Präferenzen)[11]. Abbildung 6 zeigt beispielhaft, an welchen Stellen der Benutzer in die Pipeline eingreifen könnte, und was für Auswahlmöglichkeiten er dabei hat. Da diese Pipeline nicht schon zu Beginn festgelegt ist, könnten optionale Komponenten (wie etwa die Signifikanzberechnung, in Abb. 6 gestrichelt dargestellt) auch weggelassen werden.

4.2.2 Eingabe und Ausgabe-Formate

Im folgenden Abschnitt beschreiben wir einige Vorschläge für geeignete Ein- und Ausgabeformate in XML, wie man sie in einer modular aufgebauten Pipeline, bei der jede Komponente als ein untergeordneter Web Service realisiert ist, verwenden könnte. Die vorgestellten Formate sind von den in unserer Studie verwende-

[11] Die Ausgabe von Beispielsätzen und morphosyntaktischen Informationen ist in der aktuellen Pipeline noch nicht vorgesehen, könnte aber bereitgestellt werden, indem eine weitere Extraktionskomponente auf der Grundlage des geparsten Textes hinzugefügt wird.

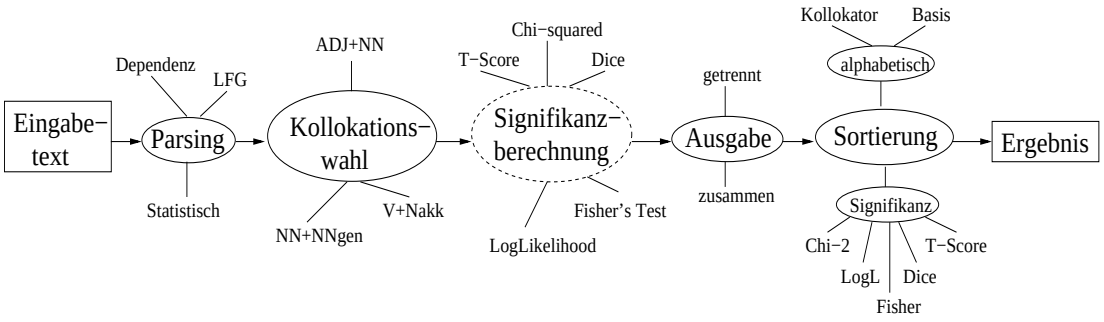


Abbildung 6: Pipeline zur Kollokationsextraktion inklusive Auswahlmöglichkeiten einzelner Module.

ten Werkzeugen inspiriert, sollten aber an Anforderungen anderer Werkzeuge angepasst werden können[12]. In der oben skizzierten aktuellen Implementierung unserer Pipeline (Vgl. Abschnitt 4.1.2) haben wir die Formatvorschläge noch nicht berücksichtigt. Das Format wird anhand eines Beispielsatzes aus dem EMEA-Korpus illustriert: *"Die zweite Studie lieferte ähnliche Ergebnisse."*. Aus Gründen der Übersichtlichkeit werden die XML-Strukturen sukzessive skizziert und vorgestellt.

Allgemeine Struktur Nach einem Durchlauf der oben vorgestellten Pipeline zur Extraktion von Kollokationen soll nicht nur eine Liste der signifikanten Kollokationen selbst zurückgegeben werden, es soll vielmehr alle Information, die während des Durchlaufs gesammelt wurde, abgespeichert werden (bspw. Zwischenergebnisse von einzelnen Komponenten). Ein Ausgabedokument, das als Ergebnis aus der Verarbeitungspipeline zurückgeliefert wird, könnte daher mit der in Abbildung 7 gegebenen Spezifikation beginnen, die u.a. Informationen über den Analysierungsgrad des eingegebenen Ursprungstextes, sowie über diesen Text selbst enthält und damit implizit die durchlaufenen Komponenten widerspiegelt (z.B. in den Belegungen der Attribute "parsing" und "mweExtraction").

```
<metadata>
  <source>IMS, Universität Stuttgart</source>
</metadata>
  <TextCorpus language=''de'' encoding=''utf8'' tokenisation=''yes'' POStagging=''STTS''
     lemmatisation=''yes'' parsing=''FSPAR'' mweExtraction=''yes''
     source=''ftp://www.ims.uni-stuttgart.de/pub/D-Spin/emea_fsparsed.xml''>
  <text>
     Die<B/>zweite<B/>Studie<B/>lieferte<B/>ähnliche<B/>Ergebnisse.<NL/>
  </text>
```

