Projektgruppe

Intelligence Service

PG Bericht

24. September 2008

Veranstalter:
Lehrstuhl 8, Universität Dortmund

Betreuer:
Prof. Dr. Katharina Morik
Dipl.-Inform. Felix Jungermann

Teilnehmer:
Baumann, Björn
Böhmer, Martin
Firstein, Roman
Fritsch, Regina
Günal, Emel
Güner, Mustafa
Kaz, Erkan
Koloch, Rafael
Kubatz, Marius
Viefhues, Alexander
Zhu, Qingchui

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Einführung

1.1 Aufgabenstellung

Ziel der PG ist das automatische Erstellen eines Pressespiegels für eine bestimmte Person (z.B. einen Politiker) oder eine bestimmte Firma aus dem Internet bzw. aus Datenbanken. Daraus sollen dann gezielt Antworten auf bestimmte Fragen extrahiert werden. Methoden zu einem solchen Intelligence Service werden untersucht und implementiert. Allerdings ist das Spektrum der Informationen für eine einzige Anfrage hierbei zu gross. Das Problem ist, die interessanten Daten zwischen den uninteressanten Daten herauszufinden. Dies ist das Problem des Information Retrieval. Der zu entwickelnde Intelligence Service soll natürlich über das Information Retrieval von Suchmaschinen hinausgehen. Das grundsätzliches Problem ist, dass Suchmaschinen nicht konkrete Antworten liefern. Vielmehr wird eine Auswahl an Dokumenten geliefert, die die Antwort zu gestellten Anfrage höchstwahrscheinlich enthält. Was man aber oft möchte, ist auf eine Frage wie:

Welcher Bundeskanzler stellte als letztes das Misstrauensvotum?
Antwort: Gerhard Schröder (zusammen mit der URL, auf der die Information gefunden wurde, zu erhalten.)

Für solche Fragebeantwortung muss man nicht nur die relevanten Dokumente finden, sondern auch die relevanten Passagen,dies ist ein weiterer Punkt, der von Suchmaschinen nicht erbracht wird. Wenn die Dokumente durch eine Auszeichnungssprache (XML) annotiert sind, ist die Suche in den relevanten Dokumenten erleichtert, so dass gezielt etwa nach Investitionen, Erfolgen, neuen Produkten, Börsenzahlen gesucht werden kann. Die meisten Dokumente sind aber nicht annotiert. Man muss also algorithmisch nach Entitäten eines bestimmten Typs (z.B. Person, Ort, Firma) suchen. Das Gebiet, das sich mit der Erkennung der Entitäten eines inhaltlichen Typs in Texten befasst, ist die Named Entity Recognition (NER) und verwendet statistische Verfahren und solche des maschinellen Lernens bzw. Data Mining. Somit ist die NER ein weiterer Bereich, mit dem sich die PG befassen muss. Die Abfolge von Anfragen sollte jedoch automatisiert erfolgen, um ein allgemein nutzbares System zu schaffen. Für Politiker bietet sich hierfür beispielsweise die Internetseite Bundestag.de an. Hier sind zu jedem Abgeordneten die jeweiligen Biographien hinterlegt. Zusätzlich zu diesen offensichtlichen Daten kann man jedoch auch noch die digital vorliegenden Drucksachen (z.B. Anträge) und Protokolle verarbeiten. Nach durchgeführter NER über diesen Dokumenten sollen dann konkrete Fragen beantwortet werden.

1.2 Methoden der Named Entity Recognition

1.2.1 Übersicht

Definition und Zielsetzung

Zielsetzung der Named Entity Recognition ist es, bestimmte Bestandteile eines natürlichsprachlich verfassten Text, also Zeitungsartikel, Dossiers oder auch E-Mail, zu erkennen und einer vorgegebenen Kategorie zuzuordnen. Diese Bestandteile nennt man auch **Named Entities**. Die Named Entities dienen in der Regel dazu wichtige Fragen, die an einen natürlichsprachlichen Text gestellt werden, zu beantworten. Was? Wann? Wo? und Wer? sind dabei typische Fragen, die mit der Named Entity Recognition geklärt werden können.
Die Besonderheit dieser Named Entities ist ihre Einmaligkeit, denn üblicherweise handelt es sich dabei um eine Person, Organisation oder einen Ort. Angaben also, die in der Regel definit sind und nur einmal vorkommen. Beispielsweise könnte dem Begriff *Angela Merkel* eindeutig die Klasse Person zugewiesen werden - es handelt sich also um ein Named Entity oder auch Eigenname, während der Begriff *Bundeskanzler/in* nicht auf eine bestimmte Person hinweist, somit also nicht klassifiziert wird. Wichtig ist also, dass nur relevante Informationen klassifiziert werden, wobei sich die semantische Relevanz auf die Domäne des Texts bezieht. Die Relevanz kann durch die Auswahl passender Trainingsdaten sowie den Tags bestimmt werden. Neben den oben aufgeführten Kategorien Person, Organisation und Ort gehören auch Zeitangaben sowie quantitative Aussagen zu den möglichen Named Entities.
Hier nun ein kurzes Beispiel für eine mögliche Ein- und Ausgabe eines NER-Systems:

Eingabe: Auch die widersprüchlichen Angaben darüber, wie viel Geld Bohlen tatsächlich am 11. Dezember 2006 gestohlen wurde, wollte das Landgericht Bochum klären.

Ausgabe: Auch die widersprüchlichen Angaben darüber,
wie viel Geld <**Person**>**Bohlen**</**Person**> tatsächlich am <**Datum**> **11. Dezember 2006** </**Datum**> gestohlen wurde, wollte das <**Organisation**> **Landgericht Bochum** </**Organisation**> klären.

Geschichte der NER

Named Entity Recognition wurde durch die von der DARPA instituierten Message Understanding Conferences (kurz: MUC) bekannt. Die Message Understanding Conferences fanden erstmals 1987 statt und hatten die Entwicklung besserer Information Extraction-Methoden zum Ziel. Dabei gibt es eine Reihe von Teams, die alle an einem genau definierten Ziel, unter Vorgabe der zu untersuchunden Texte, arbeiten. Die einzelnen Ergebnisse der Gruppen werden an den Konferenztagen vorgestellt und ermöglichen so einen umfassenden Blick über den aktuellen Stand der Technik.
In diesem Zusammenhang wurde Named Entity Recognition / Koreferenz erstmals 1995 in der sechsten MUC als Ziel definiert. Das Forschungsgebiet ist in der Informatik also relativ neu.

Evaluation

Um gute Programme zu entwickeln, benötigt man ein Scoring-verfahren um die Qualität des Programms bemessen zu können. In der Named Entity Recognition haben sich dabei drei

immer wieder verwendete Evaluationsmaße etabliert. Alle Werte gehen davon aus, dass wir den gesamten Korpus einer Datei oder eines Texts nach Named Entities durchsuchen.

Precision Unter Precision, oder zu deutsch Präzision, versteht man die Anzahl der korrekt klassifizierten Named Entities im Verhältnis zu der Anzahl der vom Recognizer gefundenen Named Entities.

$$Precision = \frac{Anzahlkorrektklassifizierter NEs}{Anzahl NEs gefunden} \tag{1.1}$$

Recall Recall ist die Ausbeute an Named Entities aus dem gesamten Dokument. Hier wird also das Verhältnis der Anzahl an korrekt klassifizierten Named Entities zu den insgesamt im Text vorhandenen Named Entities betrachtet.

$$Recall = \frac{Anzahlkorrektklassifizierter NEs}{Anzahlvorhandener NEs im Korpus} \tag{1.2}$$

F-Measure Um beide Größen, also Precision und Recall, zusammen betrachten zu können, bildet man den ungewichteten harmonischen Mittelwert beider Werte und erhält den so genannten F-Measure [Rij79].

$$F - Measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{1.3}$$

Interne / Externe Evidenz

Jedes Merkmal zur Klassifikation eines Named Entity kann ganz grob in Interne und Externe Evidenz aufgeteilt werden. So kann jedes Analyseverfahren in [1.2.1] eine der beiden Gruppen aufgeteilt werden.

Interne Evidenz Verfahren der Internen Evidenz ziehen ihr Wissen nur aus dem betrachteten Wort. Dies kann ein Eintrag in einem Lexikon (zum gesuchten Wort) sein [1.2.1], bestimmte Bestandteile eines Wortes oder auch die Großschreibung einzelner Buchstaben. Wenn wir beispielsweise das Wort DARPA betrachten, ist es wahrscheinlich, dass es sich hierbei um eine Organisation handelt, da alle Buchstaben groß geschrieben sind, folglich es sich um eine Abkürzung handeln könnte.

Externe Evidenz Viele Verfahren der Internen Evidenz gelangen rasch an ihre Grenzen. So unterscheidet sich rein syntaktisch eine Bank mit der Bedeutung einer Finanzeinrichtung nicht mit der einer (Sitz-) Bank. Auch ein Lexikon würde an dieser Stelle nur viele Möglichkeiten anzeigen, aber kein Aufschluss darüber geben, worum es sich nun wirklich handelt.

Hier muss man weitergehen und den Kontext des Wortes betrachten. Der Kontext kann einerseits der umschlossene Satz, aber auch ein ganzer Textabsatz darstellen. So ist der Kontext *Die französische Stadt...* ein eindeutiger Hinweis auf ein Named Entity der Kategorie *Ort.*Zum Kontext gehören aber auch die vorher bestimmten Named Entities, die oftmals ebenfalls ein hilfreiches Indiz für eine mögliche Einordnung liefern.

Mögliche Analyseverfahren

Tokenisierung Bevor man einen Text bearbeiten und maschinell auswerten kann, muss man ihn in seine einzelnen Bestandteile, den Token, zerlegen. Diese Aufsplittung gestaltet sich bei den meisten Sprachen, beispielsweise englisch, deutsch oder spanisch, relativ einfach. Man sucht nach Leerzeichen, die üblicherweise jedes Wort abgrenzen und erhält damit einen einzelnen Token. Wichtig bei der Tokenisierung ist allerdings auch, dass dies nicht in allen Sprachen so einfach sein muss.
Im nächsten Schritt kann man anhand von Satzzeichen ('.', ',', '?') die Struktur des Texts erfassen und bekommt damit wichtige Informationen wenn es darum geht den Kontext eines Worts zu untersuchen. Ausruf- oder Fragezeichen kann man hierbei als deutlichen Hinweis für ein Satzende ansehen. Bei einem '.'kann man sich dagegen nicht immer sicher sein, dass es sich um das Ende eines Satzes handelt. Hier ist beispielsweise auch eine Abkürzung möglich (bzw. , e.V.,usw.) und man muss anhand des Kontexts bzw. des syntaktischen Aufbaus das Satzende bestimmen.

Morphologische Analyse Die morphologische Analyse bedient sich sprachlicher Mittel um die im ersten Schritt erfassten Token für die weitere Bearbeitung zu vereinfachen. Ein wichtiger Bestandteil der morphologischen Analyse ist das **Stemming-Verfahren**. Hierbei werden Verben in ihre Stammform überführt um die Anzahl der Regeln, die in den nachfolgenden Schritten zur Erkennung der Kategorie dienen, klein zu halten. So wird aus *'schläft' 'schlafen'* und aus *'sprach' 'sprechen'*. Auch die Anzahl der Lexikoneinträge kann mit diesem Verfahren reduziert werden.
Ganz ähnlich kann man sich die Motivation für die Präfix- und Suffix-Erkennung vorstellen. Auch hier geht es darum komplex zusammengesetzte Wörter zu vereinfachen und auf ihr Lemma zurückzuführen.
Wichtig zu erwähnen ist, dass dieser Schritt bei manchen Sprachen sehr wichtig sein kann. Im Deutschen gibt es eine Vielzahl von morphologischen Veränderungen eines Wortes. Viele verschiedene Zusammensetzungen eines Wortes sind denkbar und müssen hier berücksichtigt werden. Gleiches gilt beispielsweise auch in der französischen Sprache. Im Gegensatz dazu kann man bei der Betrachtung eines englischen Textes einen Großteil dieser Verfahren wegfallen lassen und allenfalls noch ein Stemming von Verben durchführen. Die maschinelle Erfassung der Semantik eines Textes kann also in den verschiedenen Sprachen deutlich variieren.

Lexikalische Analyse Die Lexikalische Analyse in der Named Entity Recognition ist ein einfacher LookUp in einem vorbereiteten Lexikon. In den meisten Fällen kann an dieser Stelle schon eine eindeutige Kategorie für das Named Entity gefunden werden. So wird man unter dem Stichwort *Angela Merkel* sicherlich nur eine mögliche Kategorie-Zuordnung zulassen, nämlich *Person*.
Da es sich hierbei allerdings um ein Verfahren der Internen Evidenz (1.2.1) handelt, sprich es wird nur die Struktur des Wortes / Token betrachtet und nicht der Kontext, können Mehrdeutigkeiten auftreten. Diese Mehrdeutigkeiten können nur aufgehoben werden, wenn man sich den kompletten Satz, bzw. den ganzen Abschnitt ansieht und nach möglichen Relationen zur gesuchten Entität Ausschau hält. Dies wird in der Syntaktischen Analyse gemacht.
Hier einige Beispiele für Mehrdeutigkeiten:

Wort	Mögliche Deutung
Essen	Stadt in NRW oder Mahlzeit?
Bank	Finanzeinrichtung oder Sitzmöbel?
Buchen	Baum oder Imperativ von *buchen*. Bsp.: **Buchen** Sie mir einen Flug!
Hamburger	Burger oder Einwohner von Hamburg?

Tabelle 1.1: Beispiele von Mehrdeutigkeiten

Syntaktische Analyse Um Mehrdeutigkeiten in der Struktur der Wörter aufzuheben, kommt man nicht umher den Kontext eines Wortes zu betrachten. Kleinräumig ist dies ein Satz, im größeren Maßstab kann man hier aber auch einen Textabschnitt betrachten. Bei der Betrachtung des Kontexts geht es darum den Satz in syntaktische Bestandteile zu zerlegen. Es geht dabei also um die Erkennung von Nomen, Verben, Präpositionen, usw. Auch bereits erkannte und klassifizierte Named Entities gehören zum Kontext und können hilfreich sein, beispielsweise wenn es darum geht zwei gleiche Referenzen auf ein Objekt aufzulösen. So steht die Abkürzung *IBM* für *International Business Machine*. Beides sind Named Entities, die aber die gleiche Firma beschreiben. Anhand der Abkürzung könnte man dies erkennnen.
Für die syntaktische Analyse eines Satzes werden in der Named Entity Recognition **Part-of-Speech Tagger** eingesetzt. Diese weisen jedem Wort eines Satzes eine Wortklasse zu. Anhand von Kontextinformationen können hier Mehrdeutigkeiten aufgehoben werden, wobei auch unbekannten Wortphrasen eine Wortklasse zugeteilt wird. TnT ist ein relativ bekannter POS-Tagger und klassifiziert englische Wörter mit bis zu 86% und deutsche mit 89% Wahrscheinlichkeit zur richtigen Wortgattung.
Eine Alternative zum POS-Tagger wäre ein **Full-Parsing**, wo die Analyse einer Satzkonstruktion über einen Parsebaum geschieht, der an kontextfreie Grammatiken angelehnt ist und Wortgattungen aufgrund von Position und Vorkommen im Satz ableitet. Allerdings ist dieses Verfahren sehr fehlerbehaftet und benötigt viel Rechenzeit, weswegen man es in der Regel nicht anwendet [AI99].

Domänenspezifische Analyse In der Domänenspezifischen Analyse geht es darum die Klassifizierung durch Einbezug des Text-Themas - daher Domänenspezifisch - zu verbessern. Hierbei kommt insbesondere die Koreferenz-Auflösung zum Einsatz. Die Aufgabe der **Koreferenz-Auflösung** ist die Erkennung von gleichen Referenzen innerhalb eines Texts. Falls also am Anfang eines Texts von G.W. Bush die Rede ist und in späteren Abschnitten nur noch ‘Er‘vorkommt, muss erkannt werden, dass es sich hierbei um die anfangs genannte Person handelt. Auch bei Organisationen/Firmen ist die Erkennung dieser Personalpronomen wichtig um den Inhalt des Texts verfolgen zu können. Auch temporale Referenzen müssen aufgelöst werden. So bedeutet die Phrase ‘um Viertel vor Vier ‘genausoviel wie 15:45 Uhr.
Beim **Merging** geht es hingegen darum gegebene Named Entities zu verschmelzen, falls es sich dabei um ein und dasselbe Objekt handelt. Hier einige Merging-Beispiele:

Systemarchitekturen

Listenbasierte Systeme Beim Entwurf einer Systemarchitektur für ein NER-System kann man sich ganz einfach eine riesige Liste mit allen Named Entities vorstellen, die es gibt.

Abkürzung	Referenz
IBM	International Business Machine
Deutsche Bahn AG	Die Bahn entlässt Mitarbeiter
USA	United States of America
Hamburger	Burger oder Einwohner von Hamburg?

Tabelle 1.2: Beispiele für Merging-Operationen

In dieser Liste müssten dann aber auch immer alle neuen Wortschöpfungen ergänzt, sowie morphologische Varianten eines Wortes abgespeichert werden. (aus [CB03])
Während diese beiden Punkte noch relativ gut realisierbar sind, führen die möglichen Mehrdeutigkeiten dazu, dass ein Listenbasiertes Verfahren keine gute Option für die Named Entity Recognition darstellt. Mann kann zwar nachschlagen, was ‘Essen‘alles bedeuten kann, aber ohne Betrachtung des Kontexts wird es nicht möglich sein, die passende Kategorie der Entität zu finden. Rein Listenbasierte Verfahren scheiden also aufgrund der zu hohen Fehlerrate aus.

Regelbasierte Systeme Bei Regelbasierten Systemen definiert man anhand der Merkmale der einzelnen Token Regeln, um das Einsortieren in Kategorien zu ermöglichen. Dabei nutzt man morphologisches, lexikalisches, syntaktisches und domänenspezifisches Wissen aus, also alle Verfahren, die zur Bestimmung von Named Entities in Frage kommen. Hat man eine Reihe von Regeln entwickelt kann man daraus eine Grammatik (kontextfrei) entwickeln, die in Texten nach Named Entities sucht.
Im Gegensatz zu lernbasierten Verfahren benötigt man hier allerdings spezielle Linguisten für die Entwicklung der einzelnen Regeln. Diese Regeln werden dann an relativ kleinen Trainings-Datensätzen getestet und falls erforderlich korrigiert und wieder getestet. Nach einigen Iterationen hat man so eine qualitativ gute Regelmenge zur Erkennung von Named Entities zusammen. Insgesamt ist die Entwicklung von Regelbasierten Verfahren zeitaufwändiger, da man hier sehr umfassende Grammatiken entwickeln muss. Und auch hinsichtlich der Flexibilität zeigen sich Regelbasierte Systeme eher nachteilig im Vergleich zu Lernbasierten Verfahren. Hier kann man die Grammatik in der Regel nicht schnell erweitern oder gar einige Teile verändern, da viele Regeln aufeinander aufbauen und transitiv abhängig sind. Auch die Anpassung an eine neue Domäne ist damit wesentlich schwieriger. (siehe [Fel03])

Nachfolgend einige Beispiele wie solche Regeln aussehen können:

- Aufeinanderfolgende Phrasen der Form ¡Wort¿¡Wort¿ GmbH deuten mit hoher Wahrscheinlichkeit auf eine Firma / Organisation hin
- Großgeschriebene Worte können Hinweise auf eine Firma oder Organisation sein: NASA, ADAC, UNICEF
- ‘denken‘ist, unabhängig vom Tempus der Verbform, in der dritten Person immer ein starker Hinweis für ein menschliches Subjekt
- Vorkommen von ‘-burg‘, ‘-dorf‘, ‘-stadt‘deuten auf eine Ortsangabe aus dem deutschsprachigen Raum hin

Lernende / Automatische Systeme Lernende oder Automatische Systeme nutzen statistische Verfahren oder Methoden des Maschinellen Lernens um in einem Textkorpus nach Named Entities zu suchen. Im Gegensatz zu Regelbasierten Verfahren werden dabei qualitativ und quantitativ hohe Anforderungen an Trainingstexte gestellt. Douglas E. Appelt und David J. Israel beschreiben in ihrem Paper [AI99], dass man für jede Verdopplung der Trainingsdatensätze den F-Measure um 1,5 Punkte steigern kann. Dabei ist die Auswahl der Trainingsdatensätze entscheiden für das spätere Verhalten des Recognizers und kann mitunter schwieriger sein als das Entwerfen von Regeln für ein Regelbasiertes System.

Zu den wichtigsten Verfahren des Maschinellen Lernens gehören Hidden Markov Models 1.2.2, Maximum Entropy Markov Models 1.2.3, Conditional Random Fields 1.2.4 und Support Vextor Machines 1.4.1. Jedes Verfahren wird mit den möglichen Erweiterungen weiter unten beschrieben.

Eine Gefahr bei Maschinellen Lernverfahren ist das so genannte **Overfitting**. Dabei wird dergleiche Trainingstext immer und immer wieder gelernt, so dass eine Überanpassung des Systems an den Datensatz vorkommen kann. Wird nun bei der Erkennung ein thematisch anderer Datensatz verwendet, ist eine hohe Fehlerrate bei der Klassifikation von Named Entities wahrscheinlich.

Vor- und Nachteile der Systemarchitekturen

Regelbasierte Systeme benötigen aufgrund der einfachen Erstellung eines Parsebaums relativ wenig Zeit zur Auswertung, wobei die Auswertungsergebnisse auf gleicher Höhe oder etwas besser im Vergleich zu Lernbasierten Systemen sind. So ergab die Auswertung eines Wall Street Journals in der siebten Message Understanding Conference einen Vorteil der Regelbasierten Systeme von 3,3 Prozentpunkten (93,7% vs. 90,4% gemessen an der Zahl der korrekt klassifizierten NE [Fel03]).

Lernende Systeme sind hingegen wesentlich flexibler, wenn es darum geht, ein System zur Laufzeit zu verändern. Während wir bei Regelbasierten Systemen viele Regeln einzeln anpassen müssen, genügt es die Parameter zu verändern und das System mit neuen Trainingstexten zu füttern. Diese Flexibilität zeigt sich auch hinsichtlich der Anpassung an verschiedene Sprachen.

1.2.2 Hidden Markov Model

Einleitung

Namensgeber dieses Modells ist sein Entwickler Andrei Andrejewitsch Markov (1856 - 1922), der wesentliche Beiträge zur Wahrscheinlichkeitstheorie und Analysis beisteuerte. Er berechnete 1913 die Buchstabensequenzen in russischer Literatur um die Notwendigkeit der Unabhängigkeit für das Gesetz der grossen Zahlen nachzuweisen. Die Berechnungen konnten zudem als Aussage über die Wohlgeformtheit der Orthographie von Buchstabenketten interpretiert werden. Aus diesem Ansatz entwickelte sich ein allgemeines statistisches Werkzeug, der sogenannte stochastische Markov-Prozess. Basierend auf dieser Methode findet eine Anwendung in der NER[1] statt.

[1] Named Entity Recognition

Markov-Prozess

Der Markov-Prozess wird allgemein synonym zur Markov-Kette verwendet, da sich die Verteilungsfunktion zu einem beliebigen, auch nicht diskretem Zeitpunkt bestimmen lässt. In der Computertechnik kommen die spezielleren Markov-Ketten zum Einsatz.

Markov-Kette

Eine Markov-Kette ist eine spezielle Klasse von stochastischen Prozessen. Man unterscheidet bei Markov-Ketten zwischen diskreter und stetiger Zeit. Markov-Ketten in stetiger Zeit werden meistens als Markov-Prozesse bezeichnet. Das System springt bei jedem Zeitschritt von einem Zustand in den Nächsten, dem zeitlichen Fortschreiten liegt eine Übergangswahrscheinlichkeit zugrunde. Das Besondere an einer Markov-Kette ist die Eigenschaft, dass durch Kenntnis einer begrenzten Vorgeschichte genauso gute Prognosen über die zukünftige Entwicklung möglich sind, wie bei einer Kenntnis der gesamten Vorgeschichte des Prozesses. Im Falle einer Markov-Kette erster Ordnung heißt das:
Die Zukunft des Systems hängt nur von der Gegenwart (dem aktuellen Zustand) und nicht von der Vergangenheit ab. Um diese Eigenschaft anschaulich zu beschreiben sagt man auch, dass eine Markov-Kette „gedächtnislos“ ist. Ausserdem besagt die Ordnungszahl einer Markov-Kette, von wie vielen vorherigen Zuständen der aktuelle Zustand abhängt. Bei den Markov-Ketten 1. Ordnung hängt der aktuelle Zustand und seine Ausgabe nur vom seinem vorherigen Zustand ab, mathematisch ausgedrückt:

$$\begin{aligned} P(q_t &= S_j | q_{t-1} = S_i, q_{t-2} = S_k, \ldots) \\ &= P(q_t = S_j | q_{t-1} = S_i) \end{aligned}$$

Um Markov-Ketten zu modellieren, müssen sie aus den folgenden Elementen bestehen:

- die Menge der Zustände $S = \{s_1, s_2, \ldots, s_N\}$
- a_{ij} ist die Wahrscheinlichkeit, dass das System in den Zustand s_j übergeht unter der Vorraussetzung, dass das System sich im Zustand s_i befindet, wobei die Summe aller Übergangswahrscheinlichkeiten in einem Zustand bzw. einer Zeile der Matrix zusammen 1 ergeben müssen.

$$a_{ij} = P(q_{t+1} = s_j | q_t = s_i), 1 \leq i, j \leq N \tag{1.4}$$

$$a_{ij} \geq 0, \sum_{i=1}^{N} a_{ij} = 1 \tag{1.5}$$

- Anfangswahrscheinlichkeitenvektor $pi = \{\pi_i\}$, der den Anfangszustand des Systems beschreibt.

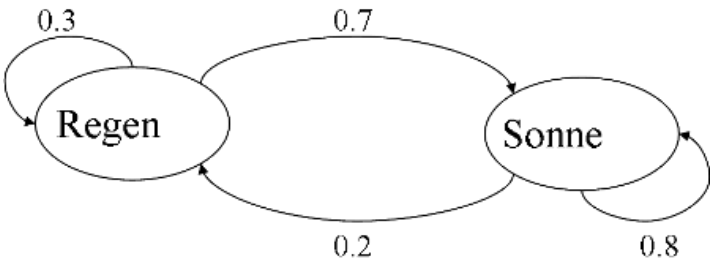


Abbildung 1.1: Wettervorhersage mit dem Markov Modell

$$\pi_i = P(q_1 = s_i), 1 \leq i \leq N \tag{1.6}$$

Ein Markov-Modell ist eine Datenstruktur, die die nötigen Werte für die Steuerung einer Markov-Kette bereitstellt. Hier beschreibt das ursprünglich einfache Modell Vorhersagen auf Grundlage indirekter Evidenz (Hidden Markov-Modell).

Hier benutze ich das populäre Beispiel vom Markov-Modell, nämlich das System der Wettervorhersage in Fig1.1, welches einfach mit zwei Zuständen aufgebaut ist: Regen und Sonne.

Übergangswahrscheinlichkeiten:

$$P(Regen|Regen) = 0.3, P(Sonne|Regen) = 0.7$$

$$P(Regen|Sonne) = 0.2, P(Sonne|Sonne) = 0.8$$

$$A = \{a_{ij}\} = \begin{pmatrix} 0.3 & 0.2 \\ 0.7 & 0.8 \end{pmatrix}$$

Anfangswahrscheinlichkeiten:

$$P(Regen) = 0.4, P(Sonne) = 0.6$$

Nehmen wir an, dass wir eine Beobachtungsfolge der Zustände in unserem Beispiel errechnen möchten: {Sonne, Sonne, Regen, Regen}.

$$\begin{aligned} P(Sonne, Regen) &= P(Sonne)P(Sonne|Sonne)P(Regen|Sonne)P(Regen|Regen) \\ &= 0.6 * 0.8 * 0.2 * 0.3 \\ &= 0.0288 \end{aligned}$$

Hidden Markov-Model

Heute finden wir die HMM-Technik überall vor, wo (Zeit)seriendaten zu modellieren, zu interpretieren und zu segmentieren sind und wo ein Erwerb umfangreicher Trainingsdatensammlungen mit passendem Aufwand zu vollziehen ist. Sie bieten eine Vielfalt von technischen Anwendungen. Einige bekannte Anwendungsgebiete sind:

- Gen-Vorhersage
- Spracherkennung und Mustererkennung
- Signal-Verarbeitung
- Neurobiologie
- ...

mit Hilfe von Hidden Markov-Modellen kann man nach bestimmten Sequenzen oder Mustern in Daten suchen. Bei allen diesen Anwendungsmöglichkeiten werden bestimmte Muster in grossen Datenmengen gesucht. Die Mustersuche in Datenbanken wird als Data-Mining bezeichnet.

Aus dem allgemeinem Markov-Modell ist mit dem Das Hidden Markov-Modell ein erweitertes stochastisches Modell entwickelt worden, das sich durch zwei Zufallsprozesse beschreiben lässt. Der erste Zufallsprozess entspricht dabei einer Markov-Kette, die durch Zustände und Übergangswahrscheinlichkeiten gekennzeichnet ist. Die Zustände der Kette sind von außen jedoch nicht direkt sichtbar (sie sind verborgen). Stattdessen erzeugt ein zweiter Zufallsprozess zu jedem Zeitpunkt beobachtbare Ausgangssymbole gemäß einer zustandsabhängigen Wahrscheinlichkeitsverteilung. Die Aufgabe besteht häufig darin, aus der Sequenz der Ausgabesymbole auf die Sequenz der verborgenen Zustände zu schließen.

Definition [LR89] ein Hidden Markov-Modell wird formal als Fünftupel $\lambda = (N, M, A, B, \pi)$ definiert:

- N ist die Menge aller Zustände $S = \{s_1, s_2, \ldots, s_N\}$ im HMM, die Zustandsvariable $Q = q_1 q_2 \ldots q_T$ annehmen kann.
- M ist Merkmalsraum (oft als Ausgabealphabet bezeichnet), also der Definitionsbereich von b_i. Dieser kann diskret, oder kontinuierlich sein. Je nach dem ist b_i eine Wahrscheinlichkeit oder eine Dichte.
- $A = \{a_{ij}\}$ Zustandsübergangsmatrix, wobei a_{ij} die Wahrscheinlichkeit angibt, dass nach Zustand q_i in Zustand q_j gewechselt wird.
- $B = \{b_1, \ldots, b_n\}$ Menge der Emissionswahrscheinlichkeitsverteilungen bzw. Dichten
- $b_i(k)$ Wahrscheinlichkeit im Zustand q_i die Beobachtung k zu machen
- $\pi = \{\pi_i\}$ Anfangswahrscheinlichkeitsverteilung mit

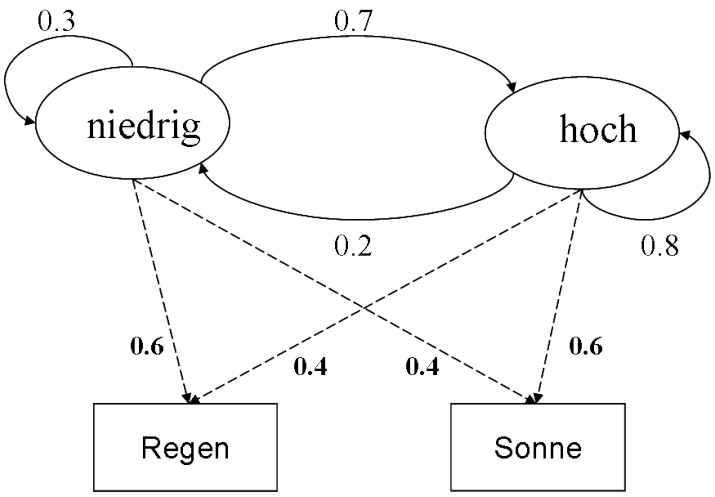


Abbildung 1.2: Wettervorhersage mit dem Hidden Markov-Modell

$$\pi_i = P(q_1 = s_i), 1 \leq i \leq N \tag{1.7}$$

Gegeben seien die passenden Werte von N, von M, von A, von B und von π. Das HMM kann als Generator verwendet werden, um eine Beobachtungs-Reihenfolge anzugeben, in der jede Beobachtung O_t eins der Symbole von M ist und T die Zahl der Beobachtungen in der Folge ist. Das unten genannte Verfahren kann verwendet werden als Generator von Beobachtungen und als Modell dafür, wie eine gegebene Beobachtungs-Reihenfolge durch ein passendes HMM erzeugt wurde. Diese Schrittfolge[LR89] kann man genauer direkt unten einsehen.

1. wähle Anfangszustand $q_i = s_i$ entsprechend Anfangszustandverteilung π
2. Setze $t = 1$
3. Wähle $O_t = v_k$ entsprechend Wahrscheinlichkeitsverteilung der Beobachtungssymbole im Zustand s_i, d.h. $b_i(k)$
4. Wechsle in den nächsten Zustand $q_{t+1} = s_i$ entsprechend Übergangswahrscheinlichkeitsverteilung a_{ij} für Zustand s_i
5. setze $t = t + 1$, wiederhole Schritt 3, falls $t < T$, sonst endet dies Verfahren

In der vorherigen Vorstellung haben wir schon gesehen, wie das System der Wettervorhersage mit Hilfe des allgemeinen Markov-Modells dargestellt wird und jetzt schauen wir uns das erweiterte System in Fig.1.2, welches aus dem Markov-Modell evolviert ist, an.

Zwei Zustände vom Luftdruck:

niedrig oder hoch

Zwei Beobachtungen:

Regen und Sonne

Übergangswahrscheinlichkeiten:

$$P(niedrig|niedrig) = 0.3, P(hoch|niedrig) = 0.7$$

$$P(niedrig|hoch) = 0.2, P(hoch|hoch) = 0.8$$

$$A = a_{ij} = \begin{pmatrix} 0.3 & 0.2 \\ 0.7 & 0.8 \end{pmatrix}$$

Ausgabewahrscheinlichkeiten:

$$P(Regen|niedrig) = 0.6, P(Sonne|niedrig) = 0.4$$

$$P(Regen|hoch) = 0.4, P(Sonne|hoch) = 0.3$$

Anfangswahrscheinlichkeiten:

$$P(niedrig) = 0.4, P(hoch) = 0.6$$

Hier als Beispiel können wir versuchen, alle möglichen Reihenfolgen der versteckten Zustände anzunehmen:

$$\begin{aligned} P(Sonne, Regen) &= P(Sonne, Regen, niedrig, niedrig) \\ &+ P(Sonne, Regen, niedrig, hoch) \\ &+ P(Sonne, Regen, hoch, niedrig) \\ &+ P(Sonne, Regen, hoch, hoch) \end{aligned}$$

Anschaulicher sollte ein Gedanke nach unserer Probe daher erzeugt werden. Die Berechnung von $P(O|\lambda)$ gemäß originaler Definition kann ziemlich groß sein, man betrachtet in diesem Fall den Rechenaufwand auf $O(N^T)$. Gibt es eine Methode, um diesen schlechten Fall zu verbessern? In den folgenden Abschnitten werden wir konkreter darauf eingehen.

Drei klassische Problemstellungen bei den Anwendungen

Es gibt drei grundlegende Probleme, die gelöst werden müssen, damit das Hidden Markov-Modell in der Real-Welt bei verschiedenen Anwendungen nützlich ist. Hier befindet sich noch ein sinnvolles Online-Tutorium [Boy] für alle Interessenten.

1. Gegeben sei die Beobachtungsfolge $O = O_1 O_2 \ldots O_T$ und ein Modell $\lambda = (A, B, \pi)$, wie berechnet man die Wahrscheinlichkeit $P(O|\lambda)$?

2. Wie bestimmt man eine Zustandsfolge $Q = q_1 q_2 \ldots q_T$, die eine gegebene Beobachtungsfolge $O = O_1 O_2 \ldots O_T$ zu einem gegebenem Modell $\lambda = (A, B, \pi)$ besitzt?

3. Wie passt man die Modellparameter $\lambda = (A, B, \pi)$ an, wenn nur die Beobachtungsfolge bekanntgegeben ist, um die Wahrscheinlichkeit $P(O|\lambda)$ zu maximieren?

Die Wahrscheinlichkeit einer Reihenfolge von Übergängen herauszufinden ist nützlich. Um die Wahrscheinlichkeit einer Reihenfolge von Beobachtungen zu finden, welche als ***Evaluation*** gekennzeichnet wird, kann in der Vorwärts- und Rückwärts- Richtung gerechnet werden (d.h. vorwärts von der ersten Beobachtung arbeiten oder von der letzten Beobachtung zurück zu arbeiten).

Nachdem das oben genannte Problem gelöst worden ist, können wir nicht absichern, welche Reihenfolge der Zustände für eine gegebene Reihenfolge von Beobachtungen vorgenommen wird. Im allgemeinen ist es am effizientesten, die wahrscheinlichste Zustandflugbahn für eine gegebene Reihenfolge von Beobachtungen zu schätzen. Dieses Problem wird häufig als ***Decodieren*** gekennzeichnet.

Eine andere allgemeine Anforderung ist, die Wahrscheinlichkeiten zu erlernen, die mit Übergängen im System verbunden sind. Durch ein Repräsentativ-training wird seine passende Reihenfolge gegeben anstatt seine Übergangswahrscheinlichkeiten direkt anzugeben. Die Idee ist, den Raum von Übergangswahrscheinlichkeiten zu finden, der die Wahrscheinlichkeit der Trainings-Reihenfolge maximiert. Dies ist das sogenannte ***Lernproblem***.

Forward- und Backward-Algorithmus

Die Wahrscheinlichkeit der Beobachtungen O für eine spezifische Zustandsreihenfolge Q ist:

$$\begin{aligned} P(O|Q,\lambda) &= \prod_{t=1}^{T} P(O_t|Q_t,\lambda) \\ &= b_{q_1}(O_1)(\times)b_{q_2}(O_2)\ldots b_{q_T}(O_T) \end{aligned} \tag{1.8}$$

und die Wahrscheinlichkeit der Zustandsreihenfolge ist:

$$P(O|\lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \ldots a_{q_{T-1} q_T} \tag{1.9}$$

Die Wahrscheinlichkeit einer Beobachtungssequenz bei vorgegebenem Modell ist die Wahrscheinlichkeit für das Auftreten einer Beobachtungssequenz bei allen möglichen Hintergrundsequenzen, die sich zur Gleichung 1.8 expandieren lässt.

$$P(O|\lambda) = \sum_{alleQ} P(O|Q,\lambda)P(Q|\lambda) \tag{1.10}$$

$$= \sum_{q_1,q_2,\ldots,q_T} \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \ldots a_{q_{T-1} q_T} b_{q_T}(O_T) \tag{1.11}$$

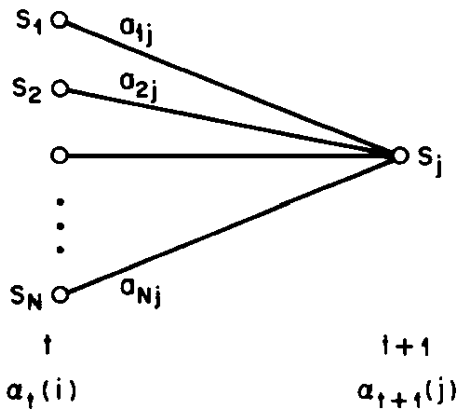


Abbildung 1.3: Forward-Rekursion

Die Forward-Variable wird definiert als:

$$\alpha_t(i) = P(O_1, O_2, \ldots, O_t, q_t = s_i|\lambda) \tag{1.12}$$

Forward Rekursion

1) Initialisierung:

$$\alpha_1(i) = P(O_1, q_1 = s_i|\lambda) = \pi_i b_i(O_1), 1{\leq}i{\leq}N \tag{1.13}$$

Die Gesamtwahrscheinlichkeit, Zustand s_i im Zeitpunkt 1 zu erreichen, ist die Anfangswahrscheinlichkeit für s_i, multipliziert mit der Wahrscheinlichkeit, dass O_1 ausgegeben wird.

2) Induktion:

$$\alpha_{t+1}(j) = \sum_{i=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}), 1{\leq}t{\leq}T-1, 1{\leq}j{\leq}N \tag{1.14}$$

Die Gesamtwahrscheinlichkeit, Zustand sj im Zeitpunkt t+1 zu erreichen, ist die Summe der Wahrscheinlichkeiten von einem beliebigen Zustand s_i in Zustand s_j unter Ausgabe des nächsten Symbols O_{t+1}.

3) Terminierung:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{1.15}$$

Die Gesamtwahrscheinlichkeit der Realisierung in Fig.1.3 ist die Summe der Wahscheinlichkeiten, zum Zeitpunkt T in einen beliebigen Zustand zu kommen. Daher ist der Rechenaufwand

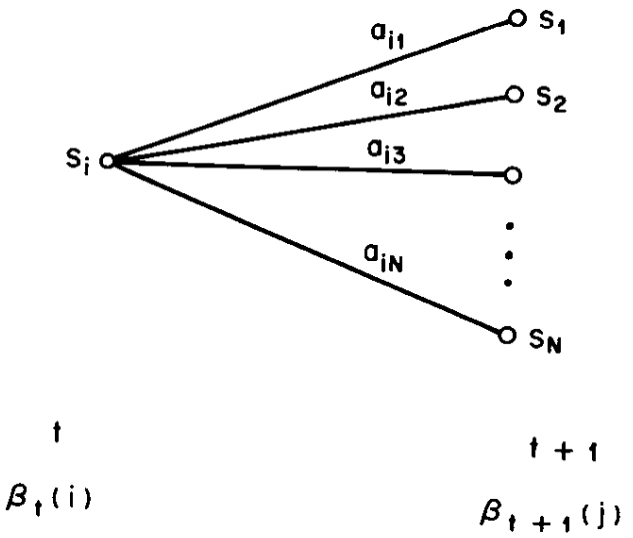


Abbildung 1.4: Backward-Rekursion

auf N^2T reduzierbar. Wir können also die Backward-Rekursion in Fig.1.4 analog machen. Die Backward Rekursion wird in der Lösung zum Problem 3 verwendet ist aber für das Problem 1 nicht notwendig.

Backward Rekursion
Wir definieren zuerst die Backward-Variable:

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, \ldots, O_T | q_t = s_i, \lambda) \tag{1.16}$$

1) Initialisierung:

$$\beta_T(i) = 1, 1 \leq i \leq N \tag{1.17}$$

2) Induktion:

$$\beta_t(j) = \sum_{j=1}^{N} \beta_{t+1}(j) a_{ij} b_i(O_{t+1}), 1 \leq i \leq N, t = T-1, T-2, \ldots, 1 \tag{1.18}$$

3) Terminierung:

$$P(O_1, O_2, \ldots, O_T) = \sum_{i=1}^{N} \beta_1(i) \pi_i b_i(O_1) \tag{1.19}$$

Viterbi-Algorithmus

Die Frage welche Hintergrundsequenz bei gegebener Beobachtungssequenz und Modell optimal ist, beschäftigt sich damit, die korrekte Zustandsabfolge zu finden. Das zweite Problem ist mit Hilfe des Viterbi-Algorithmus lösbar, wobei der $P(Q, O|\lambda)$ maximal ist.

Der Viterbi-Algorithmus in Fig.1.5 ist eine Methode aus dem Gebiet der Dynamischen Programmierung. Die Idee war, dass $\delta_t(i)$ die höchste Wahrscheinlichkeit, einen Suchpfad zum Zeitpunkt n beschreibt. Dabei werden die ersten n Beobachtungen gezählt und $\delta_t(i)$ landet schliesslich im Zustand s_i.

$$\delta_t(i) = \max_{q_1,q_2,\ldots,q_{t-1}} P(q_1, q_2, \ldots, q_t = i, O_1, O_2, \ldots, O_t|\lambda \tag{1.20}$$

Durch Induktion hat man die Beziehung

$$\delta_{t+1}(j) = [\max_i \delta_t(i)a_{ij}]b_i(O_{t+1})$$

, Um die Zustandsfolge wieder zu gewinnen, speichert man auch gleichzeitig das Argument, das die $\delta_{t+1}(j)$ für jedes t und j jeweils maximiert, in ein Feld $\Psi_t(j)$

Der gesamte Optimierungsvorgang
1) Initialisierung:

$$\begin{aligned} \delta_1(i) &= \pi_i b_i(O_i), 1{\leq}i{\leq}N & (1.21)\\ \Psi_t(i) &= 0 & (1.22) \end{aligned}$$

2) Induktion:

$$\begin{aligned} \delta_t(j) &= \max_{1\leq i\leq N}[\delta_{t-1}(i)a_{ij}]b_j(O_t), 2{\leq}t{\leq}T, 1{\leq}j{\leq}N & (1.23)\\ \Psi_t(i) &= \arg\max_{1\leq i\leq N}[\delta_{t-1}(i)a_{ij}], 2{\leq}t{\leq}T, 1{\leq}j{\leq}N & (1.24) \end{aligned}$$

3) Terminierung:

$$\begin{aligned} P^\star &= \max_{1\leq i\leq N}[\delta_T(i)] & (1.25)\\ q_t^\star &= \arg\max_{1\leq i\leq N}[\delta_T(i)] & (1.26) \end{aligned}$$

4) Optimale Zurückverfolgerung der Zustandsfolge:

$$q_t^\star = \Psi_{t+1}(q_{t+1}^\star), t = T-1, T-2, \ldots, 1 \tag{1.27}$$

EM-Algorithmus

Der EM-Algorithmus [2] ist ein allgemeines Verfahren zur Bestimmung von Maximum-Likelihood Schätzwerten von probabilistischen Modellen. Die Bezeichnung Algorithmus ist eigentlich irreführend, da in der allgemeinen Definition des Verfahrens keine genauen Anweisungen für Rechenschritte gegeben werden, wie es der Begriff des Algorithmus fordert, sondern nur eine allgemeine Beschreibung des Verfahrens und seiner mathematischen Eigenschaften. Für jedes

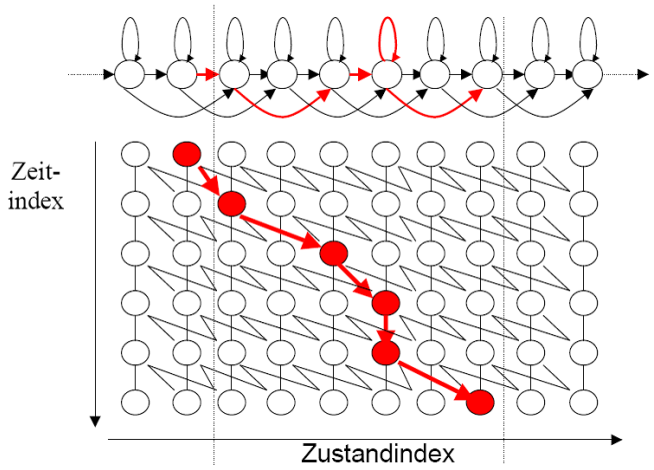


Abbildung 1.5: Viterbi-Decoder

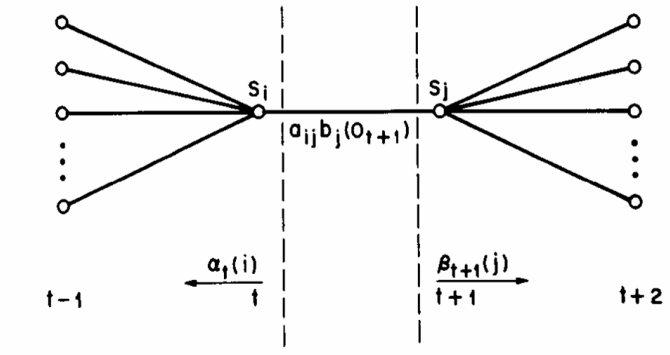


Abbildung 1.6: Baum-Welch-Algorithmus

Anwendungsgebiet muss daher ein EM-Algorithmus erfunden werden. Für Hidden Markov-Models heißt dieser Algorithmus „Baum-Welch Algorithmus“ in Fig.1.6.

$\xi_t(i,j), 1{\leq}t{\leq}T; 1{\leq}i,j{\leq}N$ sei die Wahrscheinlichkeit, dass der Übergang zwischen Zustand s_i und Zustand s_j zur Zeit t (mit Forward-und Backward-Variable geschrieben) bei einer gegebenen Realisation O genommen wird:

$$\begin{aligned} \xi_t(i,j) &= P(q_t = s_i, q_{t+1} = s_j | O, \lambda) && (1.28) \\ &= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} && (1.29) \end{aligned}$$

Weiterhin ist $\gamma_i(t) = \sum_{j=1}^{N} \xi_t(i,j)$ die Wahrscheinlichkeit, dass Zustand i zum Zeitpunkt t bezüglich O erreicht wird.

Im Estimation-Schritt werden nun die Erwartungswerte für die Anzahl der Übergänge pro Kante berechnet:

- $\sum_{t=1}^{T-1} \gamma_i(t)$ ist die erwartete Anzahl von Übergängen aus Zustand s_i bei Realisierung O.
- $\sum_{t=1}^{T-1} \xi_t(i,j)$ ist die erwartete Anzahl von Übergängen von Zustand s_i nach s_j bei Realisierung O.

Im Maximization-Schritt werden nun die Modellparameter neu berechnet anhand der Erwartungswerte:

- $$\overline{\pi_i} = \gamma_i(1) \tag{1.30}$$

  Die neuen Anfangswahrscheinlichkeiten sind die Wahrscheinlichkeiten, dass Zustand s_i zum Anfang der Zustandssequenz bezüglich O auftritt.

- $$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_i(t)} \tag{1.31}$$

  Die neue Wahrscheinlichkeit für einen Übergang von Zustand s_i nach Zustand s_j ist die erwartete Anzahl der Übergänge von s_i nach s_j dividiert durch die erwartete Gesamtzahl der Übergänge aus s_i durch die Normierung entsteht hier wieder eine Wahrscheinlichkeit $0{\leq}a_{ij}{\leq}1$.

- $$\overline{b_i k} = \frac{\sum_{t=1}^{T} \gamma_i(t) \delta_{k,O_t}}{\sum_{t=1}^{T} \gamma_i(t)} \tag{1.32}$$

[2]Expectation-Maximization Algorithmus

δ_{k,O_t} ist wie folgt definiert:

$$\delta_{k,O_t} = \begin{cases} 1; & k = O_t \\ 0; & sonst \end{cases} \tag{1.33}$$

Die neue Wahrscheinlichkeit für die Ausgabe eines Symbols k im Zustand s_i ist die erwartete Anzahl im Zustand s_i zu sein und dabei das Symbol $k = O_t$ auszugeben, dividiert durch die erwartete Anzahl, überhaupt im Zustand s_i zu sein.

Damit wurde ein neues Modell $\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\pi})$ berechnet. Wie für den EM-Algorithmus allgemein gezeigt, gilt $P(O|\overline{\lambda}) \geq P(O|\lambda)$.

Zeichenbasiertes NER mit Hidden Markov-Modellen

In einem Zeichenstrom kann mittels eines HMMs eine Named Entity erkannt und klassifiziert werden. Interessant ist der Ansatz vorallem deswegen, weil er aufzeigt, wie wertvoll bereits Teilsequenzen von Zeichen sein können für NER.

Das Ziel ist, zu einem gegebenen Text automatisch alle Wörter mit den zugehörigen Wortarten zu annotieren. Wie beschrieben kann ein HMM diese Aufgabe lösen, wenn seine Parameter (A, B, π) richtig gesetzt sind. Ein HMM muss also trainiert werden, indem man mithilfe eines vorgetaggten Textes die Parameter so anpaßt, dass das HMM für die Wortfolgen in diesem Text die richtigen Wortartfolgen mit möglichst geringer Fehlerrate ausgibt.

Als Beispiel nehmen wir an, dass man den Wörtern eines Textes ihre Wortarten zuweisen möchte. Alles was vorliegt, ist ein Text als eine Folge von Wörtern. Einen solchen Text kann man in Annäherung als eine Folge von zufälligen Ereignissen interpretieren, zwar dass auf eine Wortart mit einer bestimmten Wahrscheinlichkeit eine andere Wortart folgt, und für jede Wortart dann mit einer bestimmtenWahrscheinlichkeit ein konkretes Wort generiert wird. Betrachtet man den Text als Ergebnis des gesamten Zufallsexperiments, so ist die Abfolge der Wörter erkennbar, nicht mehr die Folge der Wortarten, die die Wörter generierte.

Ein Text kann also als ein Zufallsexperiment mit einer Menge von Zufallsvariablen $O_1, \ldots, O_T, q_1, \ldots, q_T$ gesehen werden. Die Werte der O_i ist die Folge der beobachtbaren Daten (die Realisierung), hier sind die einzelnen Wörter. Die Werte der q_i sind die versteckten Daten, nämlich die zu den Wörtern gehörigen Wortarten.

Die zugrundeliegende Verteilung ist also eine gemeinsame Verteilung der Menge von Zufallsvariablen $\{O_1, \ldots, O_T, q_1, \ldots, q_T\}$. Mit Hilfe des Hidden Markov-Modells können solche Verteilungen modelliert werden.

Der Aufbau des HMM's wurde schon erläutert. Pro Zustand, welcher vom HMM durchlaufen wird, wird ein Zeichen generiert bzw. ausgegeben. Jeder Zustand wird durch ein Tupel bestehend aus Klasse (Person, Ort, Organisation)in Fig.1.7, welche den Offset im aktuellen Wort angibt, identifiziert. Es gibt z.B. einen Zustand (Ort, 2) in welchem die Emission des dritten Zeichens aller NE vom Typ 'Ort' vorgenommen wird. Ein spezieller Endzustand (Ort, L) wird für das Leerzeichen (Wortgrenze) zwischen zwei Wörtern verwendet. Ferner wird eine Outputmodell der Form $P(v_k|v_{k-1}\ldots v_{k-N+1}, s_k)$ benutzt, wobei v_k das aktuelle Zeichen und s_k der aktuelle Zustand ist - es handelt sich also um ein (zustandsabhängiges) N-Gram.

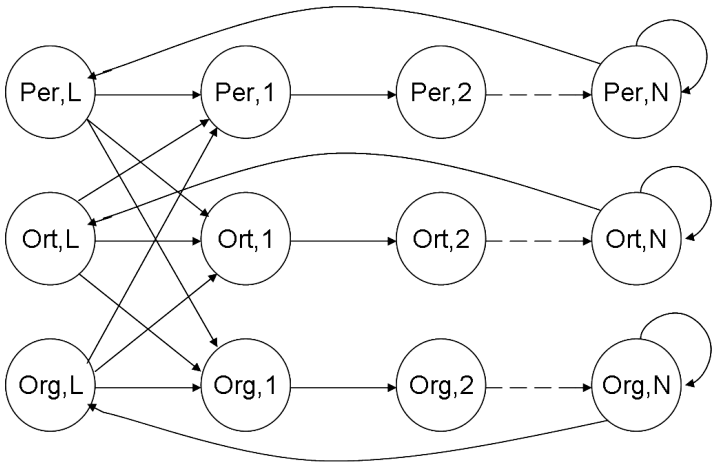


Abbildung 1.7: NER mit Hilfe des HMMs

Die Anzahl der Zustände pro Klasse ist beschränkt sich auf N 'reguläre' Zustände und dem Endzustand.

Statistische Modelle jenseits des HMMs

Local-Berechnungs-Methoden (Forward- und Backward- , Viterbi-Algorithmus) erleichtern die Analyse von großen Datenmengen, wenn eine Schätzung von Emissionswahrscheinlichkeiten vorhanden ist. Der EM-Algorithmus ist eine gute Wahl, wenn schnelle und nicht so präzise Ergebnisse gefordert sind.

- HMM sind für die Klassifikation und Modellbildung insbesondere von Zeitreihen gut geeignet.
- Durch das Modell erhält man viele Informationen über die modellierte Signalquelle.

Trotz erfolgreicher Einsätze der einschlägigen HMM-Modellierungswerkzeuge in etlichen symbolverarbeitenden KI-Anwendungen unterliegt das traditionelle Hidden Markov-Model einer Reihe offenkundiger Schwachstellen und Einschränkungen:

- das endliche Zustandsinventar führt zu dem endlichen Gedächtnis
- die fixe Abhängigkeitsstruktur der Zustandsvariablen
- die eindimensionale Organisation der Ausgabedaten
- insbesondere seine implizit exponentielle Zustandsverweildauer

welche in den nachkommden Abschnitte zur Vorstellung einiger Variationen des Themas HMM gab: Maximum Entropy Markov Models, Conditional Random Fields.

1.2.3 Maximum Entropy Markov Models

Einleitung und Motivation

Das Internet überhäuft uns mit so vielen Informationen, dass es unmöglich ist in kürzester Zeit die gebündelten und erwünschten Informationskerne zu filtern. Die Idee ein automatisches System zu entwickeln, hinter welchem eine bestimmte Struktur und verborgene Logik steckt, lässt einen Lösungsweg für das Problem aufblitzen. Ein Problem bleibt jedoch weiterhin bestehen: Der Computer kann niemals die freie Entscheidung des Menschen übernehmen. Aber ein automatisiertes System kann beliebig nah zur Menschenentscheidung approximiert werden. Einer der Versuche Informationen aus dem Internet zu bekommen, ist das so genannte Hidden Markov Model (HMM), das Informationen syntaktisch liest und semantisch bewertet. Dieses Modell ist im vorherigen Kapitel **??** erklärt worden. Wir beschäftigen uns hier mit einer Verbesserung des HMM, dem so genannten Maximum Entropy Markov Models (MEMM).

Von HMM zu MEMM

Wie kann man ein System entwickeln, so dass es Informationen aus dem Netz ausliest und auch richtig bewertet? Wie man sieht, unterteilt sich dieses Problem in zwei Bereiche: syntaktisch auslesen und semantisch bewerten. Das erste ist kein Problem, denn das Auslesen wird schon mit einer 100-prozentigen Wahrscheinlichkeit richtig ausgeführt. Das größte Problem ist, wie man ausgelesene Informationen bewertet, weil in vielen Situationen gelesene Informationen unterschiedlich bewertet werden können. Dies hängt von verschiedenen Faktoren ab. Dazu gehört auch der gesunde Menschenverstand, was eine Maschine nie lernen kann. Ein solches System ist folgendermaßen aufgebaut:

- Man macht eine Stichprobe. Das heißt man nimmt per Hand eine gewisse Information aus dem Web und bewertet sie selber. Es wird hier aber angenommen, dass der Mensch per Hand die Information richtig bewertet.
- Danach wählt man ein Schätzungsverfahren und bestimmt eine Wahrscheinlichkeitsverteilung, wie eine bestimmte Information (z.B. Zeichen, Wort, Satz) bewertet wird. Je besser dieses Schätzungsverfahren ist, desto näher kann ein automatisches System zur Menschenentscheidung approximiert werden.

Jetzt wiederholen wir kurz HMM und gehen dann zu MEMM über. Als Beobachtungen bezeichnet man die gelesene Information und als Zustände deren Bewertung. Folge von Beobachtungen und Zuständen ist dann die ganze Information, bzw. die Bewertung. HMM hat den Zusammenhang zwischen einer Beobachtung und deren Zustand, beispielsweise die Übersetzung aus dem Englischen. Eine Beobachtung wäre hier ein Wort und ein Zustand dessen Übersetzung. Es ist aber klar, dass manche Wörter verschieden übersetzt werden können. Um richtig zu übersetzen ist es eventuell notwendig den kompletten Satz und die Satzstruktur zu kennen, was dieses Modell nicht berücksichtigt. Deswegen ist es schwach. Ein Modell, welches dieses Verfahren nutzt, sind die so genannten Conditional Random Fields (siehe Kapitel **??**). Hier beschäftigen wir uns mit einer etwas schwächeren Variante, dem MEMM.

MEMM

Die Verbesserung ist die, dass die Bewertung der Beobachtung auch von der vorherigen Bewertung abhängt. Man teilt P in $|S|$ Funktionen auf: $P\left(s|s^{`}, o\right) = P_{s^{`}}(s|o)$, was so viel heißt wie: „die Wahrscheinlichkeit, dass die Beobachtung o als Bewertung s ausgewertet wird, wenn $s^{'}$ der vorherige Zustand ist“.

Maximale Entropie

Wie oben beschrieben macht man eine Stichprobe und berechnet die Wahrscheinlichkeit P für die Testdaten. Als nächstes muss die Verteilung abgeschätzt werden. Hier beschäftigen wir uns mit einer Verteilung, die maximale Entropie hat. Maximale Entropie wird bei der gleichmäßigen Verteilung erreicht. Anhand von errechneten Wahrscheinlichkeiten P für die Testdaten wird die gesamte Verteilung abgeschätzt, die mit den Testdaten nicht im Widerspruch steht, so dass eine minimale Anzahl von Annahmen getroffen wird. Dies ist die gesuchte Verteilung mit der maximalen Entropie und P wird wie folgt berechnet:

$$P_{s^{`}}(s|o) = \frac{1}{Z(o,s^{`})} \exp\left(\sum_a \lambda_a f_a(o,s)\right)$$

mit $Z(o, s^{`}) = \sum_s \exp\left(\sum_a \lambda_a f_a(o,s)\right)$. Die beiden Formeln sollten ohne Beweis angenommen werden. Das binäre Feature b ist eine binäre Funktion, die einer Beobachtung den Wert „true“ oder „false“ zuordnet. Bsp.: Die Beobachtung ist ein Großbuchstabe. Die Beobachtung besteht aus sechs Worten. Man definiere folgende charakteristische Funktion:

$$f_{\langle b,s\rangle}(o_t, s_t) = \begin{cases} 1, & \text{wenn } b(o_t) \text{ ist true und } s = s_t \\ 0, & \text{sonst} \end{cases}$$

Die Funktion hilft uns die Beobachtungen anhand ihrer Eigenschaften zu analysieren und P zu berechnen. Wenn P schon berechnet wurde, was nun? Wie kommt man an die Zustandsfolge?

Viterbi Algorithmus

Man definiert die rekursive Variable:

$$\delta_{t+1}(s) = \left(\max_{s^{`}\in S}\right)\left\{\delta_t(s^{`}) \cdot P_{s^{`}}(s|o_{t+1})\right\}$$

wobei $\delta_t(s^{`})$ die Wahrscheinlichkeit ist, dass die wahrscheinlichste Zustandsfolge bis zum $o_t$ im $s^{`}$ endet. Es werden dynamisch alle δ-Werte berechnet. Somit errechnet sich die gesuchte Zustandsfolge.

1.2.4 Conditional Random Fields

Einführung

Im großen Bereich des Maschinellen Lernens ist das sogenannte Taggen von Sequenzen von wichtiger Bedeutung. Hier geht es nicht um die Kennzeichnung von Wörtern (Parsing) sondern um die Zuordnung von Wörtern zu Entity-Klassen. Dabei handelt es sich bei der Eingabe stets

um sequentielle Daten. Hierbei werden die Wörter identifiziert und verschiedenen Entity-Klassen wie LOCATION, PERSON und anderen zugeordnet. Das sogenannte Named Entity Recognition (NER) ordnet genau diese Named Entities (NE) den Entity-Klassen zu. Die Grundlage zur Identifizierung der Entities sind meistens Lexika. Allerdings reichen diese nicht aus, da ein Wort zweideutig und somit zu mehr als eine Entity-Klasse zugeordnet werden kann. Speziell durch die Verwendung von Conditional Random Fields (CRF) zur Named Entity Recognition umgeht man bestimmte Probleme die mit vorher verwendeten Modellen existieren.

Grundlagen

Bei CRF handelt es sich um eine Weiterentwicklung der Hidden Markov models (HMM) und Maximum entropy markov models (MEMM). Dabei werden gezielt Nachteile (label bias problem) der vorherigen Modelle vermieden. Als Grundlage für CRF dient hier ebenfalls die Markov Random Fields (MRF). Dabei erfüllt ein MRF u.A. folgende Eigenschaften:

- Graphisches Modell
- Darstellbar als ungerichteter, azyklischer Graph gemäß G=(V,E)
- Knoten aus V entsprechen den Labels

Desweiteren gilt stets die Markov-Eigenschaft. Das heißt, dass ein aktueller Zustand nur von einer bestimmten Menge von Zuständen abhängt. Der in der Graphentheorie etablierte Begriff der Clique ist hier das Stichwort. Konkret kann man sagen, dass ein Knoten bzw. ein Zustand v nur von den Knoten abhängt, die in der gleichen Clique sind wie v. Formal notiert kann man schreiben:

$$P\left(y_v | y_w : v \neq w\right) = P\left(y_v | y_w : v \sim w\right)$$

Wie oben beschrieben handelt es sich bei MRF um ein ungerichtetes graphisches Modell. Somit können Vorgänger nicht eindeutig bestimmt werden. Durch eine geeignete Funktion muss deshalb die bedingten Wahrscheinlichkeiten eines Zustandes, die über die Cliquen definiert werden, errechnet werden. Dabei definiert man die Potentialfunktion über jede Clique *ci* im Graphen und erhält somit eine Menge von Potentialfunktionen gemäß:

$$\Phi = \Phi_{c_i} : A_{c_i} \rightarrow R^+$$

Die Funktion liefert also eine Zahl $\in R$, die also auch Zahlen größer als 1 liefern kann. Die Größe der Zahl skaliert mit der Wahrscheinlichkeit einer Belegung der Clique. Somit ist die Wahrscheinlichkeit einer Label-Sequenz wie folgt definiert:

$$P\left(y_1, \ldots, y_n\right) = \frac{1}{Z} * \prod_{\Phi} \Phi_{c_i}\left(y_{i_1}, \ldots, y_{i_{|c_i|}}\right)$$

Durch einen weiteren Schritt wird durch einen Normalisierungsfaktor eine Zahl generiert die Wahrscheinlichkeitskonform ist. Die Normalisierung wird durch folgende Funktion realisiert:

$$Z = \sum_{Y^n} \left[\prod_{\Phi} \Phi_{c_i}\left(y_{i_1}, \ldots, y_{i_{|c_i|}}\right)\right]$$

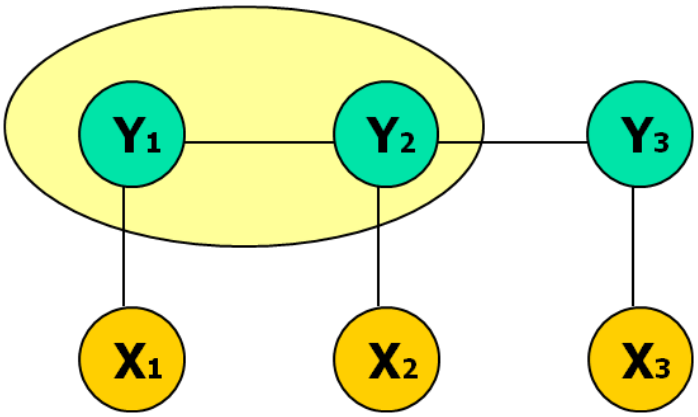


Abbildung 1.8: Graph mit Clique

Conditional Random Fields

Bei CRF handelt es sich eigentlich um MRF. Der wichtige Unterschied allerdings ist, dass hier eine gewisse Abhängigkeit existiert. Formalisiert kann man folgendes festhalten:

$$P(y_1, \ldots, y_n|x)$$

Hier kann man sehen, dass die Wahrscheinlichkeit der Label-Sequenz von x abhängt. Desweiteren wird stets angenommen, dass der als Graph visualisierte CRF einer first-order Markov-Kette entspricht. Dies bedeutet, dass eine Clique als ein Knoten interpretiert wird und einen Vor- bzw. Nachfolger hat. Bei den Cliquen selbst handelt es sich um verbundene Labels.
Die enge Beziehung zu MRF spiegelt sich auch in der Potentialfunktion wider:

$$\Phi_k(y_{i-1}, y_i, x, i)$$

Um die Verständlichkeit zu erhöhen, kann man sich auf binäre Potentialfunktionen beschränken. Die Funktion sei erfüllt wenn sie 1 liefert bzw. nicht erfüllt wenn sie 0 liefert. Ein einfaches Beispiel soll die formale Notation vereinfacht darstellen:
Sei $\Phi(y_{i-1}, y_i, x, i) = b(x, i)$ wenn $y_{i-1} = [ORG]$ und $y_i = [LOC]$ dann ist $b(x, i) = 1$, genau dann wenn Beobachtung an Position i das Wort „Deutschland“ ist, ansonsten 0.
Die bedingte Wahrscheinlichkeit gleicht der Notation wie bei den MRF. Deswegen kann man die Formel folgender Form festhalten:

$$P(\vec{y}|\vec{x}) = \frac{1}{Z(\vec{x})} \prod_k \sum_{i=1}^{t} \Phi_k \left(y_{i-1}, y_i, \vec{x}, i\right)$$

wobei stets gilt: $(\vec{x}) = (x_1, \ldots, x_n)$
Beim CRF Graphen handelt es sich stets um einen Baum. Dementsprechend kann man das Hammersley-Clifford-Theorem anwenden und die Formel für das Training mit den Sequenzen vorbereiten:

$$P(\vec{y}|\vec{x}) = \frac{1}{Z(\vec{x})} exp\left[\sum_k \sum_{i=1}^{t} \Phi_k \left(y_{i-1}, y_i, \vec{x}, i\right) \right]$$

Wie vorher erwähnt wurde, erzeugt die unbehandelte Verwendung der Potentialfunktion Zahlen, die nicht ins Wahrscheinlichkeitsuniversum passen. Erst durch den Normalisierungsfaktor Z werden Ergebnisse e [0,1] geliefert und entsprechen damit einem Wahrscheinlichkeitswert. Eine gute Label-Sequenz soll stets einer Zahl entsprechen die nahe 1 ist, eine schlechte Label-Sequenz einer Zahl die nahe 0 ist.

Training

Der eingeführte Vektor $\vec{\lambda}$, der die Potentialfunktionen unterschiedlich gewichtet, muss schließlich Trainiert werden. Nur dann kann man gewährleisten, dass die Wahrscheinlichkeiten für gegebene Trainingsbeispiele maximiert werden. Formal bildet man den maximum-log-likelihood und maximiert diese Funktion. Um die Funktion weiter zu vereinfachen, summiert man die Potentialfunktion über die Sequenzlänge:

$$F_k(\vec{y}, \vec{x}) = \sum_{i=1}^{T} \left(\Phi_k \left(y_{i-1}, y_i, \vec{x}, i\right)\right)$$

Letztendlich erhält man eine Gleichung für die log-likelihood-Funktion. Wenn es zwischen den Zuständen im modellierten Graph und den Labels eine eins-zu-eins-Beziehung existiert, ist die Konvexität der Funktion gegeben und somit auch das Finden eines globalen Maximums. Die Berechnungen erfolgen dann stets iterativ, indem bei jedem Schritt die Parameter der maximum likelihood-Lösung näher kommen, oder alternativ durch andere gradientenbasierte Methoden.
Um die Berechnungen der bedingten Wahrscheinlichkeiten effizient durchzuführen, werden Matrizen verwendet. Bei Sequenzen mit n Labels werden genau n+1 Matrizen erzeugt und betrachtet, also jeweils eine für jede Stelle und eine für die gesamte Beobachtungssequenz. Jede Matrix hat die Form $|Y| \times |Y|$, wobei Y das Zustandsalphabet darstellt. Formal kann folgendes festhalten:

$$M_i(\vec{x}) = \left[M_i(y', y|\vec{x})\right], wobei\ gilt$$

$$M_i(y', y|\vec{x}) = exp\left(\sum_k \lambda_k \Phi_k(y', y, \vec{x}, i)\right)$$

Die Formel für die Berechnung der Bedingten Wahrscheinlichkeiten kann nun folgendermaßen festgehalten werden:

$$P(\vec{y}|\vec{x}, \vec{\lambda}) = \frac{\prod_{i=1}^{T+1} M_i(y_{i-1}, y_i|\vec{x})}{Z(\vec{x}, \vec{\lambda})}$$

Natürlich sieht man hier wieder den Normalisierungsfaktor Z. Durch die Multiplikation aller Matrizen werden alle aktiven Potentialfunktionen für alle möglichen Label-Sequenzen berechnet.

1.3 Methoden der Indexierung und Information Retrieval

1.3.1 Indexierung

Einführung

Mit der Popularität des Internets sind auch die Quellen der Dokumentensammlungen (Zeitungsartikel, Webseiten, Bibliographien, usw) gestiegen, die den Nutzern für die Informationsgewinnung zur Verfügung stehen. Somit ist auch die Frage entstanden, wie man in dieser großen Informationsmenge, auf effiziente Art und Weise, Dokumente finden kann, die einem relevant erscheinen. Bewährt haben sich dafür WebSuchmaschinen, welche den Nutzern als Hilfsmittel dienen, um bestimmte Dokumente in Sammlungen zu finden. Die WebSuchmaschinen liefern einen leichten und effizienten Zugang zu Informationen. Neben der effizienten Suchfunktion sind Suchmaschinen auch fähig Indexierungen vorzunehmen. Viele gängige Suchmaschinen sind indexbasiert. Man stelle sich den Index wie das Schlagwortverzeichnis eines Buches vor. Durch stöbern im Index nach dem gesuchten Wort findet man schliesslich den passenden Verweis auf eine Seite und schlägt diese dann auf. Vorteil hierbei ist mit Sicherheit die verkürzte Suchzeit. Im Folgenden soll vorgestellt werden, wie man den Index als Datenstruktur konstruieren und verwalten kann. Zusätzlich wird auf die Indexstruktur Inverted File näher eingegangen. Da im Internet eine erheblich große Menge an Dokumenten zur Verfügung steht, können auch die Indizes dementsprechend groß ausfallen. Weiterhin soll gezeigt werden, wie man mit geeigneten Kompressionsverfahren die Indexgröße reduzieren kann. Den Abschluss bilden weitere Indexierungsverfahren: Das Signature File, das Suffix Array, sowie das Suffix Tree Verfahren.

Index

Inverted Files Ein Index ist eine Datenstruktur, welche Ausdrücke auf Dokumente abbildet, die sie enthalten. Die effizienteste Struktur, die zum Indizieren verwendet wird, ist das Inverted File Verfahren. Es ist eine Sammlung von Listen, in denen alle Ausdrücke abgespeichert werden, die in den Dokumenten vorkommen. Dabei unterscheidet man zwei Varianten des Index. Zum einen den word-level Index, der anzeigt, an welcher Position ein Ausdruck im Dokument vorkommt. Die zweite Variante ist der document-level Index, der nur die Information speichert, ob ein Ausdruck in einem Dokument vorkommt. Hierbei wird also kein Hinweis dafür geliefert, an welcher Stelle der jeweilige Ausdruck erscheint. Üblicherweise werden jedem Dokument eine Menge von Ausdrücken (Termen) zugeordnet. Doch wie der Name Inverted File verrät, findet hier eine Zuordnung der Dokumente zu einzelnen Termen statt. Gleichzeitig werden Informationen über das Vorkommen eines Terms in den Dokumenten bereitgehalten. Ein Inverted File besteht daher im wesentlichen aus einem Wörterbuch, in dem alle vorkommenden Terme eines Dokuments abgelegt werden. Das Wörterbuch bildet dann auf ein sogenanntes Posting File ab, in dem die Dokumenten-ID (doc-id), also die Termvorkommen, eventuell auch Wortpositionen, abgespeichert sind.
Der Vorteil einer solchen Datenstruktur ist der schnelle Zugriff auf Terme und Dokumente. Daraus resultiert natürlich eine kurze Suchzeit. Der Nachteil bei einer großen Anzahl an Wörtern, die in Listen abgespeichert werden müssen, ist der hohe Speicheraufwand. Durch Erweiterungen des Index wird beabsichtigt eine Suche möglichst effizient zu machen. Methoden wie Stemming oder Stopping beispielsweise sorgen dafür, den Index klein zu halten. Mit Stemming werden Wörter auf ihre Stammform reduziert. Man betrachtet dabei also nicht

die Wörter, sondern nur die Wortstämme und trennt die Endungen eines Wortes ab, so dass dann die Wörter in ihrer Grundform im Index gespeichert werden. Mit Stopping werden Präpositionen (die, von, bzw.,...) weggelassen, die wenig Aussagekraft haben. Dies ermöglicht die Einsparung von Platz bei der Speicherung des Indexes. Daraus resultiert eine effizientere Suche, da der Index kleiner ist. Eine andere Technik, die verwendet wird um die Größe des Index klein zu halten, ist der Thesaurus. Ein Lexikon, in dem unterschiedliche Wörter gleicher Bedeutung zusammengefasst werden.

Indexkonstruktion Die Konstruktion eines Indexes, auch Invertierung genannt, beinhaltet die folgenden 3 Schritte: Im ersten Schritt, dem Parsen der Dokumente, werden die Dokumente, die zu indizieren sind, von vorne bis hinten durchlaufen und eine Liste der Wörter mit der Dokumenten-id erstellt. Dabei können schon die Stopwörter, die den Index unnötig vergrößern, ausgelassen werden. Nachdem man diese Liste alphabetisch mit Hilfe von Sortieralgorithmen geordnet hat, werden Wörter, die in der Liste doppelt vorkommen, zusammengefasst. Durch die in der Liste angegebenen Dokumenten-ids ist das Vorkommen der Wörter eindeutig bestimmt. Beim Erstellen des Index können auch Probleme auftreten: Durch das Sortieren des Index müssten alte unsortierte oder neue sortierte Daten auf die Platte abgelegt werden. Dies würde kurzzeitig zu einem erhöhten Speicherverbrauch führen.

Indexverwaltung Da die Aktualisierung eines Index meistens mit Neueinfügungen von Wörtern verbunden ist und selten Wörter gelöscht werden, wächst natürlich auch die Größe des Index enorm. Der Index wird meist in Speicher verwaltet. Dabei werden kurze, invertierte Listen in Speichern fester Größe verwaltet, lange Listen werden in zusammenhängenden Speicherbereiche variabler Größe abgespeichert. Jede invertierte Liste wird in einem Bucket hinterlegt, da sie zu Beginn noch, als kurze Liste, hineinpasst. Doch mit dem Hinzufügen neuer Terme und dem Wachstum der Liste droht ein Überlauf des Buckets. Ist der Überlauf geschehen, wird die älteste darin enthaltene Liste zu einer langen Liste. Als Datenstruktur für einen Index werden meist verkettete Listen genutzt. Diese ermöglichen eine dynamische Speicherbelegung und ein einfaches Hinzufügen von Ausdrücken in Dokumenten.

Indexoptimierung Wie schon erwähnt kann der Index durch viele Einträge stetig wachsen. In Kapitel 2 wurden Verfahren erläutert, die die Größe des Index reduzieren. Das Stemming kürzt Wörter auf ihre Stammform ab und fasst diese in einem Wort zusammen (z.B. wird "neues", "neuem", "neuesten" abgekürzt zu "neu"). Wenn man in der Suchanfrage den Begriff "neu" eingibt, werden auch all die Dokumente ausgegeben, in denen das Wort neu mit verschiedenen Endungen auftritt. Durch das Auslassen von Präpositionen, Artikeln, also sogenannten Stopwörtern, verringert sich der Speicherbedarf des Index.

Kompressionsverfahren

Es gibt aber auch andere Ansätze mit denen man einen Index klein halten kann. Und zwar sind dies Komprimierungsverfahren, die mittels geeigneter Kodierung zu einem erwünschten Ergebnis führen. Diese Kodierungsmethoden ermöglichen eine Reduzierung der Textgröße auf fast 50 Prozent. Daraus resultiert eine geringere Auswertungszeit von Anfragen. Zusätzlich kann in kurzer Rechenzeit ein Index erstellt werden.

Golomb-Code Der Golomb-Code wurde 1966 von Solomon W. Golomb vorgestellt. Dabei handelt es sich um eine Darstellungsform für alle positiven Zahlen, einschliesslich der Null. Ein großer Vorteil an diesem Kodierungsverfahren ist die Abspeicherung von Daten unbekannter Größe. Es ermöglicht eine schnelle Kodierung und eine sparsame Nutzung des Speichers. Die Idee des Golomb Ansatzes besteht darin, die darzustellende Zahl n durch den Quotienten q, einen frei wählbaren Parameter b, und dem Rest r zu beschreiben. Dazu wird im ersten Schritt der Quotient der Zahl n berechnet mit der simplen Formel

$$q = \left\lfloor \frac{n-1}{b} \right\rfloor .$$

Das Ergebnis des Quotienten wird unär kodiert ausgegeben, an das Ende der Bitfolge wird eine logische 0 angehängt. Im zweiten Schritt bestimmt man den Rest r der Zahl n mit Hilfe der Formel:

$$r = n * (q * b) - 1.$$

Das Ergebnis wird nun binär ausgegeben. Zum Abschluss hängt man den zweiten Teil des Code-Wortes an das Ergebnis aus dem ersten Schritt und man erhält das fertige Code Wort. Zusätzlich zum Golomb-Code gibt es noch den Rice-Code. Der Rice-Code ist ein Spezialfall des Golomb-Code. Hierbei ist der Paramater b nicht frei wählbar, sondern eine Potenz von zwei.

Weitere Indexierungsverfahren

Selbstverständlich gibt es auch weitere Ansätze um Indexierung vorzunehmen. So sind das Suffix Array, der Suffix Tree und die Signature Files Verfahren mit denen man einen Index erzeugen kann.

Signature Files In einem Signature File können Einträge einer Datenbasis, welche eine bestimmte Bedingung erfüllen, effizient herausgefiltert werden. Vor allem große Textbestände sind innerhalb kurzer Zeit nach Schlüsselwörtern durchsuchbar. Die Signaturen selbst werden durch die Abbildung von Wörtern auf Bitstrings mittels Hash-Funktion erzeugt. Dabei werden die Dokumente in einer Textdatei, die Signaturen in einer separaten Datei abgespeichert. Ein Vorteil gegenüber dem Inverted File ist der geringe Speicherplatzbedarf. Jedoch ist dieses Verfahren ineffizient für große Datenbanken. Ein Nachteil sind die falschen Einträge, sogenannte False Drops, die herausgefiltert werden. Die Konstruktion eines Signature Files erfolgt dadurch, das Dokument in logische Blöcke zu unterteilen. Für jeden Block werden die einzelnen Signaturen gebildet, wobei jeder Block eine konstante Anzahl an Wörtern hat.

Suffix Array Ein Suffixarray ist ein Array, welches die Suffixe, in lexikographischer Reihenfolge angeben kann. Der Text ist als eine lange Zeichenkette abgespeichert. Jedes Teilstück des Textes ist eindeutig durch seine Position bestimmt, wobei die Positionen im Text frei wählbar sind. Der Vorteil des Suffixarrays als Index zu nutzen liegt darin, möglichst schnell jedes Teilwort innerhalb des Wortes zu lokalisieren. Durch die lexikographische Sortierung können die Suffixe effizient mit Hilfe der binären Suche gefunden werden.

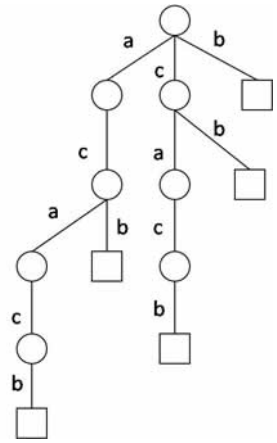


Abbildung 1.9: Suffix Baum

Suffix Tree Ein Suffix-Tree dient dazu, die Suffixe in einer baumartigen Struktur darzustellen. Die Suffixe enden als Blatt im Baum. Also hat ein Suffix-Tree soviele Blätter wie eine Zeichenkette Suffixe besitzt. Aus den inneren Knoten gehen mindestens zwei Kinder aus. Jede Kante des Suffix-Trees definiert ein Teilwort der Zeichenkette.

$$Beispiel : acacb$$

1. Suffix= cacb
2. Suffix= acb
3. Suffix= cb
4. Suffix= b

1.3.2 Suchmaschinen

Einführung

Neben den im vorherigen Abschnitt beschriebenen Indexierungstechniken zum Zugriff auf die gespeicherten Dokumente, benutzen moderne Suchmaschinen Ranking-Verfahren um die Ergebnisse einer Suchanfrage zu sortieren. Diese Verfahren werden als Page-Ranking-Verfahren bezeichnet, da sie die Linkliste der Ergebnisse einer Suchmaschine absteigend sortieren. Die Relevanz einer Page dient hierbei als Sortierkriterium. Dieses Kriterium soll möglichst objektiv und maschinell berechenbar sein und das menschliche Interesse sowie die Aufmerksamkeit widerspiegeln.

Page-Ranking-Algorithmen können in zwei Kategorien aufgeteilt werden. Semantisch unabhängige, vor dem Indizierungsschritt ausführbare Algorithmen, welche die Dokumente a priori mit einem statischen Rank versehen und semantisch abhängige Verfahren, welche nach der Indizierung die Dokumente aufwändig verarbeiten, die teilweise sogar zur Query Zeit ausgeführt werden.

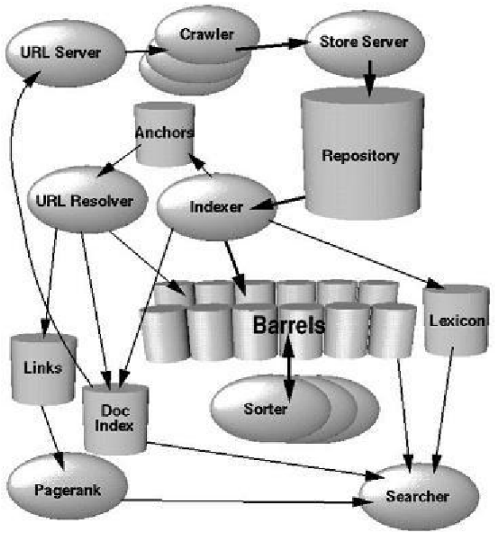


Abbildung 1.10: Google Systemarchitektur

Die statischen Ranking-Verfahren erzeugen ein globales Ranking von Webseiten durch Analyse der Verbindungsstruktur des Netzes. Sie betrachten nur die Topologie des Webgraphen und ignorieren die Inhalte und den semantischen Kontext einer Webseite. Die Verweise auf andere Webseiten sind die einzigen Informationsträger.

Die zweite Kategorie, der semantischen Page-Ranking Verfahren, nutzt Informationen der mit Meta-Daten angereicherten Dokumente. Durch aufwändiges Preprozessing extrahierte Meta-Daten der Inhalte sind hier die Informationsträger.

In diesem Abschnitt werden beide Ranking-Verfahren anhand zwei Suchmaschinen erläutert.

Google Architektur

Google ist zur Zeit die mit Abstand bekannteste und am häufigsten benutzte Suchmaschine. Diese positive Entwicklung resultiert aus dem ausgewogenem Mix aus Performance, Benutzerfreundlichkeit und Qualität der Suchergebnisse. Lawrence Page und Sergey Brin, die beiden Gründer von Google beschreiben in "Anatomy of a Large-Scale Hypertextual Web Search Engine" [BP98] den Aufbau und die Einsatzmöglichkeiten Ihrer Suchmaschine, sowie in "The PageRank citation ranking: Bringing order to the Web" [BP98] das PageRank Verfahren zur Analyse der Webtopologie.

Die System-Architektur von Google besteht aus mehreren miteinander kooperierenden Komponenten:

Die Crawler, ausgelegt auf größtmögliche Parallelität, wandeln URLs über den Domain Name Service (DNS) Requests in IP-Adressen um. Sie stellen mittels HTTP simultan, mehrere Verbindungen mit den Server her, um die erreichten Pages herunterzuladen.

Im Repository wird der komplette HTML-Code jeder heruntergeladenen Page mit zlib komprimiert gespeichert.

Die Aufgabe des Indexers besteht im dekomprimieren und analysieren der gespeicherten Pages. Jede heruntergeladene Page bekommt eine DocID. Zusätzlich werden Hits als Wortvorkommen in so genannten Hit Listen registriert. Die Hit Listen beinhalten Informationen über das Auftreten eines bestimmten Wortes in einem bestimmten Dokument, einschließlich Position, Schriftkegel und Hervorhebungen. Eine weitere wichtige Funktion des Indexers ist das Abspeichern aller Links Page in einer Anchor Datei. Die Sonderbehandlung von Link in Dokumenten ermöglicht Indexierung von Seiten, die aus Bildern, Programmen oder Datenbanken bestehen.

Für jedes Dokument werden zwei Indizes erstellt, der Forward-Index ist bereits teilweise sortiert und in einer Reihe von Barrels gespeichert. Die Barrels ermöglichen einen hohen Grad an Parallelität, der Index wird auf viele relativ kleine Barrels aufgeteilt. Jedes Barrel beinhaltet also nur eine kleine Anzahl an WordIDs. Enthält ein Dokument eine WordID, so wird seine DocID in dem entsprechenden Barrel gespeichert. Der Datensatz enthält zusätzlich eine Liste von WordIDs mit Hitlisten. Der Invertierte Index wird in den gleichen Barrels wie der Forward-Index gespeichert, allerdings ist er bereits durch den Sorter verarbeitet worden. Für jede gültige WordID enthält das Lexicon einen Zeiger auf das entsprechende Barrel. Dort befindet sich ein Zeiger auf eine DocList, die alle Vorkommen des Wortes in allen Dokumenten zusammenfasst.

Im Document-Index werden alle zusätzlichen Informationen über ein Dokument gespeichert. Dabei enthält jeder Eintrag den Dokumentstatus, einen Zeiger ins Repository, eine Dokumentprüfsumme und verschiedene Statistiken.

Die Sorter arbeiten kontinuierlich daran, den nach der DocID sortierten Forward Index in einen nach WordID sortierten Inverted Index umzuwandeln.

Während das gesamte übrige System kontinuierlich daran arbeitet, die Datenbasis zu aktualisieren und zu erweitern, ist der Searcher die einzige anfrageverarbeitende Komponente. Der Searcher verarbeitet Suchanfragen, erzeugt Ergebnislisten und sortiert sie nach Relevanz.

PageRank

Lawrence Page und Sergey Brin beschreiben in "The PageRank citation ranking: Bringing order to the Web" [BP98] das von Google verwendete PageRank Verfahren zur Analyse der Webtopologie. Dieses Verfahren basiert auf der Idee eines globalen Rankings von Webseiten, dass durch Analyse der Verbindungsstruktur des Netzes erzeugt wird. Die Autoren nehmen an, dass wichtige Seiten besonders oft von anderen Seiten verlinkt werden. Ein Webautor signalisiert durch einen Link auf eine andere Seite die Bereitschaft eine andere Quelle in den eigenen Inhalten zu zitieren.

Das Web ist ein mittels HTTP Protokoll realisiertes, virtuelles Netz aus HTML-Dokumenten, die durch Hyperlinks miteinander verbunden sind. Das Verfahren nutzt diese Struktur in dem es das Web als Graph betrachtet.

Webgraph - Es sei $G = (V, E)$ ein gerichteter Graph mit:

- V der Knotenmenge statischer Webseiten
- E der Kantenmenge aus Hyperlinks

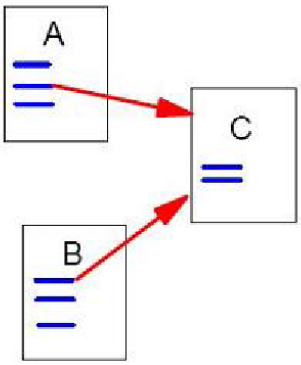


Abbildung 1.11: A und B sind Backlinks - C erreichbar durch Forwardlinks

Jeder Knoten des Webgraphen ist also ein HTML-Dokument, verbunden mit anderen Dokumenten durch Anchor Tags. Eingehende Kanten eines Dokuments bezeichnen wir als B Backlinks oder "inedges", ausgehende Kanten als F Forwardlinks oder "outedges".

Der PageRank Algorithmus betrachtet die Backlinks einer Webseite, allerdings werden nicht alle eingehenden Links gleich bewertet. Der Rank einer Seite ist die Summe der Ranks ihrer Backlinks.

PageRank Sei u eine Webseite und ein Knoten in G

- F_u die Menge aller Forwardlinks von u,
- B_u die Menge aller Backlinks von u,
- $N_u = |F_u|$ die Anzahl aller Nachfolger von u
- d soll Damping-Faktor zur Normierung des Ranks sein
- E sei eine Initialbelegung des statischen PageRanks über alle Seiten.

R ist eine Rankingfunktion von u und ihrer Vorgängerin v, der Form:

$$R(u) = d \sum_{v \in B_u} \frac{R(v)}{N_v} + dE(u)$$

Der PageRank einer Seite u wird also rekursiv aus der Summe der PageRanks derjenigen Seiten die auf u verweisen bestimmt. Allerdings wird nur eine einzige Verbindung zu u, anteilig zur der Menge aller Links betrachtet. Für die Analyse des Algorithmus betrachten wir Adjazenzmatrix A des Webgraphen G mit:

- $a_{v,u} = \frac{1}{N_v}$ falls eine Kante von v nach u existiert.
- $a_{v,u} = 0$ sonst.

PageRank Starte in $R_0 = E$ der Initialbelegung und laufe durch die folgende Schleife bis R_j konvergiert.

- $R_{i+1} \leftarrow AR_i$

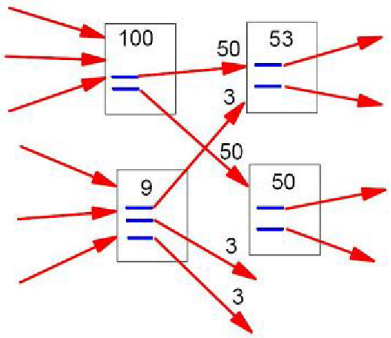


Abbildung 1.12: Eine Iteration der vereinfachten Rankingfunktion

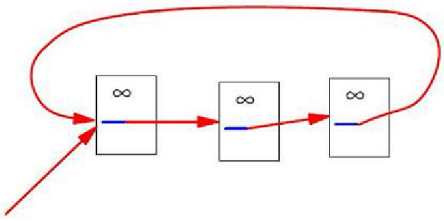


Abbildung 1.13: Ein Kreis im Webgraph: Rank Sink

- $d \leftarrow ||R_i||_1 - ||R_{i+1}||_1$
- $R_{i+1} \leftarrow R_{i+1} + dE$

Ohne die Initialbelegung und den Damping-Faktor würde der PageRank bei jeder Iteration zu den Senken fliessen. Also zu Seiten die auf keine anderen Seiten verlinken (Dangling-Links) oder auf dem Webgraph Kreise bilden (Rank Sinks). Um den negativen Effekt der Dangling-Links entgegenzuwirken, werden die PageRanks dieser Seiten erst berechnet, wenn die eigentliche Berechnung bereits abgeschlossen ist. Seiten die durch gemeinsames Verlinken

auf dem Webgraph Kreise bilden und nur auf sich selbst verweisen, erzeugen eine Schleife in der der Rank steigt aber nicht weiter verteilt wird, da dieser Kreis keine ausgehenden Kanten besitzt.

Der Damping-Faktor, der stets zwischen 0 und 1 liegt, wirkt diesem Problem entgegen und verringert den Ausmaß der PageRank-Weitergabe.

Das statische PageRank-Verfahren von Google ist Unabhängig von den Inhalten einer Webseite. Der Algorithmus konvergiert schnell und liefert selbst nach wenigen Iterationen gute Ergebnisse. Nach 45 Iterationen ist die PageRank-Berechnung von 161 Mio. Links abgeschlossen. Bei Verdoppelung der Anzahl von Links konvergiert der Algorithmus bereits nach 52,5 Iterationen. Der Algorithmus wird zur Datenaufbereitung ausgeführt und schont Ressourcen die zur Query-Zeit benötigt werden.

Zusätzlich lässt sich das Verfahren, wie in "A Unified Probabilistic Framework for Web Page Scoring Systems" von Diligenti, Gori etal. [al.04] beschrieben, auch probabilistisch als Random Surfer Modell interpretieren.

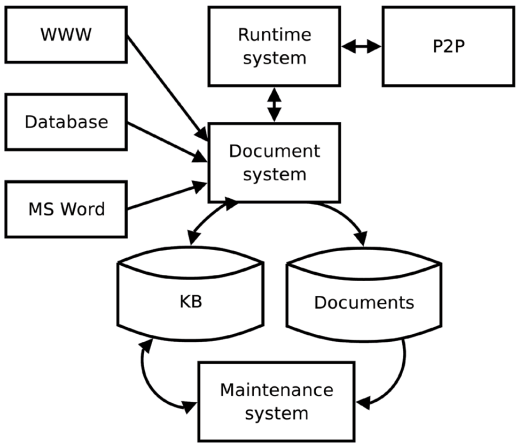


Abbildung 1.14: Architektur einer ALVIS Node

ALVIS Superpeer Semantic Searchengine

Der vorangegangene Abschnitt beschäftigte sich mit Suchmaschinen, die auf möglichst genaue Indexierung und Ranking angewiesen sind. Dieser Abschnitt beschäftigt sich mit einer der wenigen Suchmaschinen, die Methoden der Verarbeitung von natürlichen Texten nutzen um Inhalte zu analysieren und eine Wissensdatenbank aufzubauen. Das Open Source Projekt ALVIS gehört zu dieser neuen Kategorie der Suchmaschinen, die als Suchmaschinen der dritten Generation, oder semantische Suchmaschinen bezeichnet werden.

Das nach einem allwissendem Zwerg aus der nordischen Mythologie benannte Projekt wurde am 1. Januar 2004 ins Leben gerufen. Mehrere internationale Institute und Firmen beteiligten sich während der Laufzeit an der Forschung und Entwicklung. ALVIS, Superpeer Semantic Searchengine entstand unter der wissenschaftlichen Führung von Wray Buntine und wurde von der Technischen Universität Helsinki koordiniert. Das Projekt nutzt Open Source Software und wurde zum größten Teil auch als Open Source veröffentlicht. Der durchgehende Open Architecture Ansatz, der XML als Datenformat benutzt, garantiert hohe Interoperabilität mit anderen Systemen. Das Projekt wurde nach drei Jahren erfolgreich abgeschlossen und kostete über 3,5 Millionen Euro. Der gesamte Projektverlauf wurde auf www.opensourcesearchengine.org Dokumentiert und kann dort nachgeschlagen werden. [al.07]

ALVIS ist eine themen-orientierte, dezentrale, verteile Suchmaschine. Einzelne Instanzen dieser Suchmaschine, genannt Nodes, sammeln fokussiert Informationen über Dokumente und Webseiten aus ihrer Wissensdomäne. Wie in jeder anderen Suchmaschine, können diese Informationen vom Benutzer über ein übliches Web Formular abgefragt werden. Das Ergebnis besteht aus einer nach Relevanz sortierten Linkliste. Zusätzlich stellt eine Node, mittels peer-to-peer Technologie, ihr Wissen anderen Nodes zur Verfügung. Diese Form von der Zusammenarbeit über peer-to-peer ermöglicht domänenübergreifendes, verteiltes Suchen.

Abbildung 1.14 zeigt eine typische Node, die grob aus vier Subsystemen besteht. [WLB05]

- Das „Document system“ ist für die Sammlung und Indexierung von Dokumenten verantwortlich.

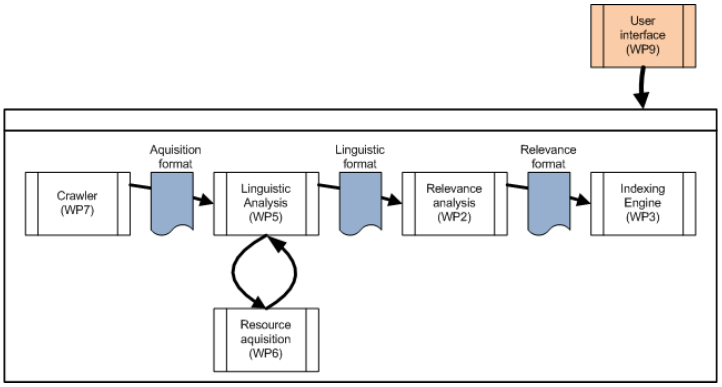


Abbildung 1.15: Document processing pipeline

- Das „Maintenance system“ erweitert und pflegt die Knowledge Base (KB).
- Das „Runtime system“ stellt Dienste die mit dem Benutzer interagieren und Fragen beantworten.
- Das „P2P system“ ist eine Schnittstelle des „Runtime system“ und stellt die Kommunikation mit anderen Nodes sicher.

Wie bereits erwähnt, findet die Kommunikation mit dem Benutzer, sowie anderen Nodes im „Runtime system “ statt. Das „Runtime system “ ist damit ein Server Dienst, der ständig zur Verfügung gestellt wird. Dieser Dienst greift, um Anfragen beantworten zu können auf das „Document system “ zu. Das „Document system “ ist für die Sammlung und Speicherung von Dokumenten verantwortlich. Die Akquise neuer Dokumente geschieht über einen fokussierten Crawler. Als Quellen dienen Webseiten, Datenbanken und MS Word Dokumente. Bei der Akquise verhält sich eine Node wie jede andere Suchmaschine, sie durchforstet die Quellen und indexiert die heruntergeladenen Daten. Diese werden in ein standarisiertes Format transformiert und im Document repository abgespeichert.

Anders, als eine gewöhnliche Suchmaschine, nutzt sie bei der Datenaufbereitung bereits erworbenes Wissen um semantische Informationen hervorzuheben. (Cite: Wray Buntine etal., Final Activity Report, 2007.) Dies geschieht durch möglichst präzises Tagging der gelesenen Webseiten. Gefundene Dokumente werden mit Hilfe der „Document processing pipeline“ verarbeitet und im „Documents repository“ abgelegt. Das „Document system“ nützt also die Knowledge Base als Ressource. Das „Maintenance system“ wird periodisch bei Bedarf eingeschaltet um aus den neu ankommenden Dokumenten Informationen zu extrahieren und sie in der Knowledge Base zu speichern. Die so erzeugten linguistischen Ressourcen werden wiederum vom „Document system“ zur Analyse der eingehenden Dokumente genutzt.

ALVIS Document processing pipeline

Das Herzstück von ALVIS ist die „Document processing pipeline“, sie spielt eine zentrale Rolle in der Verarbeitung von Dokumenten innerhalb des „Document system“. (Abbildung 1.15) Sie besteht aus vier Stufen, die jeweils, wie in einer Pipeline üblich, die Ergebnisse des

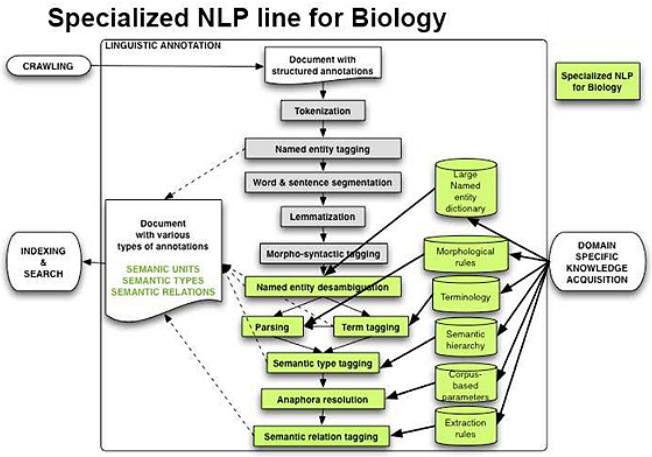


Abbildung 1.16: Eine spezialisierte NLP Line

Vorgängers nutzen. Gemäß dem dem Open Architecture Ansatz ist XML das gegebene Dokumentenformat. Jede Stufe erzeugt ein XML Dokument, welches mittels XLS Transformation in ein vom Nachfolger lesbares Format transformiert wird. Die Pipeline soll in der gesamten Verarbeitung die Dokumente anreichern und keine im Dokument enthaltenen Informationen entfernen.

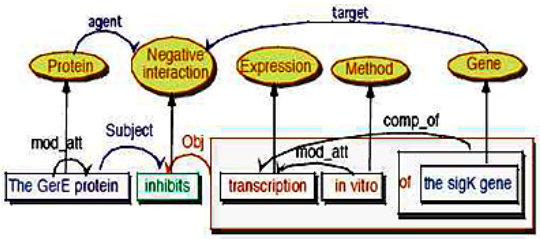
Die Document processing pipeline besteht aus vier hintereinander geschalteten Stufen. hinterfolgenden Komponenten:

WP7 Einem fokussierten Crawler, der auf ein bestimmtes Themengebiet konfiguriert werden kann, spürt auf und lädt die gefundenen Dokumente herunter. Die Inhalte werden in ein XML Dokument umgewandelt und mit ein paar Meta Daten versehen.

WP5 Der linguistischen Analyse (NLP) - Einer Komponente, welche für die linguistische Annotation eines Dokuments zuständig ist. Sie basiert auf der Natural Language Processing Platform. [AN06] Wie in der Abbildung 1.16 gezeigt, handelt es sich hierbei um einen linearen Prozess zu schrittweise Annotation von Webdokumenten unter Berücksichtigung von bereits vorhanden Ressourcen aus der Knowledge Base. Das Spektrum dieser Ressourcen sowie deren Aufbau ist sehr bereit gefächert und variiert je nach Spezialisierung der Suchmaschine. Man kann sie jedoch grob in Regelsammlungen, Ontologien und Wörterbücher aufteilen. Durch die Annotation werden Dokumente mit semantischen Informationen einer bestimmten Domäne angereichert. Die Abbildung 1.16 zeigt eine spezialisierten NLP Linie für Annotation im Bereich Biologie. Für das ALVIS Projekt wurden mehrere NLP Linien erstellt, unteranderem für die Annotation von englischen, chinesischen und slovakischen Texten. Durch die NLP Linie werden, je nach länge morphologische, syntaktische und semantische Informationen hinzugefügt.

WP6 Der Resource Acquisition, die mit WP5 kooperiert um, als semi-automatischer Sammler linguistischer und semantischer Daten, diese im Dokument repository abzulegen.

WP2 Einem Relevance System, der die Relevanz der vorhandenen Dokumente an Hand von Rankingverfahren analysiert. Zusätzlich berücksichtigt das Relevance System die linguistischen und semantischen Tags bei der Berechnung des Ranks.



Semantic relation	agent(Ger_protein, inhibit), target(sigK gene, inhibit), …
Semantic category	is_a(Ger_protein, protein), is_a(inhibit, negative_interaction), …
Syntactic dependency	subject(Ger_protein, inhibit), object(transcription, inhibit), …
Terminology	term(GerE protein), term(in vitro) term(sigK gene)
POS tag	cat(the, det), cat(GerE, nom), cat(inhibit, verbe), …
Named entity	entity(GerE, protein), entity (sigK, gene), entity (sigma K, protein)
Segmentation	word(the), word (Ger_protein), word (inhibit), word (transcription)

Abbildung 1.17: Linguistische Analyse eines Satzes

WP3 Schließlich ist die Indexing Engine für die Indexierung der Dokumente zuständig. Interessanter weise unterstützt sie die von den vorhergegangenen Stufen erzeugten und XML getaggten Dokumente.

WP9 Das User interface greift auf die von der Pipeline erzeugten Daten um Fragen zu beantworten. Wie bereits erwähnt handelt es sich hierbei um Formulare in einer Webseite.

Linguistische Analyse

Die linguistische Analyse von Dokumenten ist abhängig von der Spezialisierung der NLP, die morphologische Analyse ist zum Beispiel nur für die Slovenische NLP verfügbar. Die Named Entity Recognition ist für englische, chinesische, französische und auch für die spezialisierte, biologische NLP vorhanden. Abbildung 1.17 zeigt die einzelnen NLP Schritte in umgekehrter Rheinfolge, an Hand eines Satzes der durch die spezialisierte, biologische NLP annotiert wurde. Eine generalisierte NLP die alle verfügbaren Komponenten beinhaltet könnte folgendermaßen aussehen:

- Im ersten Schritt wird der Text üblicherweise segmentiert, also wortweise zerlegt,
- im zweiten Schritt erfolgt die NER,
- POS Tagging,
- Term Tagging,
- lexikalische Analyse und Chunking,
- dann werden die die semantischen Abhängigkeiten markiert.
- Schließlich folgt die Markierung von semantischen Typen und Relationen sowie die Auflösung von Anaphern.

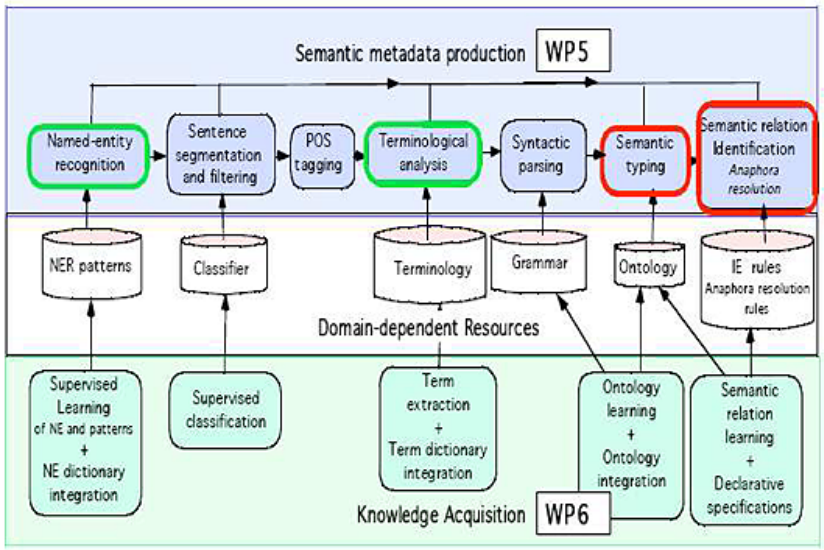


Abbildung 1.18: Zusammenspiel zwischen Document und dem Maintenance System.

Maintenance System

Wie im vorherigen Abschnitt beschrieben erzeugt das Document System am laufendem Band XML Dokummente die mit semantischen Meta-Daten annotiert sind. Auf der anderen Seite der Datenhaltung wird mit Methoden des maschinellen Lernens die vorhandene Knowledgebase ständig erweitert. Das hierfür zuständige Maintenance System erzeugt teilautomatisch die entsprechenden Datenbanken.

Das [WP6] Maintenance System besteht aus fünf Subsystemen die, wie in Abbildung 1.18 dargestellt, folgende Datenbanken mit Daten befüllen.

- Die NER Pattern Datenbank speichert alle Suchmuster die zur Name Entity Recognition benötigt werden. Das zuständige Subsystem, Supervised Learning of Name Entity patterns, arbeitet überwacht und enthält ein Wörterbuch mit möglichen Named Entities.
- Ein Classifier enthält alle Muster die zur Segmentierung von Sätzen verwendet werden können.
- Das Term extraction Subsystem erzeugt eine Terminologie.
- Um Sätze nach ihrer Syntax parsen zu können wird eine Sammlung von Grammatiken benötigt, diese wird von einem Subsystem vervollständigt, welches gleichzeitig für die Bildung der Ontologie zuständig ist. Mit der gebildeten Ontologie werden neue Ausdrücke gelernt und auf der anderen Seite die einzelnen Wörter ihrem semantischen Typ zugeordnet.
- Zusätzlich existiert eine Sammlung von Regeln zur Informationsextraktion und der Erkennung von Anaphern, diese wird von dem Semantic relation learning Subsystem erweitert.

Fazit

Eine verteilte, spezialisierte Suchmaschine die auf Peer-to-Peer Technologien aufbaut und einen gemeinsamen Index verwaltet ist eine robuste und elegante Alternative zu den monolithischen Strukturen auf dem Markt. Die Nutzung von maschinellen Lernverfahren in Bereich der Verarbeitung von Dokumenten ist neu und wird sicherlich in der Zukunft öfters in dem kommerziellen Sektor anzutreffen sein. Die ALVIS NLP Linie wurde in vielen Experimenten bestätigt. Sie verhält sich robust bei einer großen Anzahl von Dokumenten (55.000 Dokumente, mit über 100 Millionen Wörtern) selbst wenn die Dokumente in unterschiedlichen Sprachen geschrieben wurden. Die extensive Nutzung von spezialisierten Ressourcen wurde auf einem Corpus aus 300.000 wissenschaftlichen Zusammenfassungen im Bereich der Mikrobiologie getestet. Allerdings so groß wie diese Zahlen auch klingen mögen, die Zahlen sind sehr Klein im vergleich zu den Millionen an Dokumenten die im Web gefunden werden können.

Für die Verwendung innerhalb der Domäne des Bundestages müsste zunächst einmal eine spezialisierte NLP Linie für die Verarbeitung von politischen Texten erstellt werden. Dies wäre gleichbeutend mit Pionierarbeit auf zwei Gebieten, da keine deutsche NLP Linie exisitert. Zusätzlich müsste das gesamte Maintenance System auf die Erstellung von deutschsprachigen Ressourcen angepasst werden.

1.3.3 Lernende Suchmaschinen

Vorwort

Der vorliegende Abschnitt dient zu Lernenden Suchmaschinen. Er basiert auf einem Paper von Thorsten Joachims [Joa02]. Das klassische Anwendungsgebiet der Lernenden Suchmaschinen sind Literaturdatenbanken, die heute in Form Digitaler Bibliotheken zunehmend an Bedeutung gewinnen. Lernende Suchmaschinen sind besonders populär geworden durch die Anwendung in Internet-Suchmaschinen; dadurch kommt jeder Internet-Nutzer mit IR-Methoden in Berührung. Jeder, der eine dieser Anwendungen mehrmals genutzt hat, wird die wesentlichen Unterschiede zwischen Suchmaschinen-Anwendungen und deren klassischer Datenbanksysteme leicht erkennen:

- Meistens durchläuft der Prozess der Anfrageformulierung mehrere Iterationen, bis passende Antworten gefunden werden.
- Anfragen liefern potentiell sehr viele Antworten, aber nur wenige davon sind für den Nutzer interessant.
- Das vorgenannte Problem entschärft sich durch die vom System bereitgestellte Rangordnung der Antworten, wodurch potentiell relevante Antworten gehäuft am Anfang der Rangliste auftauchen (z.B. betrachten bei Internet-Suchmaschinen mehr als 90% aller Nutzer nur die ersten 10 Antworten)

Was ist eine lernende Suchmaschine?

Zur Definition des Gebietes legen wir hier die Beschreibung der Aufgaben und Ziele dar. Eine lernende Suchmaschine ist eine Suchmaschine mit künstlicher Intelligenz, die selbstständig lernt, für wen etwas interessant sein könnte, indem sie die Benutzer befragt.

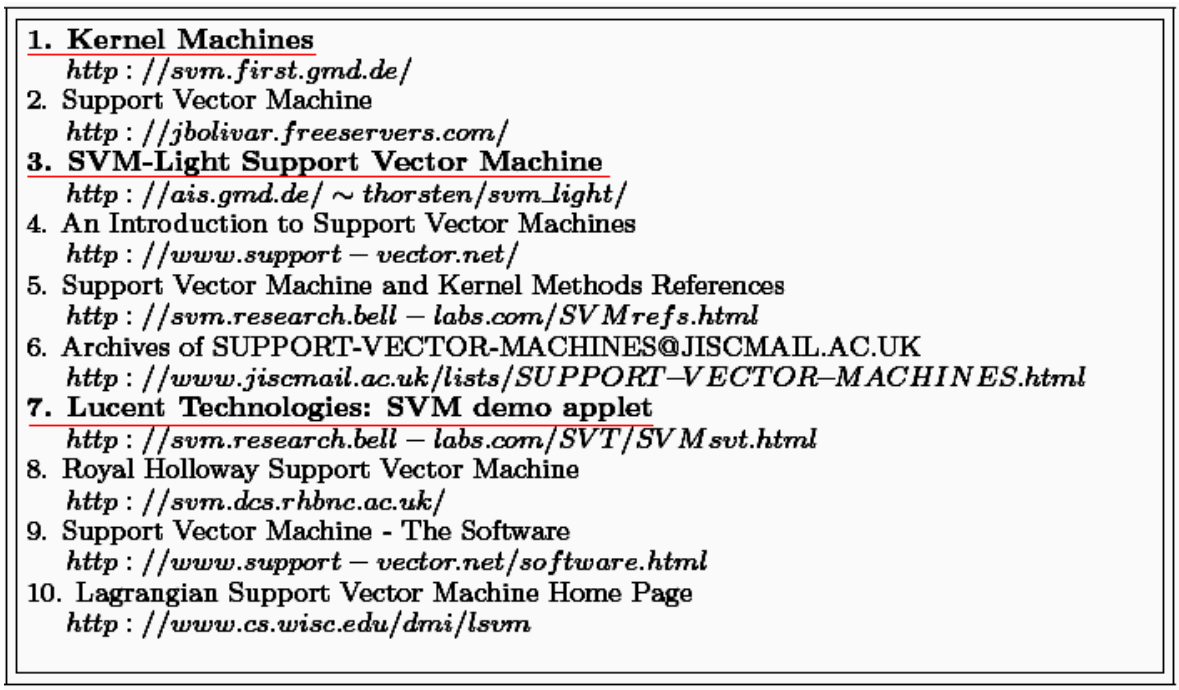


Abbildung 1.19: die Suchergebnisse mit der Anfrage 'support vector machine'

Trainingsdaten

Eine Lernende Suchmaschine braucht Trainingsdaten, damit sie lernen kann. Die Trainingsdaten können durch die Anfragen der Besucher automatisch erzeugt werden. Sie werden durch eine Datengruppe $Trio(q, r, c)$ dargestellt.

- q: Anfrage
- r: Die Rangliste der Suchergebnisse
- c: URL-Liste, die der Benutzer gewählt hat.

Um die Definition von $Trio(q, r, c)$ besser zu verstehen, sehen wir ein Beispiel mit Anfrage 'support vector machine'. Die Anfrage 'support vector machine' liefert 10 Antworten, aber nur $1, 3, 7$ sind für den Nutzer interessant (siehe Abb.1.19).
Bei dem Beispiel sind:

- q: 'support vector machine'
- r: Liste (1,2,3,4,5,6,7,8,9,10)
- c: (1,3,7)

Welche Informationen sind relevant?

Was können die lernenden Suchmaschinen durch die Trainingsdaten lernen? Bei dem Beispiel (Abb.1.19) hat der Benutzer $1, 3, 7$ gewählt. Es ist aber nicht sicher, dass $1, 3, 7$ absolut relevant sind. Aber man kann die relativ relevanten Dokumente bestimmen. Der Benutzer hat nach dem Rang2 den Rang3 gelesen. Ja, Rang3 ist wahrscheinlich relevanter als Rang2, da man ihn mit einer Relation r^* darstellen kann link3 $<r^*$ link2. Analog kann man die ganze Relation r^* aus dem Beispiel (Abb.1.19) bestimmen:

link3 $<r^*$ link2 link7 $<r^*$ link2
link7 $<r^*$ link4
link7 $<r^*$ link5
link7 $<r^*$ link6.
Die Relation r^* hat also partielle Information der gesuchten Sortierung der Suchergebnisse.

Darstellung der Relation $< r^*$

Für eine Suchergebnisliste gibt es m Dokumente $D = \{d_1, \ldots, d_m\}$. Die Relation r^* kann man mit einer Matrix M darstellen.
$((M \subset D \times D))$
$(d_i < r^* \ d_j) \Longrightarrow (M_{ij} = 1 \text{ und } M_{ji} = 0)$
Bei dem Beispiel (Abb.1.19)

$$M = \begin{pmatrix} - & * & * & * & * & * & * & * & * & * \\ * & - & 1 & * & * & * & 1 & * & * & * \\ * & 0 & - & * & * & * & * & * & * & * \\ * & * & * & - & * & * & 1 & * & * & * \\ * & * & * & * & - & * & 1 & * & * & * \\ * & * & * & * & * & - & 1 & * & * & * \\ * & 0 & * & 0 & 0 & 0 & - & * & * & * \\ * & * & * & * & * & * & * & - & * & * \\ * & * & * & * & * & * & * & * & - & * \\ * & * & * & * & * & * & * & * & * & - \end{pmatrix}.$$

(* unbekannt)

Die Funktion für lernende Suchmaschinen

Mit den Trainingsdaten müssen die lernenden Suchmaschinen eine Funktion finden, damit die Dokumente gut sortiert werden können. Um die Suchergebnisse zu verbessern, muss die Qualität der Funktion zu beurteilen sein. In der Statistik ist Kendall's τ sehr oft benutzt worden, um 2 Range zu vergleichen.
Für 2 Range $(r_a \subset D \times D)$ und $(r_b \subset D \times D)$,
P ist die Anzahl der übereinstimmenden Paare,
Q ist die Anzahl der nicht übereinstimmenden Paare, ist dann Kendall's τ definiert durch:

$$\tau = \frac{P - Q}{P + Q} = 1 - \frac{2Q}{\binom{m}{2}} \tag{1.34}$$

Um die Kendall's τ besser zu verstehen, sehen wir folgendes Beispiel für 2 Ränge r_a und r_b:
$r_a : d_1 <_{r_a} d_2 <_{r_a} d_3 <_{r_a} d_4 <_{r_a} d_5$
$r_b : d_3 <_{r_b} d_2 <_{r_b} d_1 <_{r_b} d_4 <_{r_b} d_5$.
Die Anzahl der nicht übereinstimmenden Paare Q ist 3 $(\{d_2, d_3\}, \{d_1, d_2\}, \{d_1, d_3\})$. Und die

Anzahl der übereinstimmenden Paare P ist 7. Damit kann man τ durch die Definition errechnen:

$$\tau = \frac{P-Q}{P+Q} = \frac{7-3}{7+3} = 0,4. \tag{1.35}$$

Wenn Q=0, hat τ Maximum 1. Und umgekehrt hat τ Minimum -1. $r_{f(q)}$ ist das System mit der die Frage q aus der Rangliste ausgelesen wird. $\tau(r_{f(q)}, r*)$ ist dann die Qualität der Suchergebnisse einer Funktion f nach der Anfrage q. Man kann die $\tau(r_{f(q)}, r*)$ als die Präzision der Suchergebnisse betrachten. Ist $\tau(r_{f(q)}, r*)$ groß, ist somit die Präzision der Suchergebnisse dann ebenfalls groß.
Wir sind jetzt in einer Position um das Problem näher zu definieren. Das Ziel ist die Funktion für die zu erwartenden Kendall's τ

$$\tau_{p(f)} = \int \tau(r_{f(q)}, r^*) d_{Pr(q,r^*)} \tag{1.36}$$

für eine feste, aber unbekannte Verteilung $Pr(q, r^*)$ von Abfragen und Zielgruppen eine Rangliste auf einer Dokumentensammlung D mit maimal m Dokumenten. Während das Ziel des Lerners nun definiert ist, bleibt die Frage offen, ob es möglich ist, die Funktion zu optimieren.