Abbildung 7: Vorschlag für den XML-Header der Ausgabedatei mit Repräsentation des eingegebenen Textes.

Repräsentationen auf Wortebene Die Ausgabe des Parsers enthält neben syntaktischen Relationen auch Informationen zu Wortgrenzen (Tokenisierung), zu den Grundformen der Wörter (Lemmatisierung) und zu den Wortarten (Tagging). Die entsprechenden Repräsentationen[13] dieser Informationen sind in Abbil-

[12] Im Rahmen des D-SPIN Projektes sollen einheitliche Formate entwickelt werden, die so allgemein wie möglich gehalten sind, damit sie für die Ein- und Ausgabe von möglichst vielen Werkzeugen verwendbar sind, und damit die Werkzeuge untereinander dadurch einen möglichst hohen Grad an Kompatibilität erreichen. Zum Zeitpunkt der Fertigstellung dieses Papiers waren in der Diskussion der Projektteilnehmer über das Format noch keine endgültigen Festlegungen getroffen.

[13] Hier zugunsten der Übersichtlichkeit leicht abgekürzt.

```
<tokens>
   <token id=''t1'' start=''1'' end=''3''>Die</token>
   <token id=''t2'' start=''5'' end=''11''>zweite</token>
   <token id=''t3'' start=''13'' end=''19''>Studie</token>
   ...
   <token id=''t7'' start=''52'' end=''52''>.</token>
</tokens>
<sentences>
   <sentence id=''s1'' start=''1'' end=''52''/>
</sentences>
<POStags>
   <tag tokID=''t1'' cat=''ART''/>
   <tag tokID=''t2'' cat=''ADJA''/>
   <tag tokID=''t3'' cat=''NN''/>
   ...
   <tag tokID=''t7'' cat=''$.''/>
</POStags>
<lemmas>
   <lemma tokID=''t1''>d</lemma>
   <lemma tokID=''t2''>2.</lemma>
   <lemma tokID=''t3''>Studie</lemma>
   ...
   <lemma tokID=''t7''>.</lemma>
</lemmas>
```

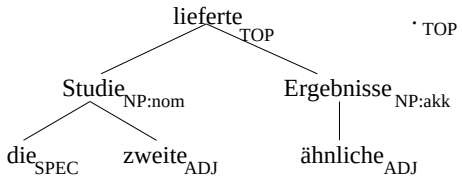
Abbildung 8: XML-Formatvorschlag für Tokenisierung, Tagging und Lemmatisierung.

dung 8 dargestellt. Zunächst wird jedes Wort (Token) mit einer Identifikationsnummer (TOKID) versehen. Außerdem wird festgehalten, aus welchen Zeichen (*"characters"*) ein Wort besteht (Vgl. "start"- und "end"-Attribute in Abb. 8). Bei der Darstellung von Wortarten (in Form von POS-Tags) und auch bei der Lemmatisierung wird auf diese TOKIDs referenziert.