Der Algorithmus für lernende Suchmaschinen

Die meiste Arbeit der lernenden Suchmaschinen im Information Retrieval ist nicht die Formulierung von Dokumenten, sondern die vereinfachte Aufgabe, eine binäre Klassifikation von Problem mit den beiden Klassen "relevantünd "nicht relevant". Eine solche Vereinfachung hat mehrere Nachteile.
Die "nicht relevanten"Dokumente haben eine starke Mehrheit. Ein Lernender muss eine Regel haben, um die Dokumente maximal automatisch genau zu klassifizieren. Aber es ist noch wichtig, und gezeigt, dass solche absolute relevante Urteile nicht existieren, und nicht verfügbar sind. Deshalb sollen wir einen Algorithmus finden.
Für identische q und ihre r^*:

$$(q_1, r_1^*), \cdots, (q_n, r_n^*) \tag{1.37}$$

wählt Lerner L eine Funktion f, so dass

$$\tau_s(f) = \frac{1}{n} \sum_{i=1}^{n} \tau(r_{f(q_i)}, r_i^*) \tag{1.38}$$

maximiert wird. Ist es mäglich, einen Algorithmus und eine Familie von Rang Funktionen F zu entwerfen, so dass eine Funktion $f \in F$ Maximierung effiziente gefunden wird, und diese Funktion weit über den Trainingsdaten verallgemeinert werden kann. Betrachten Sie die linearen Funktionen

$$(d_i, d_j) \in f_{\bar{\omega}}(q) \Leftrightarrow \bar{\omega}\varphi(q, d_i) > \bar{\omega}\varphi(q, d_j). \tag{1.39}$$

ω ist ein Gewichtsvektor, und $\varphi(q, d)$ ist ein Diagramm mit der Dimension, die die Eigenschaft zwischen Anfrage q und Dokument d beschreibt.

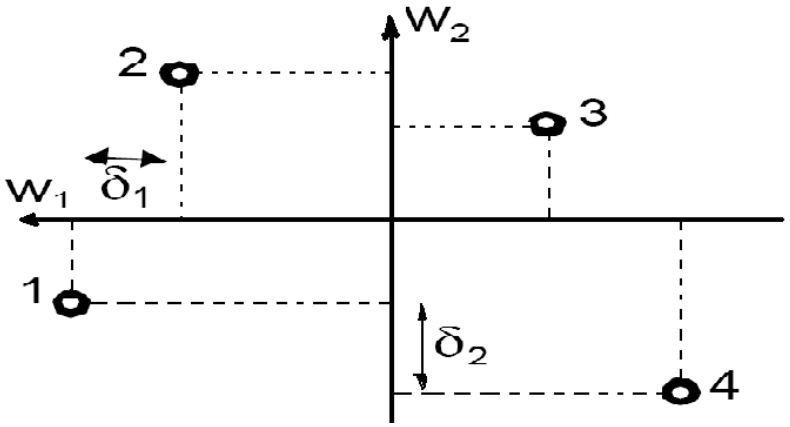


Abbildung 1.20: Anfrage - Dokument

Mit unterschiedlichen Gewichtsvektoren ω werden die Dokumente unterschiedlich sortiert (siehe Abb.1.20). Jetzt haben wir sofort eine Frage, wie kann man die Dokumente als Punkte auf einem Diagramm zeichnen? Das folgende Beispiel illustriert die Anwendung der Gewichtsvektoren bei Suchergebnissen.
Beispiel-Frage: 'side effect of drugs on memory and cognitive abilities, not aging'

t_i	ω_i	d_{1i}	d_{2i}	d_{3i}	d_{4i}
side effect	2	1	0,5	1	1
drugs	2	1	1	1	1
memory	1	1		1	
cognitive ability	1		1	1	0,5
not aging	-2		1		
Retrivalgewicht		5	2	6	4,5

Entsprechend den Retrivalgewichten werden die Dokumente in der Reihenfolge $d_3; d_1; d_4; d_2$ ausgegeben.
Die Maximierung (Formel 1.38) ist direkt äquivalent zur Minimierung der Zahl Q, der nicht übereinstimmenden Paare in der Formel (1.34). Für die Klasse der linearen Rangfunktionen (1.39)ist dies äquivalent dazu, um das Gewicht des Vektors zu finden, so dass die maximale Anzahl der folgenden Ungleichungen erfüllt ist.
$\forall(d_i, d_j) \in r_i^* : \bar{\omega}\varphi(q_1, d_i) > \bar{\omega}\varphi(q_1, d_j)$
...
$\forall(d_i, d_j) \in r_n^* : \bar{\omega}\varphi(q_n, d_i) > \bar{\omega}\varphi(q_n, d_j)$
Das Problem ist NP-Schwer. Aber genau wie bei der Klassifizierung des SVMs ist es möglich durch die Einführung von (nicht negativ) Schlupfvariablen $\xi_{i,j,k}$ und Minimierung $\sum \xi_{i,j,k}$ dieses Problem zu lösen.

(RANKING SVM)

$$\text{Min.}V(\bar{\omega}, \bar{\xi}) = \frac{1}{2}\bar{\omega} \cdot \bar{\omega} + C \sum \xi_{i,j,k} \tag{1.40}$$

mit Nebenbedingung:

$$\forall(d_i, d_j) \in r_i^* : \bar{\omega}\varphi(q_1, d_i) \geq \bar{\omega}\varphi(q_1, d_j) + 1 - \xi_{i,j,1} \tag{1.41}$$

$$\cdots \tag{1.42}$$

$$\forall(d_i, d_j) \in r_n^* : \bar{\omega}\varphi(q_n, d_i) \geq \bar{\omega}\varphi(q_n, d_j) + 1 - \xi_{i,j,n} \tag{1.43}$$

$$\forall i, j, k : \xi_{i,j,k} \geq 0 \tag{1.44}$$

C ist positive Konstante, die den Ausgleich zwischen der Minimierung von $\frac{1}{2}\bar{\omega}\cdot\bar{\omega}$ und der korrekten Klassifizierung der Trainingsbeispiele regelt. Durch die Neuordnung der Ungleichung (1.41) als

$$\bar{\omega}\varphi(q_1, d_i) - \bar{\omega}\varphi(q_1, d_j) \geq 1 - \xi_{i,j,1} \tag{1.45}$$

wird verdeutlicht, dass die Optimierung des Problems äquivalent ist zu einer Klassifikation von SVM auf paarweise unterschiedliche Vektoren $\bar{\omega}\varphi(q_1, d_i) - \bar{\omega}\varphi(q_1, d_j)$. Es kann gezeigt werden, dass die erlernten Retrieval Funktionen $f_{\bar{\omega}^*}$ immer als eine lineare Kombination der Vektoren vorkommen.

$$(d_i, d_j) \in f_{\bar{\omega}^*}(q) \tag{1.46}$$

$$\iff \bar{\omega}^*\varphi(q, d_i) > \bar{\omega}^*\varphi(q, d_j) \tag{1.47}$$

$$\iff \sum \alpha_{k,l}^*\varphi(q_k, d_l)\varphi(q, d_i) > \sum \alpha_{k,l}^*\varphi(q_k, d_l)\varphi(q, d_j) \tag{1.48}$$

Der Vektorraum und die zugehörigen Trainingsdaten sind durch eine Kernelfunktion $\sum \alpha_{k,l}^*\varphi(q_k, d_l)\varphi(q, d_i)$ in einen Raum mit so hoher Dimension zu überführen, dass sich die Trainingsdaten dort linear trennen lassen.

Experiment

Um die Qualität der verschiedenen Retrieval Funktionen vergleichen zu können, wird die Methode in [T. Joachims 2002] beschrieben. Die zentrale Idee ist es, zwei Rankings in der gleichen Zeit zu vergleichen. Man kombiniert zwei Ranglisten A und B zu C. Es muss die folgende Bedingung gelten: Die oben erwähnten L Links der kombinierten Ränge C enthalten die oben genannten K_a Links von A und der oben erwähnten K_b Links von B mit $|K_a - K_b| \leq 1$. Mit anderen Worten, wenn der Benutzer die Links von C von oben nach unten scannt, hat er an einem beliebigen Punkt fast ebenso viele Links von der Spitze von A wie von der Spitze des B gesehen.

Ein Beispiel ist in Abb.1.21 gegeben. Die von zwei Retrieval Funktionen kombinierten Ergebnisse C werden an den Benutzer weitergeleitet. In diesem Beispiel hat der Benutzer auf den Links 1, 3 und 7 angeklickt. In diesem Beispiel muss der Benutzer die oben erwähnten 4 Links aus einzelnen Rankings gesehen haben, da er auf den Link 7 in der kombinierten Rangliste C angeklickt hat. Er hat auf 3 Links in der oberen 4 Links im Ranking A (1, 2 und 4) angeklickt, sondern nur auf 1 Link in Rang B (1). Es wird daraus geschlossen, dass die oben genannten 4 Links von A besser sind als jene von B für diese Anfrage.

Am Paper von Thorsten Joachims [Joa02] wird die lernende Funktion von SVM mit Google, MSNSearch und Toprank vergleicht. Die Ergebnisse siehe man im Tab.1.3.

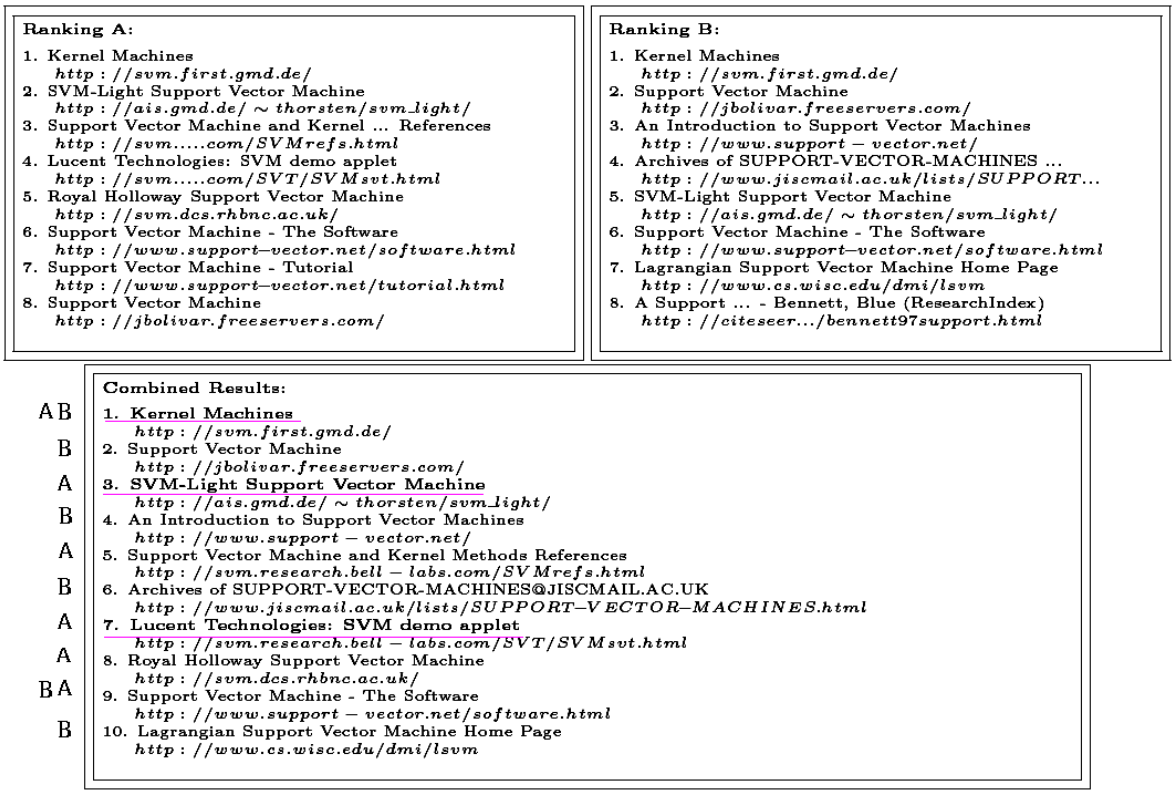


Abbildung 1.21: Ranking-merge Beispiel

Comparison	more clicks on learned	less clicks on learned
Learned vs. Google	29	13
Learned vs. MSNSearch	18	4
Learned vs. Toprank	21	9

Comparison	tie(with clicks)	no clicks	total
Learned vs. Google	27	19	88
Learned vs. MSNSearch	7	11	40
Learned vs. Toprank	11	11	52

Tabelle 1.3: Vergleich mit Suchmaschinen

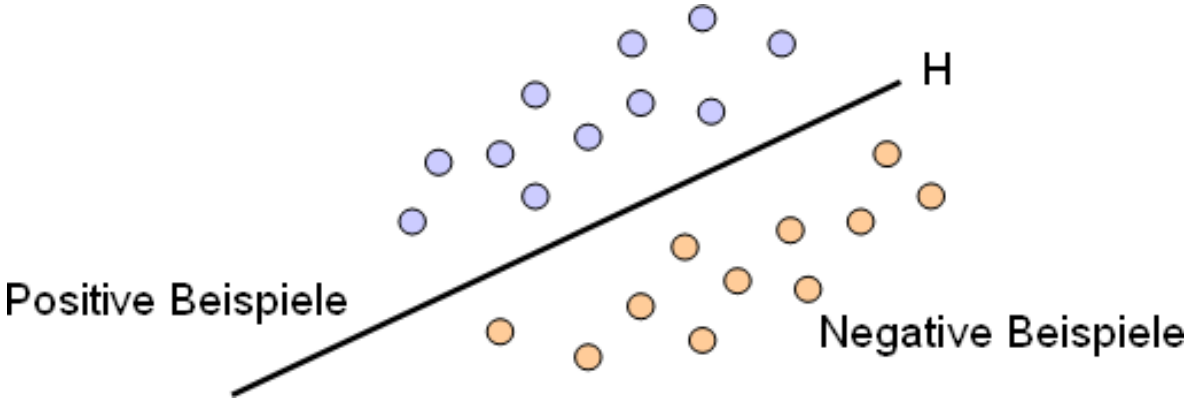


Abbildung 1.22: trennende Hyperebene

1.4 Maschinelle Lernverfahren

1.4.1 SVM

Eine Support Vector Machine (SVM) löst ein binäres Klassifizierungsproblem. Sie lernt auf einer bereits klassifizierten Menge von Trainigsbeispielen und entscheidet dann für Test-Elemente, zu welcher der zwei Klassen sie gehören. Das eigentliche Lernen besteht darin, eine Hyperebene H zu berechnen, die die Beispiele beider Klassen trennt. Danach kann anhand der Lage der Test-Elemente zur Hyperebene eine Klassifizierung erfolgen. Zunächst gilt es aber unter den vielen möglichen trennenden Ebenen die Optimale zu finden. Optimal wird dabei so verstanden, dass der Abstand von der Ebene zum nahesten Beispiel beider Klassen maximiert wird. Es liegt also ein Optimierungsproblem [Bur98] vor, welches nachfolgend formalisiert werden soll.
Es sei o.B.d.A. angenommen, dass die Beispiele in positive und negative unterteilt seien. Vorerst wird vorrausgestzt, dass die Trainingsdaten linear trennbar sind, es also eine trennende Hyperebene gibt. Der Fall linear nicht trennbarer Daten wird danach aufbauend auf diesen Erkenntnissen betrachtet.
(x_i, y_i) $(i = 1, ..., m)$ seien m Trainingsbeispiele, bestehend aus einem Punkt $x_i \in RR^d$ und der Klassifizierung[3] $y_i \in \{-1, 1\}$. $w \in RR^d$ sei der zu bestimmende Normalenvektor

[3]Als Klassifizierung wird die -1 und 1 statt z.B. 0 und 1 verwendet, weil sich damit, wie wir sehen werden,

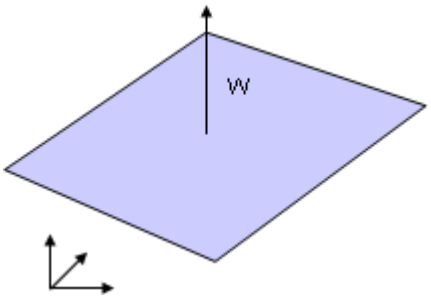


Abbildung 1.23: Hyperebene

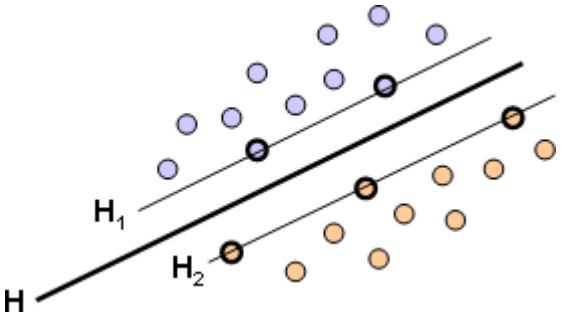


Abbildung 1.24: Breite der Hyperebene

der Hyperebene (siehe Abbildung 1.23), $b \in RR$ ihr Bias. Als Ebenengleichung wird die Variante mit dem Skalarprodukt gewählt, so dass alle Punkte x der Ebene die Gleichung $w \cdot x + b = 0$ erfüllen. Per Definition sollen die nahesten positiven Beispiele auf der Ebene H_1: $w \cdot x_i + b = 1$ und die nahesten negativen Beispiele auf der Ebene H_2: $w \cdot x_i + b = -1$ liegen (siehe Abbildung 1.24). Dies kann durch eine entsprechende Skalierung von w und b sichergestellt werden. Aus diesen Definitionen lassen sich direkt zwei Nebenbedingungen des Optimierungsproblems formulieren:

$$w \cdot x_i + b \geq 1, \qquad \forall i : y_i = 1 \tag{1.49}$$

$$w \cdot x_i + b \leq -1, \qquad \forall i : y_i = -1 \tag{1.50}$$

Die lässt sich zusammenfassen zu:

$$y_i(w \cdot x_i + b) - 1 \geq 0, \qquad \forall i \tag{1.51}$$

Ziel des Optimierungproblems ist die erwähnte Maximierung des Abstandes zu den nähesten Beispielen. Man spricht hier auch von der Breite (*margin*) der Ebene H, die bildlich den Abstand der Ebenen H_1 und H_2 aus Abbildung 1.24 repräsentiert. Dieser Abstand beträgt $\frac{2}{\|w\|}$, wobei $\|w\|$ die euklidische Länge von w darstellt. Zur Maximierung der Breite muss also $\|w\|$ minimiert werden. Eine SVM lässt sich dann durch folgendes Optimierungsproblem beschreiben:

$$\|w\|^2 \longrightarrow Min!$$
$$y_i(w \cdot x_i + b) - 1 \geq 0, \qquad \forall i$$

besser rechnen lässt.

Nun wird ersichtlich, warum die SVM *Stützvektor*-Maschine heißt: Die Hyperebene H wird durch Maximierung der Breite bestimmt. Die Breite hängt nur von den zur Ebene am nächsten gelegenen Beispielen ab, die ja die Ebenen H_1 und H_2 bestimmen. Die Beispiele, die auf einer dieser Ebenen liegen, stützen die Ebene H, da sich ohne sie die Lage (die Breite) von H verändern würde, und werden daher Stützvektoren genannt.
Um den Fall nicht linear separierbarer Daten besser handhaben zu können, ist eine Transformation des Problems in die duale Lagrange-Form nötig. Dazu werden die üblichen Lagrange-Multiplikatoren $\alpha_i \geq 0$ eingeführt und die Nebenbedingung in die Zielfunktion aufgenommen. Man erhält zunächst die Zielfunktion:

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^{m} \alpha_i y_i (w \cdot x_i + b) + \sum_{i=1}^{m} \alpha_i \tag{1.52}$$

Jedes Optimum der Zielfunktion muss auch ein Minimum der Funktion sein. Dies ist die eine notwendige Bedinung für die Eigenschaft als Optimum. Die Karush-Kuhn-Tucker (KKT) Bedingungen setzten diese Tatsachen in Gleichungen um. Damit ein Minimum vorliegt, müssen die ersten partiellen Ableitungen in Richtung der Entscheidungsvariablen w und b null sein. Die KKT Bedingungen für das Optimierungsproblem der SVM mit der Lagrange-Zielfunktion (1.52) lauten:

$$\begin{aligned}
\frac{\partial}{\partial w} L(w, b, \alpha) &= w - \sum_{i=1}^{m} \alpha_i y_i x_i = 0 \\
\frac{\partial}{\partial b} L(w, b, \alpha) &= -\sum_{i=1}^{m} \alpha_i y_i = 0 \\
y_i (w \cdot x_i + b) - 1 &= 0, \qquad \forall i \\
\alpha_i &\geq 0, \qquad \forall i \\
\alpha_i (y_i (w \cdot x_i + b) - 1) &= 0, \qquad \forall i
\end{aligned}$$

Durch Umstellen der ersten beiden Bedingungen erhält man:

$$w = \sum_{i=1}^{m} \alpha_i y_i x_i \tag{1.53}$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{1.54}$$

Einsetzen von (1.53) und (1.54) in (1.52) liefert die Zielfunktion L_D des dualen Optimierungsproblems:

$$L_D(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \tag{1.55}$$

Die Zielfunktion hängt damit nur noch von α ab und verwendet als einzige nicht elementare Rechenoperation das Skalarprodukt. Nun erhalten wir das (einfachere) duale Optimierungs-

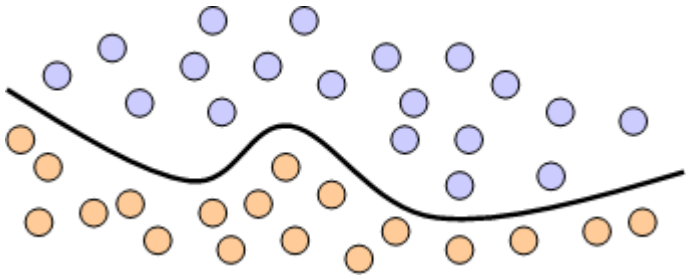


Abbildung 1.25: Nicht linear trennbare Trainingsdaten

problem für eine SVM auf Basis linear trennbarer Trainingsdaten:

$$\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (x_i \cdot x_j) \longrightarrow Max!$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0, \qquad \forall i$$

$$\alpha_i \geq 0, \qquad \forall i$$

Nach Lösung des Problems sind alle α_i bestimmt und der Normalenvektor w kann mit (1.53) berechnet werden. Den Bias erhält man, in dem man einen beliebigen Stützvektor in die Ebenengleichung von H einsetzt und nach b umformt. Normalenvektor, Bias und damit die Hyperebene sind also durch Linearkombinationen aus gerade den Beispielen x_i, für die $\alpha_i > 0$ ist, bestimmt. Diese Beispiele sind, wie wir schon gesehen haben, die Stützvektoren. Also gilt:

- $\alpha_i > 0$: x_i ist Stützvektor
- $\alpha_i = 0$: x_i ist kein Stützvektor

Nun können die Elemente der Test-Menge klassifiziert werden. Dazu genügt die Auswertung der Funktion

$$f(x) = w \cdot x + b \tag{1.56}$$

für ein Testelement x. Das Vorzeichen von $f(x)$ gibt dabei die Klassifizierung an, da nur folgende Fälle zu unterscheiden sind:

- $f(x) > 0$: x liegt im positiven Halbraum
- $f(x) < 0$: x liegt im negativen Halbraum
- $f(x) = 0$: x liegt auf H

Betrachten wir jetzt den Fall nicht linear trennbarer Daten, den Abbildung 1.25 schematisch zeigt. Was kann man tun, um dieses Problem zu lösen?

- Fehler zulassen
  Dies kann mit der Einführung einer *weich trennenden* Ebene gelöst werden, die fehlerhafte Klassifizierungen von Trainingsbeispielen bis zu einem gewissen Grad zulässt.
- Das Problem und die Daten so transformieren, dass sie linear trennbar werden.

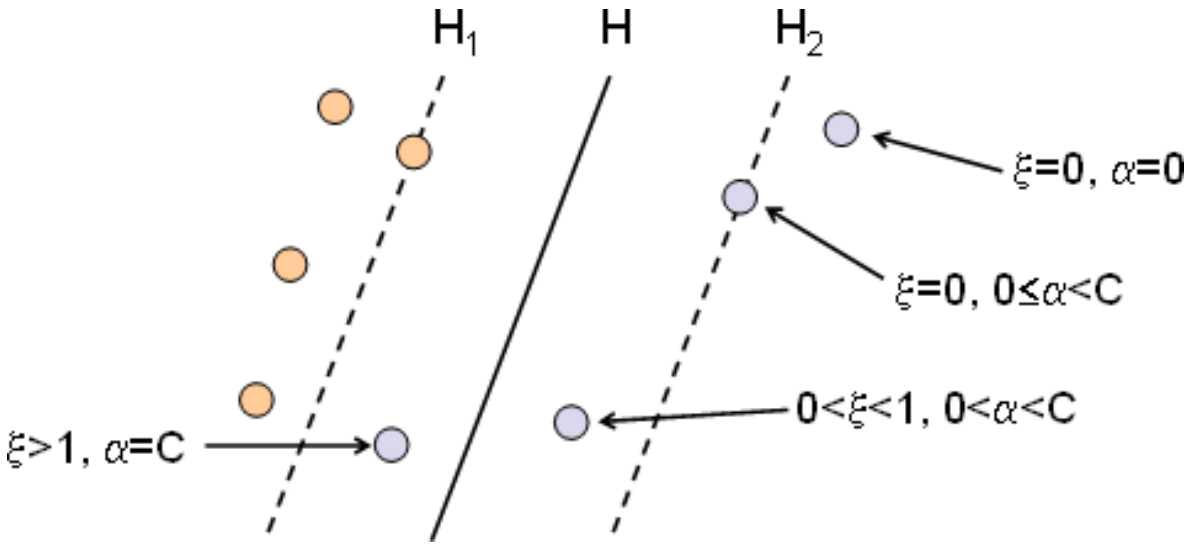


Abbildung 1.26: Beispiele für α-ξ Kombinationen

Die eine SVM mit weich trennender Hyperebene erweitert die bekannte SVM um die „Fehlerkosten“ $\xi_i \geq 0$ für jedes Beispiel. Die Nebenbedingungen 1.49 und 1.50 ändern sich dann zu:

$$w \cdot x_i + b \geq 1 - \xi_i, \qquad \forall i : y_i = 1 \tag{1.57}$$

$$w \cdot x_i + b \leq -1 + \xi_i, \qquad \forall i : y_i = -1 \tag{1.58}$$

Diese lassen sich wieder zusammenfassen zu:

$$y_i(w \cdot x_i + b) - 1 + \xi_i = 0, \qquad \forall i \tag{1.59}$$

Zu der ursprünglichen Zielfunktion werden die „Fehlerkosten“ einfach hinzu addiert. Weiterhin wird noch eine vom Benutzer zu wählende Konstante C eingeführt, die die Auswirkung der Fehlklassifikationen beeinflusst. Eine weich trenndende SVM wird dann durch folgendes Problem repräsentiert:

$$\begin{aligned} \|w\|^2 + C\sum_{i=1}^{m} \xi_i &\longrightarrow Min! \\ y_i(w \cdot x_i + b) - 1 = 0, &\qquad \forall i \\ 0 \leq \alpha_i \leq C, &\qquad \forall i \end{aligned}$$

Die Zielfunktion des dualen Problem ist wieder (1.52), da die ξ_i beim Bilden des dualen Problems verschwinden. Das bekannte duale Problem für linear trennbare Daten wird für die weich trennende SVM also nur um die Nebenbedingungen $0 \leq \alpha_i \leq C$ erweitert. Abbildung 1.26 verdeutlicht noch einmal die Bedeutung der α_i und ξ_i.
Die Transformation der linear nicht trennbaren Daten in linear trennbare gelingt ohne große Mühen, da in den entwickelten Formeln das Skalarprodukt die einzige nicht elementare Rechenoperation ist. Die Idee ist, die Beispiele durch eine Abbildung $\Phi : \ RR^m \longrightarrow \ RR^h$ mit

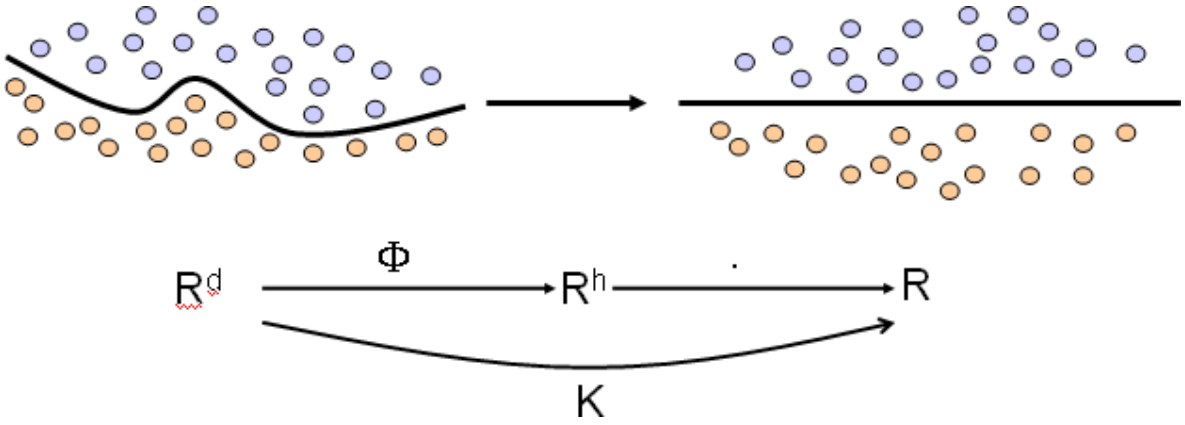


Abbildung 1.27: Der Kern-Trick

$h > m$ in einen höherdimensionalen Raum zu transformieren, wo sie dann linear trennbar sind. Die Zielfunktion (1.55) wird dann einfach angepasst:

$$L_D(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j (\Phi(x_i) \cdot \Phi(x_j)) \tag{1.60}$$

Sei $K(x, y) = \Phi(x) \cdot \Phi(y)$ eine Kernfunktion, die das Skalarprodukt zweier transformierter Trainingsdaten direkt berechne. Eine solche Funktion muss die Abbildung Φ gar nicht kennen. Die Abbildung 1.27 verdeutlicht diesen Gedanken, den man auch den Kern-Trick nennt.
Ein Beispiel:
Seien $x, y \in RR^2$ und $K(x, y) = (x \cdot y)^2$ eine Kernfunktion. Um zu zeigen, dass K das Skalarprodukt direkt berechnet, muss eine Abbildung Φ mit $\Phi(x) \cdot \Phi(y) = (x + y)^2$ gefunden werden. Wir nehmen an, dass die Funktion

$$\Phi(x) = \begin{pmatrix} [x]_1^2 \\ \sqrt{2}[x]_1[x]_2 \\ [x]_2^2 \end{pmatrix} \tag{1.61}$$

dieses tut. $[x]_i$ bezeichnet dabei die i-te Komponente von x. Beweis:

$$\begin{aligned} \Phi(x) \cdot \Phi(y) &= \begin{pmatrix} [x]_1^2 \\ \sqrt{2}[x]_1[x]_2 \\ [x]_2^2 \end{pmatrix} \cdot \begin{pmatrix} [y]_1^2 \\ \sqrt{2}[y]_1[y]_2 \\ [y]_2^2 \end{pmatrix} \\ &= [x]_1^2[y]_1^2 + 2[x]_1[x]_2[y]_1[y]_2 + [x]_2^2[y]_2^2 \\ &= ([x]_1[y]_1)^2 + 2[x]_1[y]_1[x]_2[y]_2 + ([x]_2[y]_2)^2 \\ &= ([x]_1[y]_1 + [x]_2[y]_2)^2 \\ &= (x \cdot y)^2 \end{aligned}$$

Wir sehen also, dass K das Skalarprodukt von x und y direkt berechnet, ohne dabei erst in den höherdimensionalen Raum abzubilden.

Setzt man die Kernfunktion in in die Zielfunktion (1.60) ein, erhält man zusammen mit den bekannten Nebenbedingungen das Optimierungsproblem für eine SVM mit nicht linear trennbaren Daten:

$$\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \longrightarrow Max!$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0, \qquad \forall i$$

$$0 \leq \alpha_i \leq C, \qquad \forall i$$

Mit der *Mercer's Condition* lässt sich überprüfen, ob eine Funktion $K(x, y)$ eine Kernfunktion ist:
Es gibt Abbildungen K und Φ mit $K(x, y) = \Phi(x) \cdot \Phi(y) = \sum_i [\Phi(x)]_i [\Phi(y)]_i \Leftrightarrow$
für jedes $g(x)$ mit finitem $\int g(x)^2 dx$ gilt

$$\int K(x, y) g(x) g(y) dx dy \geq 0$$

Die Lösung des Optimierungsproblems gestaltet sich als schwierig. Zwar liegt ein quadratisches, konvexes Optimierungsproblem vor, welches in $O(m^3)$ gelöst werden kann, jedoch ist dies in praktischen Anwendungen für eine sehr große Menge an Trainingsdaten langwierig. Erschwerend kommt hinzu, dass die Ungleichungen für große m nur mit numerischen Methoden gelöst werden können. Ein numerischer Solver muss als Unterprogramm aufgerufen werden. Im Folgenden werden zwei Lösungsalgorithmen vorgestellt, die das Optimierungsproblem heuristisch lösen.
Ein Verfahren ist das *Chunking*, bei dem der zentrale Gedanke darin liegt, Trainingsbeispiele, die keine Stützvektoren sind, aus der Berechnung auszuschließen, da sie, wir wir gesehen haben, die Lage der Hyperebene nicht beeinflussen. Eine Iteration läuft wie folgt ab:

1. Löse das Problem auf Basis einer Teilmenge der Trainingsdaten.
2. Betrachte die berechnten α_i-Werte:
   Ist $\alpha_i = 0$ ist x_i kein Stützvektor und wird von der weiteren Berechnung ausgeschlossen.
3. Nimm M Trainiungsbeispiele in die Berechnungsmenge auf, die die KTT-Bedingungen am schlechtesten erfüllen.

Über die Iterationen wird also versucht, die Stützvektoren zu identifizieren. In der letzten Iteration sind dann nur noch Stützvektoren in der Berechnungsmenge, so dass dann das gesamte Problem gelöst wird. Damit reduziert Chunking den Aufwand auf die quadrierte Anzahl der Stützvektoren. Dies kann erhebliche Vorteile bringen, aber auch genauso schlecht wie die Lösung ohne die Heuristik sein, wenn nahezu alle Trainingsdaten Stützvektoren sind. Definitv nachteilig ist aber, dass Chunking weiterhin nicht ohne Numerik-Solver auskommt. Genau diesem Nachteil nimmt sich das zweite Verfahren an: Die *Sequential Minimal Optimization* [Pla99], kurz *SMO*.
Während das Chunking immer mehrere α_i optimiert, beschränkt sich die SMO auf die Optimierung von genau zwei α_i in jeder Iteration. Der Vorteil ist, dass sich dies analytisch lösen lässt. Somit kann auf das Unterprogramm zur numerischen Lösung einer Gleichung verzichtet

werden. Insgesamt sind natürlich weit mehr Optimierungen nötig als beim Chunking, da alle α_i paarweise optimiert werden müssen. Trotzdem ist die Laufzeit besser als die des Chunkings, da die Numerik-Solver Aufrufe schwerer wiegen als das Mehr an Optimierungsiterationen (die alle in $O(1)$ gelöst werden können).
Der SMO-Algorithmus verwendet eine äußere und eine innere Schleife zur Auswahl der α_i-Paare. In der äußeren Schleife werden zunächst alle Beispiele durchlaufen und alle nicht gebundenen (non bound, d.h. α_i ist weder 0 noch C) Beispiele gesucht, die die KKT-Bedingungen verletzten. Das dazugehörige Alpha sei jeweils α_I, welches für die innere Schleife fest ist. Die innere Schleife findet nun unter den übrigen Beispielen ein α_{II} als Optimierungspartner. Als Auswahlkriterium wird der momentane Fehler

$$E_i = f(x_i) - y_i \tag{1.62}$$

genutzt. E_I sei der Fehler des Beispiels aus der äußeren Schleife. Die Fehler der übrigen Beispiele werden berechnet und das Beispiel, welches α_{II} liefert, wie folgt gewählt:

- $E_I > 0$: Wähle Beispiel mit kleinstem E_i
- $E_I < 0$: Wähle Beispiel mit größtem E_i

Für die genaue Vorgehensweise zur Optimierung des Paares (α_I, α_{II}) sei auf [Pla99] verwiesen. Nach der Optimierung wird die Ebene neu berechnet. Damit ändern sich auch die Ergebnisse von $f(x_i)$ bei der Fehlerberechnung aus (1.62). Die äußere Schleife läuft bis kein Beispiel mehr die KKT-Bedingungen verletzt. Danach werden noch einmal alle Beispiele durchlaufen und paarweise optimiert. Die Suche in der äußeren Schleife beginnt immer an einer Zufallsposition, damit sich die SVM nicht den ersten Trainingsdaten annähert.

1.4.2 SVM struct

Dieser Abschnitt basiert auf dem Artikel, welcher unter [THJA04] zu finden ist.

Einleitung SVMstruct soll auch komplexe Ausgaben miteinbeziehen können. Es soll eine Zuordnung von einer Eingabe $x \in \mathcal{X}$ zu einer diskreten Ausgabe $y \in \mathcal{Y}$ gefunden werden, welche auf einer Trainingsmenge von Eingabe-Ausgabe Paaren (x_1, y_1),...,$(x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$ basiert. $\mathcal{Y}$ ist dabei ein strukturierter Ausgaberaum. $y \in \mathcal{Y}$ können z.B. Sequenzen, Strings, Bäume, Gitter und Graphen sein. Derartige Probleme gibt es bei einer Vielzahl von Anwendungen. Durch eine Verallgemeinerung der Large Margin Methode (Standard Multiclass-SVM) sollen nun strukturierte Antworten gefunden werden.

Diskriminanten und Verlustfunktionen Das Funktionenlernen $f : \mathcal{X} \to \mathcal{Y}$, basiert auf einer Trainingsmenge von Eingabe-Ausgabe Paaren. Es soll eine Diskriminanzfunktion $F : \mathcal{X} \times \mathcal{Y} \to \Re$, über die Eingabe/Ausgabe Paare gelernt werden, von der Vorhersagen abgeleitet werden können, indem F über die Rückgabevariable für die Eingabe x maximiert wird. Formal ist dies wie folgt zu notieren:

$$\text{f(x;w)} = \arg\max_{y \in \mathcal{Y}} \text{F(x,y;w)} \tag{1.63}$$

mit w als ein Parameter-Vektor. F soll linear in seiner kombinierten Merkmalsrepräsentation von Ein- und Ausgaben $\Psi(x, y)$ sein, also soll gelten:

$$\mathrm{F(x,y;w)} = \langle w, \Psi(x, y)\rangle. \tag{1.64}$$

Die spezifische Form von Ψ hängt allerdings von der Art des Problems ab.

Beispiel 1 (natürlichsprachliche Syntaxanalyse) *F soll so gewählt werden, dass ein Modell entsteht, dass einer kontextfreien Grammatik entspricht. Jeder Knoten, im Syntaxbaum y für einen Satz x, entspricht einer Grammatikregel g_j mit den Kosten w_j. Alle gültigen Syntaxbäume y für den Satz x bekommen eine Punktzahl zugewiesen, die sich aus der Summe der w_j seiner Knoten berechnen lässt, also F(x,y;w) = $\langle w, \Psi(x, y)\rangle$. $\Psi(x, y)$ ist ein Histogrammvektor, der die Vorkommen jeder Grammatikregel g_j im Baum y zählt. f(x;w) kann effizient berechnet werden, indem die Struktur $y \in \mathcal{Y}$ gefunden wird, die F(x,y;w) maximiert.*

Das Lernen über strukturierten Ausgaberäumen $\mathcal{Y}$ bezieht andere Verlustfunktionen mit ein als die Standard 0/1 - Klassifikationsverluste.

Beispiel 2 (natürlichsprachliche Syntaxanalyse) *So ist es sinnvoll einen Syntaxbaum, der sich vom korrekten Syntaxbaum in nur wenigen Knoten unterscheidet, anders zu behandeln, als einen Syntaxbaum der sich von diesem vollkommen unterscheidet.*

Deshalb wird auch davon ausgegangen, dass eine beschränkte Verlustfunktion $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \Re$ vorhanden ist. Dies bedeutet, wenn y der wahre Ausgabewert ist, misst $\Delta(y, \hat{y})$ den Verlust der mit der Vorhersage $\hat{y}$ verbunden ist.

Von der SVM zur SVMstruct Zunächst wird der separierbare Fall betrachtet, dass der Trainingsfehler gleich null ist. Angenommen, dass $\Delta(y, y') > 0$ ist, wenn $y \neq y'$, und dass $\Delta(y, y) = 0$ ist. Die Bedingung, dass der Trainingsfehler null ist, soll nun kompakt dargestellt werden, als die Menge der nicht linearen Bedingungen

$$\forall i : \max_{y \in \mathcal{Y} \setminus y_i} \{\langle w, \Psi(x_i, y)\rangle\} < \langle w, \Psi(x_i, y_i)\rangle. \tag{1.65}$$

Jede dieser nicht linearen Ungleichheiten, kann durch $|\mathcal{Y}| - 1$ lineare Ungleichheiten ersetzt werden. Insgesamt gibt es also $n\,|\mathcal{Y}| - n$ lineare Bedingungen der Form

$$\forall i, \forall y \in \mathcal{Y} \setminus y_i$$

:

$$\langle w, \delta\Psi_i(y)\rangle > 0$$

, mit

$$\delta\Psi_i(y) \equiv \Psi(x_i, y_i) - \Psi(x_i, y)$$

.

Wenn die Menge dieser Ungleichheiten zulässig ist, dann gibt es typischerweise mehr als eine Lösung w^*. Um eine einzige Lösung zu erhalten, muss das w gewählt werden, für das sich die Punktzahl des korrekten Labels y_i konstant am meisten von dem nahesten zweitplatzierten unterscheidet, also

$$\hat{y}_i(w) = \arg\max_{y \neq y_i} \langle w, \Psi(x_i, y) \rangle$$

. Dies ist die Generalisierung des Maximum-Margin-Prinzips welches in der SVM angewandt wird. Das resultierende Hard-Margin-Optimierungsproblem, ist die SVM für linear trennbare Daten.

$$SVM^{linear} : \min_w \frac{1}{2}\|w\|^2,$$

mit den Nebenbedingungen

$$\forall i, \forall y \in \mathcal{Y} \setminus y_i : \langle w, \delta\Psi_i(y) \rangle \geq 1.$$

Um nun aber einen Fehler in der Trainingsmenge zu erlauben, wird eine Schlupfvariable eingeführt und zwar für jede nicht lineare Bedingung eine Schlupfvariable. Diese ist die obere Grenze des Trainingsfehlers. Das Einfügen einer Strafbedingung zu den Zielergebnissen die linear in den Schlupfvariablen ist, führt nun zur SVM für nicht lineare Daten.

$$SVM^{nonlinear} : \min_{w,\xi} \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i,$$

so dass $\forall i, \xi_i \geq 0$, mit den Nebenbedingungen

$$\forall i, \forall y \in \mathcal{Y} \setminus y_i : \langle w, \delta\Psi_i(y) \rangle \geq 1 - \xi_i.$$

Bisher gibt es aber noch keine neuen Erkenntnisse, welche es nicht schon im Kapitel 1.4.1 über SVM gab. Diese Formulierung soll jetzt aber für beliebige Verlustfunktionen Δ generalisiert werden. Dafür muss zunächst der Maßstab der Schlupfvariablen geändert werden und zwar gemäß des Verlustes der in jeder linearen Bedingung angefallen ist. Das Verletzen einer Margin-Bedingung die einen Ausgabewert mit hohem Verlust $\Delta(y_i, y)$ zur Folge hat, soll strenger bestraft werden, als ein Verstoß aus dem ein kleiner Verlust folgt. Dies wird erreicht, indem die Schlupfvariablen mit dem inversen Verlust skaliert werden. Dies führt nun endlich zur SVMstruct,

$$SVM^{struct} : \min_{w,\xi} \frac{1}{2}\|w\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i,$$

so dass $\forall i, \xi_i \geq 0$, mit den Nebenbedingungen

$$\forall i, \forall y \in \mathcal{Y} \setminus y_i : \langle w, \delta\Psi_i(y) \rangle \geq 1 - \frac{\xi_i}{\Delta(y_i, y)}.$$

SVMstruct Algorithmus Die Hauptaufgabe beim Lösen dieser verallgemeinerten SVM - Formulierung, liegt in der großen Anzahl der Margin-Bedingungen, denn die Gesamtanzahl der Bedingungen beträgt $n\,|\mathcal{Y}|$. Wobei $|\mathcal{Y}|$ in vielen Fällen extrem groß ist (z.B. beim Grammatik-Lernen). Daher sind Standardlösungen eher ungeeignet. Der SVMstruct-Algorithmus soll nun die Struktur des Maximum-Margin-Problems ausnutzen, also nur noch eine kleine Teilmenge

```
1  Input: (x_1, y_1), ..., (x_n, y_n), C, ε
2  S_i ← ∅ for all i = 1, ..., n
3  repeat
4   for i = 1, ..., n do
5    set up cost function [{z.B: SVM^s: H(y) ≡ (1 − ⟨δΨ_i(y), w⟩)Δ(y_i, y)}]
6    compute ŷ = arg max_{y∈Y} H(y)
7    compute ξ_i = max{0, max_{y∈S_i} H(y)}
8    if H(ŷ) > ξ_i + ε then
9     S_i ← S_i ∪ {ŷ}
10    α_S ← optimize dual over S, S = ∪_i S_i
11   end if
12  end for
13 until no S_i has changed during iteration
```

Listing 1.1: SVMstruct

von Bedingungen detailiert betrachten. Er soll eine kleine Menge von aktivierten Bedingungen finden, die eine ausreichend fehlerfreie Lösung garantieren. Dies ergibt eine gute und gültige Lösung, da es immer eine polynomiell große Teilmenge von Bedingungen gibt, so dass die zugehörige Lösung alle Bedingungen mit einer Genauigkeit von mindestens ϵ erfüllt. Alle anderen Bedingungen werden garantiert durch nicht mehr als ϵ verletzt, ohne dass sie dafür genauer betrachtet werden müssen. Der folgende Algorithmus kann auf alle SVM^{struct}-Formulierungen angewandt werden, die in [THJA04] vorgestellt werden. Es muss nur die Kostenfunktion in Schritt 5 angepasst werden.
Es gibt eine Arbeitsmenge S_i für jedes Trainingsbeispiel (x_i, y_i), welche in Zeile 2 definiert wird. Sie hält die aktuell ausgewählten Bedingungen fest. Beim Abarbeiten der Trainingsbeispiele wird die Bedingung gesucht, die den größten Verlust zur Folge hat (siehe Zeile 6). Wenn dessen Margin-Überschreitung den in Zeile 7 errechneten Wert von ξ_i, mit mehr als ϵ überschreitet (diese Abfrage ist in Zeile 8 zu finden), dann soll die Bedingung $\hat{y}$ zur Arbeitsmenge hinzufügt werden (vgl. Zeile 9). Dies entspricht einer sukzessiven Verstärkung des grundlegenden Problems, durch eine Schnittebene, die die aktuelle Lösung von der zulässigen Menge entfernt. Die gewählte Schnittebene entspricht der Bedingung, die den kleinsten zulässigen Wert für ξ_i bestimmt. Nachdem eine Bedingung zur Arbeitsmenge hinzugefügt wurde, muss die Lösung, in Zeile 10 (in Bezug auf S), neu errechnet werden. Der Algorithmus stoppt, wenn, wie in Zeile 13 gefordert, keine Bedingung durch mehr als ϵ verletzt wird und sich somit kein S_i geändert hat. Es ist leicht zu zeigen, dass der Algorithmus eine Lösung hat die nahe am Optimum liegt. Für eine große Klasse von Problemen konvergiert der Algorithmus in polynomieller Zeit trotz eventuell exponentiellen oder unendlichen $|\mathcal{Y}|$. Das liegt daran, dass die Anzahl der Bedingungen in S nicht von $|\mathcal{Y}|$ abhängig ist. Wenn die Zeile 6 des Codes also polynomiell ist, dann ist der ganze Algorithmus polynomiell.

Anwendungen und Experimente Dieser Ansatz ist effektiv und vielseitig. Er kann auf eine breite Klasse von Aufgaben angewandt werden, z.B. auf Multiclass Klassifikation, Klassifikation mit Taxonomien, Label Sequence Learning, Sequence Alignment und Natural Language Parsing. Um den Algorithmus an das neue Problem anzupassen muss lediglich eine Implementierung der Merkmalsabbildung $\Psi(x, y)$, der Verlustfunktion $\Delta(y_i, y)$ und der Maximierung des Verlusts (Zeile 6 des Codes) erfolgen.

Beispiel 3 (Label Sequence Learning) *Es soll eine Sequenz von Labeln $y = (y^1, ..., y^n), y^k \in \Sigma$, gegeben einer Sequenz von Eingaben $x = (x^1, ..., x^m)$, vorhergesagt werden. Diese Art der Klassifizierung wird z.B. bei der optischen Zeichenerkennung, der natürlichsprachlichen Syntaxanalyse, der Informationsgewinnung und der Bioinformatik angewandt. Hier wird das Label Sequence Learning am Beispiel NER betrachtet, welches bereits aus Kapitel 1.2.1 bekannt sein sollte. Es wird eine Sammlung von 300 Sätzen betrachtet. Die Zeichenmenge besteht aus Bezeichnungslosen und zusätzlich aus dem Anfang und der Weiterführung von Personennamen, Organisationen, Orten und diversen Namen. Insgesamt gibt es also 9 verschiedene Labels. $\Psi(x, y)$ ist in diesem Fall ein Histogramm der Zustandsübergänge und eine Menge von Merkmalen die die Ausgaben beschreiben. Tabelle 3 zeigt wie die unterschiedlichen Methoden bei diesem Datensatz abschneiden. Die Standard*

Methode	HMM	CRF	Perceptron	SVM
Fehler	9.36	5.17	5.94	5.08

Tabelle 1.4: Methodenvergleich

Hidden Markov Model Methode, welche bereits aus Kapitel 1.2.2 bekannt ist, wird von allen Lernmethoden deutlich übertroffen. Die Methoden Conditional Random Fields (welche bereits aus Kapitel 1.2.4 bekannt ist), Perceptron und SVM, schneiden fast gleich ab, wobei schon hier die SVM etwas besser ist als die anderen zwei Methoden. Die folgende Tabelle zeigt wie die SVM für nicht lineare Daten gegen die SVMstruct abschneidet. Die Ergebnisse der un-

Methode	Train Fehler	Test Fehler	Const	Verlust
$SVM^{nonlinear}$	0.2 ± 0.1	5.1 ± 0.6	2824 ± 106	1.02 ± 0.01
SVM^{struct}	0.4 ± 0.4	5.1 ± 0.8	2626 ± 225	1.10 ± 0.08

Tabelle 1.5: Vergleich der SVM Methoden

terschiedlichen SVM Methoden sind im großen und ganzen vergleichbar. Der Trainingsfehler ist für die $SVM^{nonlinear}$ etwas besser, der Testfehler hingegen ist nahezu identisch. Weniger Bedingungen werden bei der SVM^{struct} hinzugefügt und der durchschnittliche Verlust ist bei der $SVM^{nonlinear}$ etwas besser. Aber diese Unterschiede sind alle kaum nennenswert.

Ergebnis Als Resultat ist eine SVM für überwachtes Lernen mit strukturierten und voneinander abhängigen Ausgaben entstanden. Diese basiert auf einer Merkmalsabbildung über Eingabe/Ausgabe - Paare und deckt damit eine breite Klasse von Modellen ab. Das wären z.B. gewichtete kontextfreie Grammatiken, Hidden Markov Models und Sequence Alignment, um nur einige wichtige zu nennen. Dieser Ansatz ist flexibel im Umgang mit methodenspezifischen Verlustfunktionen und eine sehr positive Eigenschaft ist, dass nur ein allgemeingültiger Algorithmus an sämtliche Aufgaben herangehen kann. Was die Präzision des Ansatzes angeht, so ist diese für einen weiten Bereich mindestens vergleichbar mit den üblichen Ansätzen

und oft sogar besser. Was durch den Algorithmus aber garantiert wird ist, dass dieser nutzbar ist um komplexe Modelle zu trainieren, die ansonsten nur schwer im üblichen Rahmen behandelbar wären.

1.4.3 Clustering

Ensemble Clustering

Das Forschungsgebiet der Cluster Ensembles erarbeitet Verfahren zur Vereinigung mehrerer Objekt-Zuordnungen. Ein Clusterer ordnet jedem Objekt einer Menge maximal eine Gruppe zu. Die Aufgabe dieses Bereichs der Forschung liegt in der Vereinigung der Ergebnisse verschiedener Clusterer zu einer einzelnen Zuordnung, die mit den Originalzuordnungen eine möglichst hohe Ähnlichkeit besitzt. Besondere Bedeutung besitzen Algorithmen, die jene Aufgabe ohne Kenntnis über die Zuordnungskriterien durchführen, da derartige Verfahren wiederverwendbar und universell einsetzbar sind. Die Qualität der Verfahren kann unter verschiedenen Gesichtspunkten analysiert werden. Ein wesentliches Kriterium liegt trivialerweise in der Ähnlichkeit der neuen Zuordnung zu jenen aus der Eingabe. Wegen eventuell großen Datenmengen muss der Laufzeit in Abhängigkeit von der Eingabemenge ebenfalls eine große Bedeutung zugeordnet werden. Ein weiteres interessantes Kriterium, das nur eine untergeordnete Bedeutung besitzt, liegt in der potentiellen Verwendung dieses Algorithmus auf einem verteilten System. Diese zusätzliche Möglichkeit kann die Laufzeit des Verfahrens allerdings nur um einen konstanten Faktor senken, sollte aber dennoch nicht völlig vernachlässigt werden.

Als Basis für eine effiziente Bearbeitung dient eine Transformation der Eingabedaten in eine geeignete Form für die Verarbeitung. Hierzu werden die Eingabedaten in eine Matrix transformiert, wobei die Zeilen den einzelnen Objekten entsprechen, die zugeordnet werden sollen und die Spalten den Clusterern, die eine Zuordnung erzeugen. Die Einträge der Matrix entsprechen folglich der Gruppe, die der Clusterer dem Objekt zugeordnet hat. Wegen der Wiederverwendbarkeit des Verfahrens können beliebige Identifikatoren verwendet werden. Die Matrix kann nun durch Erweiterung einer Spalte in einen Hypergraphen umgewandelt werden. Hierzu findet eine Ersetzung eines Clusterers durch die Gruppen, die jener erzeugt, statt. Die Einträge der Matrix werden nun durch eine 1 ersetzt, falls das Objekt jener Gruppe zugeordnet ist, ansonsten durch eine 0. Eine Spalte repräsentiert folglich nun eine Hyperkante in einem Hypergraphen. Jegliche Objekte, die der gleichen Gruppe zugeordnet sind, werden durch Hyperkanten verbunden.

Der Cluster-based Similarity Partitioning Algorithm (CSPA)[SG02] basiert auf einer ziemlich einfachen Idee. Die einzelnen Objekte werden miteinander verglichen und die Ähnlichkeit dieser Objekte ermittelt. Diese paarweise Betrachtung der einzelnen Objekte kann je nach Anwendungsgebiet auch als Schwachpunkt dieses Verfahrens angesehen werden. Die Ähnlichkeit wird durch eine Matrixmultiplikation bestimmt. Hierzu wird die Matrix des Hypergraphen mit seiner transponierten Matrix multipliziert. Anschließend wird die Ergebnismatrix mit dem Inversen der Anzahl der Clusterer multipliziert. Jeder Eintrag wird somit auf einen Wert zwischen null und eins normiert, da jedes Objekt nach Definition pro Clusterer nur einer Gruppe zugeordnet werden kann und sich folglich maximal Anzahl der Clusterer Einsen pro Zeile der Originalmatrix bzw. pro Spalte der transponierten Matrix befinden. Nach Ausführung dieser Schritte wurde das Ursprungsproblem auf das Problem eines ähnlichkeitsbasierten Zu-

ordnungsalgorithmus reduziert. Die Matrix bestehend aus den paarweisen Ähnlichkeiten der Objekte kann nun mit Hilfe eines solchen Algorithmus zum Beispiel METIS in die gewünschte Anzahl von Cluster umgruppiert werden.

Der HyperGraph-Partitioning Algorithm (HGPA)[SG02] versucht die Schwachstelle des Cluster-based Similarity Partitioning Algorithm mit der paarweisen Betrachtung zu beheben. Dieser Algorithmus arbeitet, wie schon der Name verrät, auf einem Hypergraphen. Eine Spalte in der Matrix repräesentiert eine Hyperkante innerhalb dieses Hypergraphen. Aus diesem Graphen sollen möglichst wenige Kanten entfernt werden um die anvisierte Anzahl von unverbundenen Komponenten zu erreichen, wobei jede Kante die gleiche Gewichtung besitzt. Analog zum CSPA wird das Problem auf einen anderen Problemtyp reduziert, für den beispielsweise mit Hilfe des hMETIS-Algorithmus gute Ergebnisse erzielt werden können. Eine weitere Anforderung liegt in der annähernd gleichen Größe der unverbundenen Komponenten, die eine höherer Qualität des Ergebnisses garantieren soll. Eine kleine Schwäche dieses Verfahrens mit der anschließenden Verwendung des hMETIS-Algorithmus ist eine fehlende Berücksichtigung der Schnitthäufigkeit einer zertrennten Kante.

Der Meta-Clustering Algorithm (MCLA)[TJP04] basiert ebenfalls auf der Reduktion des eigentlichen Problems auf einen ähnlichkeitsbasierten Zuordnungsalgorithmus. Hierzu werden für alle Gruppen der Clusterer Knoten erzeugt. Die Kanten zwischen den einzelnen Gruppen werden hierbei mit ein Kantengewicht belegt, das die Ähnlichkeiten dieser Gruppen wiedergibt. Ein etabliertes Verfahren zur Ermittlung der Ähnlichkeit zwischen zwei Gruppen ist das Jaccard Measure. Dieser Maßstab wird durch den Quotienten aus den gemeinsamen Objekten und den Objekten, die mindestens in einer Menge vorkommen, ermittelt. Im Gegensatz zum CSPA, bei dem auf Basis der Ähnlichkeiten der Objekte neue Gruppen erzeugt werden, dient hier die Gruppenähnlichkeit als Eingabe für den Algorithmus. Nachdem die neuen Gruppen, die eine Vereinigung mehrerer ursprünglicher Gruppen sind, erzeugt wurden, wird jedes Objekt einer dieser Gruppe zugeordnet. Das Kriterium für die Zuordnung stellt der Quotient aus Anzahl der Unsprungsgruppen, in denen es vorkommt, und der Anzahl der Gruppen, die vereinigt wurden.

Beim Estimation-Maximization-Algorithm (EM-Algorithm) handelt es sich um einen k-means Algorithmus. Zu Beginn des Algorithmus werden Clusterzentren an beliebigen Position des Arbeitsraumes festgelegt. Die Anzahl der Clusterzentren entspricht der Gruppenanzahl, die erzeugt werden sollen. Jedes Objekt wird mit einer Wahrscheinlichkeit dem Zentrum mit der geringsten Distanz zugeordnet. Als Verfahren für die Ermittlung des Wahrscheinlichkeitwertes wird die Standardnormalverteilung empfohlen. Nachdem diese Zuordnung für alle Objekte vollzogen wurde, können anhand der enthaltenen Objekte und der Wahrscheinlichkeit ihrer Zugeh¨örigkeit neue Clusterzentren ermittelt werden. Mit diesen neuen Clusterzentren werden alle Objekte erneut zugeordnet. Dieser Vorgang wird wiederholt bis ein definierter Schwellenwert unterschritten wurde oder eine definierte maximale Anzahl an Wiederholungen erreicht ist. Diese zusätzliche Bedingung ist notwendig, da dieser Algorithmus ansonsten nicht zwangsläufig terminiert.

Der Cluster-based Similarity Partitioning Algorithm wird besonders bei großen Objektmengen sehr ineffizient, wenn nicht sogar unpraktikabel. Die Matrixmultiplikation ist zwar ein einfaches Verfahren besitzt jedoch eine Laufzeit von $O(n^2kr)$, wobei n der Anzahl der Objek-

te, k der Anzahl der Clusterer und r der Gruppen pro Clusterer entspricht. Zudem ist wegen der erzeugten n × n Matrix eine hohe Speicherbelastung gegeben. Im Gegensatz dazu ist der HyperGraph-Partitioning Algorithm sehr sparsam im Ressourcenverbrauch mit seiner Laufzeit von $O(nkr)$ und dürfte im Rahmen dieser Projektarbeit mit vielen Objekten eine größere Bedeutung besitzen. Der Meta-Clustering Algorithm ist mit seiner Laufzeit von $O(nk^2r^2)$ zwar langsamer als der HGPA, jedoch insbesondere bei wenigen Clusterern und wenig zugeordneten Gruppen interessant. Die Laufzeit des Estimation-Maximization-Algorithm $O(kNH)$ ist abhängig von der Anzahl der erzeugten Gruppen, Objekte und Anzahl der Wiederholungen bis zum Erreichen des Schwellenwertes bzw. der maximalen Wiederholungshäufigkeit.

Die Qualität der Ergebnisse differiert in den vorgenommenen Studien stark von den Eingabedaten. Zumal keinerlei Eingabedaten für ein vergleichbares Forschungsgebiet vorliegen, können hier nur schwer Ergebnisse prognostiziert werden.

Semi-supervised Clustering

Einleitung und Motivation In dieser Ausarbeitung zu dem Vortrag soll ein neues Verfahren für eine Clusteranalyse vorgestellt werden. Es handelt sich hierbei um ein Semi-supervised (halbüberwachtes) Clustering Verfahren, welches dem Nutzer erlaubt dem System Feedbacks zu geben und es ihm ermöglicht den Clusterprozess mitzusteuern. Die Feedbacks sind in Form von Constraints (Bedingungen) dargestellt. Es existieren natürlich schon andere Verfahren wie Supervised (überwachte) und Unsupervised (unüberwachte) Clustering.
Im ersten Abschnitt wird ein kurzer Überblick über die Clusteranalyse gegeben. In den darauf folgenden Kapiteln wird nun aufgezeigt, welche Eigenschaften Supervised, Unsupervised und Semi-supervised Clustering haben. Im sechsten Abschnitt werden die Verfahren bezüglich ihrer Performance verglichen und anschließend folgt die Zusammenfassung.

Clusteranalyse Die Clusteranalyse ist ein Verfahren um eine Menge von Objekten in passende Cluster einzuteilen. Hierbei sollen verborgene Muster und Strukturen in Daten erkannt und zusammengefasst werden. Die Standard Clusteranalyse ist das Unsupervised Clustering, bei dem Cluster automatisch gebildet werden. Dies geschieht mit Hilfe eines Ähnlichkeitskriteriums, wie z.B. dem Euklidschen oder Hammingschen Distanzmaß. Das Clusteringproblem existiert nicht nur im Web sondern auch in anderen Gebieten wie Biologie, Marketing usw..

Algorithmen Es existieren zwei große Algorithmengruppen, zum einen die hierarchischen Clustering Algorithmen wie z.B. Complete-Link-Algorithmus, Single-Link-Algorithmus und Agglomerierende, zum anderen die partitionierenden Clustering Algorithmen wie k-means oder EM.
Hierarchische Verfahren lassen sich in anhäufende Verfahren (agglomerative Clustering) und teilende Verfahren (divisive Clustering) unterscheiden. Bei den anhäufenden Verfahren werden schrittweise einzelne Objekte zu Clustern und diese zu größeren Gruppen zusammengefasst, während bei den teilenden Verfahren größere Gruppen schrittweise immer feiner unterteilt werden. Partitionierende Verfahren erzeugen nur eine einzige Partitionierung der Objektmenge und müssen meistens die Anzahl der gewünschten Cluster vorgeben.
Die k-means Methode allgemein(vgl. [Jos06]):

- Datenmenge in k Cluster partitionieren.

- Zentren von Clustern berechnen.
- Mit der euklidschen Distanz Abstände berechnen.
- Minimierung des Abstandes zum Zentrum.

Die agglomerative Methode allgemein(vgl. [Jos06]):

- Jedes Objekt bildet zu Beginn ein eigenes Cluster.
- In einer Abstandsmatrix werden paarweise Abstände gespeichert.
- In jedem Schritt werden Cluster, mit kleinsten Abstand zu einander, zu einem neuen zusammengefügt.
- (Abstandsmatrix anpassen)
- Algorithmus stoppt , falls sich alle Objekte in einem Cluster befinden.

Es sind natürlich noch andere Kriterien vorhanden wie(vgl. [Jos06]):

- Stochastische, die zufällige Prozesse verwenden. Z.B. zufällige Anfangssituation
- Deterministische, die keine zufälligen Prozesse verwenden.
- Exakte, die jedes Objekt genau einem Cluster zugeordnen.
- Fuzzy Algorithmen, die Zugehörigkeit von Objekten zu Clustern über prozentuale Werte ausdrücken.

Beispiel:

Aufgabe: Bestimme Cluster nach k-means (hier $k = 2$) Verfahren, nutze den euklidschen Abstand und wähle als Zentroiden $(6, 5)$ und $(11, 10)$.

Schritt Eins: $B = \{(2,4),(2,8),(4,9),(7,7),(11,10),(11,7),(6,5),(9,2),(12,5),(14,4)\}$
$C_1 = \{(6,5),(2,4),(2,8),(4,9),(7,7),(9,2)\}$
$C_2 = \{(11,10),(11,7),(14,4),(12,5)\}$

Berechne neue Zenroiden und fahre fort bis sich die Cluster nicht mehr ändern. Wähle neue Zentroiden

$C_1 = \{(6,5),(2,4),(2,8),(4,9),(7,7)\}$
$C_2 = \{(11,10),(11,7),(14,4),(12,5),(9,2)\}$

Supervised Clustering Bei diesem Verfahren wird angenommen, dass die Klassenstruktur schon bekannt ist. Dies wird in einer so genannten Trainingsphase ermittelt. Danach werden einige Instanzen mit Bezeichnungen (Markierungen) genommen und den jeweiligen Klassen zugeordnet. Diese Bezeichnungen werden Labels genannt und ermöglichen eine präzise und gezielte Zuordnung für neue Objekte. Somit kann der Nutzer die Beziehungen zwischen den Objekten erkennen und sieht warum diese zusammen gruppiert werden(vgl. [DC03]).

Unsupervised Clustering In diesem Verfahren kennt der Nutzer die Klassenstruktur nicht und hierbei sind keine Informationen und kein Hintergrundwissen vorhanden. Die Daten sind unbezeichnet und sollen nach einem Ähnlichkeitsgrad gruppiert werden, wobei das Auffinden eines geeigneten Kriteriums den gr"ößten Aufwand darstellt. Eine mögliche Hilfe für das Finden einer guten Gruppierung sind Beobachtungen und Experimente(vgl. [DC03]).

Semi-supervised Clustering Dieses neue Verfahren liegt zwischen den beiden oben genannten Verfahren. In die Clusteranalyse wird Hintergrundwissen integriert, um die resultierenden Cluster zu verbessern und um die Laufzeit für die Berechnung von Clustern zu reduzieren. Das Hintergrundwissen wird in Form von Constraints (Bedingungen) in die Clusteranalyse mit eingefügt. Der Nutzer gibt Constraints in das System ein und kann Angaben machen, ob Objekte in das selbe Cluster gehören oder nicht. Durch diese Einschränkungen wird der Lösungsraum begrenzt und der Suchraum reduziert. Außerdem hat der Nutzer die Möglichkeit die Clusteranalyse zu steuern, um eine möglichst gute Partitionierung der Objekte zu erzielen. Semi-supervised Clustering steht in Beziehung zum aktiven Lernen, aber es unterscheidet sich von Diesem durch die Wahl der Datenpunkte, welche die meisten Informationen liefern und somit eine gute Partitionierung ermöglichen. Der Nutzer wählt die Datenpunkte selbst und bestimmt auf diese Weise eine Anzahl von Bedingungen, wobei das System nun die Menge von Objekten nach diesen Bedingungen clustert(vgl. [DC03]).

Ein Vorteil dieses Verfahrens ist es, dass dem Nutzer die Möglichkeit gegeben wird mit dem System zu interagieren und mit den Daten zu arbeiten, um diese besser verstehen zu können. Dadurch lernt das System Kriterien, um den Nutzer zufriedenzustellen. Ein weiterer Vorteil ist es, dass das System keine Funktionseingaben des Nutzers erwartet, um die Kriterien, die der Nutzer im Kopf hat, zu erfüllen. Der Nachteil dieses Verfahrens ist, dass es viele mögliche Bedingungen gibt, die in das System eingegeben werden können (vgl. [DC03]).

Wann sollte Semi-supervised Clustering bevorzugt werden?
Existieren verschiedene und gleichwertige Clusterings, so würde ein aktiv lernendes System viele unnötige Anfragen stellen und nicht-zufriedenstellende Cluster erzeugen. Ein weiterer nennenswerter Punkt wäre, die Bevorzugung des Verfahrens, bei unbekannter Anordnung der End-Cluster, denn Constraints sind einfacher zu verstehen als Labels. Falls die Labels nicht leicht für die Clusteranalyse benutzbar sind, wäre dies ein weiterer Aspekt für die Wahl des Verfahrens(vgl. [DC03]).

Constraints Sind Bedingungen, die der Nutzer dem System vorgibt, und die auf jeden Fall eingehalten werden müssen. Es sind verschiedene Arten von Constraints vorhanden, als Beispiel sind die Instance-Level Constraints zu nennen. Diese machen Aussagen über die Beziehung der einzelnen Objekte(vgl. [Ebe06]). Noch erwähnenswert wären σ- Constraints (ein beliebiges Paar von Datenpunkten aus zwei verschiedenen Clustern, die eine Mindestdistanz σ haben) und γ- Constraints für hierarchisches Clustering (zwei Cluster, deren Zentroiden eine Distanz größer γ zueinander haben, können nicht fusioniert werden)(vgl. [Zho06]).

Haupttypen In diesem Abschnitt werden zwei Haupttypen von Instance-Level Constraints unterschieden. Zum Einen: Must-Link Constraints die festlegen, dass zwei Objekte in das selbe Cluster gehören und zum Anderen: Cannot-Link Constraints, welche festlegen, dass zwei

Objekte nicht in das selbe Cluster gehören. Aus diesen Aussagen kann man schlussfolgern, dass diese Constraints Aussagen über die paarweise Beziehung zwischen zwei Objekten der selben Menge machen (vgl. [Ebe06]).

Beispiele:

- Ein Beispiel für die Beziehungen der Objekte(vgl. [Ebe06]):
  Falls $ML(a, b)$ und $ML(b, c) => ML(a, c)$ aber auch Aussagen über CL möglich.

- Clusterprozess mit Einbindung von Constraints (vgl. [Ebe06]):
  Dazu nehme ich einen k-means partitionierten Cluster . Die Bedingungen werden in die Clusteranalyse integriert, um Verbesserungen bzgl. der Laufzeit, der Genauigkeit und Leistung zu erreichen.
  Allgemeine Werte und Ablauf der Methode:
  - Datensatz D
  - Satz Must-Link Constraints ($CON_{=}$) und Satz von Cannot-Link Constraints ($CON_{\neq}$)
  - Methode VIOLATE-CON angelegt worden, die einen Wahrheitswert zurück gibt. Wenn die Ausgabe dieser Methode true ist bedeutet das, dass durch die Zuteilung ein Constraint verletzt wurde. Sollte jedoch false zurückgegeben werden, ist die Einordnung von d_i in C_j gültig.

  Methoden Schritte:

  1. ob ein Must-Link Constraint existiert der die zu clusternde Instanz d enthält, dessen Contraint-Partner $d_{=}$ bereits klassifiziert wurde. Diese ist jedoch nicht in dem Cluster C, in das d eigentlich gruppiert werden soll. Eine solche Situation würde zu der Ausgabe true führen.
  2. Sollte jedoch ein Cannot-Link Constraint verletzt sein, dann wurde eine Instanz $d_{\neq}$ gefunden, die bereits im Cluster C platziert wurde. Das macht es unmöglich, dass d ebenfalls nach C geordnet wird.
  3. Nur für den Fall, dass kein spezifizierter Constraint verletzt wird, liefert $VIOLATE-CON$ false und damit kann COP-K-Means den Datenpunkt d_i dem Cluster C_j zuteilen. Offensichtlich kann mit dieser Methode auch der Fall eintreten, dass keines der vorhandenen Cluster so gefunden werden kann, dass kein Instance-Level Constraints aus ($CON_{=}$) oder ($CON_{\neq}$) verletzt wird. Bei solch einem Fall wird der Algorithmus abgebrochen.

```
1 Seien c_1, c_2, ..., c_k die initialen Clusterzentren.
2 repeat (für alle Instanzen d_i ∈ D):
3  ordne d_i dem Cluster C_j zu, dessen Clusterzentrum d_i am
   nächsten ist, so dass
   VIOLATE − CON(d_i, C_j, (Con_=), (CON_≠)) = false ist. WENN kein
   solches Cluster gefunden werden kann, DANN FAIL (RETURN{}).

5 Für alle Cluster C_i: Update Clusterzentrum: c_i (1/|c_i|) Σ_{d_j ∈ C_i} d_j
6 Schritt (2) und (3) solange wiederholen, bis Clustermengen
  sich nicht ändern.
7 Return \{c_1, c_2, ..., c_k\}:
```

Listing 1.2: COP-K-Means

$VIOLATE-CON$: (Datenpunkt d, Cluster C, Must-Link Constraints $(CON_=) \subseteq D \times D$, Cannot-Link Constraints $(CON_{\neq}) \subseteq D \times D$)

1. Für alle $(d, d_=) \in CON_=$: *FALLS* $d_= \notin C$: *RETURN true.*
2. Für alle $(d, d_{\neq}) \in CON_{\neq}$: FALLS $d_{\neq} \in C$: *RETURN true.*
3. Sonst *RETURN false.*

- Das Yahoo-Problem(vgl. [DC03]):
  Bei diesem Problem geht es darum 100.00 Dokumente (Texte, Artikel, usw.) in passende Gruppen zu partitionieren. Es wird auch nicht angegeben welche Klassenbezeichnungen verwendet werden sollen (z.B. Sport, Technik, Politik).
  Der Lösungsansatz lautet wie folgt:
  1. Die Dokumente in unsupervised clustering Algorithmus geben und clustern lassen
  2. User geht Cluster durch und sagt dem System welches Cluster er mag oder nicht mag.
     =¿ Nicht für alle Cluster tun sondern nur einige. Gebe Feedback hinzu:
     - das Dokument gehört nicht hier hin
     - bewege das Dokument zu diesem Cluster
     - Die Dokumente in selbe oder unterschiedliche Cluster zuordnen.

     =¿ nicht für alle sondern nur für diejenigen die am unpassensten sind
  3. Nach der Kritik, neu Clustern lassen mit Feedback.
  4. Wiederholen bis zufrieden!

Feedback Dieser Abschnitt behandelt die Verwendung von Constraints als Feedback. (vgl. [DC03]). Diese Constraints können verschiedene Ausprägungen haben:

- Zwei Dokumente gehören oder gehören nicht ins selbe Cluster.

- Ein Dokument gehört oder gehört nicht in dieses Cluster.
- Bewege das Dokument in dieses Cluster.
- Cluster zu grob oder zu fein.
- Cluster ist gut oder nicht gut.
- Diese Bedingungen werden an individuell gewählten Punkten eingesetzt und es sollte vermieden werden clusterspezifische Feedbacks zu geben.

Vergleich der Performance Es ist grundsätzlich schwierig Supervised und Semi-supervised Clustering zu vergleichen. Denn zum Einen, wird zwischen Constraints und Labels unterschieden und zum Anderen wird bei Supervised Clustering nicht die gesamte Menge der Objekte betrachtet. Die in der Trainingsphase benutzten Objekte sind schon markiert und müssen nicht mehr bearbeitet werden, mit anderen Worten, es existiert eine preprocessing Phase im Supervised Clustering. Diese berechnet die Klassenbezeichnungen, an Hand deren partitioniert wird. Dennoch versuchen wir die Verfahren miteinander zu vergleichen, indem wir die Prozentzahl der korrekt eingeordneten Objekte messen. Wobei hier noch wichtig ist, dass nach und nach Labels oder Constraints mit in die Analyse eingebracht werden. Nachdem 10 Constraints eingefügt wurden, wird die asymptotische Performance erreicht. Es werden 70-80% der Objekte korrekt partitioniert. Aber mit Zunahme von Constraints wird keine Verbesserung mehr erreicht. Um die gleiche Performance zu erreichen, benötigt Supervised Clustering 3 bis 6 fach mehr Labels. Aber desto mehr Labels verwendet werden desto besser wird die Performance dieses Verfahrens. Das Unsupervised Clustering hat eine Korrektheit von 50%, also hat es eine geringere Genauigkeit(vgl. [DC03]).

Zusammenfassung und Fazit In dieser Ausarbeitung wurde ein neues Vefahren dargestellt und die jeweiligen Vorteile, die es bei der Benutzung, mit sich bringt. Es wurden Fälle beschrieben in denen es sinnvoll ist ein bestimmtes Verfahren den jeweils anderen vorzuziehen. Grudsätzlich hat der Nutzer mit Hilfe von Hintergrundwissen in Form von Constraints die Möglichkeit sich in den Clusterprozess miteinzubinden und mit dem System zu interagieren. Dadurch wird eine möglichst gute Partitionierung der Objekte erreicht. Das System hat die Möglichkeit vom Nutzer zu lernen, um diesen zufriedenzustellen. Also kann das menschliche Vorgehen ein wegweisendes Verfahren werden, um Zusammenhänge innerhalb von Gruppen zu entdecken. Das eigentliche Ziel des Semi-Supervised Verfahrens ist es jedoch Wege zu finden, wie Feedback während des Clusteringprozesses in die Verarbeitung eingebunden werden kann.

SVM Clustering

Die Idee Die Idee bei Supervised Clustering ist, dass man fertig geclusterte Itemsets gegeben hat und jetzt die Fähigkeit erlernen möchte, weitere Itemsets richtig zu Clustern.
Der Lösungsansatz ist hierbei, dass man die SVM zum Trainieren des Cluster-Algorithmus benutzt.
Das Problem beim Clustering liegt darin zu erkennen: welche Items sind sich „ähnlich“, was bedeutete „ähnlich“ eigentlich genau, und wie definiert man es. Außerdem muss der Benutzer mit seinen Beispielsets möglichst viele Möglichkeiten abdecken, um gute Lernergebnisse zu

erzielen.
Formal:

gegeben:	n Trainingssets :	$(\mathrm{x}_i, \mathrm{y}_i)...(\mathrm{x}_n, \mathrm{y}_n) \in X \times Y$ X sind alle Items. Y alle möglichen Partitionen von X.
gesucht:	Clusterfunktion:	h:$X \to Y$

Hilfsmittel:

Verlustfunktion:	$\Delta : Y \times Y \to \mathbb{R}$ vergleicht zwei Clusterings
Trainingserror:	$\Delta(\mathrm{h}(x), y)$ für ein Beispiel (x,y) und Clusterfunktion h

Paarweise Klassifikation Alle Items werden paarweise verglichen und jedes Paar wird durch einen Eigenschaftsvektor beschrieben. Der Ausgabewert gibt an, ob die Items im selben Cluster sind oder nicht. Positive Beispiele sind die *in* einem Cluster. Negative die, die nicht in einem Cluster sind.
Wenn nicht eindeutig klar ist, ob ein Paar im gleichen Cluster ist oder nicht, kann der paarweise Klassifizierer nicht nach dem geforderten Leistungsmaß optimieren. Die Anzahl von Paaren in einem Cluster ist oft sehr gering. Deshalb können sie unterbewertet werden. Der Vorteil von Abhängigkeiten der einzelnen Paare kann nicht genutzt werden. Es gibt Methoden um diese Probleme zu beseitigen, allerdings sind bisher alle nur Heuristiken.

Cohen & Cardie:	lernen nur auf Paaren, die sich ähnlich sind.
Ng & Cardie:	basiert auf Zusammenhängen von Ausdrücken: $x_b + x_a$ werden pos. Paar, x_a ist der Ausdruck, der am nächsten vor x_b auftaucht und mit x_b in Verbindung steht. Alle Ausdrücke zwischen x_a und x_b werden mit x_b zu negativen Paaren

Modell Grundsätzlich wird an der Methode für das Supervised Clustering nichts geändert. Es wird nur das Ähnlichkeitsmaß modifiziert, so dass der Algorithmus die erwünschten Ergebnisse liefert.

Das Ähnlichkeitsmaß $Sim_{\mathbf{w}}$ bildet Itempaare auf Werte ab. Positiv = ähnlich; negativ = unähnlich.
$\forall : (x_a, x_b); \quad x_a, x_b \in \mathrm{x}; \quad x_a \neq x_b \quad \exists : \varnothing(x_a, x_b) \equiv \varnothing_{a,b}$
Ähnlichkeitsmaß: $Sim_{\mathbf{w}}(x_a, x_b) = \mathrm{w}^T \varnothing_{a,b}$
$\varnothing$: Eigenschaftsvektor der Paare.
w: Parameter für das Ähnlichkeitsmaß.

```
1  Imput: (x_1, y_1), ..., (x_n, y_n), C, ε
2  S_i ← ∅ für alle i = 1, ..., n
3  repeat
4   for i = 1, ..., n \textbf{do}
5    H(y) ≡ Δ(y_i, y) + w^T Ψ(x_i, y) − w^T Ψ(x_i, y_i)
6    berechne: ŷ = argmax_{y∈Y} H(y)
7    berechne: ξ_i = max{0, max_{y∈S_i} H(y)}
8    if H(ŷ) > ξ_i + ε \textbf{then}
9     S_i ← S_i ∪ {ŷ}
10    w ← Optimiere über S = ⋃_i S_I
11   end if
12  end for
13 until kein S_i hat sich während des Durchlaufs geändert
```

Listing 1.3: SVMStruct Version2

Für die Cluster Methode benutzt man: Corellation Clustering(Bansal 2002) Für ein Itemset $\mathbf{x}$ wird das Clustering $\mathbf{y}$ gesucht, so dass die Summe der Ähnlichkeitswerte maximal ist.

$$\text{argmax}_{\mathbf{y}} \sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} Sim_{\mathbf{w}}(x_a, x_b)$$

Durch dieses Vorgehen kann es allerdings vorkommen, dass Paare, die eigentlich nicht in ein Cluster gehören, zusammengefasst werden, da der Netzeffekt einen Gesamtvorteil liefert.

$$\sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} Sim_{\mathbf{w}}(x_a, x_b) = \sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} \mathbf{w}^T \varnothing_{a,b} = \mathbf{w}^T (\sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} \varnothing_{a,b})$$

Die Zielfunktion ist also ein Linearprodukt von $\mathbf{w}$ und einer Summe von $\varnothing$ Vektroren. Auch wenn man Corellation Clustering benutzt, ist jede Zielfunktion, die ein Linearprodukt von $\mathbf{w}$ ist, erlaubt.

$\text{argmax}_{y \in Y} H(y)$ liefert die am meisten verletze Nebenbedingung, wobei $\Delta(y, \hat{y})$ der Wertverlust zwischen dem wirklichen Cluster y und dem vorhergesagten $\hat{y}$ ist, und $\Psi(x_i, y) = \frac{1}{|x_i|^2} \sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} \varnothing(x_a, x_b)$.
Wenn die Nebenbedingung mehr als ϵ verletzt wird, wird sie übernommen und es wird neu optimiert.
Dies wird solange wiederholt, bis keine Nebenbedingung mehr hinzukommt.
Theorem 1:
Wenn $\overline{\Delta} = max_{(x_i, y_i) \in S}(max_y \Delta(y_i, y))$ und $\overline{R} = max_{(x_i, y_i) \in S}(max_y ||\Psi(x_i, y_i) - \Psi(x_i, y)||)$ dann konvergiert die SVM^{struct} nach $max\{\frac{2n\overline{\Delta}}{\epsilon}, \frac{8Cn\overline{\Delta}\overline{R}^2}{\epsilon^2}\}$
Theorem 2:
Der Algorithmus liefert ein approximiertes Ergebnis, das kleiner oder gleich dem quadratischen Programm der SVM^{struct} ist. Alle Nebenbedingungen sind im Rahmen von ϵ erfüllt.

Approximation Es stellt sich jetzt die Frage, wie schwer es ist, die meist verletzte Nebenbedingung zu finden?
$H(y) \equiv \Delta(y_i, y) + \mathbf{w}^T \Psi(x_i, y) - \mathbf{w}^T \Psi(x_i, y_i)$
Der letzte Teil bleibt immer gleich, und kann vernachlässigt werden, so dass sich das Maximum nicht ändert. Es bleibt also die Kostenfunktion, die sich aus dem Verlust zwischen korrekten Clustering y_i und dem vorhergesagten y sowie der Zielfunktion des Correlation Clusterings zusammensetzt. Das Finden des y für die Zielfunktion ist NP-vollständig (Bansal,2002) **u**nd den Verlust zu Addieren bringt keinen Vorteil. Also ist das Problem die meist verletzte Nebenbedingung zu finden auch NP-vollständig. Wir betrachten deswegen zwei Approximationsmöglichkeiten für die Clusterfunktion und damit für $\text{argmax}_y H(y)$.

Greedy Approximation C_G Zu Beginn ist jedes Item in einem eigenen Cluster, jetzt werden zwei Cluster vereinigt, so dass $H(y)$ maximal erhöht wird. Der Algorithmus terminiert, wenn keine Vereinigung eine Erhöhung von $H(y)$ bewirkt.
Korollar: Das Ergbenis der Greedy-Approximation C_G ist nicht größer als das der SVM^{struct}.

Relaxation Approximation, C_R Jedes Paar x_a, x_b hat eine Variable $e_{a,b}$ die anzeigt, zu welchem Maß die Items in das gleiche Cluster gehören.$e_{a,b} \in [0, 1]$ $e_{a,b} = e_{b,a}$ $e_{a,b} + e_{b,c} \geq e_{a,c}$.
Man erhält also ein lineares Programm: $\max_e \sum_{e_{a,b} \in \mathbf{e}} (e_{a,b}) \cdot (\mathbf{w}^T \varnothing_{a,b})$ und damit ein $\Psi(x, e) = \frac{1}{x} \sum_{e_{a,b} \in e} (e_{a,b}) \cdot \varnothing_{a,b}$. Dies ist die gleiche Funktion wie $\Psi(x_i, y) = \frac{1}{|x_i|^2} \sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} \varnothing(x_a, x_b)$, wenn $e_{a,b}$ ganzzahlig ist.
Korollar: Das Ergebnis der Relaxation Approximation C_R ist nicht kleiner als das der SVM^{struct}.

Für die Experimente wurde C_R^* benutzt. Hierbei hat zu Beginn jedes Item ein eigenes Cluster, dann wird über alle $x_a \in x$ iteriert, wenn x_a alleine im Cluster ist wähle es aus. Als nächstes sucht man alle $x_b \in x$ die auch alleine in einem Cluster sind und Vereinigt sie mit x_a, wenn $e_{a,b} > 0,7$

Verlustfunktionen Für die Experiemente stehen zwei Verlustfunktionen zur Verfügung zum Einen paarweise Δ_P und zum Anderen MITRE Verlust Δ_M.
$\Delta_P(y, \overline{y}) = 100\frac{W}{T}$ wobei, T die Anzahl von Paaren im Itemset ist und W die Anzahl von Paaren ist, bei denen y und $\overline{y}$ nicht übereinstimmen. Bei Relaxation Approximation wird $e_{a,b} \cdot (\mathbf{w}^T \varnothing_{a,b} + \frac{100}{T})$ verwendet.
$\Delta_M(y, \overline{y}) = 100\frac{2(1-R)(1-P)}{(1-R)+(1-P)}$ wobei R der MITRE recall ist und P die MITRE precision. Δ_M kann als Anzahl von nötigen Transformationsschritten angesehen werden, um von y nach $\overline{y}$ zu gelangen. Eine Schritt ist entweder das Teilen oder das Vereinigen von Clustern.

Experimente Als Testmaterial wurden zum Einen die MUC-6 noun-phrase Daten benutzt zum Anderen eine Auswahl aus den Google-News. Die MUC-6 Daten bestehen aus 60 Dokumente, wovon 30 als Traingsdaten und 30 als Testdaten benutzt wurden. Jedes Dokument

hat im Schnitt 101 Cluster mit durchschnittlich je 1,48 Wörtern. Es gibt aber auch viele Einzelwort-Cluster. Der Eigenschaftsvektor der Paare hat 53 Dimensionen, wie zum Beispiel: Ob die Worte mit dem gleichen Artikel auftauchen, wieviele Sätze sie voneinander entfert sind, ob eins der Worte ein Subjekt im Satz ist, etc.
Für die Google-News Daten wurden 30 Tage lang die Top 10 Themen der „World“ Kategorie ausgewählt und von jeder Kategorie soweit möglich 15 Artikel ausgewählt. Die ersten 15 Tage waren Trainingsdaten und die lezten die Testbeispiele. Jeder Artikel hat 30 TFIDF gewichtete Vektoren für Unigramme, Bigramme, Trigramme, etc. Der Eigenschaftsvektor ist der Cosinus zwischen den Vektoren der Artikel plus einer Eigenschaft, die immer 1 ist.

Testergebnisse bei MUC-6

	C_G	PCC	Default
Test mit C_G, Δ_M	41,3	51,6	51,0
Test mit C_G, Δ_P	2,89	3,15	3,59

Spalte C_G : - mit SVM-Cluster und Greedy Algorithmus gelernt
Spalte PCC: - mit PCC gelernt
Spalte Default: - in der ersten Zeile alle Items in einem Cluster
- in der zweiten jedes Item in eigenem Cluster

Zeile 1: getestet und optimiert mit MITRE Verlustfunktion
Zeile 2: getestet und optimiert mit paarweiser Verlustfunktion
Beide Tests benutzen den Greedy Algorithmus auf der Testmenge.

Es ist ganz klar zu erkennen, dass die SVM deutlich besser ist, als der Rest. Bei Δ_M kann man das dadurch erklären, dass die SVM für eine Verlustfunktion optimieren kann, PCC nicht. Bei Δ_P lässt sich allerdings so nicht argumentieren, da PCC quasi gleich optimiert, trotzdem ist die SVM besser.

Testergebnisse bei Google News

	C_G	C_R	PCC	Default
Test mit C_G, Δ_P	2,36	2,43	2,45	9,45
Test mit C_R, Δ_P	2,04	2,08	1,96	9,45

Bei diesen Daten sieht man, dass keine Mehtode deutlich besser ist als die andere. Man kann also schlussfolgern, dass die SVM besser zu sein scheint, wenn die Daten transitive Abhängigkeiten haben. Wenn nicht, sind beide Methoden annähernd gleich gut.

Optimierung für Verlustfunktionen

	Opt. to Δ_M	Opt. to Δ_P
Performance bei Δ_M	41,3	42,8
Performance bei Δ_P	4,06	2,89

Es gibt kaum Unterschiede bei der Δ_M Performance, aber bei Δ_P sind die Performanceunterschiede recht deutlich. Außerdem sind die Werte des für Δ_M optimierten Modells nicht besser, als die der Defaultwerte. Als Schlussfolgerung kann man also festhalten, dass die Wahl der Verlustfunktion nachhaltig die Accurracy des Modells beeinflussen kann.

Verlust bei argmax$_y H(y)$ Muss die Verlustfunktion in argmax$_y H(y)$ integriert sein oder darf man sie hier vernachlässigen?

	mit Verlusfunkt.	ohne Verlustfunkt.
MUC-6, Δ_M	41,3	41,1
MUC-6, Δ_P	2,89	2,81
Google, train C_G, test C_G	2,36	2,42
Google, train C_R, test C_R^*	2,08	2,16

Es gibt keine signifikante Änderung in den Zeilen, bei keinem Modell. Dies lässt den Schluss zu, dass die Verlustfunktion in argmax$_y H(y)$ nicht zwingend erforderlich ist.

Greedy vs. Relaxation Als letztest stellt sich die Frage, welche Approximation besser ist.

	Train C_G	Train C_R
Test C_G	2,36	2,43
Test C_R	2,04	2,08

Auch hier sind kaum Unterschiede zu erkennen. Es ist keine der Approximationen ist zu bevorzugen.

Effizienz der SVM^{cluster} Auf dem MUC-6 Material werden über 1000 Nebenbedingungen integriert und es wird 50 mal neu optimiert. Mit C_G wurde nur 1% der Zeit benötigt.Bei allen Experiementen hat keine SVM^{cluster} länger als 3-4 Stunden gebaucht, wobei der Durchschnitt eher unter einer Stunde lag. Als Vergleich dient der paarweise Klassifizierer. Dieser bearbeitete eine knappe halbe Million Paare im Trainingsset. Hierfür benötigte er eine halbe Woche und es wurden eine halbe Million Nebenbedingungen eingefügt.

Zusammenfassung Die SVM ist je nach Material gleich oder besser als PCC. Die Wahl der Verlustfunktion ist entscheidend für die Accurracy, in Abhängigkeit vom Modell. Der Verlust in argmax$_y H(y)$ kann vernachläsigt werden. Keine der Approximationen ist eindeutig besser. Fasst man alle Erkenntnisse zusammen ist die SVM^{cluster} mit einem Approximationsalgorithmus für die Geschwindigkeit die beste Wahl.

1.5 Relation Extraction auf unstrukturierten Texten

1.5.1 Einführung

Relationen sind Beziehungen, also Verknüpfungen zwischen Entitäten oder Daten. Sie sind uns aus Graphen, Datenbanken und Diagrammen bestens bekannt und vertraut. Das „in Beziehung“ setzen von Entitäten wird üblicherweise entweder durch ein Modell induziert, siehe Datenbanken, oder geschieht manuell wie im Fall von Diagrammen. Die automatische Erstellung von Relationen aus unstrukturierten Daten nennt man „Relation Extraction“.

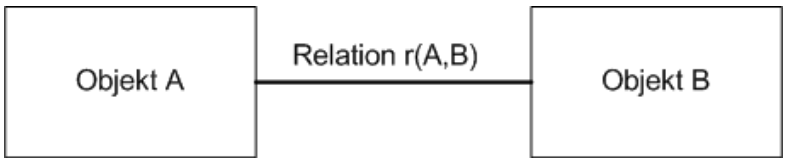


Abbildung 1.28: Relation zwischen zwei Objekten.

In der Literatur ordnet man Relation Extraction meistens als Teilgebiet der folgenden Forschungsbereiche ein:

- SRL „Statistical Relational Learning“ einer Disziplin des maschinellen Lernens
- ILP „Inductive Logic Programming“ also der konventionellen Logik Programmierung.

Wir wollen uns in diesem Teil der Ausarbeitung mit den graphischen Modellen und statistischen Kernel Methoden beschäftigen. Doch zunächst einmal eine kurze Einführung in die graphischen Darstellungen von Relationen.

Relationen sind bekanntermaßen Beziehungen zwischen Objekten und können als Graph dargestellt werden. In unserem Anwendungsfall sind die Wörter eines Satzes Objekte und die semantischen Beziehungen zwischen den Wörtern sind die zu extrahierenden Relationen, in diesem Sinne die Kanten des Graphen.
Natürlich sind nicht alle Wörter eines Satzes relevante Objekte. Im Normalfall werden nur die Nomen, in unserem Fall, die zuvor extrahierten “Named Entities„ als Objekte betrachtet. Es liegt Nahe die Relation aus den Wörtern zwischen den Objekten zu extrahieren. Die Aufgabe der Extraktion kann somit genauer als Analyse der Wörter und die gleichzeitige Bildung einer Beziehung zwischen zwei benachbarten Entitäten beschrieben werden.

Die extrahierten Relationen, besonders auf eindeutigen Entitäten, bilden einen Extraktionsgraphen der die Beziehungen zwischen den Entitäten auf dem gesamten Text aufspannt. Es existieren verschiedene Darstellungen des Graphen, Cafarella zählt eine Hierarchie der Komplexität und Mächtigkeit dieser Graphen auf:

- Die komplexeste und erschöpfende Darstellung des Grafen bilden Semantische Netze, wie sie auf den Webseiten von John F. Sowa beschrieben sind. Semantische Netze sind die klassische, aus der Psychologie stammende Abbildung kategorisierter semantischer Relationen. Die Betonung liegt hier auf den semantischen Beziehungen zwischen den Entitäten.
- Entity Relationship Diagramme, kurz ER-Diagramme sind die relaxierte Betrachtung der semantischen Netze. Die von Chen und Bachmann eingeführten Modelle sind die wohl bekannteste Darstellung von relationalen Datenbanken. Die Darstellung kennt nur wenige Kategorien der Beziehungen und konzentriert sich auf den Quantitäten der Verbindungen und deren Hierarchie.
- Die schwächste Form der Darstellung ist aus dem Internet bekannte Link Graph, auch Citation Graph genannt. Dieser Graph besteht aus Dokumenten kennt nur eine Form der Relationen, nämlich den weiterführenden Link auf ein anderes Dokument und wurde Ausführlich in den Arbeiten von Page, Brin und Kleinberg beschrieben.

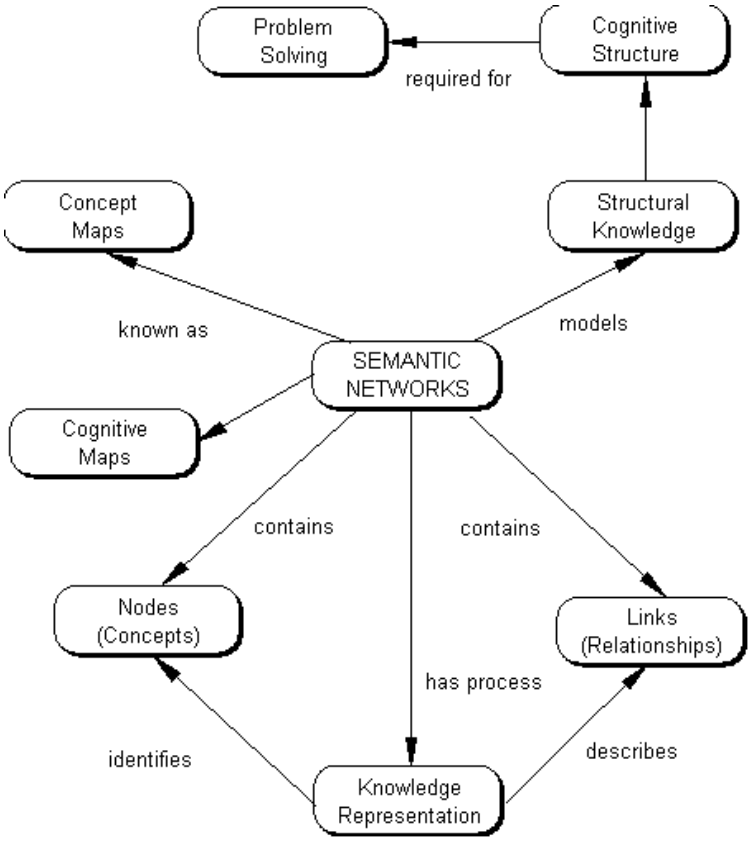


Abbildung 1.29: Semantisches Netz (www.conroeisd.net)

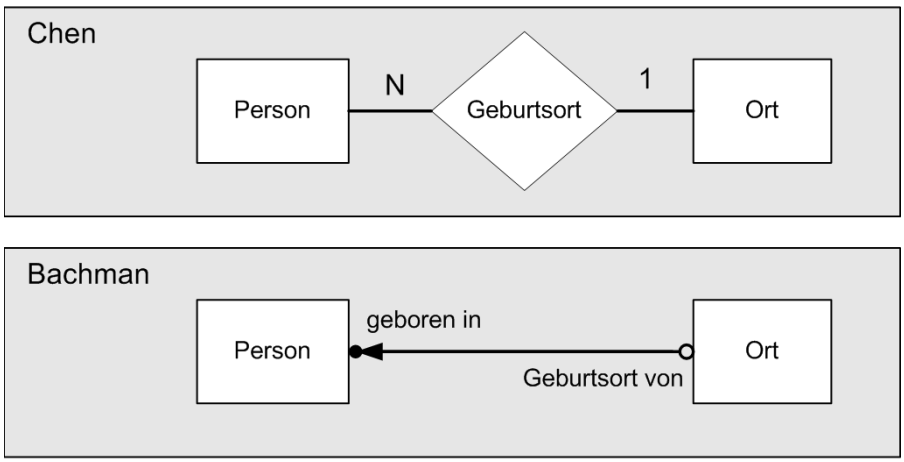


Abbildung 1.30: Chen und Bachmann Diagramme (de.wikipedia.org)

Die Defintion eines Graphen der aus Knoten, in unserem Fall NE's, besteht, sowie die Definition der Kanten als Relationen wird hier nicht aufgeführt, da sie bereits aus den vorangegangenen Ausarbeitungen bekannt sein sollte.

1.5.2 Anwendungsbeispiel: TEXTRUNNER

Ein praktisches Beispiel der Anwendung von 'Relation Extraction' Techniken ist TEXTRUNNER. Dieses Server basierte System von Cafarella, Banko und Etzioni zur relationalen Websuche, nutzt das domainunabhängige Internet als Corpus und extrahiert ohne menschliches Eingreifen Relationen aus Texten. Im Vergleich zu anderen Systemen, wie KNOWITALL [DA04] oder AskMSR [JA02] ist TEXTRUNNER auf die schnelle Verarbeitung von großen, domainunabhängigen Corpora ausgelegt. Skallierbarkeit und vergleichbar schnelle „single-pass" Verarbeitung sind zwei wichtige Argumente die dessen breite Nutzung als Suchmaschine in Aussicht stellen.

Wie bereits erwähnt extrahiert TEXTRUNNER mit einfachen, recht wenigen Methoden, Relationen aus Texten und baut aus ihnen einen „Extraction Graphen" auf. Dieser Graph ist eine textuelle Approximation des Entity Relationship Diagramms, kann also in der Hierarchie der semantischen Netze zwischen den ER-Diagrammen und Link Graphen eingeordnet werden. Der Extraction Graph bildet eine Zwischenstufe in der Repräsentation von Informationen und zeigt gerade die Informationen, die explizit nicht im Inverted Index stehen. Diese neue Suchmethode geht weit über die Stichwortsuche hinaus, was mit dieser erweiterten Datenstruktur auch nicht weiter verwundert. Beim Aufbau des Extraction Graphen wird in der nächsten Nähe der Entitäten nach den Verbindungen, ergo Relationen gesucht. Hierbei wird das Theorem aufgestellt das miteinander verbundene Entitäten auch lexikalisch nahe bei einander liegen werden. Es muss also nur der Text zwischen den Entitäten untersucht werden. Das Auffinden und Markieren von Entitäten in einem Text ein hierbei kein separater Schritt vor der eigentlichen Relation Extraction, sondern geschieht simultan.

Datenstruktur: Extraction Graph

Der Extraction Graph wird exklusiv aus dem vorliegendem Text generiert. Seine Knoten und Kanten bestehen aus Wörtern (Strings). Im Gegensatz zu einem semantischen Netz sind die vorhandenen Relationen oder Entitäten nicht Eindeutig, das bedeutet, dass die selben realen Objekte verschieden repräsentiert werden können. Es existieren keine Mechanismen um logische Inkonsistenzen zwischen Begriffen zu beheben.

Es sei ein gerichteter Graph, der, wie bereits bekannt, aus Knoten und Kanten besteht. Die Knoten bilden die Entitäten ab und die Kanten stellen logischerweise die Relationen zwischen den Entitäten dar. Die Relationen sollen als ein Tripel aus zwei Knoten und einer Kante definiert werden:

- Eine Relation im Extraction Graph ist ein Tripel $t_x = (n_i, e_{i,j}, n_j)$.
- mit $n \in V$, $V \neq \emptyset$ und Knotenmenge.
- mit $e \in E$, E als Menge der Kanten.
- Der Graph kennt nur zwei Kategorien / Typen von Kanten:

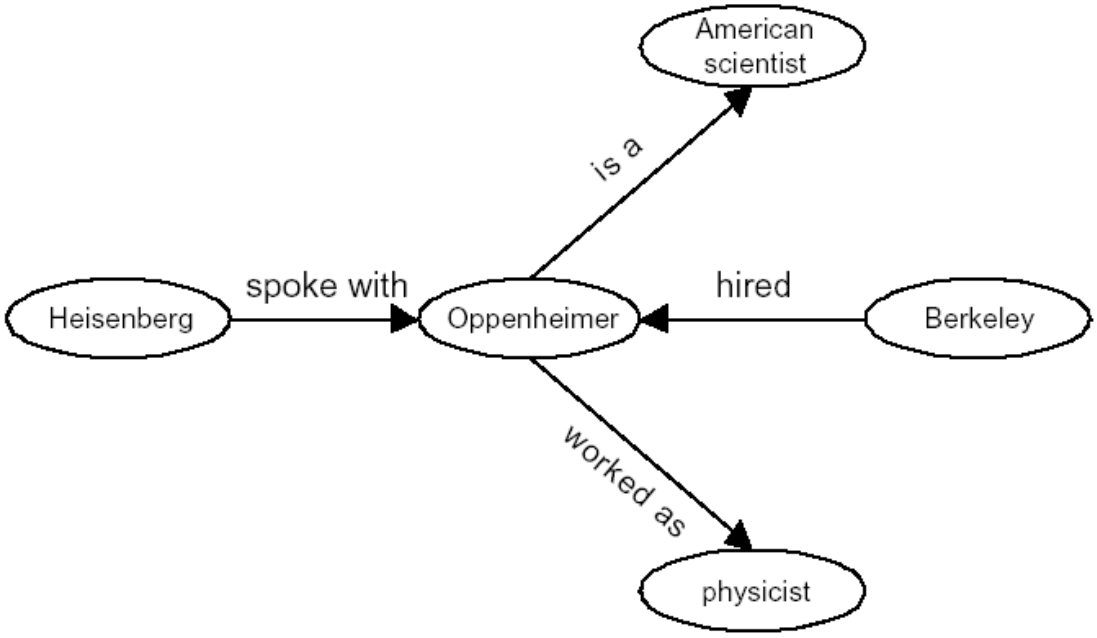


Abbildung 1.31: Darstellung des Extraction Graphen

  - Eine *isA* Relation zwischen zwei Knoten, die üblicherweise als ist-Bezeichner eines Knoten dient. „Oppenheimer - *isA* - American Sciencist“ sei ein Beispiel für eine ist Relation dieser Form.
  - Alle anderen denkbaren Relationen werden als *predicate* Relation abgebildet. Da es sich bei diesen Relationen um Text Labels handelt sind sie flexibel genug um viele Beziehungen darzustellen, allerdings kann man in diesem Zusammenhang nicht wirklich von einer semantischen Relation sprechen. „Berkeley - *hired* - Oppenheimer“ ist ein typisches Beispiel für eine derartige Kategorie.

- Zusätzlich wird zu dem Tripel die Anzahl seiner Vorkommen im Text gespeichert.
- Jedem Tripel wird sein Inhalt als String zugeordnet.

Wie die formelle Definition des Graphen zeigt, beschränkt sich die Semantik nur auf die *isA* Beziehung. Alle anderen Beziehungen werden nur sehr abgeschwächt dargestellt.

Extraktion der Kanten aus dem Text

Optimal wäre es die Extraktion in einem einzigen Lauf über das Dokument zu realisieren. Diese Vorgabe ist vermutlich der Grund für diese einfache und relaxierte Betrachtung der Kanten. Eine Kante kann trivialer-weise aus allen Wörtern zwischen zwei Entitäten bestehen. Dies würde jedoch die Suche einer Relation auf die einfache Stichwortsuche reduzieren. Die Genauigkeit der Antworten auf gestellte Queries würde erheblich leiden. Alternativ könnte man mit morphologischen Mitteln nur Verben extrahieren, was wiederum zu einschränkend

wäre. Die TEXTRUNNER Vorgehenswiese ist eine Lösung die zwischen beiden Ansätzen liegt, hierbei werden überschaubar wenige linguistische Mittel genutzt um die Kanten zu extrahieren.

- Um eine „*isA* “ Relation zu identifizieren werden die Wörter zwischen zwei Entitäten mit einer adaptierten Version der Hearst Methode [Hea99] zur Entdeckung von Hypernymen mit Hilfe von lexikalischen Mustern. Diese Art der Kante impliziert die Exsistenz einer „ist Oberbegriff von“ impliziert. Dabei werden diese Kanten automatisch kunstruiert.
- Der Algorithmus der eine „*predicate*“ Relation aus den Inhalten extrahiert findet im ersten schritt den wahrscheinlichsten Part-of-Speech Tag für die Wörter. Danach werden mit einem Ausdrucksegmentierer die Nomen in diesem Satz gesucht, die Nomen werden direkt zu Knoten umgewandelt. Die lexikalische POS Distanz zwischen den gefundenen Nomen gibt Aufschluss darüber, wie nahe die Entitäten nebeneinander liegen. Jeder betrachtete Satz wird schließlich kategorisiert, hierbei wird in zwei Kategorien eingeordnet:
  - Ein Satz besitzt das so genannte „low-content-load“ wenn er viele Satzzeichen und Stoppwörter enthält. Diese Sätze sind für die Extraktion uninteressant, sie werden als sprachlicher Ballast behandelt.
  - Sätze mit vielen Verben und Präpositionen, die aus wenigen Nebensätzen bestehen werden mit dem Label„high-content-load“ versehen. Sie sind die eigentlich interessanten Informationsträger des Textes.

  Diese Einordnung der Sätze ist eine simple Maßnahme, welche die zu betrachtende Satzmenge reduziert. Die Verbindung zwischen zwei Nomen bildet ein Muster, das auf die Form Nomen - Prädikat - Nomen reduziert wird und möglicherweise als Relation in den Graphen aufgenommen wird.
  Beispielsweise ist das Tripel „*Oppenheimer, professor of, theoretical physics*“ eines dieser gewollten Muster.

Um einen Gesamtüberblick über den Extraktionsprozess zu erhalten muss man sich folgende Reihenfolge im Ablauf der Applikation vorstellen:

- Der Self-Supervised Learner markiert a priori potentielle Sätze entsprechend ihrem content-load als vertrauenswürdige oder nicht-vertrauenswürdige Quellen für die Relationsextraktion. Vertrauenswürdig sind alle Sätze die entsprechend als „high-content-load“ klassifiziert wurden und mit wenigen eingebetteten Nebensätzen auskommen.
- Darauf folgt die gerade beschriebene „Single Pass Extraction“.
- Schließlich vereint ein Redundancy-based Assessor identische Entitäten im Graphen.

Beantwortung von Fragen

TEXTRUNNER akzeptiert Anfragen die aus Stichwörtern bestehen. Das System beantwortet die Fragen mit einer Liste aus Objekten oder mit Prädikaten die auf ein Objekt verweisen. Es können folgende Fragen beantwortet werden:

- Die Frage nach einer Liste von Objekten die ein gemeinsames Attribut besitzen.

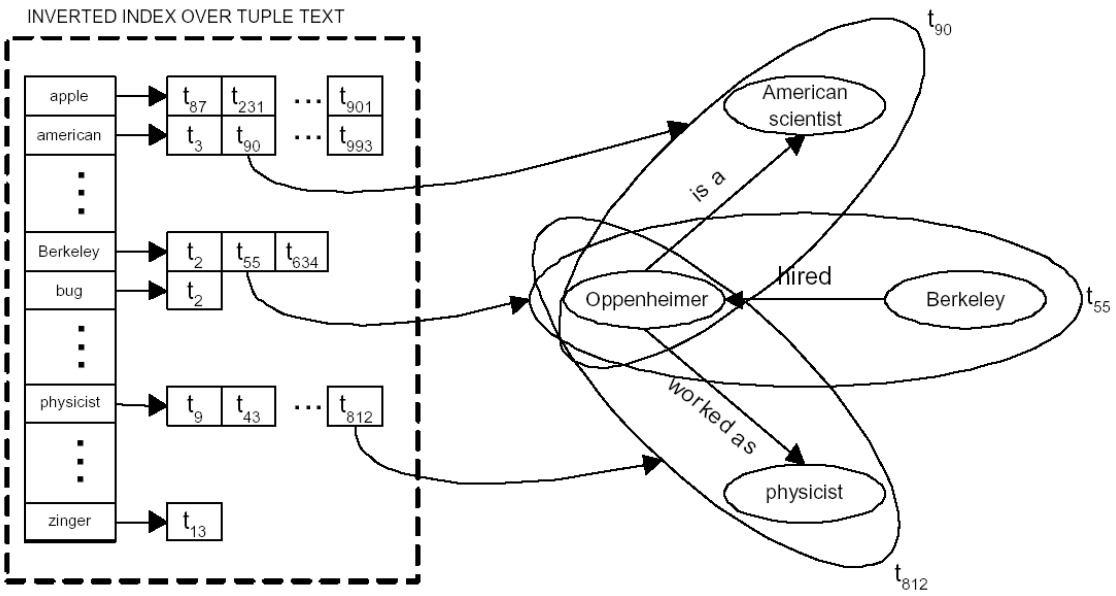


Abbildung 1.32: Inverted Index über dem Graphen

- Die Frage nach einem unbekannten Objekt. Also einem Wort welches der Benutzer finden möchte aber nicht kennt. Eine Art Kontext Suche.
- Schließlich die Frage nach einer Relation, also der Beziehung zwischen zwei Objekten.
- (Theoretisch können auch statistische Fragen mit einer Tabelle beantwortet werden, diese waren jedoch im Paper nicht zu finden.)

Eine vom Benutzer eingegebene Query wird in einzelne Wörter (Tokens) zerlegt und an das Response System übergeben. Die Frage wird also trotz dem Anspruch eine Semantische Suchmaschine zu sein, an Hand von vorhandenen Stichwörtern beantwortet. Im Gegensatz zu einer Stichwortsuche, die zu den Query Tokens die zugehörigen Dokumente ausgeben würde, liefert TEXTRUNNER Objekte zurück - also Tripel des Graphen. Um diese Tripel ausgeben zu können muss eine Zuordnung zwischen den Wörtern eines Dokuments und den Tripeln existieren. Das System baut hierzu einen Inverted Index der jedes Stichwort auf eine Menge von Tripeln abbildet. Nachdem die Eingabe in Tokens zerlegt wurde, wird eine Liste der

im Inverted Index referenzierten Tripel aufgebaut. Mit dem „spread activation“ Algorithmus werden die Knoten, also Entitäten im Graph markiert und ihre Kanten mit einem Wert„aktiviert“. Dieser Wert würde, ungebremst, die benachbarten Knoten markieren und ihre Kanten aktivieren. Die Aktivierung würde sich also wie Welle über dem Graphen ausbreiten. Diese Ausbreitung könnte man auf Kosten der Rechenzeit relativ lange gewähren lassen, TEXTRUNNER jedoch erlaubt nur eine Pfadlänge von 1. Der Wert der Aktivierung ist additiv, so dass zwei benachbarte, markierte Knoten eine gemeinsame Kante mit einem hohen Wert besitzen.

Die Tripel mit dem höchsten Wert bilden das Set aus dem die Antwort berechnet wird. Natürlich wirkt sich die Art der Frage auf die endgültige Form der Antwort aus. Dieses

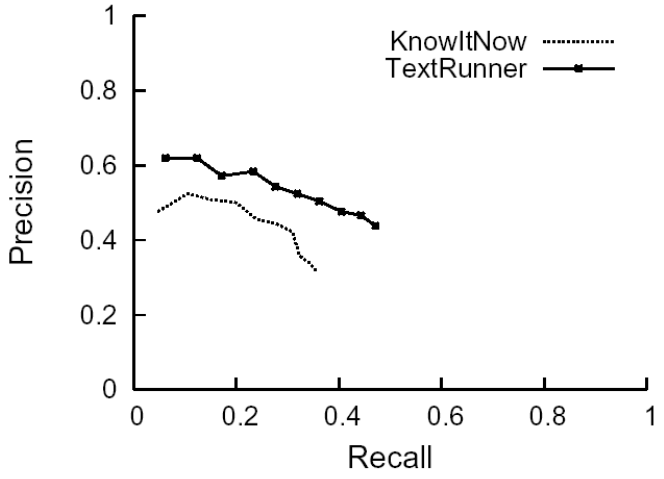


Abbildung 1.33: KNOWITNOW vs. TEXTRUNNER

Verfahren kann man gleichzeitig als eine Art von Rankingverfahren ansehen bei dem Tripel bezüglich ihrer Relevanz zur Query geordnet werden. Gleichzeitig wird ein eigenes Rankingverfahren angewendet welches die Objekte in Clustern entlang ähnlicher Prädikate anordnet.

Performanz und Ergebnisse

TEXTRUNNER nutzt acht einfache Extraktionsmuster um innerhalb 35 Stunden beinahe 1 Million Dokumente zu durchlaufen. Nichts desto trotz sind die Hardware Anforderungen relativ umfangreich. Das vorgestellte und getestete System läuft auf einem Cluster bestehend aus zwanzig Dual-Xeon Intel Servern mit einer Gesamtkapazität an Festplattenspeicher von 5 TerraByte.
Im Durchschnitt wird

- eine *isA* Relation pro 40 Sätze extrahiert,
- zusätzlich werden in 10 Sätzen 7,7 *predicate* Relationen erkannt.

Die Qualität der extrahierten Daten ist erstaunlicherweise besser als bei dem Vorgänger System KnowITNow. Offensichtlich ist ein gewisser Grad an Ungenauigkeit bei der Verarbeitung gerade gut genug.

Relevanz für die Projekt Gruppe

TEXTRUNNER hat keine wirkliche Relevanz bezüglich der Extraktion von Named Entities. Der Ansatz möglichst schnell viele Dokumente zu erfassen ist für die PG uninteressant, die Geschwindigkeit mit der im Bundestag neue Dokumente veröffentlicht werden, lässt ein großes Zeitfenster für die Verarbeitung. Die relaxierte Betrachtung des Problems mit einem weniger komplexen „Extraction Graph“ ist wesentlich interessanter und würde ganz gut zu den von uns aufgestellten Event Schemas passen. Der Ablauf der Extraktion kann auch mit wenig Aufwand

an unser Vorgehen angepasst werden. Das POS Tagging von Wörtern zwischen den Named Entities und deren Mapping auf die Trigger aus unseren Event Schemas würden zwar einen Graphen mit mehreren Typen von Relationen erstellen, allerdings bliebe die Funktionalität erhalten: Gesucht seien Kanten die an gegebene Query Tokens angrenzen. Die durch die Query implizierte Typ der Relation wäre damit nur ein Kriterium des Rankings und nicht der Suche selbst. Die Schwierigkeit liegt hier allerdings wieder beim Erkennen des vom Benutzer gemeinten Typs.

Ein weiterer Ansatz der Interessant erscheint ist der „spread activation“ Algorithmus. Bei Festgelegten „decay factor“ und einer > 1 begrenzten Pfadlänge der Ausbreitung müsste auch der Kontext einer Anfrage sichtbar werden, was zu interessanten Schlussfolgerungen führen könnte.

1.5.3 Dependency Tree

Aufbau

Ein dependency tree repräsentiert die grammatikalische Beziehung der Entitäten eines Satzes. Er wird durch ein Set von Regeln aus dem Syntaxbaum des Satzes erstellt. Solche Regeln können zum Beispiel sein:

- Subjekte zeigen auf ihre Verben
- Adjektive zeigen auf die Nomen die sie modifizieren

Anschließend wird jeder Knoten des dependenca tree mit einem Eigenschaftsvektor erweitert, dieser Vektor hat unter Anderen folgende Dimensionen:

Feature	Example
word	troops, Tikrit
part-of-speech(24 values)	NN,NNP
general-pos(5 values)	noun, verb, adj
chunk-tag	NP,VP,ADJP
entity-type	person, geo-political-entity
entity-level	name, nominal, pronoun
Wordnet hypernyms	social group, city
relation-argument	ARG_A, ARG_B

Vorverarbeitung

Zuerst müssen die Entitäten erkannt werden, hierfür gibt es eine Fülle an Verfahren die auch sehr gute Ergebnisse erzielen. Der nächste Schritt ist die Coreference resolution, hierbei werden gleiche Entitäten erkannt, also wenn eine Entität in verschiedenen Formen im Text auftaucht, muss erkannt werden, das es sich um die gleiche Entität handelt. Dies ist Thema der aktuellen Forschung, genaueres hierzu kann man bei Pasula et al.,2002; McCallum and Wellner, 2003 nachlesen.