Repräsentation auf Satzebene Das syntaktische Analyse stellt die komplexeste darzustellende Komponente dar. Die in Abbildung 9(a) gezeigte XML-Struktur enthält nur die wichtigsten syntaktischen Informationen, nämlich die Abhängigkeiten (Dependenzen) der Satzglieder untereinander[14].

```
<depparsing>
   <dependency function=''SPEC'' govID=''t3'' depID=''t1''/>
   <dependency function=''ADJ'' govID=''t3'' depID=''t2''/>
   <dependency function=''NP:nom'' govID=''t4'' depID=''t3''/>
   <dependency function=''ADJ'' govID-=''t6'' depID=''t5''/>
   <dependency function=''NP:akk'' govID=''t6'' depID=''t6''/>
<depparsing>
```

(a) XML-Format



(b) Baumrepräsentation

Abbildung 9: Repräsentation der syntaktischen Analyse (Dependenzstruktur).

Der XML-Code für dieses Format kann problemlos automatisch aus der Parsing-Ausgabe generiert werden, jedoch enthält die Ausgabe des verwendeten Parsers einige zusätzliche Informationen, die in diesem Format

[14] Zur Verdeutlichung der Dependenz-Struktur ist daneben in Abbildung 9(b) eine Baumrepräsentation des Satzes gegeben.

verloren gehen. Diese Art von Information sollte in einer zukünftigen (erweiterten) Version des Formates jedoch unbedingt berücksichtigt werden. In Zukunft muss noch untersucht werden, inwiefern die syntaktischen Repräsentationen in diesem XML-Format von der Art des Parsings abhängig sind (z.B. Dependenzparsing vs. Konstituentenstruktur-Parsing) und ob es Möglichkeiten gibt, die Ergebnisse unterschiedlicher Parser (in Form von XML-Repräsentationen) ineinander zu überführen.

Repräsentation von Wortkookkurrenzen Für die signifikanten Mehrwortterme, die das Ergebnis unserer Extraktionspipeline darstellen, soll ebenfalls eine geeignete XML-Struktur gefunden werden. Abbildung 10 zeigt ein mögliches XML-Format für Mehrwortterme ("mwe" = *Multiword Expressions*), in dem die errechneten Signifikanzmaße (hier: "loglik") und die absolute Frequenz ("absfreq") der Terme mitberücksichtigt sind (siehe auch Abb. 10(b) für eine Darstellung im Tabellenformat).

```
<mwes>
   <mwe type='v_nobj'>
      <k1 refid='16'/>
      <k2 refid='14'/>
      <loglik value='22.82'/>
      <absfreq value='7'/>
   </mwe>
   <mwe type='adj_nn'>
      <k1 refid='12'/>
      <k2 refid='13'/>
      <loglik value='273.44'/>
      <absfreq value='183'/>
   </mwe>
   <mwe type='adj_nn'>
      <k1 refid='15'/>
      <k2 refid='16'/>
      <loglik value='357.20'/>
      <absfreq value='64'/>
   </mwe>
</mwes>
```

(a) XML-Format

Nomen	Verb	Frqz	LogL
Ergebnis	liefern	7	22.82

Adjektiv	Nomen	Frqz	LogL
zweite	Studie	183	273.44
ähnlich	Ergebnis	64	357.20

(b) Tabellen-Format

Abbildung 10: Repräsentation von Kollokationen.

5 Zusammenfassung

Das vorliegende Papier ist eine Momentaufnahme unserer laufenden Arbeiten zur Bereitstellung computerlinguistischer Werkzeuge und Prozessierungspipelines durch Web Services. Anhand eines Beispielszenarios wurde eine übliche Verarbeitungspipeline vorgestellt und teilweise realisiert. Dabei sind wir auf einige grundlegenden Fragen gestoßen, die es in Zukunft noch zu untersuchen gilt. Ein zentrales Problem bei der Einbindung von NLP-Werkzeugen in Web Services stellen die für Web Services unüblichen langen Antwortzeiten der Werkzeuge dar (Vgl. Abschnitt 4.1.3, FSPAR: 30 Minuten für 10 Millionen Wortformen). Weiterhin sind die (Zwischen-) Ergebnisse z.T. sehr umfangreich (die FSPAR-Ausgabe eines Textes mit etwa 10 Millionen Wortformen hat ein Volumen von 414MB) und sollten daher in Zukunft komprimiert an den Benutzer zurückgegeben werden. In Bezug auf Möglichkeiten der Benutzerinteraktion mit dem Web Service gibt es ebenfalls noch einige offenen Fragen zu klären. Damit der Benutzer an möglichst vielen Stellen in den Verarbeitungsprozess eingreifen kann, müssen die einzelnen Komponenten einer NLP-Pipeline jedoch zunächst stärker modularisiert werden. Wir werden uns in Zukunft verstärkt auf die Definition geeigneter Ein- und Ausgabeformate konzentrieren um eine solche Modularisierung zu ermöglichen.