Relation Extraction

Für die Relation Extraction wird über alle Paare von Entitäten in einem Satz iteriert. Für jedes Paar einen Erweiterten dependency tree erstellen, der die beiden Entitäten enthält und

alle Knoten, die zwischen den Entitäten liegen. Anschließend wir mit der Kernelfunktion die Ähnlichkeit der Bäumen berechnen. Je Ähnlicher ein Baum einem Aus dem Trainingsdatensatz ist, desto wahrscheinlicher ist es, dass die Entitäten in der gleichen Beziehung zueinander stehen.

Kernelfunktion

Die Kernfunktion besteht aus meheren Teilfunktionen:
$m(t_i, t_j) = \begin{cases} 1 \text{ if } \varnothing_m(t_i) = \varnothing_m(t_j) \\ 0 \text{ sonst} \end{cases}$ ist 1 wenn die Eigenschaftsvektoren der Knoten t_i und t_j mindestens in einer Eigenschaft den gleichen Wert haben
$\mathrm{s}(t_i, t_j) = \sum_{v_q \in \varnothing_s(t_i)} \sum_{v_r \in \varnothing_s(t_j)} C(v_q, v_r)$ Ähnlichkeitsfunktion zwischen zwei Knoten.
$C(v_q, v_r) = \begin{cases} 1 \text{ if } v_q = v_r \\ 0 \text{ sonst} \end{cases}$ Vergleichsoperationen der Eigenschaften.
Zusammen ergibt sich dann folgende Kernfunktion:
$\mathrm{K}(\mathrm{T}_1,\mathrm{T}_2) = \begin{cases} 0 \text{ if } m(r_1, r_2) = 0 \\ s(r1, r2) + K_c(r_1[c], r_2[c]) \text{ sonst} \end{cases}$ mit T_1 und T_2 erweiterte dependency trees.
Diese Kernfunktion gibt ein normalisiertes, symetrisches Ähnlichkeitsmaß zurück.

Experimente

Für die Experimente wurden verschiedene Kernel Varianten eingesetzt. Als Datensatz wurde Automatic Content Extraction(ACE) vom Intitut for Standards an Technology(NIST) benutzt. die ist ein Datensatz mit über 800 annotierte Texte.

	normal	Kernel Option. 1	Kernel Option. 2
Recall	26,3	51,8	35,0
Recision	70,3	81,2	67,5
F1	38,0	63,2	45,8

Beispiel

Für den Satz: "Troops advanced near Tikrit."

Optimierungsvarianten

Um die Performance zu verbessern berechnet man nicht die vollständigen dependeny trees, sondern nur die kürzesten Pfade zwischen den beiden Entitäten, die betrachtet werden → bessere Laufzeit, da deutlich weniger Knoten betrachtet werden.Nach Razan C. Buescu and Raymond J. Mooney

1.5.4 SVM Methoden

Eine Support Vector Machine (SVM) ist ein Klassifikator. Eine Support Vector Machine unterteilt eine Menge von Objekten so in Klassen, dass um die Klassengrenzen herum ein möglichst breiter Bereich frei von Objekten bleibt; sie ist ein sogenannter Large Margin Classifier.

Trainingsdaten

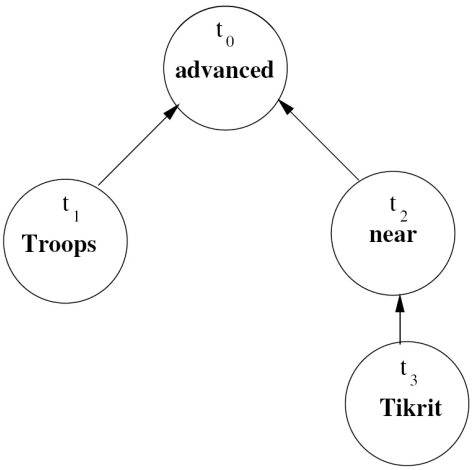


Abbildung 1.34: Dependecytree

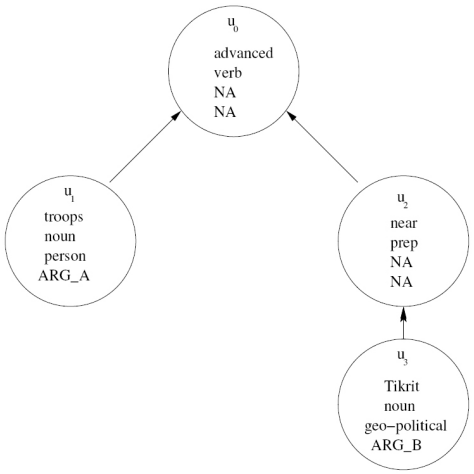


Abbildung 1.35: Erweiterter Dependecytree

SVM Methode braucht Trainingsdaten, damit es lernen kann. Die Trainingsdaten können menschenlich erstellen. Es kostet aber viele Arbeitskräfte. Sie können auch semi-automatisch erzeugt werden. Diese Methode erfordert keine Experten aus verschiedenen Bereichen. Es braucht nur die Leute mit fachlichen Kenntnissen, um die Beziehung zwischen Entitäten zu beurteilen. Dann werden die Daten als Trainingsdaten gespeichert. Die Daten können auch für andere Learning Algorithmen verwenden.
1. Die Relationen zwischen zwei Entitäten werden gegeben.
2. Menschen entscheiden ja und nein.
3. Die Daten werden als Trainingsdaten gespeichert.

Vektor Bildung

Für die Verarbeitung natürlicher Sprache Probleme mit SVM ist es ein wichtiges Kettenglied, wie man die Relationen zwischen zwei Entitäten als Vektor bilden. Ein Vektor ist ein Daten mit der Dimension, die die Eigenschaft zwischen zwei Entitäten in dem Dokument beschreibt. Die Vektorelemente können binären sein. 1 oder 0 bedeutet, ob die Eigenschaft existiert ist. Die Wert der i-te Dimension des Vektors wird durch Funktion

$$f_i : H \times T- > \{0, 1\} \tag{1.66}$$

errechnet. Wobei sind H die alle Eigenschaft in dem Dokument für die zwei Entitäten. T sind alle Eigenschaft (Dimension), die man definiert. Dann ist die i-te Vektorelement

$$x_i = f_i(h, t). \tag{1.67}$$

Beispeil : Man definiert eine Vektorraum (Es gibt Persons, Es Organizations, Es gibt Physical, $\cdots$). Für "Herr Ackermannünd Äcor GmbHïn "Herr Ackermann ist ein Manager von Acor GmbH"kann man dann ein Vektor (1,1,0,$\cdots$) bilden.

Gewichte Vektor

Mit der Trainingsdaten wird eine gewichte Vektor errechnet. Die gewichte Funktion:

Abbildung 1.36: SVM

$$y = \overrightarrow{\omega}\,\overrightarrow{x} + b, \tag{1.68}$$

wobei $\overrightarrow{\omega}$ die gewichte Vektor ist.

Klassifikation

Mit der gewichte Vektor werden neue Daten entscheidet. Mit einem Fenster werden die neue Daten extrahieren. Allgemeine stehen die Entitäten einer Relation nahe. Aber wenn das Fenster zu groß ist, ist die Laufzeit lange. Wenn Fenstergroß 7 ist, gilbt es maximal 7 Entitäten 21 Paar der Entität. Zwischen ein Paar (E_i, E_j) steht Relation? Egal, Es ist nicht wichtig. Wichtig ist, ob das Paar (E_i, E_j) gehört zu Klasse der Relation A ist. Um es zu entscheiden, muss zuerst das Paar als ein Vektor $(\overrightarrow{x})$ gebildet werden. Durch die gewichte Funktion (1.68) wird die Wert y ermittelt. Wenn y positive ist, ist das Paar (E_i, E_j) zu Klasse A gehört.

Abbildung 1.37: Die Darstellung der gewichte Funktion

1.5.5 Kernel Methoden

Kernelfunktion wird ersten in SVM für nichtlineare Erweiterung verwendet. Vergleich mit SVM-Methode braucht man für Kernel-Methode ohne Vektor zu bilden. Man braucht nur zwei Objekte zu vergleichen. Wenn sie ähnlich sind, sind sie gehört zu gleich Klasse. Das Klassifikationsproblem ist jetzt das Ähnlichkeitsproblem. Für das Ähnlichkeitsproblem sind nur typische Exemplare notwendig anstatt riesigen Trainingsdaten.

Subsequenz Kernel

Sind die zwei Sequenzen der Wörter ähnlich? Es wird eine Kern-Funktion definiert. Für eine Sequenz $X = x_1x_2x_3 \cdots x_s$ bedeutet $i = [i_1 i_2 \cdots i_n]$ eine Index von X, wobei $1 \leq i_1 < i_2 < \cdots < i_n \leq n$. x_i ist ein Wort. s ist hier die Länge der Sequenz. Dann ist $X[i]$ eine Teilsequenz von X.
Z.B. X ='ACDBABC' ist eine Sequenz, wobei A ein Wort ist. Für eine Teilsequenz 'ADB' gebt es zwei Index $\underline{A}C\underline{DB}ABC$ ([134]), $\underline{A}C\underline{D}BA\underline{B}C$ ([136]).
Wir definieren die Länge $L(i)$ der Index-Sequenz i in X als $i_n - i_1 + 1$. Z.B. für $i = [134]$ ist $L(i) = 3$; für $j = [136]$ ist $L(j) = 5$.
Für zwei Sequenzen X und Y ist $\sum^n$ die Menge der gemeinsame Teilsequenzen. Die Kern-Funktion ist dann:

$$K_n(X, Y) = \sum_{u \in \sum^n} \sum_{i:u \prec X[i]} \sum_{j:u \prec Y[j]} \lambda^{L[i]+L[j]}, \tag{1.69}$$

wobei u die gemeinsame Teilsequenz ist, und λ ein verfallenden Faktor ist ($\lambda < 1$). λ macht Kern-Funktion umgekehrt proportional zu die Länge $L(i)$.
Für die Verarbeitung natürlicher Sprache ist nur Subsequenz Kernel nicht genug. In der Kern-Entscheidfunktion muss auch die Semantik-Ähnlichkeit errechnet werden.

1.6 Semantic Role Labeling

1.6.1 Semantische Rollen

Eine *semantische Rolle* ist „die Bedeutungsfunktion eines Satzteils auf den ganzen Satz“[Sem08] oder genauer „die semantische Relation der Satzbestandteile zum Prädikat“[EV06]. Das Konzept der semantischen Rollen gibt es bereits seit Ende der 60er Jahre und stammt aus der Linguistik. Semantische Rollen werden von vielen neuartigen Grammatikmodellen genutzt, um Syntax und Semantik in einem Modell zu erfassen und so etwas wie eine Universalgrammatik für alle Sprachen darzustellen.[Sem08]
Es gibt viele unterschiedlich spezielle Rollenaufteilungen für semantische Rollen. Im Folgenden sollen die semantischen Rollen nach Fillmore (1971) beschrieben werden, welche eine eher generellere Rollenaufteilung beschreiben.

- ***Agent*** - Dieser führt die Handlung aus.
  Bsp.: Die nächste Frage stellt *der Kollege Burgbacher*.
- ***Experiencer*** - Dieser nimmt etwas wahr oder fühlt etwas.
  Bsp.: *Sie* haben den Vorschlag gehört.
- ***Instrument*** - Dieses ist ein Mittel, mit dem eine Handlung ausgeführt wird.
  Bsp.: Die Wahl findet *mit verdeckten Stimmkarten*, also geheim, statt.
- ***Object*** (oft auch *Theme*) - Dieses verändert sich durch die Handlung.
  Bsp.: Die Regierung hat *ein Denkmal* errichtet.
- ***Time*** - Dies ist die Zeit des Geschehens.
  Bsp.: Darüber werden wir *morgen* beraten.
- ***Location*** - Dieses ist der Ort des Geschehens.
  Bsp.: Sie dürfen Ihre Stimmkarte nur *in der Wahlkabine* ankreuzen.
- ***Goal*** - Dieses ist der Ort zu dem sich etwas bewegt.
  Bsp.: Wir sind wieder *in die Mitte Europas* gerückt, was das Wachstum anbelangt.
- ***Path*** - Dies ist der Weg des Geschehens.
  Bsp.: Im Übrigen haben die USA hervorragend mit den menschenverachtenden Taliban verhandelt, und zwar über eine Gaspipeline *durch Afghanistan*.
- ***Source*** - Dies ist der Ort von dem aus sich etwas bewegt.
  Bsp.: Herr Kollege Brüderle, setzen Sie sich doch einmal aufs Fahrrad und fahren *von Ihrem Heimatort aus* in Richtung Westen nach Frankreich.

[Kas08, Kri04]

Oberflächenkasus

Die semantischen Rollen werden häufig auch als Tiefenkasus bezeichnet, das auf der Syntax beruhende Gegenstück ist der Oberflächenkasus.
Der Oberflächenkasus hilft die semantischen Rollen zu bestimmen. Wie in Tabelle 1.6 an den kursiven Stellen zu sehen ist, ist der Oberflächenkasus aber leider nicht eindeutig. Weitere

	Maskulinum	Femininum	Neutrum	Plural
Nominativ (wer oder was)	der - er	*die - sie*	*das - es*	*die - sie*
Akkusativ (wen oder was)	den - ihn	*die - sie*	*das - es*	*die - sie*
Dativ (wem)	dem - ihm	der - ihr	dem - ihm	den - ihnen
Genitiv (wessen)	des - seiner	der - ihrer	des - seiner	der - ihrer

Tabelle 1.6: Oberflächenkasus

Übereinstimmungen zwischen verschiedenen Kasus sind ebenfalls in Tabelle 1.6 abzulesen. Wenn nicht alle Formen unterschiedlich ausgeprägt sind, so nennt sich dies auch *Synkretismus*. Würde kein Synkretismus im Oberflächenkasus vorliegen, so wäre es eine einfache Aufgabe die semantischen Rollen zu bestimmen, die allerdings auch wieder sprachspezifisch wäre. [EV06]

Part of speech tagging

Beim *part of speech (POS) tagging* wird jedem Wort in einem Satz eine Wortart zugeordnet. Außerdem wir auch jedem Satzteil eine Satzphrasenart zugeordnet. Die Ergebnisse des POS taggings lassen sich gut durch einen *Syntaxbaum* visualisieren. Abbildung 1.38 zeigt das Beispiel eines Syntaxbaumes.

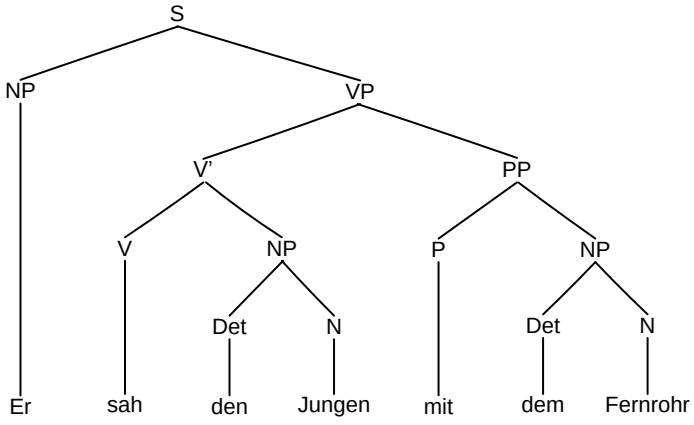


Abbildung 1.38: Syntaxbaum

S: Satz
NP: Nomenphrase
VP: Verbphrase
PP: Präpositionale Phrase (Adverbiale Phrase)
N: Nomen
V: Verb
Det: Determinantien (z.B. Artikel)
P: Präposition
V': Verbphrase die einer Verbphrase (VP) untergeordnet ist (auch V" usw. möglich)

Im Beispiel aus Abbildung 1.38 wird auch direkt ein Problem der Wortartenzuweisung deutlich. Denn dieser Satz ist auf zwei Weisen interpretierbar.

1. Er sah den Jungen durch sein Fernrohr.
2. Er sah den Jungen, der das Fernrohr hat.

Der Syntaxbaum aus Abbildung 1.38 stellt die 1. Variante dar. In der 2. Variante wäre der Satzteil „mit dem Fernrohr" keine eigenständige Präpositionale Phrase der Verbphrase mehr, sondern wäre als präpositionale Phrase an die Nomenphrase „den Jungen" angeschlossen.
Allerdings sind Syntaxbäume trotzdem sehr hilfreich, um auf semantische Rollen schließen zu können. So beschreibt eine NP vor einer VP meist ein Subject und dieses ist meist der Agent.

Eine NP in einer VP hingegen beschreibt meistens ein Object. Und eine PP beschreibt häufig Informationen über Ort und Zeit. [Mey02]

1.6.2 Automatisches Zuweisen von semantischen Rollen

Das automatische Zuweisen von semantischen Rollen, auch Semantic Role Labeling genannt, bezeichnet die Gruppierung von Wörtern eines Satzes in einer Weise, die es anschließend ermöglicht den Wortgruppen eine semantische Rolle zuzuweisen. Das verbreiteteste Vorgehen dabei ist, das Verb eines Satz als zentrales Element zu untersuchen. Zu jedem Verb gehören eine Reihe von verallgemeinerten Argumenten, die in ihrer konkreten Ausprägung dann einen Satz bilden. Die Argumente entsehen hauptsächlich durch den Oberflächenkasus. So könnte man z.B. Sätze mit dem Verb *befürworten* durch die Argumente *Befürworter* und *Befürwortetes* gliedern, so dass sie die Struktur <*Befürworter*> *befürwortet* <*Befürwortetes*> erhalten. Eine Ausprägung dieser Satzform wäre dann zum Beispiel: „Die Fraktion der SPD befürwortet den Antrag XYZ.“.
Wie man erkennen kann, sind solche Argumente Träger semantischer Rollen. Ziel ist es alo einen Satz so erfassen, dass alle Wörter eines Satzes auf die Argumente des Verbs abgebildet werden und so ihre semantischen Rollen erhalten. Diese Abbildung wird auf der Basis von Informationen über die Charakteristik des Satz vorgenommen. Als Charakteristika (auch *features* genannt) dienen z.B. syntaktische Informationen, wie der Oberflächenkasus, oder Part-of.Speech-Tags, wie sie oben vorgstellt wurden.

Probabilistisches Semantic Role Labeling

Probabilistisches Semantic Role Labeling soll wie ähnliche Probleme behandelt werden, z.B. POS, Syntaxanalyse...). Hierfür sollen statistische Techniken ausgenutzt werden.
Die Zuweisung semantischer Rollen wird in zwei Teilaufgaben aufgeteilt. Zum einen die Identifizierung der Frameelementgrenzen und zum anderen die Zuweisung einer semantischen Rolle zu jedem Frameelement. Frameelemente sind Bereiche im Satz. Für verschiedene Frames sind unterschiedliche Frameelemente möglich. In Abbildung 1.39 sind Frames und Frameelemente aus der FrameNet-Datenbank zu sehen. Außerdem sind an die Frames noch die jeweiligen Triggerverben und -nomen angeheftet. Findet man ein solches Triggerwort im Text, so weiß man, dass das Frame zu diesem Triggerwort in diesem Bereich vorliegen muss. Nun kann nach den definierten Frameelementen gesucht werden.

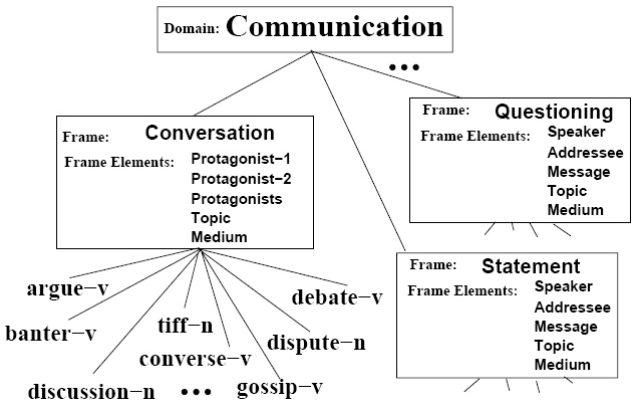
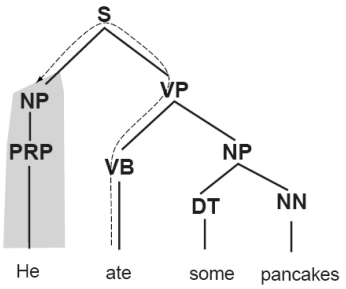


Abbildung 1.39: Die Domain *Communication* aus der FrameNet-Datenbank

In der folgenden Beschreibung wird davon ausgegangen, dass die Grenzen bereits per Hand annotiert worden sind. Die Ergebnisse von automatisch erkannten Grenzen sind in etwa gleich.



Frequency	*Path*	*Description*
14.2%	VB↑VP↓PP	PP argument/adjunct
11.8	VB↑VP↑S↓NP	subject
10.1	VB↑VP↓NP	object
7.9	VB↑VP↑VP↑S↓NP	subject (embedded VP)
4.1	VB↑VP↓ADVP	adverbial adjunct
3.0	NN↑NP↑NP↓PP	prepositional complement of noun
1.7	VB↑VP↓PRT	adverbial particle
1.6	VB↑VP↑VP↑VP↑S↓NP	subject (embedded VP)
14.2		no matching parse constituent
31.4	Other	

Abbildung 1.40: Das Parse Tree Path Merkmal

Um nun die semantischen Rollen zuweisen zu können, sind die folgenden *Merkmale* gegeben:

1. **Phrase Type (pt):** Dieser beschreibt den syntaktischen Typ des Satzteils, also z.B. NP, VP oder S.

2. **Governing Category (gov):** Von einer NP ausgehend wird der nächste Vorfahr S oder VP gesucht. Wenn der nächste Vorfahr das S ist, so bekommt die NP die Governing Category *Subjekt*, ist der nächste Vorfahr eine VP, so wird der NP die Governing Category *Objekt* zugeordnet. Für andere Typen als NP als Ausgangsbasis hat die Governing Category nur wenig Effekt, weshalb dieses Merkmal auch nur für NP's genutzt wird.

3. **Position:** Diese gibt an, ob die Komponente vor oder hinter dem Prädikat angesiedelt ist .

4. **Parse Tree Path:** Dieses Merkmal beschreibt den Pfad vom Zielwort zur Komponente der Frage. Im Beispiel aus Abbildung 1.40 ist das Zielwort „He“ und beantwortet die Frage nach dem „wer“. VB↑VP↑S↓NP bedeutet also, dass vom Verb (VB) nach oben (↑) zur Verbphrase (VP), von dort nach oben (↑) zum Satz (S) und nun nach unten (↓) zur Nomenphrase (NP) durch den Baum gelaufen wird. Wenn wir diesen Pfad durchlaufen haben, so handelt es sich um ein Subjekt. Dies ist in der Tabelle aus Abbildung 1.40 abzulesen. Dort werden auch die Häufigkeiten der unterschiedlichen Pfade im Bezug auf den Datensatz (FrameNet Datenbank, mit ca. 8148 Beobachtungen) aufgezeigt.

5. **Voice:** Dieses Merkmal unterscheidet zwischen aktiven und passiven Verben. Im vorliegenden Datensatz sind ca. 5 % der Verben passiv.

6. **Head Word (h):** Hier wird das Hauptwort des Satzteils makiert. Mit diesem kann die grammatische Funktion bestimmt werden.
   - Bsp. NP: the old fashioned *door*
   - Bsp. VP: might be *hit*
   - Bsp. PP: *on* the table

Um die Wahrscheinlichkeiten direkt über der vollen Menge der Merkmale zu berechnen, also mit

$$P(r \mid h, pt, gov, position, voice, t) = \frac{\#r, h, pt, gov, position, voice, t}{\#h, pt, gov, position, voice, t}, \tag{1.70}$$

mit r = semantische Rolle und t = target word / Verb,
sind leider zu wenig Daten vorhanden. Deshalb wird hier die lineare Interpolation benutzt und die Wahrscheinlichkeiten werden somit unterschiedlich kombiniert.

$$\begin{aligned} P(r \mid Komponente) = {} & \lambda_1 P(r \mid t) + \lambda_2 P(r \mid pt, t) + \lambda_3 P(r \mid pt, gov, t) + \\ & \lambda_4 P(r \mid pt, position, voice) + \lambda_5 P(r \mid pt, position, voice, t) + \\ & \lambda_6 P(r \mid h) + \lambda_7 P(r \mid h, t) + \lambda_8 P(r \mid h, pt, t) \\ & mit \sum_i \lambda_i = 1 \end{aligned} \tag{1.71}$$

Distribution	*Coverage*	*Accuracy*	*Performance*
$P(r\|t)$	100.0%	40.9%	40.9%
$P(r\|pt,t)$	92.5	60.1	55.6
$P(r\|pt,gov,t)$	92.0	66.6	61.3
$P(r\|pt,position,voice)$	98.8	57.1	56.4
$P(r\|pt,position,voice,t)$	90.8	70.1	63.7
$P(r\|h)$	80.3	73.6	59.1
$P(r\|h,t)$	56.0	86.6	48.5
$P(r\|h,pt,t)$	50.1	87.4	43.8

Abbildung 1.41: Wahrscheinlichkeitsverteilungen in der finalen Version

In Abbildung 1.41 sind die acht Wahrscheinlichkeitsverteilungen aus Formel 1.71 abgebildet. Accuracy steht hier für den Anteil an Testdaten, für die die korrekte Rolle vorhergesagt wurde. Coverage ist der Anteil der Testdaten, für die das bedingte Ereignis in den Trainigsdaten gesichtet wurde. Performance beschreibt das Produkt von Accuracy und Coverage.
An Abbildung 1.41 ist zu erkennen, dass es einen Tradeoff zwischen spezifischen Verteilungen, also Verteilungen mit einer hohen Accuracy und einer niedrigen Coverage, und unspezifischen gibt. So ist die Güte (Accuracy) für die unspezifischen Verteilungen deutlich höher, jedoch ist die Abbdeckung im Datensatz nicht mehr so hoch. Bei den spezifischen Verteilungen hat man eine sehr gute Abdeckung im Datensatz jedoch lässt die Güte hier zu wünschen übrig.
So entstand die Idee eines Rankings, um die λ-Werte aus Formel 1.71 festzulegen. Es wurde ein Netz gebildet von spezifischen zu unspezifischen Verteilungen, welches in Abbildung 1.42 zu sehen ist.
In diesem Netz wird zunächst immer versucht eine der obersten und somit unspezifischten Verteilungen zu benutzten. Erst wenn dies nicht möglich ist, da keine Beispiele hierzu vorhanden sind, so wird eine Ebene tiefer im Netz gegangen. Ein solches Netzwerk wird auch *BackOff-Kombination* genannt.

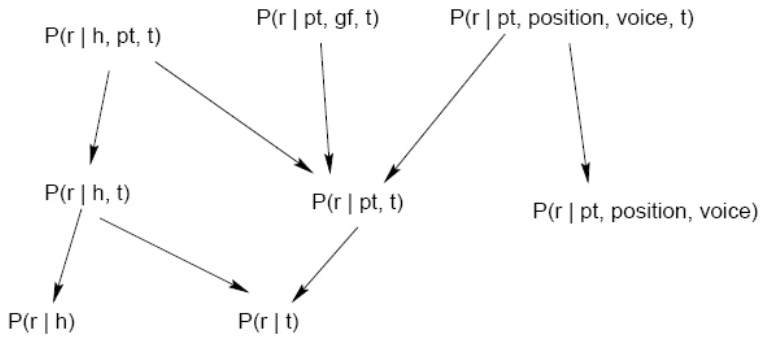


Abbildung 1.42: Verteilungsnetz / BackOff-Kombination

Andere Schemata als die BackOff-Kombination zur Wahl der λ-Werte, haben relativ wenig Effekt gezeigt, denn die Bewertung hängt nur vom Ranking der Wahrscheinlichkeiten ab und nicht von deren exakten Werten.
In Tabelle 1.7 sind die Ergebnisse des automatischen Zuweisens semantischer Rollen mit der probabilistischen Methode festgehalten. Die Basis ist hier entstanden, indem immer die spezifischste Verteilung, also $P(r \mid t) = \frac{\#(r,t)}{\#(t)}$ mit Accuracy = 40,9, wie in Abbildung 1.41 zu

Kombinationsmethode	korrekt
Lineare Interpolation	79,5%
Backoff, Lineare Interpolation	80,4%
Basis: häufigste Rolle	40,9%

Tabelle 1.7: Ergebnisse des probabilistischen Semantic Role Labeling

sehen ist, gewählt und damit die semantische Rolle zugewiesen wurde. Wie bereits erwähnt wurde, wurde als Korpus die FrameNet Datenbank mit 8148 Beobachtungen verwendet. [GJ00, GJ02a]

Semantic Role Labeling mit SVM

In Kaptiel 1.4.1 wurde die SVM bereits detailliert vorgestellt. Hier soll nun ihre Anwendung für die Methoden des Semantic Role Labeling demonstriert werden. Das Ziel ist, die SVM so zu trainieren, dass sie ein Wort nach seiner semantischen Rolle klassifiert. Da die SVM aber ein binärer Klassifizierer ist und die Anzahl der semantischen Rollen überlicherweise größer als zwei ist, muss das Konzept der SVM zunächst in das eines Multiklassen-Klassifizierers geändert werden. Dazu bieten sich zwei Ansatze [PHK+05].
Der *One-versus-All-Ansatz* (OVA) sieht für jede Klasse (die hier einer semantischen Rolle entspricht) eine SVM vor, die entscheidet, ob das einzuordnende Wort zu genau dieser eine Klasse gehört oder zu der Menge der übrigen. Für n Klassen sind damit höchstens n SVMs auszuwerten. Ein Nachteil dieser Lösung ist der hohe Daten- und Lernaufwand. Jede SVM muss durch das OVA-Prinzip immer auf allen vorhandenen Daten lernen.
Eine Alternative ist der *paarweise Vergleich*. Für n Klassen (semantische Rollen) werden $\frac{n*(n-1)}{2}$ SVMs benötigt, so dass für jedes Klassenpaar eine Entscheidung zugunsten einer der Klassen des Paars entsteht. Die entgültig zugeordnete Klasse ist die, die die meisten Einzelentscheidungen für sich gewinnen konnte. Vorteil dieses Ansatz ist die im Vergleich zum OVA-Prinzip kleine Trainingsdatenmenge, da mit jeder SVM immer nur zwei Klassen verglichen werden. Dieser Vorteil kann sich aber auch durch die viel größere Anzahl an SVMs ins Gegenteil kehren. Die genauen Auswirkungen hängen von jeweiligen Anwendungsfall ab, so dass die Methode zur Erweiterung der SVM zum Multiklassen-Klassifizierer gut überlegt sein sollte.
Das SRL geschieht durch die Bewertung von Ausprägungen verschiedener Merkmale (*features*). Im vorherigen Abschnitt wurden bereits einige Merkmale, wie Pharse Type, Governing Category oder Position vorgestellt. Letzteres ist ein numerisches Feature, die beiden zuerst genannten nicht. Da die SVM aber ein numerischer Klassifizierer ist, besteht das Problem, nicht numerische Features passend zu codieren [KM00]. Dies kann durch die Verwendung von binären Feature-Vektoren geschehen. Solch ein Vektor besitzt für jede mögliche Ausprägung jedes Features eine Komponente. Ist eine Ausprägung vorhanden, so wird die Komponenete auf 1 gesetzt, sonst auf 0. Bei dieser Art der Darstellung erhält ein Vektor schnell mehr als 100.000 Dimensionen. Ein Trick macht ihn aber trotzdem effizient einsetzbar und wenig speicherintensiv. Der Vektor kann nämlich sehr gut kompaktiert werden, indem nur die Indizies der Komponenten gespeichert werden, die mit einer 1 besetzt sind. Die Anzahl dieser Komponenten ist in den allermeisten Fällen sehr klein [Joa98]. Man spricht bei derartigen Vektoren aus von spärlich oder dünn besetzten (*sparse*) Feature-Vektoren.

Eine Alternative zu den binären stellen TF-IDF basierte Vektoren dar, auf die aber hier nicht weiter eingegangen werden soll.
Unter Umständen liegen für das SRL keine oder nur begrenzte syntaktische Informationen vor. Dies kann zum Beispiel bei sehr komplizierten Sprachen oder bei solchen, für dies es noch keine umfassenden Daten gibt, der Fall sein. Mit der Methode des *Chunking* lassen sich dann aber trotzdem sogenannte oberflächliche/flache syntaktische Informationen gewinnen (shallow parse). Ein vollständiges syntaktsisches Parsen wird dagegen als Tiefenparsen (deep parse) bezeichnet. Das Chunking bezeichnet das Erkennen von zusammenhängenden Wörtern in einem Satz, die dann durch die Zuweisung Chunk Labels, z.B. in der IOB-Notation (Inside, Outside, Begining), die flachen Syntaxinformationen darstellen.
Ein Beispiel:

> Eingabe:
>
> $[_{PPER}$ Ich] $[_{VAFI}$ habe] $[_{VVPP}$ ausgesagt] , $[_{ART}$ die] $[_{ADJA}$ schwarzen] $[_{NN}$ Koffer] $[_{PTKNEG}$ nicht] $[_{VVPP}$ genommen] $[_{PTKZU}$ zu] $[_{VAINF}$ haben].
>
> Ausgabe:
>
> $[_{B-NP}$ Ich] $[_{B-VP}$ habe] $[_{I-VP}$ ausgesagt] , $[_{B-NP}$ die] $[_{I-NP}$ schwarzen] $[_{I-NP}$ Koffer] $[_{O}$ nicht] $[_{B-VP}$ genommen] $[_{I-VP}$ zu] $[_{I-VP}$ haben].

Auch für dieses Verfahren lässt sich die SVM, analog zur Verwendung beim SRL, nutzen. Eine fertige Implentierung bietet YamCha (Yet another multipurpose Chunk Annotator) [Yam] [KM00], bester Chunk Annotator im CoNLL2000 Shared Task im Bereich Chunking. Als Basis nutzt YamCha eine SVM mit paarweiser Klassifizierung, die über das Chunk Label eines Wortes entscheidet. Die Eingabe ist ein Text mit POS-Tags. Als Features werden die umgebenden Wörter und POS-Tags, je mit einer Fenstergröße von zwei, sowie die beiden letzten vergebenen Chunk Labels verwendet.
Anhand der unterschiedlichen Tiefen der syntaktischen Analyse unterscheidet man drei unterschiedliche Verfahren des SRL:

- constituent-by-constituent (c-by-c)
  Für dieses Verfahren dient ein vollständiges syntaktisches Parsen als Grundlage. Der Syntaxbaum eines Satzes wird zu einer Kette seiner Komponeneten linearisiert und dann werden komponentenweise (constituent-by-constituent) die semantischen Rollen bestimmt.

- pharse-by-pharse (p-by-p)
  Der Betrachtungsraum ist wie beim c-by-c Verfahren der ganze Satz. Da jedoch keine vollständigen syntaktsichen Informationen vorliegen, kann nicht wie beim c-by-c vorgegangen werden. Aus dem Kontext des Satzes müssen andere Informationen zur Klassifizierung herangezogen werden. Um doch syntaktische Informationen zu verwenden, findet das Chunking hier Anwendnung.

- word-by-word (w-by-w)
  Klassifierungen werden beim w-by-w Labeling nur auf Basis des zu klassifierenden Wortes und der umgebenden Worte durchgeführt. Es werden also keine Satzgrenzen berücksichtigt. Auch hier lässt sich das Chunking sinnvoll einsetzten, wenn keine syntaktischen Informationen vorhanden sind.

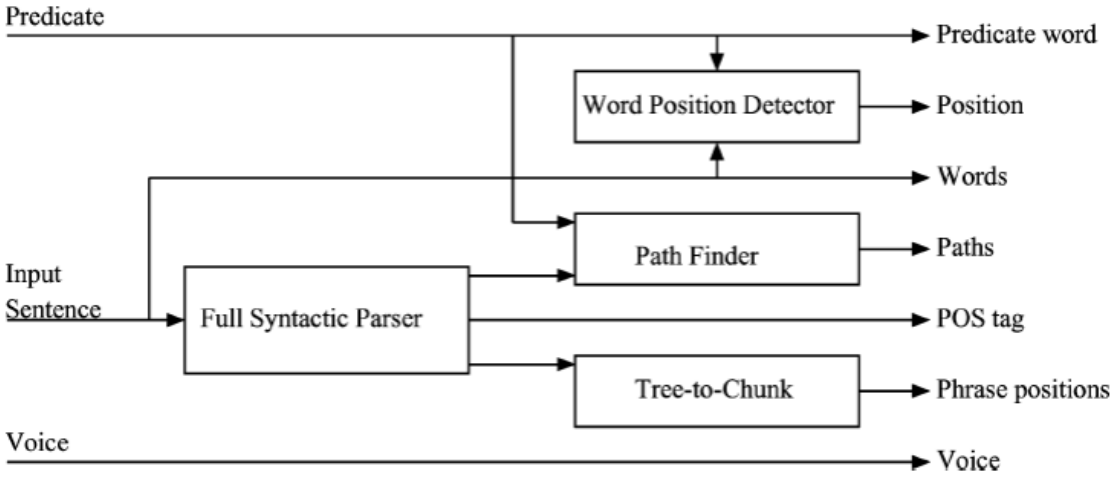


Abbildung 1.43: Deep-Parse-System aus [PHK+05]

[PHK+05] vergleicht das Deep-Parsing und das Shallow-Parsing in drei Ausprägungen: Ein c-by-c Verfahren mit vollständigem Syntax-Parsing (Deep-Parsing) und zwei w-by-w Verfahren, einmal mit syntaktischem Deep-Parsing und einmal ohne. Ingesamt ist [PHK+05] eine Weiterentwicklung des Ansatzes von [GJ02a], der die dort verwendeten Wahrscheinlichkeiten (siehe Kapitel 1.6.2) durch SVMs ersetzt und neue Features ergänzt. Zu den neuen Features zählen z.B. die NER (Person, Organization, Location, ...), das POS-Tag des Headwords und die *Verb Sense Information*. Letzere erkennt und merkt sich, welche Bedeutung eines mehrdeutigen Verbes (mit gleichen Argumentanzahlen) gerade gemeint ist. Ein Beispiel ist das Verb *wiegen*. Einerseits bezeichnet es das Wiegen eines Gewichtes mit den Argumenten *Wiegende(r)* und *Gewogenes*, andererseits das Wiegen eines Kindes in der Wiege mit den Argumenten *Wiegende(r)* und *Kind*.
Die Abbildungen 1.43 und 1.44 zeigen die beiden zu vergleichenden grundlegenden Systemarchtiekturen. Das Deep-Parse-System verwendet den syntaktischen Parser von Charniak [Cha00]. Im Shallow-Parse-System wird dieser durch einen POS-Tagger und YamCha als Chunker ersetzt.
Als Korpus für den Vergleich dient der PropBank-Korpus von Juli 2002. Wie allgemein üblich, wurden die Sektionen 02-21 zum Lernen, die Sektion 00 zur Entwicklung und Debugging und alle übrigen zum Testen verwendet.

System	Precision	Recall	F_1
Deep-Parse, c-by-c	80, 6	67, 1	73, 2
Deep-Parse, w-by-w	70, 7	60, 5	65, 2
Shallow-Parse, w-by-w	66, 2	54, 9	60, 0

Tabelle 1.8: Vergleichsergebnisse aus [PHK+05]

Tabelle 1.8 zeigt die konkreten Ergebnisse des Vergleichs von [PHK+05]. Es fällt auf, dass die Verfahren mit Tiefen-Parsing überlegen sind und ein weiter gefasster Kontext (c-by-c) sich postiv auf die Ergebnisse auswirkt. Der Hauptgrund für die großen Leistungsunterschiede ist

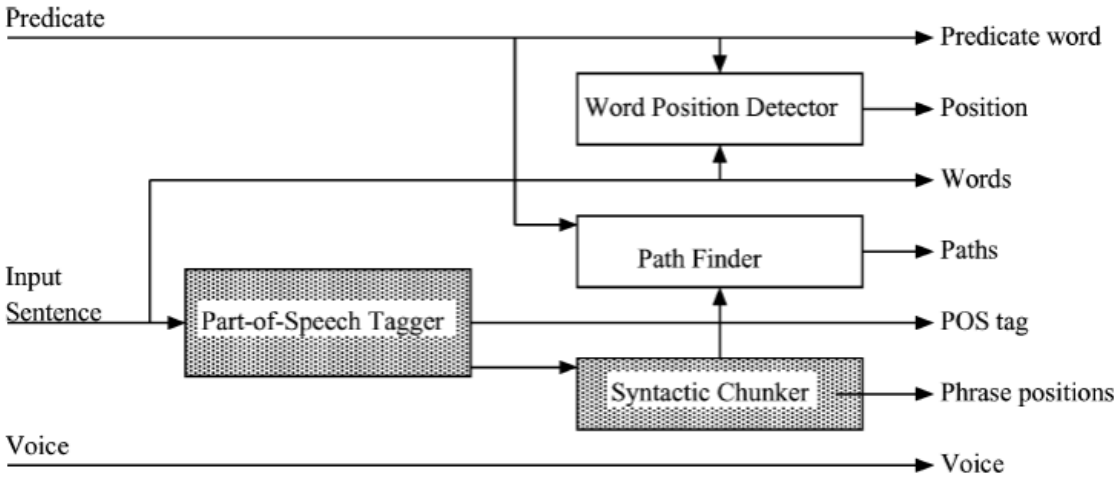


Abbildung 1.44: Shallow-Parse-System aus [PHK+05]

in der unterschiedlichen Bildung eines der wichtigsten Features, des Pfades (Path), zu finden. Bei einer syntaktischen Tiefenanalyse enthält der Pfad mehr und genauere Information als bei einer flachen Analyse.
Im Rahmen dieses Vergleichs wurde auch eine Bewertung des Beitrags einzelner Features zu Gesamtergebnis ermittelt. Dabei kommt dem Path-Feature, wie schon erwähnt, eine der wichtigsten Rollen zu. Noch wichtiger ist nach [PHK+05] das Headword. Positiv wirken sich auch das POS-Tag des Headwords und die Verb Sense Information aus. Überraschender Weise verschlechtert die NER allerdings die Ergebnisse. Dies liege, so die Autoren, daran, dass viele Komponenten Named Enities enthielten, aber dabei keine Charakteristika des Prädikats seien. Somit wirkt die NER eher als Störquelle.

Semantic Role Labeling mit Maximum Entropy

Einen guten Einstieg in diese Thematik liefert [Rat97]. Die grundlegende Idee beim SRL mit Maximum Entropy Models (MEM) ist, das Auftreten einer einer Klasse a in einem Kontext b statistisch abzuschätzen. Eine Klasse kann dabei ein POS-Tag oder eine semantische Rolle sein. Der Kontext in dem diese auftreten sind Wörter oder Sätze. Diese allgeimeine Formulierung lässt erahnen, dass MEM ein weites Anwendungsgebiet haben, das aber hier auf SRL begrenzt werden soll.
Sei $\mathcal{A}$ die Menge aller Klassen, $\mathcal{B}$ die Menge aller Kontexte und $x = (a, b)$ eine Klasse-Kontext-Kombination mit $a \in \mathcal{A}$ und $b \in \mathcal{B}$. Weiter sei $\mathcal{E} = \mathcal{A} \times \mathcal{B}$ die Menge aller solcher Kombinationen x. Die Wahrscheinlichkeit für ein Auftreten von x wird mit der Wahrscheinlichkeitsverteilung $p(a, b) = p(x)$ bezeichnet. Für das Modell gibt es zwei Wahrscheinlichkeitsverteilungen: $\tilde{p}(x)$ ist die Verteilung, die als Parameter für das Modell fungiert und sich aus konkreten Beobachtungen speist. $\mathcal{S} \subseteq \mathcal{E}$ sei dabei die Menge der Beobachtungen. $\tilde{p}(x)$ gilt also nur über $\mathcal{S}$. Die zweite Verteilung, $p(x)$, ist eine vom Modell errechnete, verallgemeinerte Verteilung über $\mathcal{E}$, also über allen möglichen Kombinationen (a, b).
Die Beobachtungen, die $\tilde{p}(x)$ definieren, werden die mittels einer Feature-Funktion $f : \mathcal{E} \rightarrow \{0, 1\}$, die für alle Klasse-Kontext-Kombinationen x angibt, ob sie für das Feature relevant

sind. Es sei hier von k Features mit je einem dazugehörigen f_i $(i = 1, ..., k)$ ausgegangen. Die Features werden durch die folgende Bedingung in das MEM integriert:

$$E_p f_i = E_{\tilde{p}} f_i \qquad \forall i = 1, ..., k \tag{1.72}$$

E bezeichnet dabei den Erwartungswert der jeweiligen Verteilungsfunktion. So ist $E_{\tilde{p}} = \sum_{x \in \mathcal{S}}^{m} x \cdot \tilde{p}(x)$ und $E_{p(x)}$ analog. Mit dieser Bedingung erreicht man, dass die vom Modell errechnete Wahrscheinlichkeitsverteilung konsistent mit den Beobachtungen ist.
Das MEM für SRL ist nun ein Optimierungsmodell

$$p* = \underset{p \in \mathcal{P}}{\text{argmax}}, H(p)$$
$$H(p) = -\sum_{x \in \mathcal{E}} p(x) \log p(x)$$
$$\mathcal{P} = \{p | E_p f_i = E_{\tilde{p}} f_i, i = 1, ..., k\}$$

welches aus allen gültigen Verteilungen die Verteilung $p*$ findet, die die Entropie H maximiert. Eine Verteilung p ist gültig, wenn sie die Gleichung (1.72) erfüllt.
Die optimale Verteilung ist aber anders darstellbar [Rat97]:

$$p* = \pi \prod_{j=1}^{k} \alpha_j^{f_j(x)} \tag{1.73}$$

Dabei sind die α_j die einzigen Variablen, die sich durch den Generalized Scaling Algorithmus [Rat97] relativ leicht brechnen lassen.
Abschließend soll noch ein Beispiel die Formulierung der Features und der Gleichung 1.72 verdeutlichen: Sei $\mathcal{A} = \{x, y\}$ und $\mathcal{B} = \{0, 1\}$. Als Feature soll das Vorkommen des Kontextes $b = 0$ modelliert werden, dessen Häufigkeit man insgesamt mit 60% beobachtet habe. Dies stellt Tabelle 1.6.2 dar.

Tabelle 1.9: Ausgangsdaten für SRL-MEM Beispiel

$p(a, b)$	0	1
x	?	?
y	?	?
$\sum$	0,6	?

Das Feature lässt sich durch die Funktion

$$f_i = \begin{cases} 1, & \text{wenn } b = 0 \\ 0, & \text{sonst} \end{cases}$$

darstellen, so dass sich für die Bedingung 1.72 ergibt:

$$\begin{aligned} E_{\tilde{p}} \cdot f_1 &= E_p \cdot f_1 \\ &\Leftrightarrow \\ p(x,0) \cdot 1 + p(y,0) \cdot 1 &= E_p \cdot f_1 \\ &\Leftrightarrow \\ 0,6 &= E_p \cdot f_1 \end{aligned}$$

Das Ergebnis, das die Entropie maximiert, ist in Tabelle 1.6.2 dargestellt. Gültig wären auch alle Ergebnisse deren erste Spalte in Summe 0,6 und deren zweite Spalte summiert 0,4 ergeben. Diese maximieren aber die Entropie nicht.

Tabelle 1.10: Ausgangsdaten für SRL-MEM Beispiel

$p(a,b)$	0	1
x	0,3	0,2
y	0,3	0,2
$\sum$	0,6	0,4

Unsupervised Semantic Role Labeling

Beim *Unsupervised Semantic Role Labeling* soll eine Zuordnung der semantischen Rollen erfolgen, ohne dass manuell getaggte Daten vorliegen. Dies wird erzielt, indem initial Zuordnungen anhand eines Verblexikons gemacht werden, wenn eine Zuordnung eindeutig möglich ist. Anhand dieser Zuordnungen soll dann ein Wahrscheinlichkeitsmodell aufgestellt werden, mit dem die restlichen Daten zugeordnet werden sollen.
Beim Iterieren über den Daten wachsen die annotierten Daten immer weiter an und um am Ende alle semantischen Rollen getaggt zu haben, wird die Schwelle des Wahrscheinlichkeitsmodells immer weiter herabgesetzt.
Das Verblexikon nimmt in dieser Methode eine sehr zentrale Rolle ein. Es listet mögliche Rollen für alle Argumente eines Verbs auf. In Abbildung 1.45 ist ein beispielhafter Eintrag des Verlexikons für das Verb „whisper“ dargestellt.

whisper
Frames: Agent V Agent V Prep(+dest) Recipient Agent V Topic
Verben in der selben (Sub-)Klasse: [bark, croon, drone, grunt, holler, ...]

Abbildung 1.45: Der Eintrag whisper aus dem Verblexikon

Dieser Eintrag ist wie folgt zu verstehen: Das Verb *whisper* hat drei mögliche Frames, die erste mögliche wäre *Agent V*. Frames bestehen wiederum aus Slots, die durch semantische Rollen gefüllt werden sollen. Slots sind alle nicht Verben eines Frames, z.B. Subjekt und Objekt. Bei dem Frame *Agent V Topic* würden die z.B. die Slots Subjekt und Objekt durch die Argumente Agent und Topic gefüllt. Desweiteren sind die Verben aus der selben Subklasse, die hier „*Geräusche erzeugen*“ heißen könnte, angegeben. Diese werden benötigt, falls im späteren Verlauf dieser Methode die Verben generalisiert werden sollen.

Im *Frame-Matching* Schritt sollen für jedes Verb die möglichen Rollen seiner Argumente berechnet werden und die vorhandenen Argumentslots, wie Subjekt, Objekt, indirektes Objekt und Adverbial, vorhergesagt werden. Die Argumente haben nun eine Menge an möglichen Rollen, welche die Argumentslots füllen könnten. Wenn diese Menge einelementig ist, so kann die Rolle zugeordnet werden. Diese Daten, die eindeutig zuzuordnen waren, bilden die gelabelten Anfangsdaten, mit welchen dann das Wahrscheinlichkeitsmodell trainiert werden kann.

Mögliche Frames für V	vorhergesagte Slots			%Frame	%Sent	Score
	Subj.	Obj.	Prbj.			
Agent V	Agent			100	33	133
Agent V Theme	Agent	Theme		100	67	167
Instrument V Theme	Instr.	Theme		100	67	167
Agent V Theme P Instr.	Agent	Theme	Instr.	100	100	200
Agent V Recipient Theme	Agent	Recip.		67	67	133

Tabelle 1.11: Der Frame-Matching Schritt am Beispiel

In Tabelle 1.11 wird der Frame-Matching Schritt am Beispiel genauer dargestellt.
%Frame stellt hier den Anteil der zuzuordneden Argumente dar, die im Frame belegt werden konnten. Für 4 der 5 möglichen Frames konnten alle Argumente in einen Slot einsortiert werden. Diese bekommen also alle den %Frame-Wert 100. Nur für das letzte Frame konnte das Argument Theme nicht zugeordnet werden, da hierzu ein weiterer Objekt-Slot nötig gewesen wäre. Es konnten also nur zwei von drei Argumenten einem vorhergesagten Slot zugeordnet werden. Deshalb bekommt dieses Frame nur den %Frame-Wert 67.
%Sent ist der Anteil von den vorhergesagten Slots, die gefüllt werden konnten. Nur das vierte Frame füllt hier alle vorhergesagten Slots und bekommt somit einen %Sent-Wert von 100.
Die Score eines Frames ist nun einfach die Addition von %Frame und %Sent.
Nun müssen die maximalen Score-Werte betrachtet werden. Hat, wie in unserem Beispiel, nur ein Frame einen maximalen Score-Wert, so kann dieser Frame mit den semantischen Rollen gelabelt werden und schafft somit Anfangsdaten, mit denen dann in einem späteren Schritt die restlichen Daten annotiert werden können.

Um Slots zu füllen, bei denen mehrere Möglichkeiten zur Füllung bestehen, muss nun ein Wahrscheinlichkeitsmodell aufgestellt werden. Dieses sogenannte *BackOff-Modell* ist wie folgt definiert,

$$P(r \mid v,s,n) \quad \longrightarrow \quad \begin{array}{ll} \lambda_1 P_1(r \mid v, sc) & + \\ \lambda_2 P_2(r \mid v, nc) & + \\ \lambda_3 P_3(r \mid vc, s) & + \end{array} \quad \longrightarrow \quad P(r \mid sc) \tag{1.74}$$

mit r = Rolle, v = Verb, s = Slot, n = Nomen im Slot,
sc = Slotklasse, nc = Nomenklasse, vc = Verbklasse.

Wenn die Wahrscheinlichkeit einer Kandidatenrolle einen Schwellenwert *minEvidence* nicht erreicht, so soll dieser noch zu füllende Slot im nächsten BackOff-Level erneut untersucht werden.
Wie in Formel 1.74 zu sehen ist, gibt es drei BackOff-Level. Das erste ist das spezifischste, bei welchem das Verb, der Slot und das Nomen im Slot genutzt werden. $P(r \mid v,s,n)$ ist also die

Wahrscheinlichkeit, wenn die Kombination Verb, Slot und Nomen auftritt, dass dann die Rolle r vorhergesagt wird. Im zweiten BackOff-Level wird schon eine starke Verallgemeinerung vorgenommen. Hier werden drei Wahrscheinlichkeiten additiv kombiniert. In jeder der drei kombinierten Wahrscheinlichkeiten wird eine Sache verallgemeinert. So wird in der dritten Wahrscheinlichkeit, wie bei Abbildung 1.45 bereits erläutert wurde, das Verb auf die Verbklasse verallgemeinert. Die λ-Werte sind in dieser Methode bisher gleichverteilt. Im dritten BackOff-Level ist dann nur noch eine ganz starke Verallgemeinerung vorhanden. Hier wird eine Rolle nur noch unter Berücksichtigung der Slotklasse zugeordnet. So sind nur noch sehr allgemeingültige Zuordnungen, wie Subjekt impliziert Agent, möglich.
Um nun eine Rolle auszuwählen, wenn mehr als zwei Rollen die Schranke minEvidence erreichen, so werden die besten zwei von diesen ausgewählt. Um die Güte dieser beiden möglichen Rollenzuweisungen zu berechnen, wird der Logarithmus des Verhältnisses gebildet. Nun muss ein Schwellenwert log_ratio erreicht werden. Wird auch dieser von beiden möglichen Zuweisungen erreicht so wird die Rolle mit der höheren Wahrscheinlichkeit zugewiesen. Dieser Schwellenwert wird hoch initialisiert und wird mit der Zeit immer weiter herabgesetzt.
Diese Methode wurde nun auf zufällig ausgewählte 20% des *British National Corpus* angewandt. Die Ergebnisse hiervon sind in Tabelle 1.12 zu sehen.

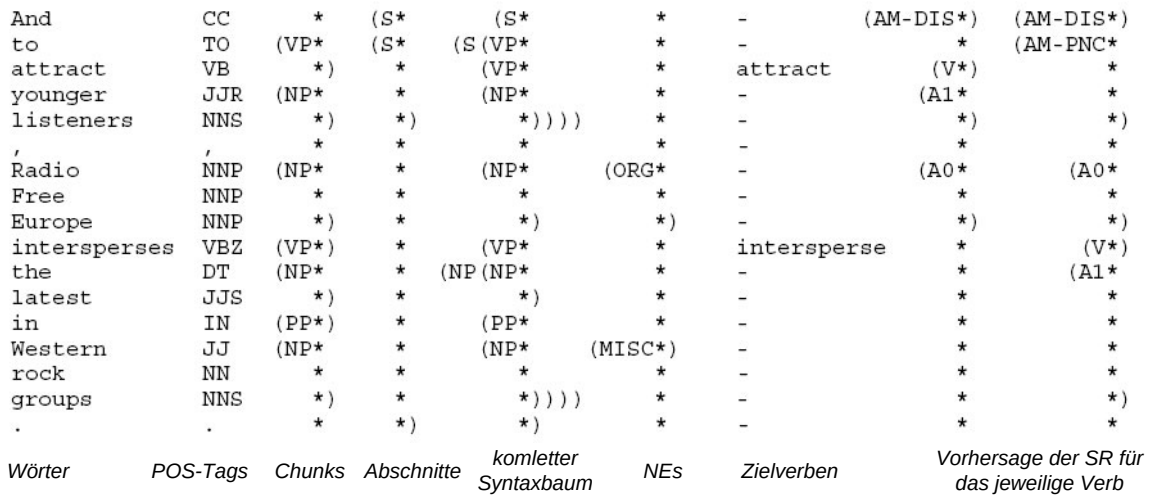
Rollenzuordnung		identifizierte Argumente			alle Zielslots		
		Baseline	Algorithmus		Baseline	Algorithmus	
			eindeutig	final		eindeutig	final
zugeordnet	korrekt	77.3	76.4	90.1	63.7	75.9	87.2
	falsch	22.7	2.7	7.0	36.3	6.8	10.4
nicht zugeordnet	möglich	0	17.1	0	0	14.1	0
	unmöglich	0	3.8	2.9	0	3.1	2.4

Tabelle 1.12: Ergebnisse des Unsupervised Semantic Role Labelings

Unter *Rollenzuordnung*, *nicht zugeordnet* bedeutet *unmöglich*, dass keine Kandidatenlisten vorhanden ist. In diesem Fall ist es natürlich unmöglich eine Rolle zuzuweisen. Der Unterschied zwischen *identifizierte Argumente* und *alle Zielslots* ist, dass bei *identifizierte Argumente* Fehler die in Vorverabeitungsschritten gemacht wurden nicht miteinbezogen werden, bei *alle Zielslots* hingegen werden diese miteinbezogen. Dies ist sinnvoll, da so zunächst die Güte des Verfahrens alleine unter *identifizierte Argumente* betrachtet werden kann. Allerdings sind die Werte unter *alle Zielslots* die letztlich relevanten, da das Benutzen der Vorverarbeitungsschritte nun einmal unumgänglich ist. Die *Baseline* wird durch das dritte BackOff-Level, welches in der Formel 1.74 zu sehen ist, beschrieben.
Was sehr auffällig ist, ist dass viele Zuordnungen bereits während des Frame-Matching Schritts gemacht werden. Dies ist in der Spalte *eindeutig* festgehalten. So wird die Baseline letztlich sogar bereits durch den Frame-Matching Schritt überschritten. Deshalb kann davon ausgegangen werden, dass der Frame-Matching Schritt auch vielen supervised Methoden einen großen Gewinn bringen könnte. [SS04, SS05]

1.6.3 CoNLL2005 - Shared Task

An der *CoNLL2005 - Shared Task* (Computational Natural Language Learning 2005 - Shared Task) haben 19 Teams teilgenommen. Diese hatten 3 Monate Bearbeitungszeit zur Verfügung.



Wörter	POS-Tags	Chunks	Abschnitte	komletter Syntaxbaum	NEs	Zielverben	Vorhersage der SR für das jeweilige Verb
And	CC	*	(S*	(S*	*	-	(AM-DIS*) (AM-DIS*)
to	TO	(VP*	(S*	(S(VP*	*	-	* (AM-PNC*
attract	VB	*)	*	(VP*	*	attract	(V*) *
younger	JJR	(NP*	*	(NP*	*	-	(A1* *
listeners	NNS	*)	*)	*))))	*	-	*) *)
,	,	*	*	*	*	-	* *
Radio	NNP	(NP*	*	(NP*	(ORG*	-	(A0* (A0*
Free	NNP	*	*	*	*	-	* *
Europe	NNP	*)	*	*)	*)	-	*) *)
intersperses	VBZ	(VP*)	*	(VP*	*	intersperse	* (V*)
the	DT	(NP*	*	(NP(NP*	*	-	* (A1*
latest	JJS	*)	*	*)	*	-	* *
in	IN	(PP*)	*	(PP*	*	-	* *
Western	JJ	(NP*	*	(NP*	(MISC*)	-	* *
rock	NN	*	*	*	*	-	* *
groups	NNS	*)	*	*))))	*	-	* *)
.	.	*	*)	*)	*	-	* *

Abbildung 1.46: Das Eingabeformat für die CoNLL-2005 an einem Beispielsatz

Auch im Jahr 2004 war das Thema der CoNLL Semantic Role Labeling, wodurch ein direkter Vergleich möglich war. Als Merkmale für den Lernschritt waren die Wörter, POS-Tags, Chunks (Basiszerlegungen), Abschnitte im Start-End-Format, Named Entities und die Zielverben gegeben. Und in der 2005er Ausgabe der Shared Task war zusätzlich noch der komplette Syntaxbaum gegeben. Des Weiteren ist noch die Argumentzuordnung zu den semantischen Rollen gegeben, diese ist aber natürlich für die Lernmenge nicht verfügbar gewesen. Ein Beispielsatz in diesem Eingabeformat ist in Abbildung 1.46 zu sehen.
Was außerdem neu in der 2005er Shared Task war, ist dass die Trainingsdaten vergrößert wurden und eine neue Testmenge in Form des Brown-Korpus hinzugefügt wurde. Die Trainingsdaten stammten aber weiterhin aus dem PropBank Korpus, sowie dem WSJ, welcher ein Teil des PennTreeBank Korpus ist.
In Tabelle 1.13 sind einige ausgewählte Systeme mit ausgewählten Merkmalen zu sehen. Die Tabelle gibt an wie die einzelnen Systeme vorgegangen sind.
Unter der Zeile F_1-*Rang* ist die Güte der jeweiligen Systeme im Bezug auf die anderen Systeme zu sehen. Wir haben hier die Positionen 1, 2, 4 und 17 ausgewählt, um anhand dieser, einige Merkmale, die genutzt wurden, zu erklären. In der zweiten Zeile (*Team*) steht der Kürzel des Teams, welches das System entwickelt hat. So kann später in den Ergebnissen einfach nach dem Kürzel gesucht werden.

F_1-Rang	Team	ML-Method	Kombination	NE	Argument			Verb	
					Typ	PP	Dic	Ver	Sub
1	pun	SNoW	ja	+	+	+	-	+	+
2	hag	ME	ja	-	+	+	+	+	+
4	pra	SVM	ja	+	+	+	-	+	+
17	pon	DT	nein	+	+	-	-	+	-

Tabelle 1.13: Auszug der Merkmalstypen ausgewählter Systeme in der CoNLL-2005

In der Spalte *ML-Method* ist vermerkt, welche Lernmethode zum Einsatz gekommen ist. An dieser Stelle könnten auch mehrere Methoden stehen, bei unseren ausgewählten Systeme ist es aber immer nur eine. Mit Absicht wurden hier vier Systeme gewählt, die mit unterschiedlichen Lernmethoden arbeiten. So sind in unserem Ausschnitt Dependency Tree (DT), SVM (Support Vector Maschine), Maximum Entropy (ME), sowie Sparse Network of Winnows (SNoW) als Lernmethoden genutzt worden.
Unter *Kombination* ist vermerkt, ob eine Kombination von verschiedenen Ausgaben genutzt wurde. Verschiedene Ausgaben konnten unter anderem durch verschiedene Lernalogrithmen, verschiedene syntaktische Eingabestrukturen (es gab verschiedene Parser) oder durch nutzen der n-besten Parsing-Kandidaten entstehen. Unter *NE* ist vermerkt, ob Named Entities genutzt wurden.
Dann folgen die Merkmale des Arguments. Dort ist unter *Typ* vermerkt, ob der syntaktische Typ des Arguments genutzt wurde. *PP* (präpsitionale Phrase) gibt an, ob sezifische Merkmale der PP genutzt wurden. Unter *Dic* wird angezeigt, ob semantische Wörterbücher verwendet wurden. Ein solches Wörterbuch sammelt Wörter, die in der Trainingsmenge beispielsweise häufig im Temporal oder Lokativ standen.
Dann gibt es noch die Merkmale, die das Verb betreffen. *Ver* zeigt hier an, ob die standard Verbmerkmale, wie Voice (aktiv/passiv Unterscheidung), POS-Tag oder Form, benutzt wurden. Unter *Sub*, also Subkategorisierung, ist vermerkt, ob die syntaktische Regel, die den Elter des Prädikats erweitert, als Merkmal genutzt wurde. Ein Beispiel hierfür wäre die Regel VP $\rightarrow$ V NP.
Grundsätzlich kann zu den Systemen gesagt werden, dass die Systeme einen besseren F_1-Rang hatten, die mehr Merkmale verwendet haben. Außerdem hat sich herausgestellt, dass Kombination sehr wichtig ist. So haben 8 von 19 Teams eine Kombination verwendet und von diesen sind bis auf einen alle unter den besten zehn Teams.
In Abbildung 1.47 ist eine vollständige Abbildung aller Systeme und ihrer F_1-Werte auf den unterschiedlichen Korpus zu sehen. Die Baseline ist durch ein einfaches System aus der CoNLL2004 gegeben.
Es wurde eine maximale F_1-Score von circa 80 erreicht. Dies ist ein Anstieg um 10 Punkte im Vergleich zu den Ergebnissen des Vorjahres. Diese Verbesserung entstand durch die 5-fache Vergrößerung der Trainingsmenge, die zusätzliche Eingabe der kompletten Syntaxbäume und die verfeinerten Kombinationsverfahren. Allerdings sind diese Ergebnisse immer noch zu weit weg von Wunschergebnissen. Am Brown-Korpus ist zu sehen, wie gut ein SRL-Modul in einer realen Anwendung abschneiden würde und dies ist etwa 10 F_1-Punkte schlechter und entspricht so nur einer F_1-Score von 70. [CM04, CM05a, CM05b]

1.6.4 Fazit

In diesem Kaptiel wurden verschiedenste Verfahren zum Semantic Role Labeling vorgstellt. Damit ging auch eine Vielzahl von Features einher. Deutlich wurde aber, dass das Path-Feature und das Headword-Feature die dominantesten davon sind. In den Vergleichswettberweben sind diese Verfahren gegeneinander angetreten und es hat sich gezeigt, dass kombnierte Verfahren, also solche, die mehr als ein „Grundverfahren“ nutzen, die erfolgreichsten sind. Die Wettbewerbe haben aber auch hervorgehoben, dass mit einem F_1 von ca. 70% noch viel Forschungs- und Optimierungspotenzial bestehen. Keines der Verfahren verwendet domänenspezifische Features. Für diese Projektgruppe ergibt sich daraus die Perspektive, den F_1-Wert durch die Anpassung der Verfahren auf die Bundestags-Domäne zu verbessern.

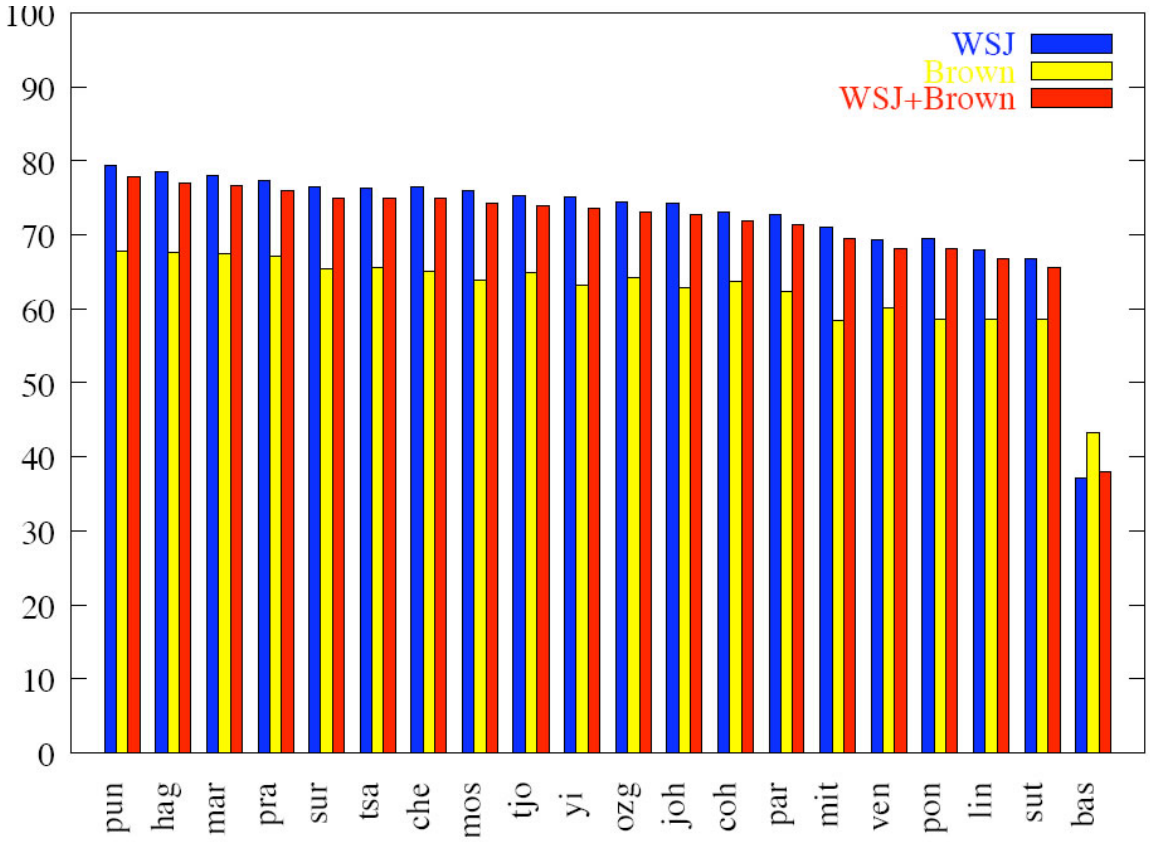


Abbildung 1.47: Vergleich der Systeme der CoNLL2005

我们在做项目520(Wir machen jetzt PG520)

aber nicht so 我们 在 做 项目520

Abbildung 1.48: Chinesiche Wörter

Die Techniken des SRL kann die Projektgruppe vor allem in den Bereichen der Fragebeantwortung und des Information Extraction nutzen. Eine grob eingegrenzte Antwort kann so weiter zergliedert werden. Das eröffnet unter Umständen die Möglichkeit, dem Fragenden eine präzise Antwort zu präsentieren, die nur noch relevante Informationen enthält und kein nachträgliche Suche des Fragenden im Antwort-Textschnipsel erfordert.

1.7 Treebank

1.7.1 Einführung

Sprachverstehen wird oft als ein KI-vollständiges Problem gekennzeichnet, weil für natürliche Spracherkennung umfangreiches Wissen über der Außenwelt zu verlangen ist, ausserdem soll es auch in der Lage sein, Wissen zu manipulieren und verwenden. Die Definition von "Verstehenïst inzwischen ein Hauptproblem in NLP. Aber A paar Probleme können in dem Sprachverstehen(NLP[4]) immer betroffen werden:

- **Wörtersegmentierung:** Einige geschriebene Sprachen wie z.B. Chinesisch, Japanisch und Thailändisch haben keine Einzelnen Wortgrenzen, deshalb verlangte jeder bedeutungsvoller Text die Identifikation der Wortgrenzen, bevor der analysiert wird. diese ist oft eine nicht-triviale Aufgabe.
  Beispiel siehe Abbildung 1.48

- **Disambiguierung:** Viele Wörter haben mehr als eine Bedeutung, so dass wir die Bedeutung, die den meisten Sinn in Kontext macht, auswählen müssen.

  Bsp.: The old *can can* hold the water.
  erste „can“: Behälter
  zweite „can“ modales Verb

- **Syntaktische Zweideutigkeit:** Die Grammatik für natürliche Sprachen ist zweideutig, es können sich oft viele verschiedenen Sytnaxbaeume aus einem Satz ergeben, analysieren Sie Bäume für einen gegebenen Satz. Um dem Geeignetesten auszuwählen, verlangt man normalerweise semantische und kontextuelle Informationen. Zur Bestimmung der Problembestandteilen von Syntax auf einem Satz müssen wir uns mit ganzen Kontext herumgucken, danach könnte man sich den Informationen entsprechendenden Analysenbaum herausfinden.
  Beispiel siehe Abbildung 1.49

[4]Natural language processing

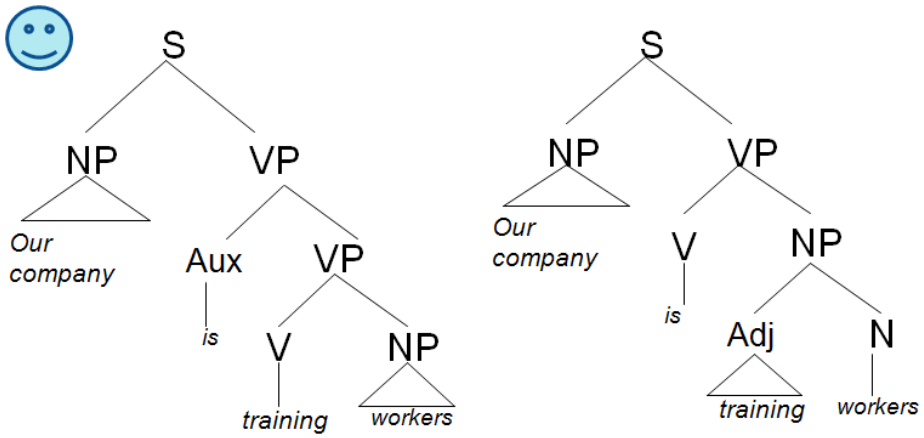


Abbildung 1.49: Syntaxmehrdeutigkeit

Das NLP befassen sich natürlich nicht nur mit den oben sogenannten Problemen, aber in diesem Abschnitt wollten wir soeben davon ausgehen. Um solche problematik zu verbessern, kann man hier eine Mehtode „Baumbank“ verwenden.

1.7.2 Verallgemeinerte Treebank

Baumbanks können in Korpus-Linguistiken für das Lernen syntaktischer Phänomene oder in computational linguistic für die Training- oder Test-Parser benutzt werden.Genauso kann ein Textkorpus von Linguisten ausgewertet werden, um Regelmäßigkeiten in dieser Sprache beschreiben zu können. Mehrsprachige Korpora können zur (teil-)automatischen Übersetzung oder für vergleichende Betrachtungen der Sprachen genutzt werden.

Definition

Eine Baumbank ist ein Korpus, in dem jeder Satz mit syntaktischen Strukturen kommentiert worden ist. Diese syntaktische Struktur wird im Allgemeinen als Baumstruktur dargestellt.[AT00]

Vom Anfangen an ist die Wortarten-Annotation zu erledigen. Sie sollte möglichst mit einem geringen manuellen Korrekturaufwand verbunden und stellt heute eine Mindestanforderung an ein Textkorpus dar.

Wortarten

Unter Wortart(englisch. POS[5]) versteht man die Klasse von Wörtern einer Sprache auf Grund der Zuordnung nach gemeinsamen grammatischen Merkmalen. Die Wortartlehre versucht eine Klassifizierung der lexikalisch-grammatischen Einheiten einer Sprache. Auf der Liste in der Abbildung 1.50 stellen teilweise Wortarten aus der Penn Baumbank dar. Die Erfassung und Kennzeichnung der Wortarten wurde ursprünglich manuell durchgeführt, im Laufe der Zeit wurde das Verfahren zunehmend durch die Computerlinguistik automatisiert. Durch die

[5]part of speech

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	+,%, &
CD	Cardinal number	*one, two, three*	TO	"to"	*to*
DT	Determiner	*a, the*	UH	Interjection	*ah, oops*
EX	Existential 'there'	*there*	VB	Verb, base form	*eat*
FW	Foreign word	*mea culpa*	VBD	Verb, past tense	*ate*
IN	Preposition/sub-conj	*of, in, by*	VBG	Verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	Verb, past participle	*eaten*
JJR	Adj., comparative	*bigger*	VBP	Verb, non-3sg pres	*eat*
JJS	Adj., superlative	*wildest*	VBZ	Verb, 3sg pres	*eats*
LS	List item marker	*1, 2, One*	WDT	Wh-determiner	*which, that*
MD	Modal	*can, should*	WP	Wh-pronoun	*what, who*
NN	Noun, sing. or mass	*llama*	WP$	Possessive wh-	*whose*
NNS	Noun, plural	*llamas*	WRB	Wh-adverb	*how, where*
NNP	Proper noun, singular	*IBM*	$	Dollar sign	$
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	#
PDT	Predeterminer	*all, both*	“	Left quote	(‘ or “)
POS	Possessive ending	*'s*	”	Right quote	(’ or ”)
PP	Personal pronoun	*I, you, he*	(	Left parenthesis	([, (, {, <)
PP$	Possessive pronoun	*your, one's*	)	Right parenthesis	(],), }, >)
RB	Adverb	*quickly, never*	,	Comma	,
RBR	Adverb, comparative	*faster*	.	Sentence-final punc	(. ! ?)
RBS	Adverb, superlative	*fastest*	:	Mid-sentence punc	(: ; ... – -)
RP	Particle	*up, off*			

Abbildung 1.50: Liste des POS-Taggers

state-of-art Techniken kann über 96% Genauigkeit für die meisten erfolgreichen Ansätze realisiert werden.

Bsp.:	The	representative	put	chairs	on	the	table
	DT	NN	VBD	NNS	IN	DT	NN

Automatische Wortartenannotation

Das automatische Annotation des Wortarts ist ein Bereich von der natürlichen Sprache Verarbeitung. Im Wesentlichen sind zwei unterschiedliche Methoden(Abb. 1.51), die sowohl statistisch als auch regelbasiert sind, können entwickelt werden. Aber statistische Techniken scheinen sich erfolgreicher als regelbasierte Methoden zu sein. Die Tatsache, dass sehr wenig handgefertigte Wissen im System aufgebaut werden muss. Demgegenüber sind die Richtlinien in den regelbasierten Systemen normalerweise schwierig zu konstruieren und sind nomalerweise nicht sehr robust. Von der ausführlichen Störung Analyse kann es geschehen, dass das regelbasierte Tagging mehr Probleme mit unbekannten Wörtern als das statistische Tagging hat. Durch die Forschung von der Universität Zurich [Mit98] sind uns bekanntgegeben, wie durch die Beiden automatische Taggingerstellung deutscher Sprache auszuwirken ist. Wenn die unbekannten Wörter in die Taggers mithilfe eines externen Lexikons eingezogen werden, fällt die Fehlerrate des regelbasierten Tagging auf 4.7% ab und wobei die Fehlerrate der statistischen Taggingsauf 3.7% senkt.

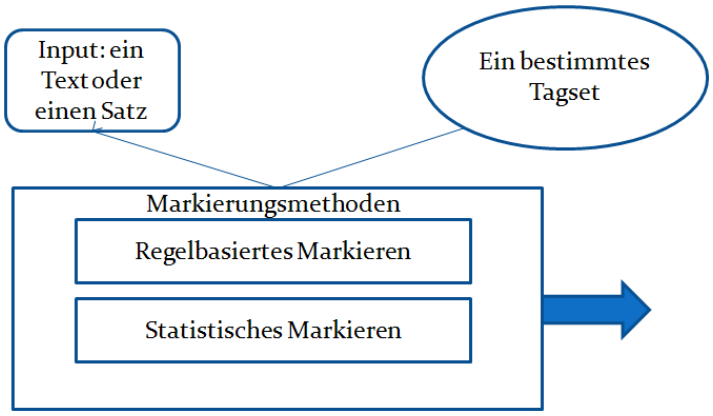


Abbildung 1.51: Prozess des POS-Taggings

Regelbasierte Annotation

Bei regelbasierte Annotation[Bri92] benutzt man eine große Datenbank von handgefertigten Disambiguierungsregeln oder Regeln aus Baumbank, Um diese Methode einfach zu verstehen, kann sich der ganze Annotationsprozess auf zwei Schritte gliedern:

Erster Schritt: Zuerst benutzt man ein Wörterbuch, um jedem Wort des Textes eine Liste potenzieller POSs zuzuweisen.

Zweiter Schritt: Diese Listen lassen sich durch die handgefertigte Disambiguierungsregeln weiter verfeinern, danach kann man mit Hilfe der allen einzelnen POS-Taggings jedes Wort trennen.

hierunter stellt ein "that"Beispiel mit einem einfachen Verfahren dar. Aber hier das einfache Wort scheint sich nicht mehr einfach zu sein, da das Wort that unterschiedliche Syntaxfunktionen gib's.

Adverbial-That Regel

`Input: that`

if (+1 → A/ADV/Pron/Noun)//falls adj, adv ,...direkt folgen
then eliminate ADV;

else if (+n → Clause)//falls als Anfang von Klausel
then eliminate non-ADV

Ergebnis:

ADV: he isn't that man!

Non-ADV: I am glad that you are satisfied with your job.

Statistische Annotation
Viele Statistische Modelle können im Trainingskorpus beitragen. Die Idee ist einfach und klar, man kann z.B. mit HMMs, MEMMs, CRFs die Wahrscheinlichkeit bei der Annotation von einem gegebenen Wort berechnen. Danach können die beste Zustandsequenz generiert werden, um später als Muster alle ungetaggte Wörter zu überprüfen und klassifizieren. Außerdem kann man auch mit Hilfe von SVMs den Korpus automatisch annotieren. Aber schwer ist, wie man die effiziente Kernfunktion auskriegen kann. In diesem Bericht erwähnen wir das Verfahren mit Bigram-HMM.

$$t_i = \arg\max_j P(t_j|t_{i-1}, w_i) \tag{1.75}$$

t_i ist die beste Wahrscheinlichkeit für den Tag eines Wort w_i, wobei i die aktuelle Stelle im Satz und j das Index über alle mögliche Tags sind. Durch die Markov-Annahme können die oben genannte Formel umgeformt werden.

$$t_i = \arg\max_j P(t_j|t_{i-1})P(w_i|t_j) \tag{1.76}$$

Annotationsvorgang
wie im Abschnitt 1.2 beschrieben sind, dass Viterbi-Algorithmuss hier zu verwenden ist.

1) Initialisierung:

$$\delta_1(k) = \pi_k P(w_1|t_k), 1{\leq}k{\leq}J \tag{1.77}$$
$$\Psi_i(j) = 0 \tag{1.78}$$

2) Induktion:

$$\delta_i(j) = [\max_i \delta_{i-1}(k)P(t_j|t_k)]P(w_i|t_k), 2{\leq}i{\leq}n, 1{\leq}k{\leq}J \tag{1.79}$$
$$\Psi_i(j) = \arg\max_{1\leq j\leq J}[\delta_{i-1}(k)P(t_j|t_k)] \tag{1.80}$$

$\max_i \delta_{i-1}(k)P(t_j|t_k)]P(w_i|t_k)$ stellt den Forward-Algorithmus dar. In der Arrayliste $\Psi_i(j)$ werden dann die beste Wahrscheinlichkeit vom Tag k zum Tag j ausgesucht.

3) Terminierung:

$$X_n = \arg\max_{1\leq i\leq N}[\delta_n(j)] \tag{1.81}$$

$\delta_n(j)$ sind die Wahrscheinlichkeiten aller möglichen Tagssequenzen vom ersten bis zum letzten Wort.

4) Optimale Zurückverfolgerung der Zustandsfolge:

for i := n-1 to 1 Step -1 do

$$X_i = \Psi_{t+1}(X_{t+1}) \tag{1.82}$$

t = T-1, T-2, ..., 1

Bsp.:	The	sportler	is	expected	to	win	tomorrow
	DT	NNP	VBZ	VBN	TO	VB	NN

Gegeben sind: $P(VB|TO) = 0.34 \quad P(win|VB) = 0.00003$
$P(NN|TO) = 0.021 \quad P(win|NN) = 0.00041$

$\surd$ $P(VB|TO)P(win|VB) = 0.34 * 0.00003 = 0.0000102$

$\times$ $P(NN|TO)P(win|NN) = 0.021 * 0.00041 = 0.00000861$

Die verschiedenen statistischen Methoden können nicht nur zur Annotation der Wortarten dienen, ferner können sie bei der Analyse von PCG[6] weiterhelfen, da am Anfangen wir schon erwähnt haben, dass nicht nur das problem mit Disambiguierung von Wort, sondern auch mit Syntaxmehrdeutigkeit vorkommt. Das ganze Verfahren bei automatischem Syntaxaufbau ist ähnlich wie POS-Tagging.

Syntaktische Struktur

Bisher haben wir schon etwas über das POS-Tagging kennengelernt. Aber wie sieht aus Baumbanks deren „Bäume“ aus? schauen wir die Struktur von einem Beispielsatz an. Für Syntaktische Analyse ist durch kontextfreie Grammatiken leicht zu formalisieren.

S → NP VP

NP → Pronoun — Noun — Det Adj Noun —NP PP

PP → Prep NP

V → Verb — Aux Verb

VP → V — V NP — V NP NP — V NP PP — VP PP— VP S —V S

Mit Hilfe von der oben genannte Grammatik kann ein Syntaxbaum eines Beispielsatzs(Abb. 1.52) erstellt werden. Durch viele solche indizierte Satzbäume in einem Korpus entsteht die endliche Produkt ”Baumbank”.

[6] Probabilistic Context-Free grammar

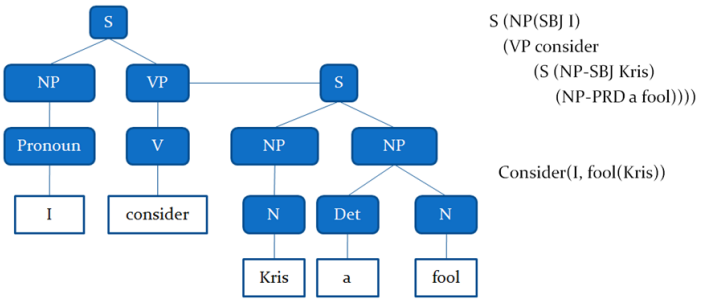


Abbildung 1.52: Syntaxbaum und Prädikate-Argument-Struktur

1.7.3 PropBank

Geschichte der Penn Baumbank

Die bekannteste Baumbank ist die Penn Baumbank [AT00]. Die Entwicklung dieser Baumbank gliedert sich in drei Phasen. In der ersten Phase (1989-1992) annotierte ein symbolischer Parser die Sätze vor, die anschließend manuell korrigiert wurden. Bedingt durch den hohen manuellen Aufwand musste in dieser Phase ein vereinfachtes Phrasenstruktur-Modell verwendet werden, das auch als Parsing-Skelett bezeichnet worden ist. Da eine so detaillierte Annotation nur teilweise durch statistische Methoden unterstützt werden kann, ist der Anteil manueller Korrekturen sehr hoch. Zum Ende der zweiten Phase der Penn Treebank (1993-1995) sind eine Million Wortformen der ersten Phase auf diese Weise aufbereitet worden. In der dritten Phase (1996-1999) ist der Umfang der detaillierten Annotation nochmals um eine Million Token erweitert worden.Diese Baumbank umfasste jedoch beachtenswerte 3 Millionen Token und stellte damit eine hervorragende Ausgangsposition für computerlinguistische Anwendungen dar.

Prädikate-Argument-Struktur

Obwohl man mit einer Grammatik von dem Quellsatz die komplette Worte analysieren kann, die Informationextraktion war häufig noch ungenaurig. Es fehlt noch im NLP etwas, das ist genau die predicate-argument Struktur. Diese Notwendigkeit zeigt offenbar durch eine neue Auswertung der Penn Baumbank [Kin02]. Die Qualität des semantischen Tagging konnte durch die wichtigste Faktor genauso die Prädikate-Argument-Struktur beeinflußt worden. Man kann die auch als Klammerstruktur bezeichnen, deren Schreibweise sieht ähnlich wie Beschreibungslogik aus. Die Eingabe ist Syntaxbaum und Die Ausgabe ist, wie in Abbildung 1.52 ein einfaches Beispiel darstellt. Das Verb consider spielt im Beispiel die wichtigste Rolle, da Ohne dies Wort der ganze Satz nicht mehr gekoppelt wird, d.h. macht der ganze Satz keinen Sinn. Außerdem Obwohl das Wort fool hier Substantiv ist, kann man mit Hilfe von NomBank, die im nächsten Abschnitt vorgestellt wird, leicht verstehen, warum es die Prädikate vom Subsatz ist.

Penn Baumbank++

Die Erfassung von linguistischen Regelmäßigkeiten ist immer schwer, da der semantischen Annotation, die in einer normalen Treebank nicht zur Verfügung stehen. aber jetzt sieht die

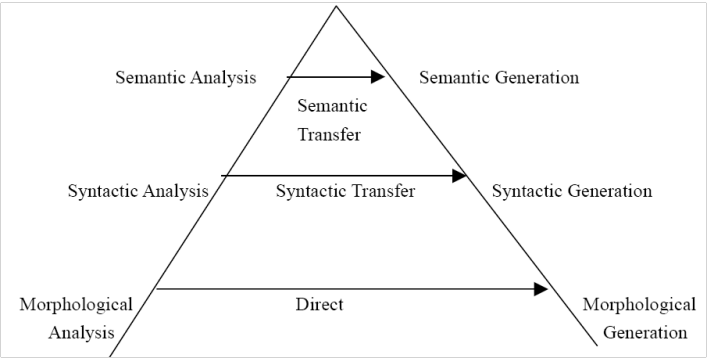


Abbildung 1.53: Pyramide des NLPs

Situation andere aus, weil mit Hilfe von der Prädikate-Argument-Struktur die Penn Baumbank endlich vor dem *Semantic-Role-Labeling* stehen lässt, d.h. sie steigert sich von der syntaktischen Annotation auf der semantischen Annotation (Abb.1.53). Man kann auch die PropBank *Penn Baumbank++* nennen. Hier gib's die Linke zu diesem großartigen Projekt. Die Anwendungen von PropBank im NLP können zur Maschinelle Übersetzung und Information Extraktion dienen.

Frames Files

Argumentmerkmale Argumentmerkmale waren sinnvoll in PropBank zum Einsatz. Die konnten auf die Eigenschaften eines Satzes leicht abgebildet werden. Sie wurden in den meisten modernen Theorien der Argumentstruktur benutzt. Außerdem können Sie eigene Argumentmerkmale, die nicht besonders verpflichtet auf einer bestimmten Theorie basiert, erzeugen.

Primäre Labels: Argumentenelemente sind $ARG[0 \rightarrow 5]$, diese bieten höhere Korrelation zwischen Argumenten und Prädikaten als Adjuncts an.

Sekundäre Labels: (Annotation von Modifiern) für Adjuncts: TMP, LOC, PRP, MNR, ...

Frames File

Jeder Eintrag im Annotationslexikon ist ein Frames File, dieses hat:

- Eine Beschreibung des Verbsinnes ist im Frameset gekennzeichnet. Ein Frameset ist eine Menge von syntaktischen Subkategorie Frames, die einige erwartete Argumente oder semantische Rollen enthalten.
- Eine Beschreibung der Menge von semantischen Rollen, mit der ein Frameset assoziiert wird(Abb. 1.55).
- Eine Beschreibung der Subkategorie Frames (Predikate + Realisierte Argumente) für jedes Frameset. Bsp.: In Tabelle 1.14
- Ein Mapping(Abb. 1.54) zwischen den syntaktischen Funktionen und den Argumentlabeln für jedes Subkategorie Frame.

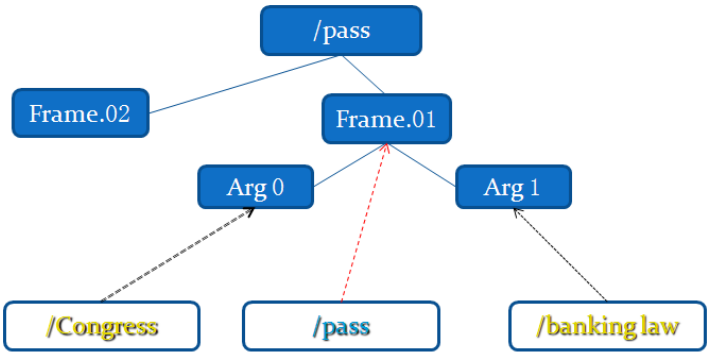


Abbildung 1.54: Das Mapping für eine Verbinstanz zu einem Frameset

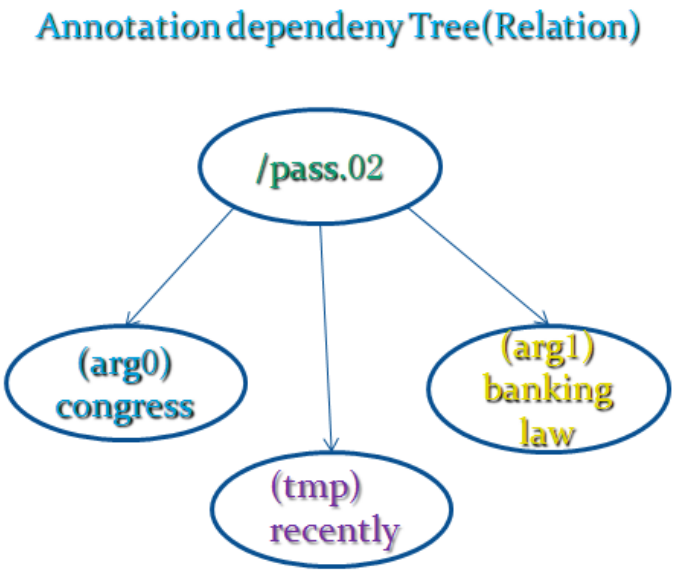


Abbildung 1.55: Dependency Tree

Verb: pass	Verb: pass
Frameset. 01= "move/go" Semantic roles: Arg0: passer Arg1: passing through Frame 1: "The train is passing through the tunnel." PropBank Annotation: Rel: passing Arg0: train Arg1: tunnel Frame 2 ...	Frameset.02 = "bill becomes law" Semantic roles: Arg0: lawmaker Arg1: bill, regulation, law, etc. Frame 1: "The Congress passed the banking law recently." PropBank Annotation: Rel: passed Arg0: passer Arg1: passing through ArgM-TMP: recently Frame 2 ...

Tabelle 1.14: Ein Beispiel vom Frameset

Überprüfung bei Interannotationen

Die Annotatoren selbe sind aus einer Vielzahl der Hintergründe,die von den Student bis zu PhDs[7], Linguisten, Informatiker und andere sind, gekommen worden. Deswegen bevor die Annotation startet, sollt bei allen Annotatoren über die Annotationregeln der PropBank ein gutes Training erforderlich gewesen sein. Die Dauer des Trainings sind jedoch unterschiedlich, bei einigen ist sehr schnell und erfolgreich, aber auch einige andere Leute haben immer Problem. Nach der Annotation der PropBank wird ein Vergleichtest zusätzlich stattfinden, um die Übereinstimmung des Interannotators zu überprüfen. Hierzu ist Kappastatistik verwendbar. Beim Test werden Alle PropBank-Korpa von zwei Tagger annotiert.

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)} \tag{1.83}$$

„role identification(role vs. non-role)“ bedeutet, dass bei der Annotation nur für jedes Wort oder jede Wortgruppe aus dem Korpus entscheidbar ist, ob die Rolle sind, oder keine. „role classification (Arg0 vs. Arg1 vs. ...)“ sind verfeinertere Entscheidungen als die oben sogenannte Rolle-Identifikation. Die P(A) ist die erwartete Übereinstimmungswahrscheinlichkeit der Interannotation und P(E) ist die zufällig erwartete Übereinstimmung. Kappastatistik für diese verschiedenen Argumententscheidungen wird gezeigt in Abbildung 1.56. Die Übereinstimmung über Rollenannotation ist unter beiden Behandlungen von ArgMs sehr hoch auf (0.99. Beruhigend ist das Kappaergebnis für die schwierigere Rollenklassifikationaufgabe auch hoch auf 0.93, schließlich sind alle Arten ArgM und nummerierte Argumente zu betrachten. Kappa berechnet auf der kombinierten Annotation und der Klassifikationentscheidung über allen Knoten in Baumbank war 0.91, einschließlich von ArgM und 0.93 über nummerierte Argumenten.

[7]Doctor of Philosophy

Interannotator agreement.

		$P(A)$	$P(E)$	κ
including ArgM	role identification	.99	.89	.93
	role classfication	.95	.27	.93
	combined decision	.99	.89	.91
excluding ArgM	role identification	.99	.91	.94
	role classfication	.98	.41	.96
	combined decision	.99	.91	.93

Abbildung 1.56: Übereinstimmung des Interannotators

Vergleich zum FrameNet

Das PropBank Projekt und das FrameNet, welches vom International Computer Science Institute entwirkelt wurde, teilen die Dokumentation der syntaktischen Realisierung der Argumente von den Prädikaten des allgemeinen englischen Lexikons mit einander. Die beiden annotieren einen Korpus mit semantischen Rollen. Trotz der zwei Ähnlichkeiten von beiden sind ihre angewendete Methoden unterschiedlich. FrameNet wird auf semantische frames gerichtet, die als vereinfachte Darstellung der Situationen, der verschiedene Teilnehmern und anderer Begriffsrollen definiert werden. Obwohl die beiden Performancen beim SRL[8] fast gleich sind, bietet PropBank die Einfachkeit der Annotation und mehr Lernfähigkeit an.

In Propbank sind Argumentlabel verbspezifisch. Das $Arg0$ ist aus einem Frameset vom buy, hingegen in FrameNet werden die die Verben instanziert. Die Frame Elemente sind verbklassen-spezifisch. Z.B. Buyer aus dem Frame Commerce über eine Klasse der Verben: buy, lease, purchase, rent, ...

PropBank		FrameNet
Buy	Sell	Commerce
ARG0: buyer	Arg0: seller	Buyer:
Arg1: thing bought	Arg1: thing sold	Seller:
Arg2: seller	Arg2: buyer	Payment:
Arg3: price paid	Arg3: price paid	Goods:
Arg4: benefactive	Arg4: benefactive	Rate/Unit:

Bsp.: FrameNet Annotation(1) und PropBank Annotation(2)

(1) [Buyer: Tim] bought [Goods: a car] [Seller: from Tom] [Payment:for $1000].

(2) [Arg0: Tim] bought [Arg1: a car] [Arg2: from Tom] [Arg3: for $1000]

Automatisierung des Semantic-Role-Labeling

Das Ziel des PropBank ist, die Trainingsdaten für automatische Rollentags zur Verfügung zu stehen. Eine von PropBank wichtigen Eigenschaften als praktisches Hilfsmittel ist, dass die Sätze, die für Annotation aus dem Wall Street Journal-Korpus gewählt sind, werden für das ursprüngliche Penn Baumbank Projekt benutzt. Und folglich sind manuell-geprüfte syntaktische Analyse-Bäume für den gesamten Datensatz vorhanden .

[8] semantic role labeling

[GJ02b] beschreibt ein statistisches System, das auf den Daten vom FrameNet Projekt verwendet wurde, um semantische Rollen automatisch zuzuweisen. Das System hat zuerst Sätze durch einen automatischen Grammatikparser [Col99] bestanden. Es extrahierte syntaktische Eigenschaften und schätzte die Wahrscheinlichkeiten für semantische Rollen von den syntaktischen und lexikalischen Eigenschaften. analysiert Training und Testsätze automatisch, wie keine per Hand-Annotation Bäume waren für das Korpus analysieren vorhanden. Im Abschnitt 1.5 sind schon beschrieben, wie durch die Merkmale von den semantischen Rollen mit dem Back-Off Modell die Wahrscheinlichkeiten eines Satzbestandteils als die erste Teilaufgabe abzugrenzen sind (englisch. chunk). Zum zweiten werden die Relation zwischen allen abgrenzten Wörterstücken durch die CCG[9] analysiert [GH03].

Ereignissevariablen

Man sucht im Bereich des NLPs ein reiches Werkzeug für das Analysieren von Verbenbedeutung. Jetzt per eine Ereignisvariable lässt eine direkte Darstellung der logischen Form von den adverbialen Modifikatoren und der Darstellung der Substantivwörtern, die auf Ereignisse beziehen, zu.
Satz.: Mr. Bush met him privately, in the White House, on Thursday.
a. PropBank Annotation

Rel: met	Arg0: Mr. Bush
ArgM-MNR: privately	ArgM-LOC: in the White House
ArgM-TMP: on Thursday	

b. Logische Darstellung mit einer Event Variable
$\exists$ e meeting(e) & Arg0(e, Mr. Bush) & Arg1(e, he) & MNR(e,privately) & LOC(e, in the White House) & TMP(e, on Thursday)

PropBank II

PropBank II sind an der Parallel chinesisch-englische Übersetzung angewendet und außerdem fasst Ereignissevariablen, nominale Hinweise um. Es liegt hier eine einfache Darstellung zur Parallel chinesisch-englische Übersetzung (Abb. 1.57) vor.

1.7.4 NomBank

NomBank ist ein Teilprojekt der Penn Baumbank II, um die zusätzlich logischen und semantischen Rollen der Penn Baumbank hinzufügen. Sie dient zur Annotation der Nominelle Prädikat-Argument-Struktur. Deren Framen sind im Allgemeinen gleich wie PropBank($ARG0$, $ARG1$, ...) und arbeitet mit der PropBank zusammen und führt zur Verbesserung der Analyse des Korpus. Zur Zeit stellt NomBank Argumentstruktur für Fälle von ungefähr 5000 allgemeinen Substantivwörtern zur Verfügung. Hier ist die Linke zur Webseite der NomBank. mit dem folgenden Beispiel kann man leicht verstehen, was die NomBank grundsätzlich tut.

Analyse der Substantivphrasen

[9] combinatory categorial grammar

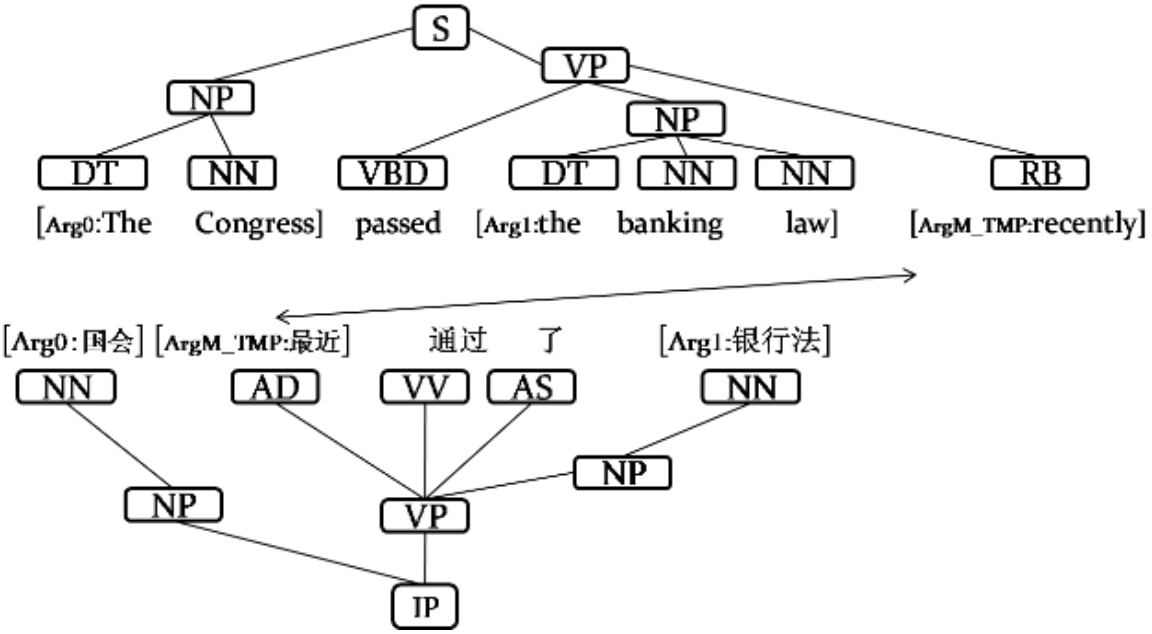


Abbildung 1.57: englisch-chinesische PropBank

Beispiel.: the museum's director
REL = director, ARG0 = director, ARG2 = the museum's

Verschiedene Lexika

Die verschieden Lexika sind die grundlegenden Bausteine, sie unterstützen sich mit einanderen und stellen die Modularität von der Nombank dar. Die folgende Liste sind die wichtigste Resources von Hand-kodierten elektronischen Lexika[MRM+04]:

- NOMLEX ist eine Wörterbuchauflistung, darin gib's 1000 Nominalisierungen als Substantivphrase.
- NOMLEX-PLUS ist eine Erweiterung von NOMLEX, um Substantivargumentenstruktur zu identifizieren.
- COMLEX ist ein syntaktisches Wörterbuch von Substantiven, Verben, Adjektiven und Adverbien
- COMLEX-PLUS ist eine Anreicherung der Substantivkomplementstruktur vom COMLEX
- CATVAR ist ein Wörterbuch über die Erklärung lexikaler Felder durch derivational-Morphologie
- ADJADV ist ein Wörterbuch, das adverbiale Verwendungen der Adjektive definiert.

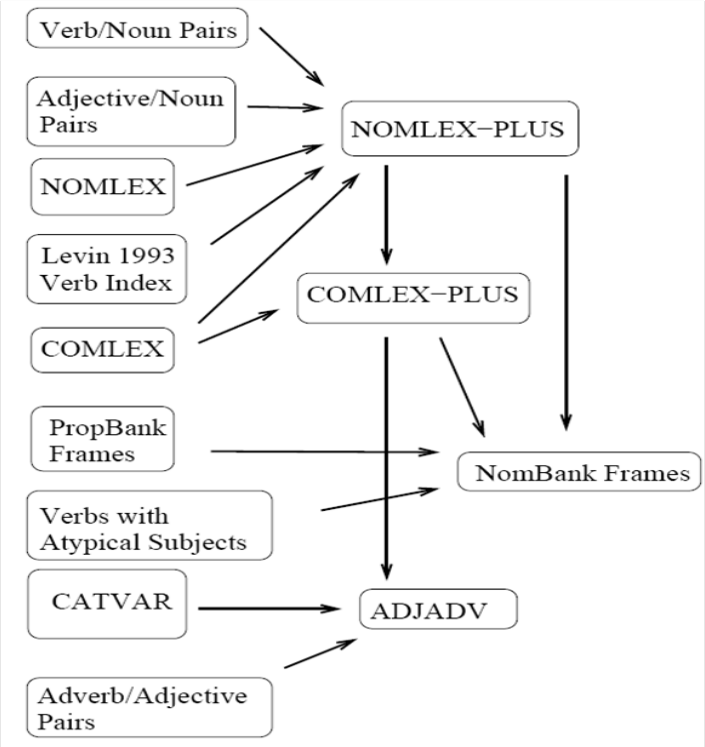


Abbildung 1.58: Die Relation zwischen Lexika

ME-Modell für SRL

Die SRL[10]-Aufgabe der NomBank wird als Klassifikationproblem behandelt und gliedert sie sich in zwei Teilen: Argumentidentifizierung und Argumentklassifikation [JN06]. Opennlp maxent[11]ist eine Implementierung von dem maximalen Entropie Modell , es wird als das Klassifikationwerkzeug verwendet.

$$p(l|x) = \frac{\exp(\sum_{i=1}^{n} \lambda_i f_i(l,x))}{Z_x} \tag{1.84}$$

λ_i ist der Gewichtsparamter für jede EigenschaftFunktion$f_i(l,x)$. $f_i(l,x)$ist eine Eigenschaftsfunktion, um die Abbildung des Merkmals l und der Eingabe x entweder 0 oder 1 zu beschriften. Z_x ist ein Normalisierungfaktor, damit die Summe gleich 1 wird. Im Identifizierungsmodell wird Merkmal l entweder „Argument“ oder „Nicht-Argument“ gekennzeichnet. Die Klassifikationsausgabe ist das Merkmal l mit der höchsten Wahrscheinlichkeit $p(l|x)$.

$$Max(T) = \max\{\begin{matrix} NONE(T) + \sum(Max(child)) \\ ARG(T) + \sum(NONETree(child)) \end{matrix} \tag{1.85}$$

Der Algorithmus, durch den die Argumente nicht zu überlappen sind, ist benötigt beim Test. So kommt der maximale Logwahrscheinlichkeit -Algorithmus vor, um ME-Modell zu verfei-

[10]semantic role labeling
[11]Die Linke zur Resource

nern. $Max(T)$ ist die maximale Log-Wahrscheinlichkeit von einem Baum. „$NONE(T)$“ und „$ARG(T)$“ sind die Log-Wahrscheinlichkeit der „None“ und „ARG“ vom Argumentidentifikationsmodell zu Baumknoten T. Und „$NONETree(child)$“ is the Log-Wahrscheinlichkeit von jeder Knote, die unter Kinderknote ist, werden auch dann alle als „NONE“ gekennzeichnet. In Abbildung 1.59 stellt dieser Algorithmus per Pesudo-Code ausführlich dar.

Algorithm Maximizing the probability of an SRL tree

Input p{syntactic parse tree}
Input m{argument identification model, assigns each constituent in the parse tree log likelihood of being a semantic argument}
Output score{maximum log likelihood of the parse tree p with arguments identified using model m}

MLParse(p, m)
if parse p is a leaf node **then**
  return $max(Score(p, m, ARG), Score(p, m, NONE))$
else
  $MLscore = 0$
  for each node c_i in $Children(p)$ **do**
    $MLscore$ += $MLParse(c_i, m)$
  end for
  $NONEscore = 0$
  for each node c_i in $Children(p)$ **do**
    $NONEscore$ += $NONETree(c_i, m)$
  end for
  return $max(Score(p, m, NONE)$+$MLscore$, $Score(p, m, ARG)$+$NONEscore)$
end if

NONETree(p,m)
$NONEscore = Score(p, m, NONE)$
if parse p is a leaf node **then**
  return $NONEscore$
else
  for each node c_i in $Children(p)$ **do**
    $NONEscore$ += $NONETree(c_i, m)$
  end for
  return $NONEscore$
end if

Subroutine:
$Children(p)$ returns the list of children nodes of p.
$Score(p, m, state)$ returns the log likelihood assigned by model m, for parse p with $state$. $state$ is either ARG or NONE.

Abbildung 1.59: Maximale Log-Wahrscheinlichkeit Algorithmus

Eine gemischte Darstellung

In kombinierten PropBank und NomBank ist eine grafische Darstellung wie in der Abbildung 1.60 vorgekommen. In der jede Rolle einer Bogenline entspricht. Z.B. wir können zuerst die Argumentstruktur des Substantivwort ovation genau wie des Verb applaud annehmen. Danach entsprechend unserer Analyse sind (they) die Geber von (the ovation) und the chefs sind dagegen die Empfänger. Außerdem hat (gave) und (ovation) zwei eindeutige Richtungsrelationen.

PropBank:

REL = gave
ARG0 = they
ARG1 = a standing ovation
ARG2 = the chefs

NomBank:

REL = ovation
ARG0 = they
ARG1: Präposition
SUPPORT = gave

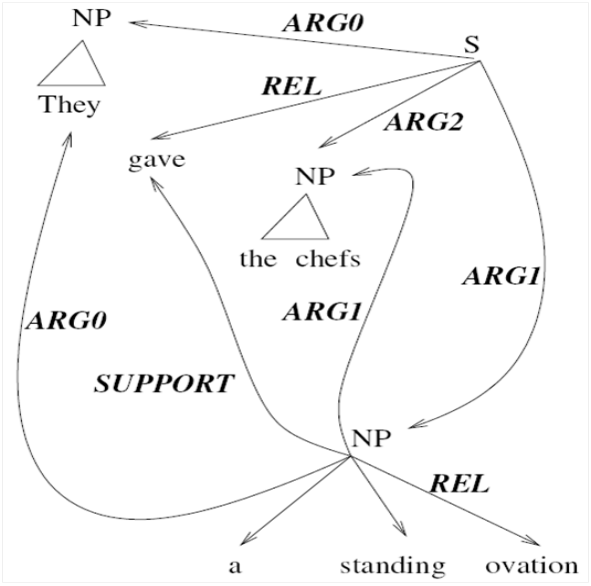


Abbildung 1.60: PropBank und Nombank

SUPPORT-Verben sind Verben, die Substantivwörter bis eins (oder mehr) ihrer Argumente per das Argumentteilen anschließen. Z.B. *John took a walk*, das Verb *took* teilt sein Subjekt mit dem Substantivwort *walk*. Aber SUPPORT-Verben können aus einigen Gründen problematisch sein.[BR04]

Fehler und Problem

Mit Hilfe der NomBank hat die Genaurigkeit der Interannotation schon auf etwa 85% gesteigert. Aber es gibt noch einige Ursachen, Durch die das Ergebnis der Annotation negativ auszuwirken ist: a)Widersprüche hinsichtlich der SUPPORT-Verben und die geteilten Argumente, die zu ihnen gehören; b) Unterschiede zwischen den Annotatoren könnten durch Störungen(Tippfehler, schlecht geformte Ausgabe, u.s.w) verursachen.

Zum Punkt a) sollte man mögliche Eigenschaften von SUPPORT-Verben formalieren und betrachten[BR04]:

- Paare der SUPPORT Verb/Substantivwort können idiosynkratisch angeschlossen werden. Einige Forscher nennen sie Idiome oder Redensart
  Z.B. *take a walk*

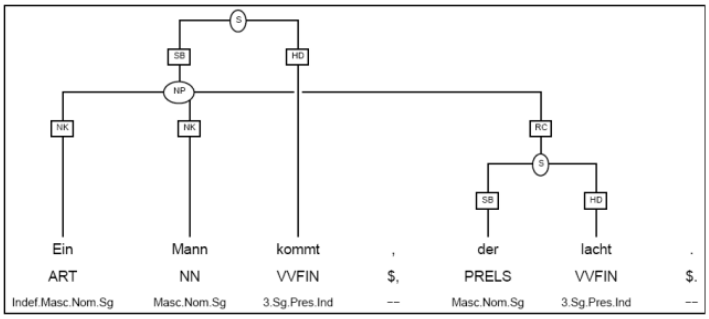


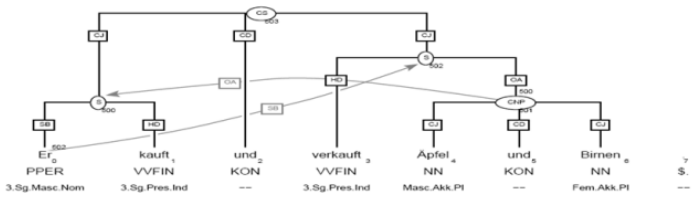
Abbildung 1.61: NEGRA Format

- Das Verb kann wesentlich leeren Sinn sein.
  Z.B. *make an attack*
- Die (Verb/Substantiv) Kombination kann eine bestimmte Argumentenmenge nehmen, aber die Bedeutung ist ganz anderes von jedem einzelnen Wort.
  Z.B. *take advantage of*
- Einige SUPPORT-Verben teilen das Subjekt von jeder mögliche Nominalization in einem bestimmten Argument.
  Z.B. *He attempted an attack*
- In einigen Fällen sind SUPPORT-Verb und das Substantiv aus den ähnlichen semantischen Kategorien.
  Z.B. *fight a battle*

1.7.5 TIGER Baumbank

Die TIGER Baumbank ist ein gemeinsames Projekt von dem Department of Computational Linguistics and Phonetics in Saarbrücken, dem Institut von Natural Language Processing(IMS) in Stuttgart und dem Institute für Germanistik in Potsdam. Sie setzt auf den Ergebnissen des Projekts NEGRA auf. Wobei NEGRA Projekt eine erste deutsche Baumbank ist. Die Ziele des TIGER-Projekts sind die Verfeinerung und Erweiterung der Negra-Annotationsrichtlinien und die Erstellung einer Baumbank mit 40.000 Sätzen. Die Annotation erfolgt dabei halbautomatisch. Ein Wortarten-Tagger und ein partieller Parser schlagen für jeden Satz die Wortarten- bzw. Phrasenannotierung vor, die dann vom Annotierer korrigiert werden muss. Während im Englischen aufgrund der festen Wortstellung Basispositionen und damit die Position von Spuren leicht zu bestimmen sind, ist dieses im Deutschen wesentlich schwieriger. Die Basisposition extraponierter Komplementsätze beispielsweise ist bis heute umstritten. Daher sind im TIGER-Korpus Prädikat-Argument-Strukturen als Teile eines ungeordneten Baums, d.h. eines Graphen mit kreuzenden Kanten, vgl. Abb.1.61 annotiert worden.

Die Basis des TIGERS Baumbank sind Texte aus dem deutschen Zeitung Frankfurter Rundschau, um eine ausgedehntere Strecke der Sprachveränderungen zu umfassen. Nur komplette Artikel werden benutzt, die wurden von aller Arte des domains(regionale, sportliche. . .) angenommen wurden.



Wortform/ Nummer	Wortart/ Kategorie	morphologische Information	Kanten- label	Eltern- knoten	sek. Kante	Ziel- knoten
Er	PPER	3.Sg.Masc.Nom	SB	500	SB	502
kauft	VVFIN	3.Sg.Pres.Ind	HD	500		
und	KON	-	CD	503		
verkauft	VVFIN	3.Sg.Pres.Ind	HD	502		
Äpfel	NN	Masc.Akk.Pl	CJ	501		
und	KON	-	CD	501		
Birnen	NN	Fem.Akk.Pl	CJ	501		
.	$.	-	-	0		
#500	S	-	CJ	503		
#501	CNP	-	OA	502	OA	500
#502	S	-	CJ	503		
#503	CS	-	-	0		

Abbildung 1.62: TIGER-Format

Zunächst schauen wir um, wie die Annotation im NEGRA-Format aussieht. Sie unterscheidet drei Ebenen (vgl. Abb. 1.61):

1. Tags auf Wortebene, die die Wortart und morphosyntaktische Information angeben (z.B. ART Masc.Nom.Sg für Artikel, Maskulinum, Nominativ, Singular).
2. Knotenbeschriftungen für phrasale syntaktische Kategorien (z.B. NP für Nominalphrase)
3. Kantenbeschriftungen für syntaktische Funktionen (z.B. HD für Kopf, SB für Subjekt).

Alle Kreuzende Kanten werden über die Beschreibung von Argumenten hinaus auch zur Kodierung von weiteren linguistischen Abhängigkeiten verwendet. In der Abbildung 1.61 liegt beispielsweise ein extraponierter Relativsatz vor, der über eine kreuzende Kante mit der Nominalphrase des Bezugsnomens verknüpft wird.

Im Vergleich zm NEGRA-Format stellt eine weitere Besonderheit des Datenmodells, die so genannten sekundären Kanten sind, dar. Sie werden bei der Behandlung der Koordination eingesetzt (vgl. Abb. 1.62. Die koordinierten Elemente werden zusammengefasst, um eine neue Konstituente zu bilden (hier: CNP für koordinierte Nominalphrasen und CS für koordinierte Teilsätze). Dabei kann es vorkommen, dass Konstituenten in einem der Konjunkte fehlen. In diesem Fall werden sekundäre Kanten gezogen. Im vorliegenden Beispiel ist das Personalpronomen Er sowohl Subjekt des ersten als auch des zweiten Konjunkts; Äpfel und Birnen ist entsprechend das Akkusativobjekt beider Konjunkte.

Es gibt zwei Annotationsmethoden in TIGER:

- Die verallgemeinerte Annotation läuft im gleichen Format wie die PropBank, das NEGRA Projekt von der Uni- Saarbrücken wird verwendet. Das enthält:

Wort	Preterminalsymbol	F-Struktur
Brei	nomen	Kasus Akkusativ Genus Maskulin Numerus Singular
Die	artikel	Kasus Nominativ Genus Neutrum Numerus Plural
essen	verb	Tempus Präsens Subjekt: Kasus Nominativ; Numerus Plural Objekt: Kasus Akkusativ; Numerus Plural
Kind	nomen	Kasus Nominativ Genus Neutrum Numerus Singular
Kinder	nomen	Kasus Nominativ Genus Neutrum Numerus Plural

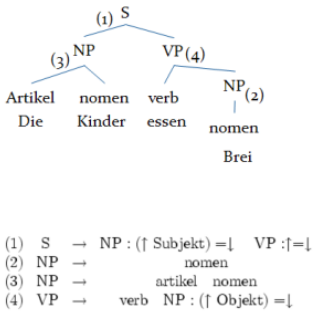


Abbildung 1.63: Lexikalisch-funktionale Grammatik

1. POS Tags sind Stuttgart-Tübingen-Tagset (STTS)
2. Morphologische Analyse sind die erweiterten STTS
3. Die grammatische Funktion in der direkt dominierenden Phrase
4. ie Kategorie von nichtterminalen Knoten (Phrasen)

- zweistufige LFG[12]-Annotation:

1. Der TIGER-Korpus wird durch die LFG-Grammatik analysiert und die Ausgabe der LFG-Grammatik wird halbautomatisch disambiguiert.
2. Die ausgewählte Ausgabe wird dann automatisch zum TIGER-Exportformat(TigerXML) konvertiert.

Lexikalisch-funktionale Grammatik

LFG ist Eine Transformationsgrammatik für die Syntax, Morphologie und Semantik. Im Gegensatz zur Chomskys Grammatik, die durch Transformationen verbundene separate Ebenen der linguistischen Darstellung beinhaltet, basiert auf zwei sich gegenseitig einschränkenden Strukturen:

- Ein Konstituentenbaum (C-Struktur):
  - Konstituenten sind Wortfolgen, die in einem inneren Zusammenhang stehen. Sie werden auch als Phrasen bezeichnet. Ein Satz enthält Informationen über Morphologie, Wahlbereich und linearer Ordnung. Um die C-Struktur zu verstehen, betrachten wir zunächst folgende einfache formale Grammatik(abb.). die Wörter, werden nicht in der Grammatik angegeben, sondern stehen in einem Lexikon von Terminalsymbolen.

Da C-Struktur alleine nicht ausreichend, wird F-Struktur als zusätzliche Eigenschaft genutzt. Hier gibt's ein sinnvolles Beispiel 1.63 aus Wikipedia.