Literatur

[Ahmad et al. 1992] Khurshid Ahmad, Andrea Davies, Heather Fulford and Margaret Rogers (1992): "What is a term? The semi-automatic extraction of terms from text", in: Mary Snell-Hornby et al.: *Translation Studies – an interdiscipline*, John Benjamins Publishing Company (Amsterdam/Philadelphia)

[Bartsch 2004] Sabine Bartsch (2004): "Structural and functional properties of collocations in English", in: A corpus study of lexical and pragmatic constraints on lexical co-occurence, Tübingen, Narrr

[Dunning 1993] Ted Dunning (1993): "Accurate Methods for the Statistics of Surprise and Coincidence", in: Computational Linguistics, 19/1.

[Evert 2004] tefan Evert (2004): "The Statistical Analysis of Morphosyntactic Distributions", in: *Proceedings of LREC 2004 Lisbon*, Portugal, 2004, pp. 1539-1542

[Evert 2005] Stefan Evert (2005): "The Statistics of Word Cooccurrences: Word Pairs and Collocations", PhD. Thesis, Universität Stuttgart.

[Hausmann 2004] Franz Josef Hausmann (2004): "Was sind eigentlich Kollokationen?", in: Wortverbindungen – mehr oder weniger fest, DeGruyter, Berlin

[Heid et al. 2008] Ulrich Heid, Fabienne Fritzinger, Susanne Hauptmann, Julia Weidenkaff and Marion Weller (2008): "Providing corpus data for a dictionary of German juridical phraseology", in: Angelika Storrer, Alexander Geyken, Alexander Siebert and Kay-Michael Würzner: *Text Resources and Lexical Knowledge* (= Proceedings of the 9th Conference on Natural Language Processing, KONVENS 2008).

[Heid/Weller 2008] Ulrich Heid and Marion Weller (2008): "Tools for Collocation Extraction: Preferences for Active vs. Passive", in: *Proceedings of LREC-2008, Marrakesh, Morocco*

[Ivanova et al. 2008] Kremena Ivanova, Ulrich Heid, Sabine Schulte im Walde, Adam Kilgarriff and Jan Pomikálek (2008): "Evaluating a German Sketch Grammar: A Case Study on Noun Phrase Case", in: *Proceedings of LREC-2008, Linguistic Resources and Evaluation Conference, Marrakesh, Morocco*,

[Kilgarriff et al. 2004] Adam Kilgarriff, Pavel Rychlý, Pavel Smrz and David Tugwell (2004): "The Sketch Engine", in: Proceedings of EURALEX-2004, Lorient, France

[Richardson/Ruby 2007] Leonard Richardson and Sam Ruby (2007): "RESTful Web Services", O'Reilly

[Schiehlen 2003] Michael Schiehlen (2003): "A Cascaded Finite-State Parser for German", in: Proceedings of the Research Note Sessions of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03), Budapest, April, 2003.

[Schmid et al. 2004] Helmut Schmid, Arne Fitschen and Ulrich Heid (2004): "A German Computational Morphology Covering Derivation, Composition and Inflection", in: Proceedings of the IVth International Conference on Language Resources and Evaluation (LREC 2004).