[12]Lexikalisch-funktionale Grammatik

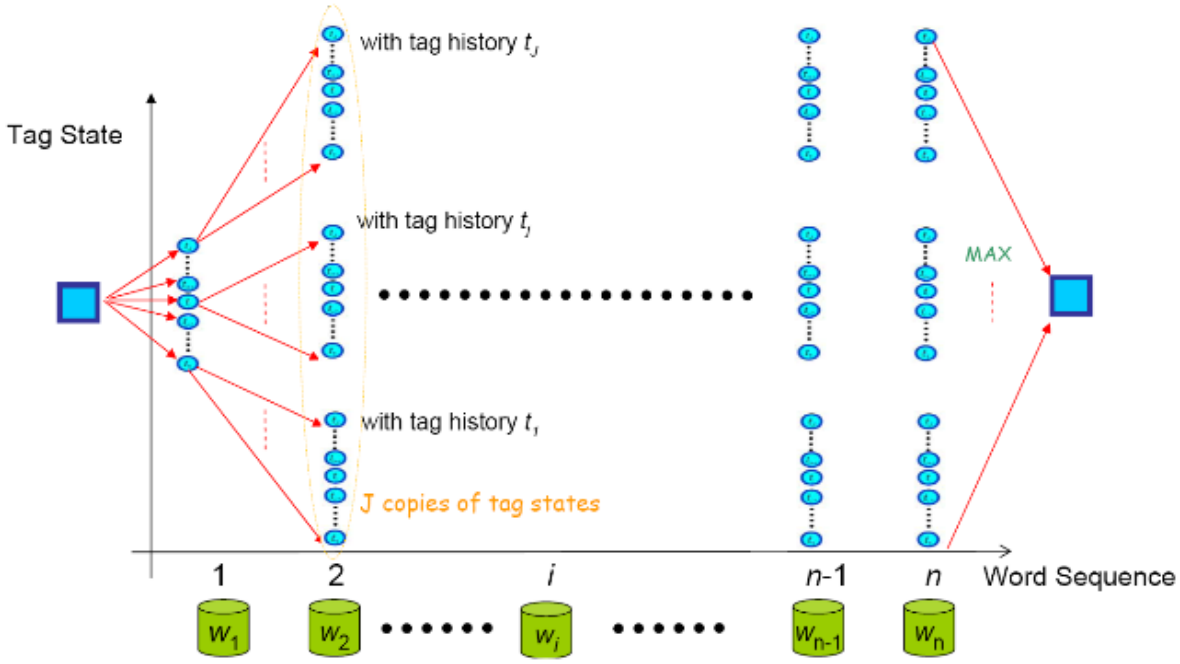


Abbildung 1.64: Trigram-HMM

- eine Merkmalstruktur (F-Struktur):
  - Sie stellt Informationen der Prädikatargumentenstruktur, über Modifikation, Zeit, Stimmung u.s.w. dar.
  - Bei einer $\uparrow{=}\downarrow$ Zuordnung wird die F-Struktur des übergeordneten Knotens der C-Struktur mit der F-Struktur des untergeordneten Knotens unifiziert.
  - Gleichung $(\uparrow Subject) = \downarrow$ unifiziert den Knoten Subjekt der F-Struktur des übergeordneten Knotens der C-Struktur mit der F-Struktur des untergeordneten Knotens.
  - Die Notation der Unifikationsstruktur erfolgt in Attribut-Wert-Matrizen.

automatisierte Annotation vom TIGER

Wortartenannotation-TnT In TIGER Baumbank wird ein Tagger, der TnT heißt, für automatisierte Annotation benutzt. TnT ist die Abkurzung von Trigrams'n'Tags, Dieser Tagger in Abbildung 1.64 ist eine Implementierung des Viterbi-Algorithmus für Zweiordnungen Markov Modell. Unbekannte Wörter werden durch ein Suffix und eine aufeinander folgende Abstraktion behandelt [Bra00].

Cascaded Markov Modell Die grundlegende Idee der Cascaded Markov Models ist der Syntaxbaum Schicht für Schicht zu erstellen, erste Strukturen(POS-Tagging) von Tiefe eins, dann eine Ebene hoch ist von Tiefe zwei gekennzeichnet und so geht's weiter(Phrasenvereinbarung). Für jede Schicht stellt ein markov Modell den besten Satz von Phrasen fest. Diese neuen festgestellten Phrasen sind wiederverwendbar und für die folgende Schicht als Eingabe. Eine T die eine weitere Schicht addiert. die Phrase, die in jeder Schicht angenommen wird, wird

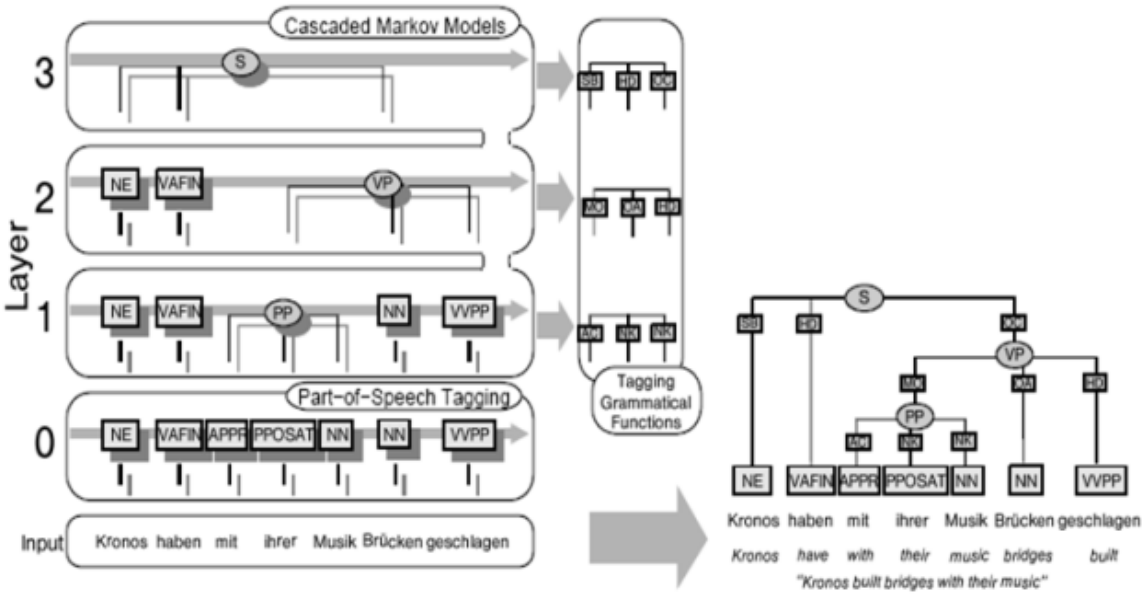


Abbildung 1.65: Cascaded Markov Modell

nach dementsprechenden stochastischen kontextfreien Grammatikrichtlinien erzeugt(die Ausgaben des Markov Modells) und nachher von links nach rechts durch Markov Modelle gefiltert [Bra99].

1) Initialisierung:

$$\Delta_{0,t}(q) \quad = \quad P(q|q_s)\delta_{0,t}(q) \tag{1.86}$$

2) Induktion:

$$\Delta_{t,t'}(q) = \max_{(t'',t,q')\in Lattice} \Delta_{t'',t}(q')P(q|q')]\delta_{t,t'}(q), 1{\leq}t < T. \tag{1.87}$$

3) Terminierung:

$$\max_{Q\in Q^\star} P(Q, Lattice) = \max_{(t,T,q)\in Lattice} \Delta_{t,T}(q)P(q_e|q). \tag{1.88}$$

Korpusanfragesprache

Eine vollständige Beschreibung der Konzeption der Sprache befindet sich in [Lez02]. Diese Anfragesprache besteht aus drei Niveaus. Auf dem ersten Niveau können Knoten durch Boolesche Ausdrücke über Attributwerte beschrieben werden. z.B. bei der Abfrage [word="lacht"& pos="VVFIN"] wird genau dann die Terminalknote lacht ausgewählt. Auf dem zweiten Niveau werden Knoten über Relationen miteinander verknüpft. hier listen wir beispielweise a paar Zeichens von Relationen auf:

$>$	direkte Dominanz
$>*$	allgemeine Dominanz
.	lineare Präzedenz
.*	allgemeine Präzedenz
$	Geschwister
$.*	Geschwister mit Präzedenz

Auf der Graphbeschreibungsebene werden schließlich Knotenrelationen durch boolesche Ausdrücke verknüpft. Um die Identität zweier Knoten auszudrücken, werden Variablen eingesetzt. Zur Zeit wird diese Anfragesprache in TIGERSearch(Query) genutzt und implementiert.

TigerXML

Baumbanken müssen zunächst in das Korpusdefinitionsformat konvertiert werden. Aber wenn dies Format nicht standardisiert ist, führt zu Schwierigkeiten bei der Entwicklung, oder Erweiterung von Konvertierungsprogrammen. Außerdem werden bei der Korpusanfragung die gezielten Anfrageergebnisse beeinflusst. Deswegen lassen sich diese Probleme durch XML-Datenformat lösen. In Abbildung 1.66 stellt dies TigerXML-Format bzw. dessen gezielte Information aus TIGER-Korpus dar. Es dient dazu, dass zum einem eine Validierung der gespeicherten Daten und zum anderen eine grafische Darstellung des TIGER-Korpus zu ermöglichen, ferner ist die Korpusanfrage in TIGER auch möglich. Im folgenden Abschnitt TIGERSearch wird direkt vorteilhaft beschrieben, warum mit XML und wie gut ist sie.

1.7.6 TIGERSearch

Das TIGERSearch-Werkzeug(abb.1.67) besteht aus drei Einzelprogrammen: mittels TIGER-Registry ,TIGERSearch und TIGERGraphViewer. Einen Überblick über die Funktionsweise und Systemarchitektur stellen das [EK03] und in Abbildung 1.68 dar. Außerdem um schneller das Tool zu erlernen, können Sie unter diese Linke aus Projektgruppe des TIGERs eine detailierte Vorstellung vom TIGERSearch einsehen.

TIGERRegistry

Für den Import diverser Korpus-Formate überführt TIGERRegistry das zu importierende Korpus zuerst in das eigene TIGER-XML-Format. Aus der XML-Zwischendarstellung wird dann der Korpusindex erzeugt. Durch die Generierung einer XML-Zwischendarstellung ist der

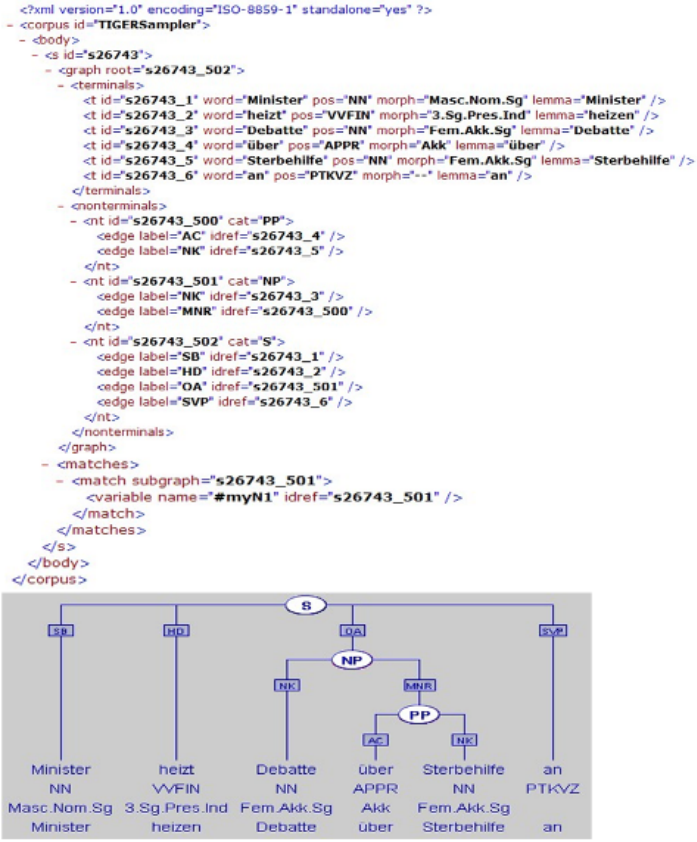


Abbildung 1.66: TigerXML im Vergleich zum Syntaxbaum

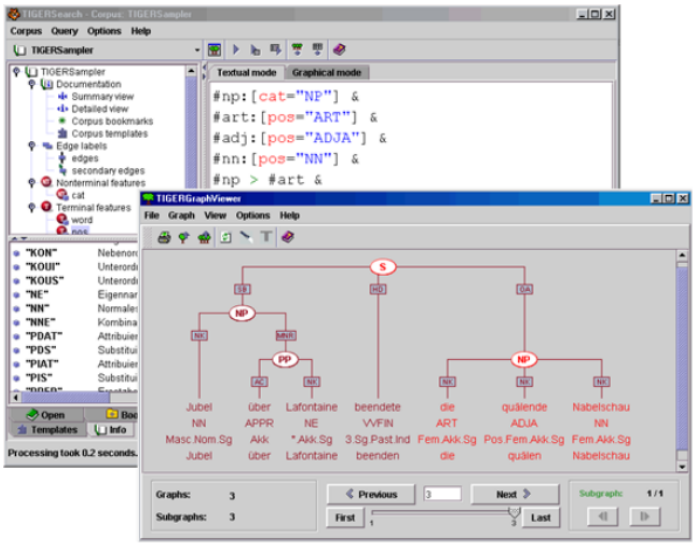


Abbildung 1.67: TIGERSearch GUI

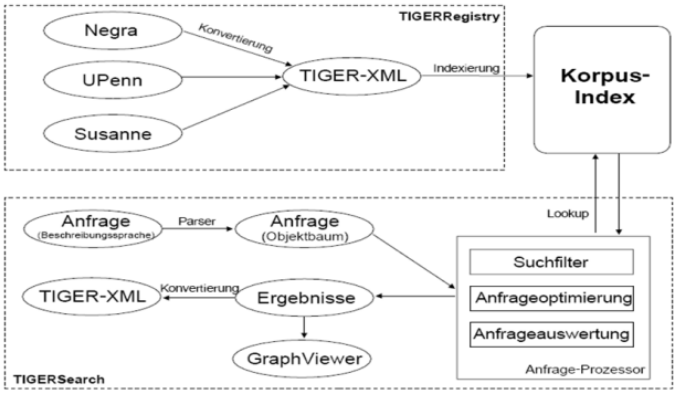


Abbildung 1.68: Architektur des TIGERSearchs

Indexierungschritt unabhängig vom Importfilter und erleichtert so dessen Erstellung. Dass bei der Zwischendarstellung das XML-Format gewählt wurde, hat zudem noch die Vorteile, dass zur Bearbeitung und Validierung eines Korpus XML-Werkzeuge eingesetzt werden können und dass - falls ein zu importierender Korpus in einem XML-Format vorliegt - die Erstellung eines XSLT-Stylesheets zur Überführung in das TIGER-XMLFormat ausreicht.

TIGERSearch

Eine textuelle Suchanfrage ist nach dem Laden eines Korpus mit TIGERSearch möglich. Im Frame linksunter werden alle Attribute des geladenen Korpus angezeigt. Die möglichen Werte mit Erklärungen, falls Erklärungen in der Korpusdefinition enthalten sind, lassen sich in einem Pulldown- Menü einsehen. Die Anfrage muss dabei in der speziellen TIGERSearch-Anfragesprache erfolgen. Nach Absenden der Anfrage mit dem SSearchKnopf wird, für den Benutzer nicht sichtbar, die Anfrage normalisiert, d.h. in eine disjunktive Normalform gebracht. Um die Anfragebearbeitung zu beschleunigen, wird danach eine Anfrageoptimierung durchgeführt. Nachdem alle relevanten Sätze durchsucht worden sind, wird der TIGERGraphViewer zur Ergebnisvisualisierung gestartet. Während der Suche wird der Benutzer über den aktuellen Fortschritt durch einen Statusbalken informiert. Die Suche lässt sich jederzeit abbrechen.

TIGERGraphViewer

Der TIGERGraphViewer kann zum einen die Phrasenstruktur-Bäume der einzelnen Sätze eines Korpus anzeigen, zum anderen dient er zur Visualisierung der Anfrageergebnisse. Dabei werden die gefundenen Strukturen farblich hervorgehoben. Unterhalb der grafischen Baumdarstellung des Satzes befinden sich eine Anzeige, die über Anzahl der Sätze und Treffer Auskunft gibt, und Bedienelemente zur Navigation. Mit den beiden Knöpfen im Feld SSubgraph"navigiert man innerhalb eines Satzes, d.h. es wird nicht zu einem anderen Satz gesprungen, sondern lediglich zur nächsten Fundstelle innerhalb des Satzes, wobei nun andere Strukturen eingefärbt werden, die ebenfalls zu einem Treffer geführt haben. Die Baumdar-

stellung einzelner Sätze lässt sich in den Formaten JPG[13], PNG[14], PDF[15] und TIF[16] als Bitmap-Grafik oder im SVG-Format[17] als Vektor-Grafik speichern. Außerdem lassen sich alle oder eine Sequenz der Bäume in dem XML-basierenden SVG-Format mitsamt Navigation speichern, die es erlaubt, interaktiv zu den einzelnen Bäumen zu springen.

1.7.7 Event Extraction

Einleitung

Der Begriff Information Extraction wird definiert als die automatische Identifikation und strukturierte Repräsentation von Entities, Relationen oder Events in natürlich- sprachlichen und meist unstrukturierten Texten. Ziel ist dabei, aus einer vordefinierten Domäne Wissen zu erzeugen.
Anwendungsbereiche für die Information-Extraction sind z.B. ·Analyse von Zeitungsartikeln nach relevanten Wirtschaftsdaten · Zusammenfassen von größeren Texten · Erstellung eines Index
Es gibt zwei Typen von Information Extraction. Zum einen die name- extraction, auch bekannt unter dem Namen "Named Entity Recognition"[näheres dazu in: Kapitel 1.2] und zum anderen die event-extraction, welches hier näher erläutert wird.

Einleitung

Die Event Extraction wird bezeichnet als die Extraktion von Entitäten und anderen semantischen Einheiten aus natürlich sprachlichen Texten. Konkretes Ziel hierbei ist es, die Textelemente in eine wesentlich bessere maschinenlesbare Form zu bringen. Abspeicherung dieser Elemente können in Datenbanken oder XML- Strukturen geschehen wie von Grishman vorgeschlagen wird [Mit03]. Die einheitliche Extrahierung der Informationen sorgt für eine leichte Durchsuchung und dient einer effizienteren Weiterverarbeitung.
Ein weiterer Begriff in der Information Extraction ist der des Relation Extraction. Relation Extraction kann dazu verwendet werden Beziehungen zwischen Named Entities und andere semantischen Einheiten zu extrahieren. Man kann also die Relation Extraction dazu verwenden, Events zu erkennen. Ziel der Relation Extraction ist die Named Entities und andere Informationen, welche extrahiert worden sind, in Beziehung zu bringen um die Events zu stützen. [näheres dazu im Abschnitt Relation Extraction]

Grundlagen

Da die Eingabe für die Event Extraction natürlich sprachliche Texte sind, die nicht alle eine einheitliche Struktur aufweisen könnten, kann sich die Herangehensweise für die Event Extraction auch in einigen Schritten unterscheiden. Textstrukturen werden klassifiziert in:

- Strukturierte

[13] Joint Photographic Experts Group (JPEG) File Interchange Format
[14] Portable Network Graphics
[15] Portable Document Format
[16] Tagged-Image File Format
[17] Scalable Vector Graphics

- Halb-Strukturierte
- Unstrukturierte

Wenn der Text im günstigsten Fall strukturiert vorliegt, ist die Event Erkennung mithilfe von Mustern möglich, ohne viel linguistische Kenntnisse zu benötigen. Für die Event Extraction in halb-strukturierten Texten ist eine Kombination aus Mustern und der erhöhten Kenntnis der Linguistik nötig. Im ungünstigsten Fall in welchem der Text völlig unstrukturiert ist, reichen einfache linguistische Kenntnisse nicht aus. In diesem Fall ist eine Layout-Analyse durchzuführen. Um einen strukturierten Text zu erhalten, kann man die Methode der Named Entity Recognition durchführen. Diese dient der Bestimmung und Klassifizierung von Eigennamen. Mithilfe von regulären Ausdrücken ist die Bestimmung von Named Entities möglich.

Beispiel

Der Satz:
„Max Mustermann ist Mitglied der SPD „ wird maschinenleserlich abgespeichert in Form von:
<NAME type=person> Max Mustermann <NAME> ist Mitglied der
<NAME type=partei> SPD <NAME>

Methoden

Man kann Triggerwörter erstellen, um später die Events leichter zu identifizieren. Ermöglicht wird dies mit einfachen regulären Ausdrücken. Jedoch besteht hier der Nachteil das die Events nicht immer erkannt werden können. Mehrere Sätze können semantisch dasselbe aussagt, aber syntaktisch ganz anders aufgebaut sein. Daher ist ein etwas aufwändigeres Konzept und die Kombination von regulären Ausdrücken nötig. Wenn man die regulären Ausdrücken kombiniert, bilden diese sogenannte Muster (pattern). Wenn ein Text abgearbeitet wird, werden die Wörter klassifiziert um später die Event-Felder zu füllen. Je nachdem zu welchem Muster die Wortgruppe korrespondiert, wird es der jeweiligen Klasse zugeordnet. Die Methode, welches das Klassifizieren der Satzbestandteile nach einem bestimmten Muster ermöglicht, ist das Pattern Matching. Auch hier entstehen die Nachteile, das z.B. ein Satz bei gleichbedeutender Semantik, verschieden aufgebaut sein kann. Zudem kann sich auch die Information auf mehrere Sätze aufteilen. Somit sind diese einfachen Muster nicht mehr effizient und man benötigt kompliziertere Muster. Vorgegebene Sätze in ihre Bestandteile (noun phrases oder Verbgruppen) zu unterteilen und sie somit zu untersuchen ist hier von Vorteil. Nach jedem Schritt werden für die Klassifizierung Pattern Matcher eingesetzt um die Satzbestandteile zu identifizieren. Siehe dazu Partial Parsing.

Partial Parsing

Partial Parsing ist eine Methode zur Event Extraction, wo ein Satz Schritt für Schritt in seine Bestandteile aufgeteilt wird. Nach jedem Schritt werden Patternmatcher für die Klassifizierung eingesetzt. Dieses Verfahren erkennt Events arbeitet Satzweise. Partial Parsing kann in folgende Schritte aufgeteilt werden:

- NER
- Nounphrase Recognition

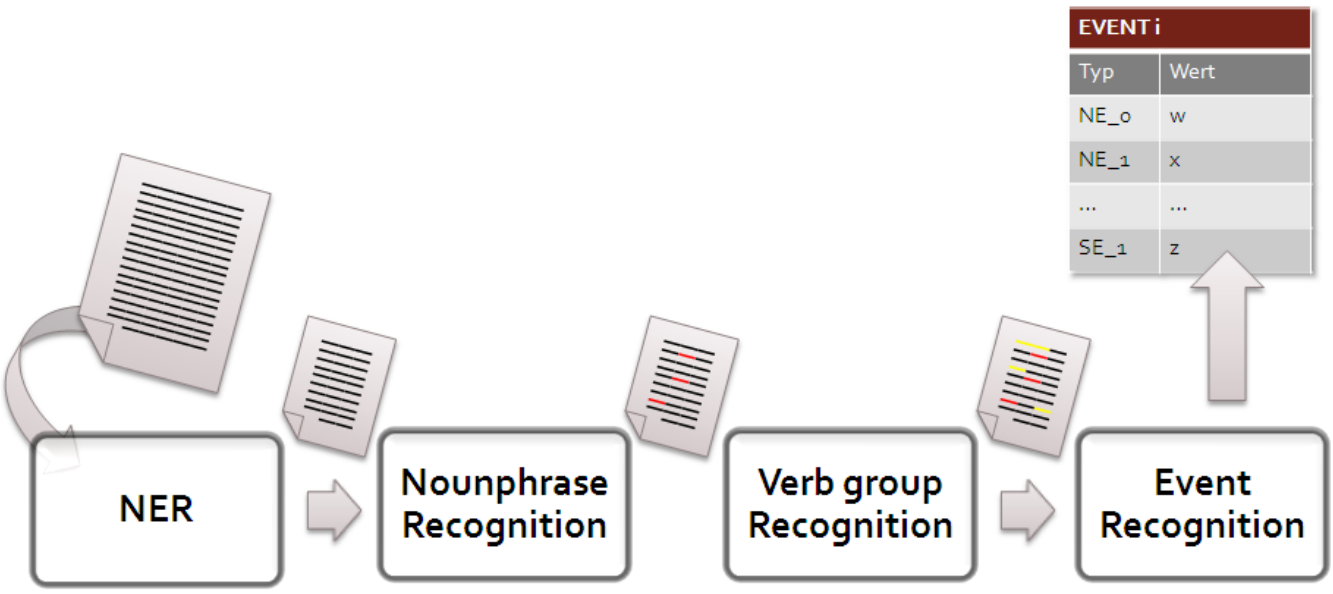


Abbildung 1.69: Schritte des Partial Parsings

- Verb group Recognition
- Event Recognition

Jeder Schritt benutzt die Ausgabe aus dem vorherigem Schritt und stellt die eigene Ausgabe dem nachfolgendem Schritt zur Verfügung. Beim NER handelt es sich um das übliche Verfahren um Eigennamen zu erkennen und zu markieren. Nounphrase Recognition wird dazu verwendet die Nomengruppen zu erkennen. Es werden zum Teil die Informationen aus dem NER-Schritt benutzt. Nounphrase Recognition ist gut geeignet Schlüsselwörter zu erkennen. Mit Verb group Recognition wird versucht die Verb-Gruppen zu erkennen und zu normalisieren. Die Verben sind wichtig, da so NE's und andere semantische Einheiten in Beziehung gebracht werden können. Nach der Event Recognition hat man ein Event-Format gefüllt. Das Event-Format ist ein Modell mit verschieden Typen. Ziel der Event-Extraction ist es die Typen mit Werten zu füllen und es geeignet abzulegen. Das besodere an dieser Art der

Event-Extraction ist, das z.B. nicht nur die NE's extrahiert werden. Durch vorher definierte Patternmatcher können auch Informationen extrahiert werden, die eigentlich so nicht im Eingabe-Text vorhanden sind. Durch die geeignete Extaktion der Informationen sind weitere Schritte vorstellbar, wodurch man Informationen erzeugen kann die eigentlich dem Menschen vorenthalten sind. So können auch hier Patternmatcher erstellt werden wodurch die neuen Informationen erzeugt werden können.

Probleme

Auch wenn die Event Extraction mit dieser Heransgehensweise Events erkennt, bestehen auch hier Probleme. Das Problem der Wiederholung von Referenzwörtern ist nicht nur hier ein Problem, sondern ein allgemeines Problem in der Information Extraction. Der Grund liegt in der Arbeitsweise der Methoden. Der Text wird einfach Satzweise bearbeitet was zum erwähnten

Problem führt. Es existieren aber Lösungsansätze in Form von Coreference Analysis. Coreference Analysis dient dazu, die referenzierten Wörter aufzulösen. In der Event-Extraction ist dies als Vorverarbeitungsschritt denkbar. Ein weiteres Problem ist die große anzahl von Satzstrukturen die in einem Freitext vorkommen. Um viele Events zu erkennen müsste für jeden Satztyp ein Muster geschrieben werden. Diesem Problem kann man entgegenwirken, indem man versucht Muster zu generieren [Gri97].

Pattern-Generierung

Mit Pattern-Generierung versucht man Pattern, also Muster, zu Generieren und auf Sätze anzuwenden. Ziel ist es die benötigten Muster automatisch zu generieren und auf den Eingabetext anzuwenden. Dazu versucht man u.A. die bei einfachen Sätzen gebildeten Muster auf kompliziertere anzuwenden. Ähnlich wie beim Partial Parsing sind auch hier folgende Schritte nötig:

- NER
- Nounphrase Recognition
- Verb group Recognition

Anschließend wird ein bestimmten Musters auf Grundlage der Bestandteile entworfen. Das erstellte Muster muss nochmal von Hand überprüft werden weil die Korrektheit nicht garantiert ist. Dabei werden die Muster klassifiziert. Ein Muster kann demnach in folgende Kategorien klassifiziert werden:

- Akzeptieren
- Verwerfen
- Vereinfachen

Bei schon annotierten Texten ist auch Vereinheitlichung von Patterns möglich. Dabei wird ein durchschnitt von mehreren Patterns zu einem gebildet. Das neue Muster bildet eine Verallgemeinerung zu den Quell-Mustern und kann nach Überprufung akzeptiert werden. Dieses vorgehen macht dann Sinn, wenn verschieden Muster z.B. nur ein einer stelle verschieden sind.

Fazit

Im allgemeinen kann Event Extraction dort angewandt werden, wo Informationen noch manuell gefunden werden müssen. Dabei können es sich um Medizinische Texte handeln oder auch Texte über Firmen und deren Tätigkeiten. Die Erfolgsquote in der Event-Extraction ist leider noch nicht so hoch wie z.B. im Bereich der reinen NER. Trotzdem ist eine Einschränkung auf eine bestimmte Domäne von Informationen unerlässlich.

1.8 Theorie der Fragebeantwortung

1.8.1 Einleitung

Fragebeantwortung ist die Aufgabe des Findens einer Antwort auf eine natürlichsprachlich-gestellte Frage. Der Korpus dieser Antwort ist zumiest sehr groß, etwa das Suchen im World Wide Web oder in riesigen Dokumentenarchiven. Besonders aufgrund der großen Pupularität des Internets, werden genau diese Systeme immer wichtiger und bekommen eine steigende Bedeutung. Auch bei einer Suchmaschine wäre es wünschenswert auf eine konkret gestellte Frage, genau eine Antwort zu bekommen.
Fragebeantwortung teilt sich grob in drei Schritt auf:

1. Analyse der Frage
2. Suchen relevanter Dokumente/Text-Passagen
3. Antwort-Extraktion

Der schwierigste Teil ist in diesem Zusammenang die konkrete Analyse der gestellten Frage. Hier müssen Antworttypen bestimmt werden und auch die Schlüsselwörter für die Anfrage an einen Index.

1.8.2 Fragekomplexität

Jede Frage kann einer so genannten Frageklasse (nach [PH01])zugeordnet werden. Je nach Komplexität der Frage ist der Aufwand zur Beschaffung einer Antwort mehr oder weniger hoch. Klasse 1 stellt Fragen nach bestimmten Attributen, einfache Nennung von Events oder auch Bedingungen von Events. In diesem Fall ist die Antwort ein einfacher String. Zur Klasse 2 gehören Fragen, die anhand einer Vorlage generiert werden. Die Antworten auf die gestellte Frage befinden sich dabei in mehreren Quellen, wobei sich diese Antworten um ein gemeinsames Stichwort drehen. Stellt man die Frage „Wer ist Bill Gates?“ könnten Antworten lauten: „Ein Mann, der am 28. Oktober 1955 in Seattle geboren wurde.“, „Gründer von Microsoft“ oder auch „Drittreichster Mann der Welt“. Alle drei Antworten sind richtig und werden Frageklasse 2 zu einer Vorlage zusammengefasst. In Frageklasse 3 sind die Antworten noch mehr gestreut und orientieren sich nicht mehr an einem Charakteristikum. Stattdessen kann diese Klasse als eine Menge von Vorlagen gesehen werden, die sich um verschiedene Themen kreisen. Diese Frageklasse ist daher am ehesten vergleichbar mit einem Dossier und Fragen können hier sehr offen gestellt sein. Bei einer Frage „Welche Konsequenzen hat die Veruntreuung der UNICEF-Spenden gehabt?“ wären daher viele Antworten möglich, darunter z.B. „Rücktritt UNICEF-Chef“ oder „Spendenrückgänge“.
Frageklassen dienen also der groben Einteilung der Schwierigkeit einer Frage.

1.8.3 Antworttypen-Modell

Einer der wichtigsten Aufgaben bei der Fragebeantwortung ist die Bestimmung des richtigen Antworttyps. In der Regel basiert der Antworttyp auf einer syntaktischen Analyse der Frage. Zunächst wird dabei der Satzbau analysiert und eventuell vorhandene Stoppwörter entfernt. Anschließen bestimmt man den Antworttyp anhand des Questions Stems und den Konzepten der Frage. Alle Kombinationen mit den jeweiligen Antworttypen sind dabei vordefiniert in einer Tabelle gegeben. Die nachfolgende Abbildung zeigt ein Beispiel. Nachfolgend

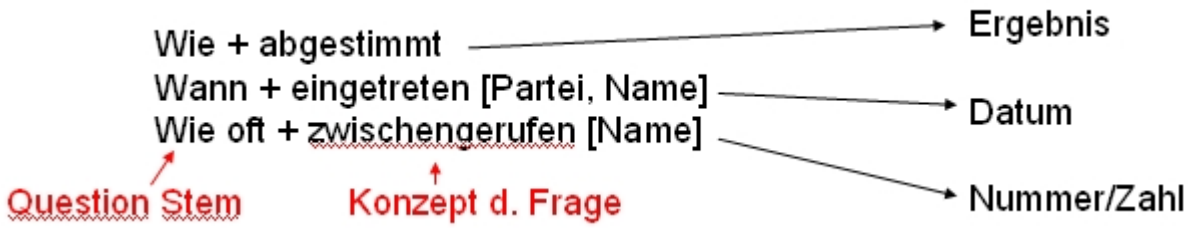


Abbildung 1.70: Beispiel für die Zuordnung der Antworttypen

wollen wir ein Antworttypenmodell vorstellen, das Frageklassen, Antworttypen und Formate berücksichtigt. Das Antworttypenmodell ist ein 4-Tupel mit folgenden Attributen:

$$\text{AT} = [\text{Kategorie, Abhängigkeit, Nummer, Format}]$$

An erster Stelle unseres Tupel-Modells beschreibt Kategorie die Art der Antwort, die sich wiederum nach den, im vorherigen Abschnitt beschriebenen, Frageklassen richtet. Hier ist entweder ein vordefinierter Antworttyp, eine Definition, eine Vorlage oder eine Zusammenfassung möglich. An zweiter Stelle beschreibt Abhängigkeit das Verhältnis zwischen der Kategorie und den Antworttypen. Fragen wir also „Wann wurde Kennedy erschossen?“, beschreibt die Abhängigkeit, dass wir als Antworttyp ein Datum erwarten. Nummer steht an dritter Stelle unseres Tupels und ist ein einfacher Boolean-Wert (Flag), der angibt, ob ein einzelner Wert oder eine Liste von Elementen gesucht ist. Schließlich definiert Format ein spezielles Ausgabeformat, bspw. wäre dies für ein Datum „dd.mm.yy“.
Kommen wir nun zu den einzelnen Kategorie-Typen:

Antworttyp Antworttypen werden, wie bereits beschrieben, aus Question Stem und Konzepten der Frage gewonnen. Beide bestimmen zusammen den Antworttyp, der zumeist in einer Tabelle vordefiniert ist. Oft handelt es sich bei den Antworttypen um Named Entities, die so auch direkt in den Texten vorkommen. Hier ist allerdings zu beachten, dass bei der Abbildung Antworttyp zu Named Entity eine 1 zu n bzw. n zu 1-Beziehung bestehen kann. So ist es möglich, dass man dem Antworttyp „Person“ die Named Entities „Person“ und „Name“ zuweist, andererseits kann man aus den Typen „Geschwindigkeit“, „Dauer“ und „Menge“ den Named Entity „Quantität“ abbilden.

Definitionen Antworten zu Definitionen ergeben sich direkt aus der Fragestellung. Auf eine Frage „Was *ist/war* [ObjektName]?“ kann man eine Antwort „ [ObjektName] *ist/war* [Antwort]“ automatisch generieren.

Vorlage Besteht eine Antwort aus verschiedenen Antworttypen, so ergibt sich daraus automatisch der Typ Vorlage. In diesem Fall muss eine Vorlage generiert werden, die aus einer Menge von Antworttypen (Slots) besteht. Dies passiert in drei Schritten:

1. Für eine Frage wird zunächst nach möglichen Antworten gesucht. Für jede Antwort wird ein Frageabhängigkeitsgraph erstellt und darin nach gemeinsamen Merkmalen gesucht.

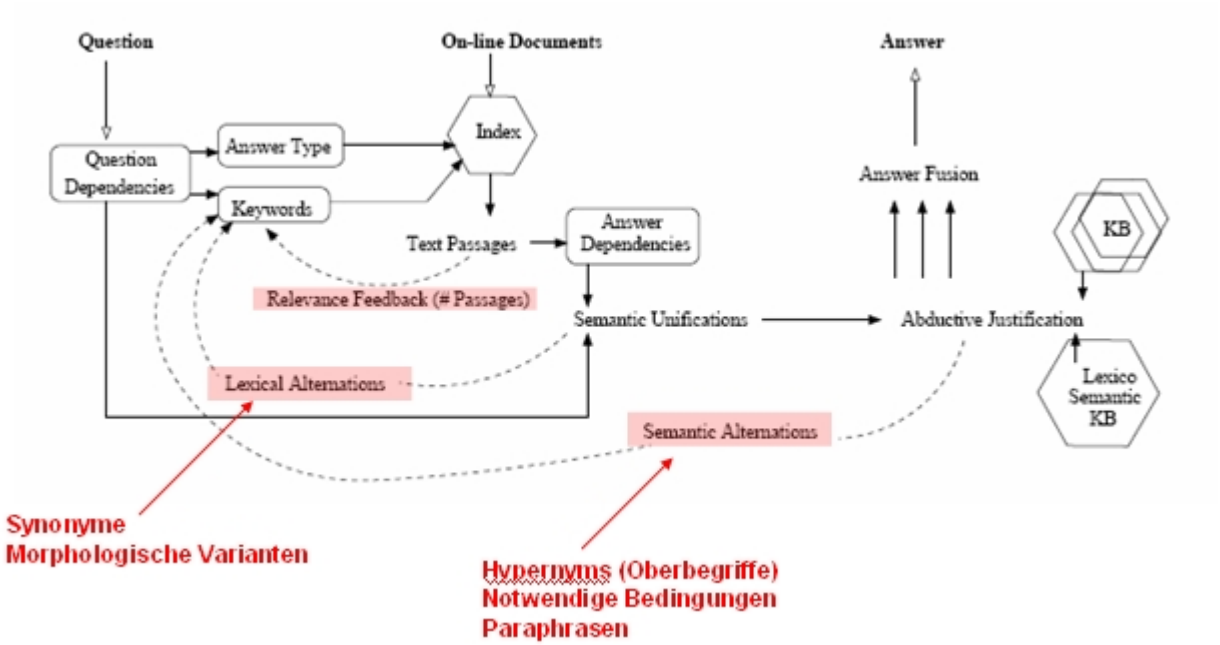


Abbildung 1.71: Ablauf der Fragebeantwortung mit verschiedenen Feedback-Loops

2. Die häufigsten Merkmale, die in den Antworttypen genannt werden, werden identifiziert.
3. Gemeinsame Merkmale werden in der Vorlage gesammelt und diese als Namen für Slots genutzt.

Zusammenfassung Zusammenfassungen enthalten verschiedene Vorlagen, die nicht aus gemeinsamen Merkmalen bestehen. Daher können diese nicht weiter zusammengefasst werden und eine Sammlung von verschiedenen Vorlagen wird als Antwort ausgegeben.

1.8.4 Ablauf der Fragebeantwortung

Abschließend wollen wir ein Beispiel für ein komplexes Q/A-System geben. Die Besonderheit dieses Systems sind die so genannten Feedback-Loops, die die Ergebnisse verfeinern und spezialisieren können.
Eine Anfrage besteht zunächst aus einem Antworttyp und einer Menge von Schlüsselwörtern. Diese werden durch Frageabhängigkeiten zusammengeführt und als Query an einen Index geschickt. Dieser liefert eine Menge von Dokumenten, die auf die Schlüsselwörter und den Antworttypen passen. Aus diesen Dokumenten werden relevante Text-Passagen extrahiert. Hier kommt nun der erste Feedback-Loop ins Spiel. Werden zuviele Ergebnisse zurückgeliefert, muss die Menge der Schlüsselwörter spezialisiert werden, so dass es bei einer erneuten Anfrage weniger Treffer gibt. Andersherum werden bei zu wenigen Treffern neue Schlüsselwörter hinzugenommen. Dieses Verfahren endet, wenn keine Schlüsselwörter hinzugefügt oder gelöscht werden können.
Bei der Prüfung der Antwortabhängigkeiten können sich im nächsten Schritt neue lexikalische Alternativen ergeben. Synonyme oder morphologische Varianten könnten daraufhin zu den

Schlüsselwörtern hinzugefügt werden. Sobald die Antwort- und Frageabhängigkeit semantisch gleich ist, kann das Ergebnis in einem letzten Schritt geprüft werden („Abductive Judtification“). Dies geschieht durch eventuell vorhandene Wissensbasen. Falls in diesem Schritt Oberbegriffe oder Paraphrasen gefunden werden, könnte auch hier wiederum ein Feedback-Loop erfolgen. Dies wird durch die dritte Rückwärtskante „Semantic Alternations“ ausgedrückt.
Für die Ausgabe der Antwort kann nun eine Antwortfusion erfolgen. Die daraus resultierende Antwort wird letztlich ausgegeben.

1.9 TREC

1.9.1 Was ist TREC?

TREC ausformuliert ist die Text Retrieval Conference, dies ist ein Workshop des National Institute of Standards and Technology (NIST genannt). Die Konferenz wird seit 1999 abgehalten, mit dem Ziel Techniken zur Fragebeantwortung zu entwickeln. Seitdem wurde die Komplexität der Fragen und die Anzahl der Fragetypen ständig erhöht.
Zu Beginn dieses Workshops wurden nur Fakten-Fragen (Factoid questions), seit 2003 aber auch Listen- und Weitere-Fragen beantwortet. Zu diesen Fragetypen kamen 2006 die komplexen, interaktiven Fragen hinzu (complex, interactive QA auch ciQA genannt). Der Hauptunterschied zu den anderen Fragetypen ist hier, dass eine Interaktion mit dem Nutzer existiert. Die Quelldokumente die verwendet wurden waren bis 2006 ausschließlich Zeitungsartikel. Ab 2007 kamen zu diesen auch Blogs wie z.B. Tagebücher, Journale im Web hinzu.

1.9.2 Durchführung

Das Ziel der TREC-Durchführung ist es für eine bestimmte Interessentengruppe eine Liste von Fragen zu beantworten. Es werden dabei die Fragen in Serien, die zu einem bestimmten Thema mit unterschiedlichen Fragetypen zusammengefasst. Mögliche Themen für Frageserien sind z.B. Person, Organisation, Objekt oder ein Event. Für die Auswertung der Systemantworten sind die Mitglieder der NIST verantwortlich. Danach haben die Teilnehmer 2 Wochen Zeit um automatisch generierte Antworten zu erstellen. Aber es muss die Reihenfolge der Fragen so abgearbeitet werden wie sie vorher vorgegeben war. Als Hilfe dürfen innerhalb einer Serie vorherige Fragen/Antworten genutzt werden aber nachfolgende nicht.

Beispiel: Frageserie Es muss innerhalb einer Serie beachtet werden, dass es mehrere Fakten-Fragen, eine oder zwei Listen-Fragen und genau eine Weitere-Frage erlaubt ist.

1.9.3 Welche Fragetypen gibt es?

Die Fragen werden in 3 Fragetypen unterteilt.

Fakten-Fragen Fakten-Fragen, bei denen die Antwort ein 2-Tupel [dok-id, Antwort-String] oder Nil (es konnte keine passende Antwort gefunden werden). Die Prüfung der Antworten erfolgt durch eine Jury mit den Bewertungen:
- incorrect: Die gelieferte Antwort ist Falsch.
- not suported: Die Antwort ist korrekt, wird aber vom Dokument nicht unterstützt.
- not exact: Die Antwort ist korrekt, wird auch vom Dokument unterstützt aber es existieren unterschiedliche Antworten auf die Anfrage.

145	John William King convicted of murder		
	145.1	FACTOID	How many non-white members of the jury were there?
	145.2	FACTOID	Who was the foreman for the jury?
	145.3	FACTOID	Where was the trial held?
	145.4	FACTOID	When was King convicted?
	145.5	FACTOID	Who was the victim of the murder?
	145.6	LIST	What defense and prosecution attorneys participated in the trial?
	145.7	OTHER	
185	Iditarod Race		
	185.1	FACTOID	In what city does the Iditarod start?
	185.2	FACTOID	In what city does the Iditarod end?
	185.3	FACTOID	In what month is it held?
	185.4	FACTOID	Who is the founder of the Iditarod?
	185.5	LIST	Name people who have won the Iditarod.
	185.6	FACTOID	How many miles long is the Iditarod?
	185.7	FACTOID	What is the record time in which the Iditarod was won?
	185.8	LIST	Which companies have sponsored the Iditarod?
	185.9	OTHER	
212	Barry Manilow		
	212.1	FACTOID	What year was he born?
	212.2	FACTOID	How many times has he married?
	212.3	FACTOID	What is the name of the musical that he wrote about the Harmonistas?
	212.4	FACTOID	What music school did he attend?
	212.5	FACTOID	For what female singer was he the musical director and pianist in the 70's?
	212.6	FACTOID	What record label did he sing for in 2000?
	212.7	LIST	List the songs he recorded.
	212.8	OTHER	

Abbildung 1.72: Beispiel für Frageserie

- locally correct: Die Antwort ist korrekt, wird auch vom Dokument unterstützt aber der Prüfer findet eine bessere Antwort innerhalb des Dokumentes.
- globally correct: : Die Antwort ist korrekt, wird auch vom Dokument unterstützt und es gibt keine bessere Antwort.

Listen-Fragen Bei Listen-Fragen ist die korrekte Antwort eine unsortierte Liste mit unterschiedlichen Antworten. Die Bewertung der einzelnen Antworten ist identisch zu den Fakten-Fragen. Wobei die global korrekten Antworten in Äquivalenzklassen zusammengefasst werden. Mit Hilfe von F-Score werden die Listen-Antworten bewertet. Wie bekannt setzt sich F-Score aus Recall und Precision zusammen.

Weitere-Fragen Weitere-Fragen, sind ein ungeordnetes Set aus [dok-id, Antwort-String]. Der Antwort-String muss hier relevante Stichworte (Nuggets) enthalten. Diese werden vor der Anfrage durch den Prüfer ausgewählt und festgelegt. Die Bewertung des Fragetyps wird dadurch relativ erschwert. Die Bewertung läuft folgendermaßen ab:

a) Die Prüfer wählen eine Liste von „Vital“ und „Non-Vital“-Nuggets aus. (Vital bedeutet, dass das Stichwort gut ist. Non-Vital , dass diese akzeptabel ist.)

b) Prüfer gehen die Ausgabe durch und markieren Nuggets(doppelte zählen nur einmal). Das Bewertungsmaß ist also: 0,1, *nuggets*

c) F-Score bestehend aus Recall(wie bisher) und einem veränderten Precison-Wert - längenbasierter Precision-Wert

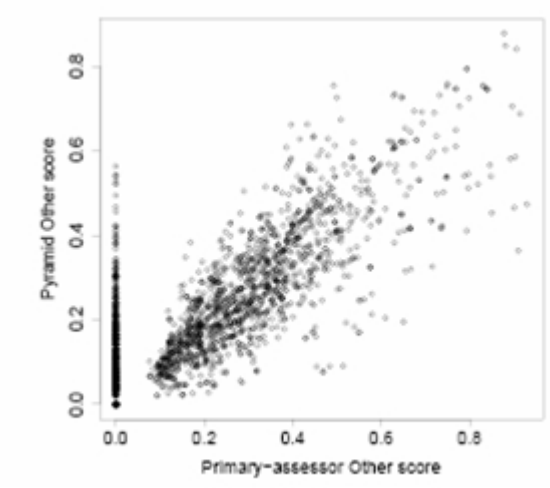


Abbildung 1.73: Vergleich Einzelperformance vs. Nugget Pyramid-Performance

- 100 Zeichen sind pro Vital/Non Vital Nugget erlaubt
- Falls kompletter Antwortstring unter 100 Zeichen ausgibt: - Precision = 1.0, sonst wird die Gleichung $1 - \frac{Laenge - Begrenzung}{Laenge} < 1$ angewendet.

1.9.4 Nugget-Pyramide

Das bisherige Scoring wird hier erweitert, aber es gibt ein Problem und zwar haben nur Vital Nuggets Einfluss auf den Recall-Wert. Bei den Antworten gibt es aber nur sehr wenige Vital Nuggets. Um dieses Problem zu lösen, entscheiden jetzt mehrere Prüfer ob ein Stichwort Vital oder „nur“ okay (non-Vital) ist.
Wir sehen in Abbildung 1.73, dass dort wo vorher bei einem Prüfer keine Antwort in der Liste gefunden wurde, nun aber mit Hilfe der Nugget-Pyramide mit einer 60%igen Wahrscheinlichkeit eine Antwort gefunden wird.

1.9.5 ciQA-Durchführung

Die Ziele dieser Durchführung sind zum einen, eine Ausgabe von komplexeren Antworten bzgl. einer Nutzer-Anfrage zu bekommen, wobei es sich nicht um eine Fakten Frage handeln soll und zum anderen sollen keine einfachen in einem Schritt erhaltenen Antworten ausgegeben werden, sondern Antworten die nach einem interaktiven Dialog mit dem Nutzer liefern. Das Konzept der möglichen Fragen innerhalb dieser Durchführung beschreibt, dass eine Entität in Beziehung zu einer anderen gesetzt werden soll. Die dabei entstehende Beziehungs-Frage setzt sich aus 2 Teilen zusammen:

1. Der Vorlage (Template) mit fixer Struktur und freien Slots
2. Einem Bericht mit der Intention der Frage

Die Bewertung dieser Fragen ist dieselbe wie bei den „Weiteren-Fragen“, aber mit einem höheren (Zeichen-) Grenzwert.

→ *Vorlage mit Intention*

Template: What evidence is there for transport of [drugs] from [Mexico] to [the U.S.]?
Narrative: The analyst would like to know of efforts to curtail the transport of drugs from Mexico to the U.S. Specifically, the analyst would like to know of the success of the efforts by local or international authorities.

→ *5 Vorlagen aus TREC 2006 ciQA*

What evidence is there for transport of [goods] from [entity] to [entity]?
Example: What evidence is there for transport of [drugs] from [Mexico] to [the U.S.]?

What [relationship] exist between [entity] and [entity]?
(where [relationship] ∈ {"financial relationships", "organizational ties", "familial ties", "common interests"})
Example: What [financial relationships] exist between [drug companies] and [universities]?

What influence/effect do(es) [entity] have on/in [entity]?
Example: What effect does [aspirin] have on [coronary heart disease]?

What is the position of [entity] with respect to [issue]?
Example: What is the position of [John McCain] with respect to [the Moral Majority or the Christian Coalition]?

Is there evidence to support the involvement of [entity] in [event/entity]?
Example: Is there evidence to support the involvement of [China] in [human organ transplants from Chinese prisoners]?

Abbildung 1.74: TREC2006 liefert uns 5 Vorlagen mit der gearbeitet werden kann

1.9.6 Interaktion mit Nutzer

Damit die Interaktion mit dem Nutzer ablaufen kann, muss ein Framework entwickelt, dass die Interaktion mit dem QA-System erlaubt. Bis 2007 konnten auch lokale Applikation wie Java verwendet werden, jetzt aber ist die Interaktion webbasiert. Die Nutzung des Systems soll für jedes Thema nicht mehr als 5 Minuten in Anspruch nehmen.

Beispielhafter Ablauf des Experimentes - Jeder Teilnehmer übermittelt 2 Initialläufe und die URL's an das NIST.
- Die Prüfer haben nun 3 Tage Zeit um mit jedem System zu interagieren. Die Daten stehen den Teilnehmern sofort zur Verfügung.
- Die Teilnehmer haben danach 2 Wochen Zeit um die Endresultate zu bestimmen

1.9.7 Ergebnisse

Um die Ergebnisse der ciQA2007 mit 7 Teilnehmern und 12 Läufen auszuwerten werden die Baseline und die Endresultate der Teilnehmer betrachtet. Baseline ist dabei die Anfrage an Lucene, welche mir die Top20 Antworten liefert. Diese Antworten werden erst dann Ausgewertet wenn keine Stoppwörter mehr enthalten sind, es wird also wird eine Tokenisierung und eine Entfernung der Stoppwörter durchgeführt. Wir sehen, dass die Baseline in manchen fällen sogar besser ist, als die Endresultate der Teilnehmer. Daraus lässt sich schließen das manuelle Abläufe besser sind als komplett automatisch generierte.

Organization	Type	Run tags		Pyramid F-Score	
		Baseline	Final	Baseline	Final
Michigan State U.	automatic	MSUciQAiHeu	MSUciQAfCol	0.359	0.361
Michigan State U.	automatic	MSUciQAiHeu	MSUciQAfInt	0.359	0.370
RMIT	automatic	rmitrun2	rmitrun5	0.361	0.343
RMIT	automatic	rmitrun2	rmitrun6	0.361	0.333
U. Mass	automatic	UMassBaseAut	UMassIntA	0.318	0.347
U. Mass	manual	UMassBaseAut	UMassIntM	0.318	0.503
U. Maryland	automatic	UMD07iMASCa	UMD07iMASCb	0.182	0.156
U. Maryland	automatic	UMD07MMRa	UMD07MMRb	0.333	0.334
U. NC and Yahoo!	automatic	UNCYABL30	UNCYAEX2	0.062	0.374
U. Strathclyde	manual	sicka	sicka2	0.410	0.394
U. Waterloo	manual	UWinitWIKI	UWfinalMAN	0.388	0.386
U. Waterloo	automatic	UWinitWIKI	UWfinalWIKI	0.388	0.380
baseline	automatic	baseA	baseB	0.327	0.327

Abbildung 1.75: Ergebnisse TREC 2007 QA-Track

1.10 Dictionaries

1.10.1 Einleitung

Was sind Dictionaries

In dem Umgang mit Information kann es vorkommen, dass einige Information sich im Laufe der Zeit nicht ändert, also konstant bleibt. Solche Informationen nennen wir sichere Informationen. Als Beispiel können wir das Datum des Eintretens Angela Merkels in die Rolle der Bundeskanzlerin anführen. 22.11.2005 ist dann die sichere Information. Unseres Ziel ist, diese geeignet zu speichern und auch den bestmöglichen Zugriff zu gewährleisten. Dazu benutzen wir Dictionaries. Dictionaries sind Listen für die Speicherung von solchen sicheren Information. Beispielsweise wäre eine Liste von Namen von Abgeordneten ein Dictionary. Ein bisschen komplizierter wird es, wenn man ganze Sätze in Dictionaries speichern will. Wenn man auf die Idee kommt, die wie eben zu speichern, also sprich eine große Liste von Sätzen, dann muss man damit rechnen, dass eine Anfrage an die Dictionaries nicht so schnell läuft, sogar sehr langsam. Eine bessere Idee wäre, einen Satz zu zerteilen und Verben und Adjektive als Relationen bzw. Events gelten zu lassen. Schauen wir uns am besten das folgende Beispiel genau an: Sichere Informationen: Merkel ist Bundeskanzlerin. Teilen wird in zwei Teile

- Merkel
- Bundeskanzlerin

Die Relation zwischen Merkel und Bundeskanzlerin : *ist* Also, man hat hier zwei Listen. Eine Liste mit Namen und eine mit Ämter. Was natürlich fehlt ist die Verbindung dazwischen. Wie oben beschrieben, definieren wir eine Funktion zwischen den Listen.

Dictionaries für unsere PG

Für unsere Zwecke in der PG brauchen wir folgende Dictionaries:

- Datum
- Drucksache
- Name
- Person
- Partei
- Reaktion
- Ort
- Ergebnis (Abstimmung)
- Amt/Rolle

Diese 10 Listen sind NERs, auf die wir uns geeignet haben. Dazu kommen noch:

- Wahlperiode
- Gehalt
- Religion
- Anzahl Kinder
- Sternzeichen
- usw.

Bei den komplizierten Sätzen funktioniert das Speicher genau so. Z.B.: „Gerhard Schröder kommt aus Berlin und hat gegen Angela Merkel bei der Bundeskanzlerwahl gestimmt, obwohl er mit ihr befreundet ist“. Wir haben hier zwei Listen und drei Relation.
Listen:

- Name
- Ort

Relationen:

- befreundet
- stimmt gegen
- kommt aus

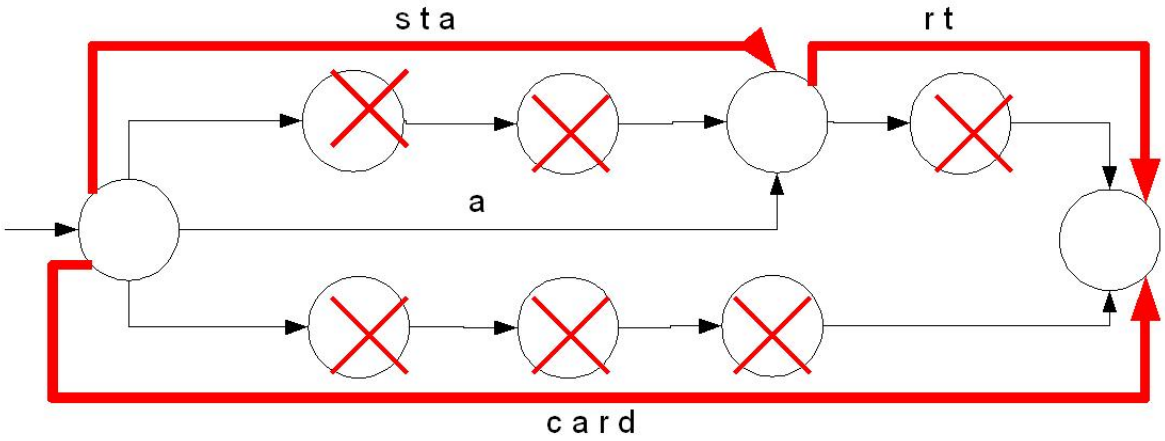


Abbildung 1.76: Automat mit Transition Jamming

Methoden zum Halten von Dictionaries

Jetzt werden wir uns damit beschäftigen, das Dictionary zu speichern und einen Eintrag hinzuzufügen. Stellen wir uns das an dem Beispiel „Schröder kommt aus Berlin“ da. Also man möchte den oben genannten Satz in das Dictionary einfügen. Dafür nehmen wir zwei Listen, die erste mit Namen und die zweite mit Orten. In der Liste Name und Ort wird entsprechend Schröder bzw. Berlin hinzugefügt und der entsprechender Index i und j markiert. Dann wird der Hashwert h von dem Namen unter Berücksichtigung der Relationen berechnet. Dann wird an der h-ten Stelle in der Zwischentabelle der Index von Berlin, also j gespeichert. Es ist leicht nachzuvollziehen, dass die Suche im Dictionary analog läuft. Also das Aussuchen von „Woher kommt Herr Schröder?“ wird wie folgt berechnet. Der Hashwert h von Schröder wird unter Berücksichtigung der Relation berechnet. In der h-ten Stelle in der Zwischentabelle wird der j-Wert ausgelesen, welcher an den Index von Berlin werweist. Dies ist die erste Möglichkeit, also mit Hashtabellen zu arbeiten. Eine andere Möglichkeit ist, die Wörter in der Hashtabelle nicht als eine Liste von Wörtern zu speichern, sondern durch einen Automaten zu realisieren. Diese Darstellung hilft uns die gleichen Buchstaben nicht doppelt zu speichern.
Rot gekennzeichnet sind die Buchstaben, die in Wörtern start und art vorkommen, deswegen brauchen wir die nicht doppelt speichern. Die Automaten helfen uns den Speicherplatz zu verringern und der Automat kann gleichzeitig eine Hashfunktion realisieren. Wir können den Automaten noch optimieren, so dass wir mit weniger Zuständen auskommen, in dem wir im Bild noch alle die Zustände löschen, die genau eine ein- und ausgehende Kante haben, weil die Zustände keine Information beinhalten. Dieses Verfahren nennt sich Transition Jamming. Das Bild zeigt die gesamte Struktur noch mal an.

Beispiele

Für umfangreiche Dictionaries gibt es in der Praxis bereits diversa Beispiele. Eine sehr bekannte Klasse der Dictionaries bilden die sogenannten „Gazetteers“. Dabei handelt es sich um Datenbanken, in denen Informationen zu geografischen Koordinaten gespeichert werden. Diese Daten reichen von Ausdehnung, über Name des Ortes auf der Erde, bis zu spezifischen

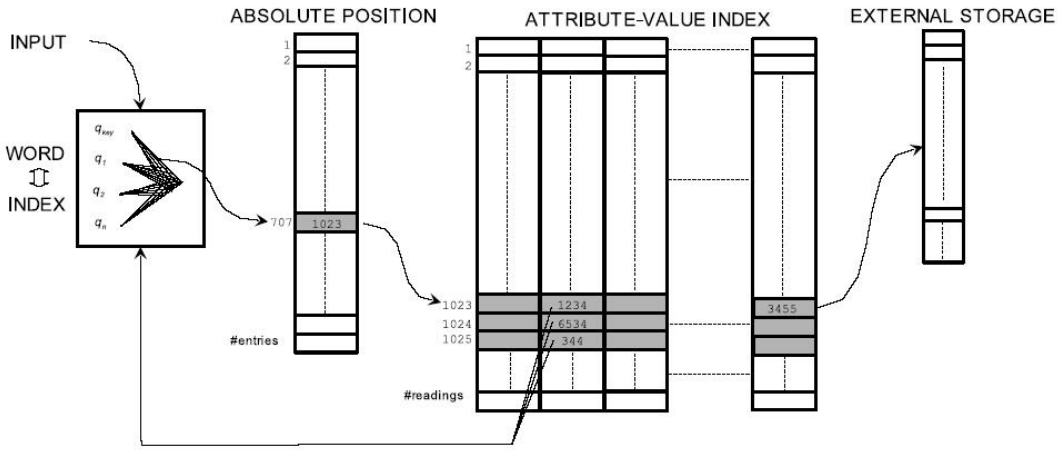


Abbildung 1.77: Gesamte Struktur

Attributen wie Bevölkerungszahl. Der bekannteste Gazetteer ist der ADL (Alexandria Digital Library), der auch über ein öffentlich zugängliches WebService Interface verfügt. Weitere nennenswerte Vertreter sind z.B. die „Metacarta GazDB“, in der Daten mehrerer Gazetteers kumuliert und in einem relationalen Datenbankschema abgelegt werden, mit spezieller Optimierung auf schnelle Aktualisierbarkeit und Durchsuchbarkeit.
Eine weitere Klasse von Dictionary ähnlichen Strukturen bilden die Nom- bzw. Propbanks, die an einer anderen Stelle dieses Dokuments behandelt werden.

Quellen für Dictionaries

Zum initialen Befüllen von Dictionaries müssen geeignete Datenquellen gefunden werden. Diese Datenquellen sollten einen möglichst breiten und verlässlichen Datenbestand bieten. In vielen Arbeiten hat sich die freie Enzyklopädie „Wikipedia“ als geeignet erwiesen, da dort bereits viele Dictionary spezifische Strukturen realisiert sind. Zum einen ist Wikipedia geeignet, um Mehrdeutigkeiten aufzulösen, also erlaubt bei Suchanfragen die Entscheidung welches, der potentiell vielen Suchergebnisse gemeint war. Zum anderen bietet Wikipedia eine Reihe von Daten bereits vorstukturiert an. Wie zum Beispiel alle Eckdaten einer Person oder eines Ortes.
Denkbar sind natürlich auch andere Quellen, jedoch ist eine gewisse Vorstukturierung sehr vorteilhaft.

Einsatzgebiete

In der NER können Dictionaries in verschiedenen Kontexten eingesetzt werden. Die bekanntesten Beispiele sind Wortfilter, wie Stop-Word Listen, oder Listen von Abkürzungen. Auch vorgefertigte Listen von Nomen sind denkbar, zum Beispiel um Mehrdeutigkeiten von groß

Abbildung 1.78: Auflösung von Mehrdeutigkeiten

geschriebenen Worten am Satzanfang aufzulösen. Generell lassen sich drei Klassen von Dictionaries bilden:

- Allgemeine Wortlisten (z.B. Stop-Words)
- Spezifische Wortlisten (z.B. bereits bekannte Entities)
- Trigger Wortlisten (z.B. GmbH, AG als Trigger für Firmen)

Dictionaries dieser Typen lassen sich auch innerhalb eines NER Vorgangs kombinieren und sogar kaskadieren, um die Qualität der gewonnenen Daten zu verbessern. Meist werden dabei die Allgemeinen Wortlisten in frühen Schritten, oder sogar im Preprocessing der Daten angewandt. In folgenden Schritten werden dann sichere Treffer erkannt. Zuletzt werden mit Hilfe von Triggern neue Entities entdeckt. Auch eine logische Verknüpfung mehrerer Dictionaries ist denkbar. So könnte man Beispielsweise ein Wort, das in der Liste von Personennamen enthalten ist, aber nicht in der Liste häufiger Nomen und weiterhin von einem „Herr“ eingeleitet wird, sehr sicher als einen Eigennamen identifizieren. Bei logischer Negation in einer solchen Verknüpfung spricht man auch von einem „Anti-Dictionary“.
Hier eine beispielhafte Aufstellung der verschiedenen Typen:

Optimierung

Es gibt diverse Techniken, um den Einsatz von Dictionaries zu optimieren. Da es sich bei Dictionaries um Wortlisten mit fester Schreibweise handelt, ist ein Lookup meist nur bei Einhaltung der exakt gleichen Schreibweise erfolgreich. Um diesen Effekt zu minimieren gibt

Abbildung 1.79: Personendaten

Table 2: List lookup features.

Features	Examples
General list	- General dictionary (see section 3.2.1) - Stop words (function words) - Capitalized nouns (e.g., January, Monday) - Common abbreviations
List of entities	- Organization, government, airline, educational - First name, last name, celebrity - Astral body, continent, country, state, city
List of entity cues	- Typical words in organization (see 3.2.2) - Person title, name prefix, post-nominal letters - Location typical word, cardinal point

Abbildung 1.80: Typen von Dictionaries

benjamin brown smith	benjamin-brown-s.	b. brown s.	bbs
benjamin-brown smith	benjamin-b. s.	b. b. smith	bs
benjamin brown-smith	benjamin-smith	b. brown-s.	
benjamin-brown-smith	benjamin smith	benjamin	
benjamin brown s.	b. brown smith	b. smith	
benjamin-b. smith	benjamin b. s.	b. b. s.	
benjamin b. smith	b. brown-smith	b. s.	
benjamin brown-s.	benjamin-s.	brown	
benjamin-brown s.	benjamin s.		

Figure 5
Names variants created from the name "Benjamin Brown Smith"

Abbildung 1.81: Reverse Stemming

es Techniken wie „stemming“, wobei die Eingabedaten in eine normalisierte Schreibweise überführt werden, aber auch Techniken, die man als „reverse-stemming“ bezeichnen könnte. Dabei werden z.B. alle Schreibweisen eines Namens gebildet, um weitere Vorkommen desselben Namens im Text finden zu können. Ein Beispiel hierfür sieht folgendermaßen aus:
Eine weitere Möglichkeit ist ein Lookup über ein Ähnlichkeitsmaß, satt über einen direkten String-Vergleich. Dabei wird für ein Eingabewort die Ähnlichkeit mit jedem Wort im Dictionary gebildet. Wird dabei ein bistimmter Schwellenwert der Ähnlichkeit erreicht, wird angenommen, dass dieses Wort in die Wortklasse des Dictionaries einzuordnen ist. Diese Methode hat leider einen sehr hoher Rechenaufwand zur Folge und erhöht das „noise“ in den Daten, also die Anzahl der Falschtreffer.

Fazit

Dictionaries können an vielen Stellen der NER eingesetzt werden und können dazu beitragen das Ergebnis der NER deutlich zu verbessern. So sind Vorteile in der Geschwindigkeit, so wie in der Verlässlichkeit der Treffer möglich. Allerdings stehen dieser Verbesserung auch Nachteile wie hoher Speicherbedarf und die Abhängigkeit von externen Datenquellen gegenüber. So muss dafür gesorgt werden, dass die Daten verlässlich sind und stets aktuell bleiben. Es bleibt also abzuwägen ob und in welchem Umfang der Einsatz von Dictionaries Sinn macht.

1.11 Frageformen

1.11.1 Ergänzungsfrage

Wird gestellt, um Auskunft über Personen, Sachen oder die Umstände eines Geschehens oder Zustandes zu erhalten um dadurch eine Wissenslücke zu schließen. Dabei kann es sich um Umstände wie Ort, Zeit, Ursache, Art und Weise oder Zweck handeln.
Merkmale sind Fragewörter am Satzanfang:
Beispiele:

- Wer, welcher, welche, welches? (Fragen nach dem Subjekt)
- Wen? (Fragen nach dem direkten Objekt)
- Wem? (Fragen nach dem indirekten Objekt)
- Wessen? (Fragen nach dem Attribut oder einem indirekten Objekt)
- Wo, wohin, woher? (Fragen nach dem Ort oder der Richtung)
- Wann, wie lange, wie oft? (Fragen nach der Zeit)
- Wie? (Fragen nach der Art und Weise)
- Weshalb, warum, weswegen, wieso? (Fragen nach der Ursache)

Beispiele für Frageaufbau:

Fragewort	erwartete Antwort	Beispiel Frage
wann	am[Datum]	Wann war das letzte Misstrauensvotum?
wie	[String]	Wie ist die Reaktion von..?
		Wie oft sind Zwischenrufe bei Reden von ..?
wer	am[Datum],[String]	Wer ist der Parteivorsitzender der ..?
welche	[String]	Welche Parteien sind im Bundestag?
welcher	[String]	Welcher Partei gehört .. an?
wird	[String]	...
wo	[String]	Wo wird die Bundeswehr zurzeit eingesetzt?
wieso	[String]	[Event Extraction]
		Wieso ist Deutschland gegen den EU-Beitritt der Türkei?
warum	[String]	[Event Extraction]
		Warum ist die BRD gegen Türkei Beitritt?
weshalb	[String]	[Event Extraction]
		Weshalb...?

Diesen Typ von Fragen behandeln wir in unserer PG. Die Fragetypen die hier aufgezeigt werden, sind zum einen „Stichwort-Fragen“ und zum anderen „Ereigniss-Fragen“, wobei letzteres noch nicht behandelt wurde.

1.11.2 Entscheidungsfragen

Sind Fragen, die nur mit ja oder nein beantwortet werden können.
Merkmal: Benötigt keine Fragewörter aber beginnen mit der finiten (gebeugten) Verbform.
Beispiele:

- hat, hast, haben
- kommt, kann, können
- sind, ist
- gibt, besitz
- werden, wurden, wurde

→ erweiterbar

Beispiele für Frageaufbau:

Fragewort	erwartete Antwort	Beispiel Frage
hat	[String]	Hat der Abgeordnete .. Kinder?
ist	[String]	Ist der Abgeordnete .. verheiratet?
sind	[String]	Sind...
gibt	[String]	Gibt es Abgeordnete mit mehr als 3 Kindern?
wurden	[String]	Wurden die Diäten in der 16 Wperiode erhöht?
werden	[String]	Werden Anträge von ..abgelehnt?
wurde	[String]	Wurde ein Misstrauensvotum in der 14.Wahlperiode gestellt?
besitzt	[String]	Besitzt der Abgeordnete .. einen Nebenjob?

Hier liefern wir die Antwort und dazu kommt eventuell eine Liste mit Namen oder nur ein Integerwert.

1.11.3 Direkte Fragen

Enden im Deutschen immer mit dem Fragezeichen ? „ Im Grunde ist dies eine Frage, welche sich mit der Hilfe der ersten beiden Frageformen beantworten lässt“.
Beispiele: siehe den ersten beiden Frageformen

1.11.4 Indirekte Frage

Sind Fragen die auch indirekt gestellt werden können, ohne die grammatische Form einer Frage zu haben.
Beispiele: siehe den ersten beiden Frageformen aber hier kann vor der Frage etwas anderes stehen wie z.B. Ich will wissen, wann....

1.11.5 Offene Fragen

Der Fragesteller verknüpft seine Frage mit weitergehenden „Erkundungsabsichten“ und will damit bewirken, dass der Befragte ausführlich antwortet.
Merkmal: Beginnen gewöhnlich mit einem Fragewort. Also alle W-Fragen (was, wer, wie, usw.)
Beispiele für Frageaufbau:

Fragewort	erwartete Antwort	Beispiel Frage
was	am[Datum],[String]	Was hat Schröder im Bundestag am...gesagt?

→ weitere Beispiele siehe Ergänzungsfrage

1.11.6 geschlossenen Fragen

Merkmal: Diese Fragen haben ihre Bedeutung in Situationen, wo es auf präzise und eindeutige Antworten ankommt, beginnen gewöhnlich mit einem Verb.
Sehen wir wie offene Fragen und es wird eine Liste mit Antworten zurückgegeben.

1.11.7 weitere Fragen

Dann gibt es noch Fragen die als Antwort eine Liste von z.B. Namen zurückgeben.
Beispiele für Frageaufbau:

Fragewort	erwartete Antwort	Beispiel Frage
gib	[String]	gib alle Abeordneten mit Kindern aus!
liefere	[String]	suche alle weiblichen Abgeordneten der ...
in welchen	[String]	in welchen Ländern sind deutsche Soldaten Stationiert?

→ beliebig erweiterbar!
Für unsere PG haben wir uns entschlossen die Fragemöglichkeiten des Nutzer soweit es geht einzuschränken, da der Nutzer zu viele Fragestellungsmöglichkeiten hat. Zudem wird die Analyse und Auswertung unnötig erschwert, deshalb wir uns für einen anderen

Abbildung 1.82: Startseite des Deutschen Bundestages.

1.11.8 statistischen-Fragen

Merkmal:
Wir erwarten hier als Rückgabe z.B. eine Prozentzahl, Anzahl, ein Wert usw.

Fragewort	erwartete Antwort	Beispiel Frage
wieviele	[Integer]	Wieviele Anträge werden durchschnittlich abgelehnt?
wieviel	[Integer]	Wieviel Anträge werden durchschnittlich abgelehnt?

→ erweiterbar!

1.12 Anwendungsbereich

1.12.1 Aufbau und Themen

Der Deutsche Bundestag

Der Deutsche Bundestag ist das oberste demokratische Staatsorgan in Deutschland. Es ist das Zentrum des politischen Lebens. Der Arbeitsalltag des Deutschen Bundestages umfasst die öffentlich geführten, politischen Debatten, die Arbeit an Gesetzentwürfen, Anträgen und Anfragen. Sowie die Beratungen der Fachpolitiker in den Ausschüssen und politische Arbeit in den Fraktionen.

Der Deutsche Bundestag als Organ der Legislative hält folgende Aufgaben und Funktionen inne:

- Gesetzgebung
- Bundeshaushalt

- Kontrolle der Regierung
- Wahl des Bundeskanzlers
- Rechtliche Grundlagen

Die Webseiten des Deutschen Bundestages können über das Portal unter der URL: www.bundestag.de erreicht werden. Die Verantwortung für das Internet Angebot trägt das Referat Online-Dienste und Parlamentsfernsehen (Referat PuK4). Die HTML Webseiten werden mit dem Content-Management-System: Infopark CMS Fiona generiert. Das Portal wird im Serverpark der Babiel GmbH betrieben.

Das Portal stellt neben Texten und Bildern aus dem Betrieb des Dt. Bundestages auch Dokumente und Audio/Video Streams zur Verfügung.

Sitemap

Das Portal des Deutschen Bundestages weist folgende Struktur auf.

- Startseite: Willkommensseite des Portals, die sowohl die Navigation, einen Veranstaltungskalender und die aktuellen Newsbeiträge darstellt.
- Sitemap: Gesamtstruktur des Webangebots.
- Architektur und Kunst: Einen Abschnitt, der sich mit Bauwerken, Künstlern und Kunstwerken rund um den Dt. Bundestag und das Parlamentsviertel beschäftigt.
- Aktuell: Pressemitteilungen, Meldungen, Newsletter, RSS
- Parlament: Überblick über Funktion und Aufgaben, sowie Informationen über das Präsidium, Gremien, Fraktionen und Wahlen.
- Abgeordnete: Eine Fülle aus Informationen, Zahlen und Listen über die Abgeordneten des Bundestages seit der 13. Wahlperiode.
- Ausschüsse: Liste und Funktion der jeweiligen Ausschüsse.
- Wehrbeauftragter: Aufgaben, Funktion und Befugnisse des Wehrbeauftragten.
- Dokumente: Die unvollständige Informationsdatenbank zu Drucksachen, Protokollen, Analysen und Gutachten des Bundestages. In diesem Abschnitt können zusätzlich Dokumente zum Stand der Gesetzgebung, Informationen über die Parteienfinanzierung, sowie das Datenhandbuch des Dt. Bundestages eingeholt werden. Das Datenhandbuch ist ein Sammelsurium statistischer Daten über den Bundestag, allerdings nur bis zur letzten Wahlperiode.
- Wissen: Dieser Abschnitt bietet eine große Anzahl an begleitenden Informationen die zum besseren Verstehen der Vorgänge unabdingbar sind. Neben weiterführenden Links, Symbolen des Staates und der Bibliothek, befinden sich hier Wissenschaftliche Dienste, ein Abkürzungsverzeichnis und ein alphabetisch sortiertes Glossar.

- Live: Multimediales Archiv der Fernsehaufzeichnungen und gleichzeitig das Parlamentsfernsehen Web-TV
- Europa und Internationales: Informationen rund um die Rolle Deutschlands in Europa und das Europäische Parlament.
- Service: Kontaktmöglichkeiten, Formulare, Anträge und Stellenangebote des Dt. Bundestages. Hier befinden sich auch einige Dienste, z.B. die Volltextsuche, die nur auf das Internetangebot des Bundestages beschränkt sind.
- Publikationen, Jugend, Ausstellungen, Parlamentspreise
- Geschichte: Chronik des Parlaments und ein Abriss über Dt. Parlamentarismus.
- Blickpunkt Bundestag: Überblick über die aktuellen Themen des Bundestages sowie ein Rückblick auf die Jahre 1998 - 2004.

1.12.2 IR relevante Dienste

Das Portal des Dt. Bundestages stellt einige Dienste zur Verfügung, welche für die Lösung unserer Aufgabe im Bereich Information Retrieval interessant sind. Aus den bereitgestellten Diensten sollen Daten gewonnen werden und in Datenbanken gespeichert werden. Die Daten sollen von dem System zur Erstellung des Dossiers verarbeitet werden. Da es sich bei dem Portal des Deutschen Bundestages um ein HTTP Internet Server handelt, werden grundsätzlich nur zwei Methoden benötigt um Dienste anzusprechen.

- HTTP GET - Einer gewöhnlichen Anfrage an den Webserver, um an Hand der URL, oder der in der URL angegebenen Parameter ein HTML Dokument oder eine Datei zu erhalten. Die überwiegende Mehrheit der Dienste kann so angesprochen werden.
- HTTP POST - Einer in einem HTML Formular eingebetteten Anfrage. Hierbei müssen die Parameter der Anfrage in den Formularfeldern übergeben werden. Zu diesem Zweck müsste im Vorfeld ein Robot geschrieben werden, der die HTML Dokumente, die Formulare enthalten, mit den entsprechenden Parametern versieht und an den Server, wie ein gewöhnlicher Browser zurücksendet. Glücklicherweise ist diese Vorgehensweise für unsere Zwecke nicht relevant, diese Formulare liefern als Ausgabe die selben Dokumente, die genauso über die HTTP GET Anfragen erreichbar sind.

WWW

Der WWW Dienst des Webservers liefert aus dem CMS erzeugte HTML Dokumente. Diese beinhalten Informationen in Textform, welche, üblich für HTML, innerhalb der Tags der Dokumentstruktur eingeschlossen sind. Um Daten aus den HTML Dokumenten zu extrahieren,

müssen die relevanten Passagen mit Hilfe von regulären Ausdrücken in der Dokumentstruktur und innerhalb des Textes gefunden und ausgelesen werden. Das relevante Datum wird an Hand der es umgebenden Wörter erkannt. Abbildung 1.2 zeigt eine beispielhafte Markierung im Abgeordnetenprofil.

Im Rahmen des Information Retrieval können zum Aufbau der Datenbanken folgende Bereiche des Portals als Quellen betrachtet werden:

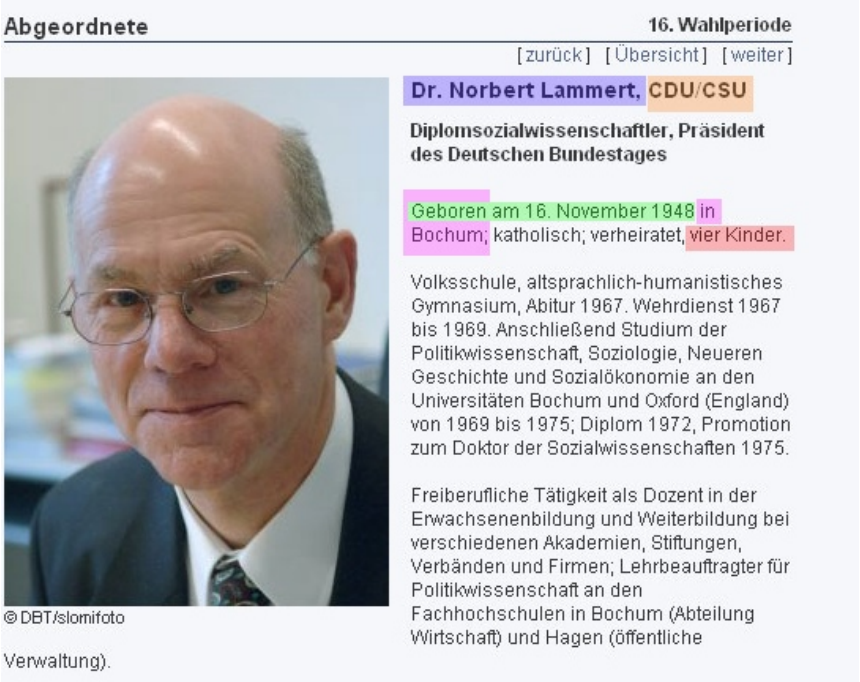


Abbildung 1.83: Markierung von relevanten Daten im Profil eines Abgeordneten.

- Abgeordnete Liefert eine Fülle von Informationen und statistischen Daten über die Abgeordneten des Bundestages. Neben Biographien sind diverse statistische Auflistungen nach verschiedenen Kriterien und die Offenlegung der Nebentätigkeiten, interessante Ansatzpunkte zum Aufbau eines Dossiers.
- Ausschüsse Beinhaltet die vollständige Auflistung der aktuellen Ausschüsse des Deutschen Bundestages. Informationen über Aufgaben und Funktionen einzelner Ausschüsse können hier extrahiert werden. Allerdings dienen diese nur der Beantwortung von Fragen, die direkt Funktion oder Aufgabe eines Ausschusses betreffen. Die Auflistung der Mitglieder eines Ausschusses kann auch aus den Profilen der Abgeordneten aufgebaut werden. Dieser Bereich ist daher rein als optional zu betrachten.

Abbildung 1.84: Der Bereich: Abgeordnete

Abbildung 1.85: Mögliche Abfragen des DIP (8-15 Wahlperiode)

- Parlament Hier liegt das Hauptaugenmerk auf den Rubriken Gremien, Wahlen und Verwaltung. Die Relevanz dieses Bereiches gestaltet sich ähnlich wie im Vorherigen. Es bedarf spezifisch formulierter Fragen zu den Themen, damit man sich überhaupt mit diesem Bereich beschäftigen muss.
- Glossar und Abkürzungsverzeichnis Im Bereich Service befindet sich das elektronische Nachschlagewerk des Parlaments. Schlagwortregister, Abkürzungsverzeichnis und Glossar beinhalten Fachbegriffe und deren Erklärung. Diese Quellen können zur Erstellung einer Terminologie der parlamentarischen und politischen Begriffe genutzt werden.

Sach- und Sprechregister : DIP

Die Sach- und Sprechregister des Deutschen Bundestages bestehen aus zwei Datenbanken. Hinter der Abkürzung DIP verbirgt sich das „Dokumentations- und Informationssystem für parlamentarische Vorgänge“ des Deutschen Bundestages. Das System ist in zwei Versionen verfügbar. Der Ersten, über dip.bundestag.de erreichbaren, älteren Version, die teilweise unvollständig Dokumente der Wahlperioden 8-15 beinhaltet. Der zweiten, neuen Version, die über dip21.bundestag.de erreichbar ist und Dokumente der 16-ten Wahlperiode beinhaltet.

Beide Datenbanken führen die Drucksachen und Plenarprotokolle im pdf-Format auf. Dokumente, älter als die 14-te Wahlperiode, sind entweder nur als Text Dateien oder in pdf eingebettete Fotokopien vorhanden. Die aktuellen Plenarprotokolle sind über die Webseite im gleichnamigen Bereich abrufbar und werden schon während der laufenden Plenarsitzung als vorläufige Protokolle eingestellt und später in die Datenbank eingepflegt. In der älteren Version des DIP existiert ein zusätzlicher separater Bereich zum Stand der Gesetzgebung. Dieser zeigt an, in welcher Stufe des Prozesses der Gesetzgebung sich ein Prozess aktuell befindet. Letztgenannte Funktion ist in der neuen DIP21 Version nicht mehr vorhanden und wurde in die normale Suche integriert.

Die Dokumente beider Datenbanken können direkt über die URL erreicht werden. Die Datenbanken weisen eine sehr einfache, nach Dokumenttyp und Wahlperiode aufgebaute Verzeichnisstruktur auf. Dies erleichtert das Crawling und den Download der Dokumente, so dass sich die aufwändige Herstellung eines Robots, der die Formulare beider Informationssysteme bedient, erübrigt.

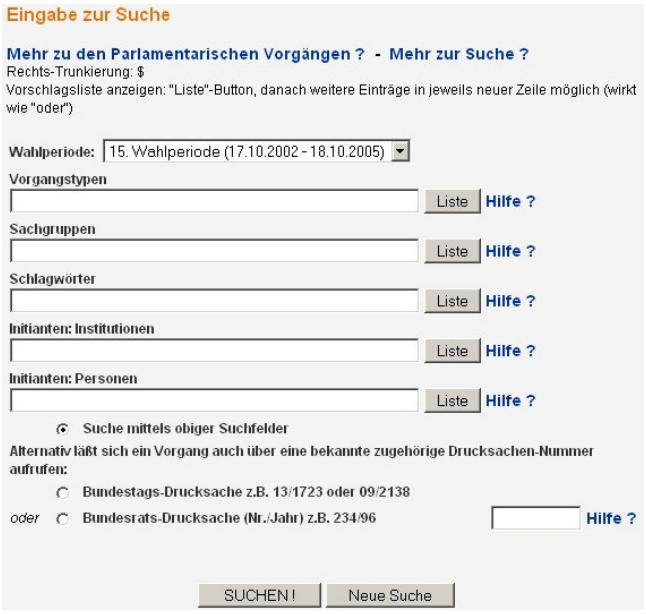


Abbildung 1.86: Formular zur Suche innerhalb der Vorgänge

Die möglichen Abfragen auf der Datenbank im älteren DIP überschneiden sich auf den Dokumenten. Wie wir später in der Analyse der Dokumente sehen werden, existieren zwei grobe Gruppen von Dokumenten, die eigentlichen Dokumente und die Plenarprotokolle. Die vier in Abbildung 1.4 dargestellten Abfrage-Möglichkeiten führen zu separaten Formularen, die wiederum eine Linkliste der relevanten Dokumente zurückliefern.

- Stand der Gesetzgebung verweist auf alle Drucksachen die von einem Antrag ausgehen und sich im Prozess der Gesetzgebung befinden. Plenarprotokolle mit Redebeiträgen und Abstimmungen zu einem Antrag sind hierbei nicht eingeschlossen.
- Parlamentarische Vorgänge im Bundestag/Bundesrat verweist auf alle Vorgänge, durchsucht also sowohl Protokolle, wie auch Drucksachen, allerdings sind die Inhalte von Fragestunden nicht indexiert. Über Parlamentarische Vorgänge kann man also Anträge, Redebeiträge in Beratungen und Anfragen finden.
- Parlamentarische Aktivitäten von Personen, hierbei werden Anträge, Anfragen aus Drucksachen und Redebeiträge aus den Protokollen berücksichtigt.
- Fragen für die Fragestunde, durchsucht die eingegangenen Drucksachen nach Fragen für eine Fragestunde und ordnet diese dem entsprechendem Protokoll zu.

Das ältere System arbeitet auf mehreren Sets aus Meta-Daten, die für jede Anfrage-Möglichkeit eine andere Belegung brauchen. Es drängt sich die Vermutung auf, dass es sich hierbei um vordefinierte Views auf eine SQL Datenbank handelt. Die Abfrage-Möglichkeiten greifen also auf den gesamten Pool der pdf-Dokumente zu, die Views schränken die Auswahl auf bestimmte Dokumenten Typen ein, die über ganz bestimmte Meta-Daten verfügen. Dies ist besonders deutlich durch die Unterteilung der Plenarprotokolle auf diejenigen, die Fragestunden enthalten und alle übrigen, in denen teils Abstimmungen und Redebeiträge vorhanden

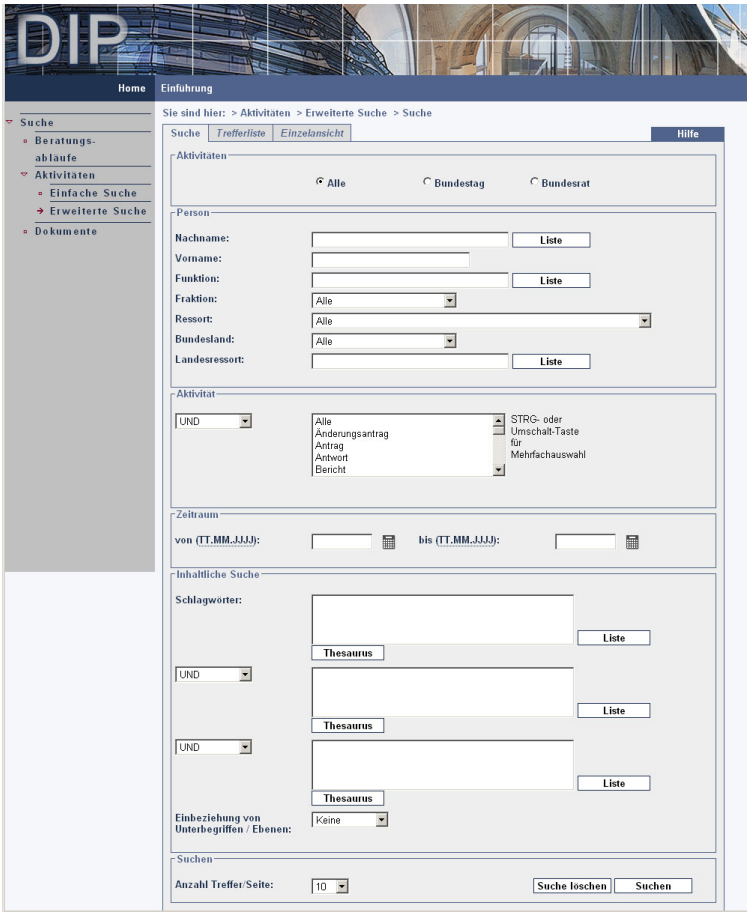


Abbildung 1.87: Erweiterte Suche in den Aktivitäten im DIP21

sind. Sowohl die Suche auf den Parlamentarischen Vorgängen wie auch die Suche auf den Aktivitäten können auf das selbe Protokoll verweisen, wenn der Redner (Suchkriterium für einen Vorgang) gleichzeitig eine Frage in der Fragestunde stellte.

Das neuere DIP21 stellt grundsätzlich drei Abfrage-Möglichkeiten zur Verfügung.

- Nach Themengebieten sortierte Beratungsabläufe in der Gesetzgebung ersetzen die Vorgänge und die GESTA der Vorgängerin.
- Die Aktivitäten, also eine Übersicht aller Aktivitäten im Bundestag.
- Eine umfangreiche Suche über allen Dokumenten des Bundestages.

Es wird wie in der vorherigen Version der gesamte Dokumenten Pool durchsucht, jede Abfrage-Möglichkeit steht mit zwei Formularen zur Verfügung, der Einfachen und Erweiterten Suche.

Bis auf die Zusammenfassung der Abfrage-Möglichkeiten unterscheidet sich die DIP21 Suche in der Anzahl und Art der abgefragten Meta-Daten. Die neuere Version des DIP bietet zwar eine kleinere Anzahl an Möglichkeiten, auf der anderen Seite erhöht sie jedoch die Anzahl der Parameter, die der Suche über Formularfelder mitgegeben werden.

Abbildung 1.88: Der Bereich: Abgeordnete

Datenhandbuch

Das „Datenhandbuch zur Geschichte des Bundestages 1994-2003“ ist ein Dokument im pdf-Format, das Umfangreiche Registerrecherche und Volltextsuche in den Dokumenten des Dt. Bundestages aus der Zeitperiode erlaubt. Das Dokument beinhaltet die für pdf-Dokumente übliche Suchfunktion mit der nach Stichwörtern gesucht werden kann. Zusätzlich sind in den Sach- und Personenregistern die Informationen über Ausschüsse, Gremien und Abgeordnete, mit dem Stand bis 2003, gesammelt. Um diese Daten erfolgreich extrahieren zu können, muss das Dokument mit einem pdf-Parser ausgelesen und die relevanten Stellen mit Hilfe von regulären Ausdrücken ausgelesen werden.

Suche

Die Volltextsuche des Portals ist in zwei Formulare unterteilt. Zum einen gibt es einen Suchagenten, den Bundesadler, der hier nicht als Staatssymbol, sondern eher als Maskottchen des Portals dargestellt wird. Dieser Agent verarbeitet einen im Formular übermittelten Text und lenkt den Benutzer über eine Link-Sammlung auf Inhalte, die der Benutzer in seinem Text gemeint haben könnte. Zusätzlich gibt der Adler einen, manchmal nicht ganz schlauen Kommentar von sich. Für die Zwecke der PG ist der Adler-Agent nur im Sinne seiner technischen Realisierung spannend. Die durch Ihn bereitgestellte Suchfunktion ist zur Navigation innerhalb der Webseiten des Bundestages für die von der PG erstellte Software ohne belang.

Die eigentliche Suchfunktion des Portals ist eine gewöhnliche Stichwortsuche, die eine nach Relevanz sortierte Link-Liste liefert. Das auf der HTTP POST Methode basierende Formular bedarf eines im Vorfeld dargestellten Robots, der das Formular mit Stichwörtern versorgt und die Ergebnisse mit einem Crawler und einem auf regulären Ausdrücken basierten Parser abtastet. Wie schon bei dem vorherigen Suchformular ist die Relevanz für das Projekt zu gering, im Vergleich zu dem Aufwand den man betreiben müsste, um diesen Dienst zu nutzen. Zum einen gehen wir davon aus, dass die Webseiten des Deutschen Bundestages aus Gründen der Wiedererkennung und Konsistenz eine feste organisatorische Struktur behalten werden.

Daher bleiben die relevanten Bereiche unverändert. Der Einsatz eines CMS impliziert indirekt die Konsistenz der Daten. Zum Anderen wäre der Einsatz von generischen aber sehr mächtigen Parsern notwendig um die Ergebnisse der Suche zu verarbeiten. Der Einsatz von spezialisierten Parsern, die in einem eingeschränktem Bereich der Webseite eingesetzt werden, erscheint uns als sinnvoller.

Digitaler Bilderdienst

Der digitale Bilderdienst des Deutschen Bundestages stellt Aufnahmen der Sitzungen, Versammlungen, Abgeordneten und Gebäude der Öffentlichkeit zur Verfügung. Die Aufnahmen stehen als Dateien zum Download im jpg-Format bereit. Da sich diese PG nicht mit der Informationsextraktion von Daten aus digitalen Bildern beschäftigt, kann dieser Dienst, der strenggenommen nur ein weiterer Bereich innerhalb des WWW Dienstes ist, vernachlässigt werden.

Newsletter

Der Newsletter Dienst ist ein auf dem SMTP Protokoll basierter Dienst der an Abonnenten Emails in text-Form verschickt. Aktuelle Meldungen des Bundestages können hier abonniert werden. Dieser Dienst kann dazu genutzt werden, um brandaktuelle Themen des Bundestages zu speichern. Allerdings ist die Halbwertszeit dieser Informationen relativ eingeschränkt. Zum einem können die selben Informationen direkt aus dem WWW Dienst extrahiert werden, zum Anderen werden diese Daten mit einer leichten Verzögerung in den Protokollen und Dokumenten des Bundestages zu finden sein.

RSS

Das Abonnement des RSS Dienstes des Bundestages hat ähnliche Merkmale wie das Abonnement des Newsletters und birgt die selben Redundanzen. Strenggenommen ist der RSS Dienst auch nur ein WWW ähnlicher Dienst der XML Dokumente anstatt von HTML ausliefert. Der einzige Vorteil des RSS Feeds ist seine kompakte Form, die einheitliche Nutzung von HTTP GET Anfragen und wesentlich einfachere Dokumentenstruktur.

1.12.3 Vorgänge

Gesetzgebung

Der Vorgang der Gesetzgebung ist ein mehrstufiger Prozess, der mit entsprechenden Dokumenten des Bundestages begleitet wird und der folgende Schritte beinhaltet:

- Die Gesetzesinitiative geht in der Mehrheit der Fälle von der Bundesregierung aus. Manchmal wird sie auch vom Bundesrat oder aus der die Mitte des Bundestages, also von Abgeordneten, Parteien und Fraktionen eingebracht.
  - Der Prozess wird im Normalfall schriftlich initiiert und dem Bundestag in einer Rede vorgestellt.
  - Referenzen auf diesen Prozess sind in der GESTA und den Plenarprotokollen zu finden.

- Die Beratung und Beschlussempfehlung erfolgt auf die eingebrachte Gesetzesinitiative. Der Schritt wird meistens in mehreren Lesungen, maximal jedoch drei, abgehandelt. Als Ergebnis liegt dem Bundestag eine Beschlussempfehlung zur zur Verabschiedung vor oder sie wird den Ausschüssen zu weiteren Beratung vorgelegt, falls besondere, in den Ausschüssen vorhandene Expertise, benötigt wird.
  - Daten sowie die Ergebnisse dieses Schrittes sind in Protokollen und Dokumenten zu finden.
- Ein Gesetz wird in einer Sitzung durch den Bundestag in einer Abstimmung verabschiedet. Die Referenzen auf diesen Schritt befinden sich in den Protokollen und als Beschluss in den Dokumenten.
- Der Beschluss des Bundestages wird an den Bundesrat zur Verabschiedung weitergeleitet.
- Bundesrat behandelt den Gesetzesbeschluss in einer Sitzung. Sollte das Gesetz durch den Bundesrat in einer Abstimmung verabschiedet werden, wird es an den Bundespräsidenten weitergeleitet. Sonst wird es an den Vermittlungsausschuss zur Kompromissfindung weitergegeben.
- Gegenzeichnung durch die Bundesregierung ohne weitere Beratung oder Ankündigung im Bundestag nötig.
- Ausfertigung durch den Bundespräsidenten.
- Schließlich verkündet der Bundespräsident das verabschiedete Gesetz. Der Prozess ist damit abgeschlossen und das Gesetz tritt entweder unverzüglich oder zu einen festgelegten Zeitpunkt in Kraft.

Eine Abstimmung

Der mit einer Abstimmung verbundene Prozess ist recht einfach und kann als Unterprozess im Prozess der Gesetzgebung gesehen werden. Die zu einem Sachverhalt geforderte Abstimmung hat einen Initiator, einen die Abstimmung überwachenden Akteur und die Menge der Abstimmenden. Eine Abstimmung wird angekündigt, initiiert und ausgezählt. Das Ergebnis der Abstimmung wird bekanntgegeben.

Anträge

Anträge sind für gewöhnlich Gesetzesinitiativen aus der Mitte des Bundestages. Die Vorgänge dieser Form werden im Bundestag mit den folgenden Bezeichnungen in der im Verzeichnis der Drucksachen geführt:

- Antrag
- Änderungsantrag
- Entschließungsantrag

Beschlussempfehlung und Bericht

In einer Beschlussempfehlung begründen die Initiatoren eines Gesetzes ihr Vorhaben. Eine Beschlussempfehlung ist in vier Artikel gegliedert.

- A. Problem, in dem ein Missstand, welchen dieses Gesetz beheben soll, ausführlich skizziert und damit die Einleitung des Prozesses justifiziert wird.
- B. Lösung. Eine ausführliche Schilderung, wie das zuvor beschriebene Problem durch das zu beschließene Gesetz gelöst werden soll.
- C. Alternativen. Hier werden, meist kurz, die möglichen Alternativen zur Lösung angeboten.
- D. Kosten. Dieser Teil führt die vermuteten Kosten auf, die nach dem in Kraft treten des Gesetzes entstehen werden. Es werden meistens Kosten für die von dem Gesetz direkt betroffenen Personengruppen und Institutionen aufgeführt.

Mit dem Passus: „Der Bundestag wolle beschliessen:“ werden der eigentliche Gesetzestext oder die Änderungen an einem vorhandenen Gesetz eingeleitet.

Im Anhang des Dokuments befindet sich ein Bericht zu dem Beratungsprozess, der im Vorfeld stattgefunden hat. Ein an dem Prozess beteiligter Abgeordneter wird mit dem Bericht immer dann beauftragt, wenn die Beschlussempfehlung zur Beratung einem Ausschuss vorgelegt wurde. In dem Bericht werden vom Berichterstatter die Stellungnahmen aller mitarbeitenden Gremien und explizit der beteiligten Ausschüsse aufgeführt.

Übersicht

In einer Übersicht werden Beschlussempfehlungen über einen Zeitraum tabellarisch zusammengefasst. Eine Übersicht ist eine Roadmap über die Abstimmungen des Bundestages und beginnt mit dem Passus: „Der Bundestag wolle beschliessen:“

Gesetzesentwurf

Ein Gesetzesentwurf ist ein Dokument, mit dessen Veröffentlichung der Gesetzgebungsprozess des Bundestages abgeschlossen wird. Das Gesetz wird im nächsten Schritt dem Bundesrat zur Abstimmung vorgelegt. Neben den Initiatoren des Gesetzes und dem Auszug aus dem Entwurf in dem das Problem, die Lösung, die Alternativen und die Kosten kurz skizziert werden, enthält das Dokument den folgenden Passus „Der Bundestag hat das folgende Gesetz beschlossen:“. Darauf folgt der eigentliche Gesetzestext. Auf den letzten Seiten des Gesetzesentwurfs befindet sich eine ausführliche Begründung zu jedem Artikel des Gesetzes.

Anfragen

Eine Anfrage ist eine Sammlung schriftlich gestellter Fragen zu einem Themenbereich an die Bundesregierung. Eine Anfrage hat einen Initiator, also die Person, Partei oder Fraktion, welche diese Frage vorbringt und einen Antwortenden. Jede Anfrage bedarf einer schriftlichen Antwort.

- Große Anfrage ist ein an die Bundesregierung gestellter Fragenkatalog zu einem Problemthema. Die Fragen werden in Kategorien zu den Aspekten des Problemthemas gruppiert, was zu einer großen Anzahl an Fragen führt.
- Kleine Anfrage ist ebenfalls eine schriftlich formulierte Anfrage, die einer schriftlichen Antwort bedarf. Die kleine Anfrage betrifft einen kleineren Themenbereich zu dem nur wenige Fragen gestellt werden.
- Die Antwort der Bundesregierung ist eine Drucksache die auf eine kleine oder große Anfrage die entsprechenden Antworten liefert.

Unterrichtung

Eine Unterrichtung ist eine Sammlung bedeutsamer Erkenntnisse, die dem Bundestag zur Entscheidungsfindung mitgeteilt werden. Eine Institution, die den Bundestag über einen Sachverhalt oder den Verlauf einer Prüfung unterrichtet, gliedert ihre Ausführung in drei Teile: in den Vorbemerkungen werden der Zweck und Schwerpunkte der Unterrichtung dargestellt. In der Zusammenfassung werden die kurz gefassten Ergebnisse aufgelistet. Schließlich folgt die detailierte, in mehrere Kapitel aufgeteilte, Ausführung.

Fragen zu einer Fragestunde

In einer Sitzung des Deutschen Bundestages werden als Punkt der Tagesordnung Fragen von Abgeordneten beantwortet. Die Fragen werden als Dokument zusammengefasst und dem Bundestag vorgelegt. Der Adressat einer Frage beantwortet diese im Rahmen einer Fragestunde. Die im Vorfeld eingereichten Fragen sind also als Drucksache des Bundestages zu finden, die Antworten sind im Plenarprotokoll der Sitzung zu finden, in der diese Fragestunde stattfindet.

Wahlen und Wahlvorschläge

Die Abgeordneten einer Fraktion können schriftlich Wahlvorschläge für die zu besetzenden Posten und Ämter im Bundestag einreichen. Diese werden als Drucksachen mit der Vorgangsbezeichnung Wahlvorschlag geführt. Die darauf folgende Abstimmung findet in einer Sitzung des Bundestages statt und die Ergebnisse sind in dem entsprechenden Plenarprotokoll zu finden.

1.12.4 Dokumente

In diesem Abschnitt beschäftigen wir uns mit der Struktur und Form der Dokumente, die auf der Webseite des Bundestages zu finden sind. Aus diesen Daten sind im Verlauf der PG die für die Erfüllung der Aufgabenstellung notwendigen Datenbanken erstellt worden.

Die für die PG relevanten Dokumente stammen aus den Wahlperioden 14-16 und sind ausschliesslich im pdf-Format vorhanden.

Die Abbildung 1.89 zeigt das Beispiel einer Drucksache (die Antwort der Bundesregierung auf eine kleine Anfrage) mit entsprechenden Markierungen, welche den Zweck des Dokumentes verdeutlichen sollen. Diese Markierungen wurden von einem Menschen angebracht, sollen aber

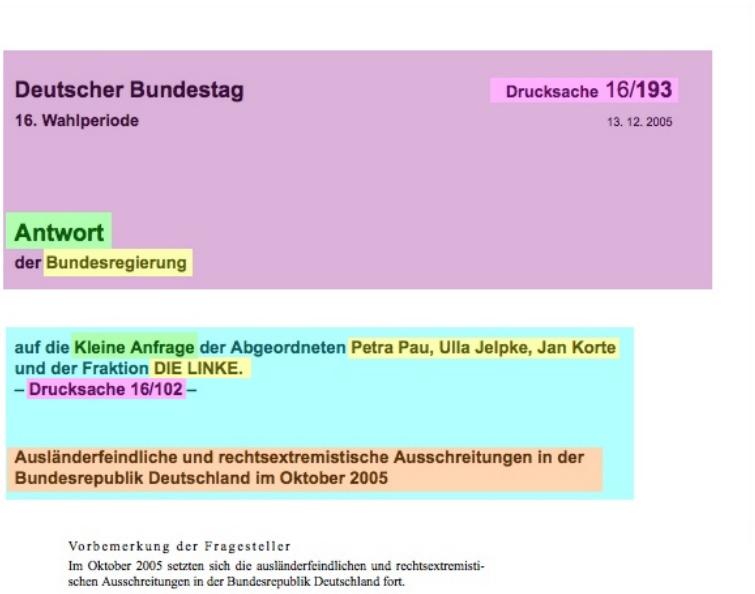

Deutscher Bundestag
Drucksache 16/193
16. Wahlperiode
13. 12. 2005
Antwort
der Bundesregierung
auf die Kleine Anfrage der Abgeordneten Petra Pau, Ulla Jelpke, Jan Korte und der Fraktion DIE LINKE.
– Drucksache 16/102 –
Ausländerfeindliche und rechtsextremistische Ausschreitungen in der Bundesrepublik Deutschland im Oktober 2005
Vorbemerkung der Fragesteller
Im Oktober 2005 setzten sich die ausländerfeindlichen und rechtsextremistischen Ausschreitungen in der Bundesrepublik Deutschland fort.

Abbildung 1.89: Beispiel einer Drucksache mit relevanten Bereichen.

im Zuge der PG von der Maschine automatisch angebracht werden. Die zwei groben Bereiche, die mit violett und hell-blau markiert wurden, identifizieren das Dokument selbst (violett) und das Dokument, auf das sich dieses bezieht (hell-blau). Die Typen der Dokumente wurden grün markiert, die beteiligten Personen und Institutionen gelb. Der Titel wurde mit orange abgesetzt, die Nummern der Drucksachen sind rosa. An diesen markierten Bereichen und Wörtern werden die Dokumente identifiziert und mit Hilfe der markierten Daten verarbeitet.

Eine detailierte Auflistung der Markierungen und Verarbeitungstechniken folgt im weiteren Verlauf dieses Abschlussberichts. Für den Anwendungsbereich führen wir noch die Übersicht über die vorhandenen Arten der Dokumente sowie deren herausstechenden Merkmale:

- BTD
  Drucksachen des Deutschen Bundestages
  - Pfad:
    Die im pdf-Format gespeicherten Dateien befinden sich unter
    http://dip.bundestag.de/btd/[Wahlperiode]/[fortlaufend nummeriertes Unterverzeichniss]/[XXYYYYY].pdf
  - Dateiname und Nummerierung:
    Dateiname ist siebenstellig, Die ersten beiden Stellen sind identisch mit der Wahlperiode (XX), die nächsten fünf Stellen (YYYYY) sind für die laufende Nummer reserviert. Diese entspricht der Drucksachennummer. Die Nummerierung innerhalb des Dokuments hat immer die Form Wahlperiode/Drucksachennummer.
  - Layout:
    Das Layout der Drucksachen ist abhängig vom Vorgang. Anträge, Beschlüsse, Anfragen und Anfragen sind meistens ein oder zweispaltig. Fragen zur Fragestunde sind immer tabellarisch angeordnet.

  - Vorgänge:

    * Anträge
    * Kleine und große Anfragen
    * Antworten
    * Beschlussempfehlungen, Berichte, Empfehlungen und Ergänzungen
    * Gesetzentwürfe
    * Übersichten
    * Unterrichtungen
    * Fragen für eine Fragestunde
    * Wahlvorschläge

- BTP
  Plenarprotokolle des Deutschen Bundestages
  - Pfad:
    Die im pdf-Format gespeicherten Protokolle befinden sich unter
    http://dip.bundestag.de/btp/[Wahlperiode]/[fortlaufend nummeriertes Unterverzeichniss]/[XXYYY].pdf
  - Dateiname und Nummerierung:
    Dateiname ist fünfstellig, Die ersten beiden Stellen (XX) sind identisch mit der Wahlperiode, die folgenden drei Stellen (YYY) sind für die laufende Sitzungsnummer reserviert. Wird in einem Schriftstück auf ein Plenarprotokoll verwiesen, so hat die Plenarprotokoll Nummer immer die Form Wahlperiode/Sitzungsnummer.
  - Layout:
    Das Layout der Plenarprotokolle ist immer zweispaltig mit vielen Einrückungen.
  - Vorgänge:

    * Abstimmungen
    * Fragestunden
    * Reden (Beratungen, Aussprachen)

- IDX
  Verhandlungen des Deutschen Bundestages: Drucksachen Bänder

  - Pfad:
    Die im pdf-Format gespeicherten Bänder der Drucksachen befinden sich unter
    http://dip.bundestag.de/btd/[Wahlperiode]/idx/[XX]IN[YYY].pdf
  - Dateiname und Nummerierung:
    Dateiname ist fünfstellig, Die ersten beiden Stellen sind identisch mit der Wahlperiode (XX), darauf folgt die Abkürzung IN für Index. Die laufende Nummer des aktuellen Bandes füllt die folgenden drei Stellen (YYY)aus.

– Layout:
  Das Layout der Drucksachen Bänder beginnt immer mit einem tabellarischen Abkürzungsverzeichnis und dann einer fünfspaltigen, tabellarischen Auflistung aller Drucksachen diesen Bandes.
– Vorgänge:
  Die Bänder beinhalten keine Vorgänge, sie listen alle Drucksachen mit Titel, Datum Urheber und Dokumenttyp auf.

2 Systementwurf

2.1 Das System in prozessorientierter Sicht

Die Ziele der Projektgruppe spezifizieren die Erstellung eines intelligenten Software-Systems zur Beantwortung von Benutzerfragen. Das Software-System soll auf Basis von bestehenden Standards in der Software-Technologie als eine Java Anwendung realisiert werden. Durch den Einsatz dieser Anwendung, soll in erster Linie zeitaufwändige Recherche verkürzt und vereinfacht werden. Neben der in der Projektaufgabenstellung beschriebenen Fragebeantwortung, sollen auf Wunsch Teile der gesammelten Informationen dargestellt werden. Die Anwendung kann also in die Kategorie der Wissens- und Informationssysteme eingeordnet werden und weist einige Merkmale einer Suchmaschine auf.

Eine derartige Anwendung setzt sowohl die Existenz einer graphischen Benutzeroberfläche (GUI) die dem fragestellendem Benutzer zugänglich ist, wie auch die Existenz eines umfangreichen Datenverarbeitungssystems im Hintergrund. Wir unterscheiden also in der architektonischen Sicht auf das System, zwischen dem Benutzer zugewandtem Frontend und dem datenverarbeitendem Backend.

Die komplexen Vorgänge im Backend sollen in erster Linie von einem Administrator/Entwickler überwacht und gesteuert werden, eine vollständige Automatisierung ist denkbar einfach, aber nicht im Sinne der Projektgruppe. Eine eigene GUI ermöglicht bessere Präsentation und Kontrolle der Datenverarbeitung, schließlich stecken die größten Aufwände gerade in dem für den Frontend-Benutzer unspektakulären Backend.

Die Funktionalität des Gesamtsystems kann grob in drei Prozessen: Wissensentdeckung, Extraktion und Fragebeantwortung zusammengefasst werden. Daraus ergibt sich die Modellierung der drei Prozesse, die eine logische Aufteilung des Systems erlauben. Wenn wir die Wege der Daten in dem System betrachten, wird auf der einen Seite der Datenweg aus dem Internet zur Datenbank und auf der anderen Seite, der Weg der Daten aus der Datenbank zum Benutzer deutlich sichtbar.

Diese sinnvolle Aufteilung des Software-Systems führte zu der Entscheidung zwei Anwendungen zu erstellen. Beide haben jeweils den Charakter eines „standalone“ Subsystems.

Abbildung 2.1 zeigt das System in eben dieser prozessorientierten Sicht. Sie stellt gleichzeitig den Weg der Daten durch das Gesamtsystem und die Interaktion der Benutzer mit den beiden Anwendungen dar. Diese etwas ungewöhnliche Sicht als SDL Diagramm auf die Anwendungen dient der Orientierung innerhalb des Systems und stellt gleichzeitig den gröbsten Funktionsumfang dar.

2.1.1 Akquisition- und Extraktionssystem (AES)

Das Akquisition- und Extraktionssystem ist eine modulare, aus Komponenten bestehende Anwendung, die Wissen aus Dokumenten und Webseiten extrahieren soll. Das Internet, spe-

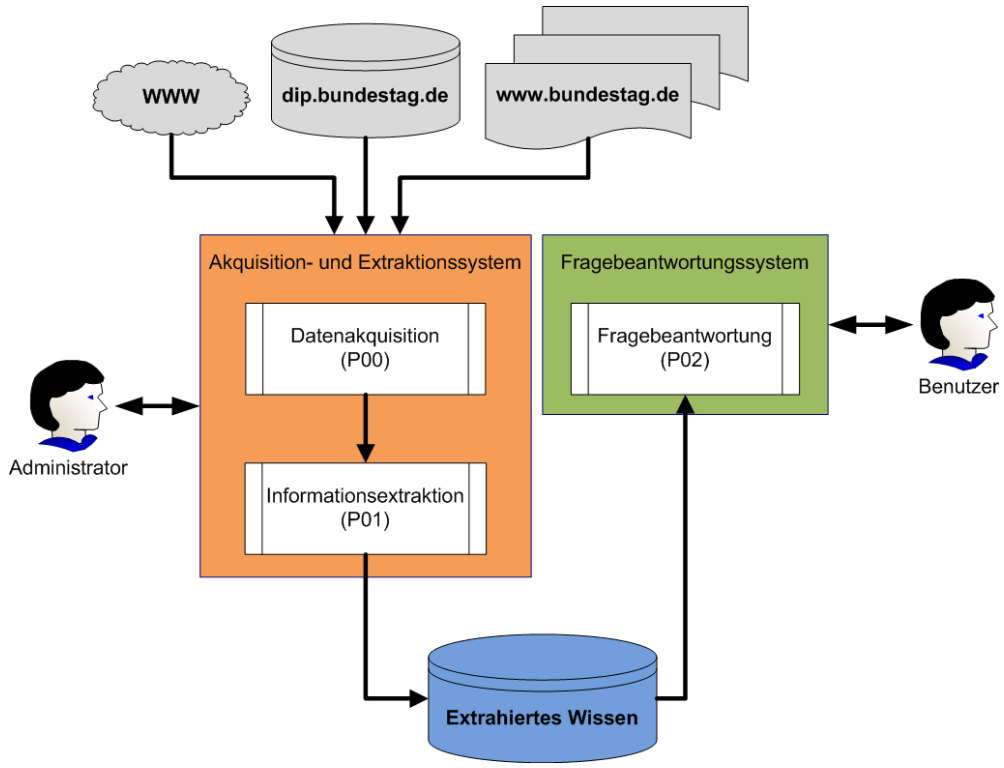


Abbildung 2.1: Zwei Anwendungen in der prozessorientierten Sicht.

ziell die Seiten des Deutschen Bundestages werden als eine umfangreiche und heterogene Quelle genutzt. Die dort vorhandenen Dateiformate der Quellen wurden bereits im Kapitel Anwendungsgebiet beschrieben.

AES speichert die akquirierten Dateien im lokalen Dateisystem und extrahiert Informationen aus deren Inhalten. Neben einer Benutzeroberfläche mit der ein Administrator/Entwickler die Anwendung steuern kann, existiert eine Anbindung an die Datenbanken in denen das extrahierte Wissen gespeichert wird. Der Weg der aus den Inhalten gewonnen Daten kann wie folgt als Prozesse beschrieben werden:

P00 Datenakquisition.
Die Datenaquisition ist ein Synonym für die Erstellung einer gemeinsam genutzten Basis von Dokumenten, die zur späteren Verarbeitung im Dateisystem des Rechners abgelegt wird. Alle zur Informationsextraktion notwendigen Vorverarbeitungsschritte sollen im Verlauf dieses Prozesses abgeschlossen werden.

Dieser Prozess veranlasst also das Herunterladen von Dateien aus dem Dokumenten-Archiv und der Webseite des Bundestages. Da die Dokumente im pdf-Format vorliegen, sollen sie zunächst einmal abgespeichert werden. Um diese Dokumente verarbeiten zu können, müssen sie von pdf zu plain text konvertiert werden. Der Prozess hält gleichzeitig nach, welche Dateien, von welcher Lokation bereits heruntergeladen worden sind und an welcher Stelle sie im Dateisystem abgespeichert wurden. Abschliessend werden alle heruntergeladenen Dokumente indexiert.

P01 Informationsextraktion.
Im Zuge dieses Prozesses werden interessante Daten aus den gesammelten Daten extrahiert. Diese Informationen, also alle für die Fragebeantwortung benötigten Daten werden strukturiert gespeichert. Die vorhandenen Dokumente werden im Verlauf dieses Prozesses mit zusätzlichen Meta-Informationen versehen. Die Informationsextraktion beinhaltet nicht nur die Markierung von „Named Entities“ sondern auch der semantischen Relationen zwischen den Entites. Zusätzlich sollen Informationen, die mit Hilfe von regulären Ausdrücken herauslesen wurden, Strukturiert in XML Dateien abgelegt werden.

2.1.2 Fragebeantwortungssystem (FBS)

Das Fragebeantwortungssystem, kurz FBS, ist ebenso eine modulare Anwendung welche Benutzerfragen entgegen nehmen soll. Sie soll die in den Datenbanken vorhandenen Informationen nutzen um diese Fragen zu beantworten. Dies wird in der Abbildung 2.1 durch den folgenden Prozess dargestellt:

P02 Fragebeantwortung.
Der Benutzer kann über eine grafische Oberfläche Fragen an das System stellen. Abhängig von der gestellten Frage werden unterschiedliche Methoden der Beantwortung ausgeführt, welche unterschiedliche Komponenten des Systems ansprechen. Die Komponenten berechnen auf Grund der vorhandenen und a priori extrahierten Informationen eine Antwort, die wiederum an den Benutzer ausgeliefert wird.

2.1.3 Prozessorientierung und UML

Bei der Beschreibung der Software wird eine Kombination aus Prozessdiagrammen und UML benutzt. UML Diagramme alleine hätten den Umfang dieser Beschreibung vervielfacht und bilden die prozessorientierte Interaktion der Komponenten nur unzureichend ab. Getreu dem Motto, dass eine Beschreibung selbst von einem Laien schnell lesbar sein sollte verzichten wir auf aufwändige Sequenz-, Zustand- und Komponentendiagramme und beschreiben die eben erwähnte Interaktion als Prozesse.

Unsere Aufteilung in die oben beschriebenen drei Prozesse ist natürlich nur die oberste Ebene und eine Abstraktion. Jeder dieser groben Prozesse zerfällt in eine Menge aus feineren, teilweise von einander unabhängigen Prozessen. Die im weiteren Verlauf des Kapitels genauer beschrieben werden.

2.2 Architektur

2.2.1 Design Entscheidungen

Entsprechend dem Anwendungsgebiet klassifizieren wir beide Subsysteme als prozessorientierte, datenverarbeitende Anwendungen, ohne explizite Benutzerintervention [Som04]. Der Benutzer löst nur die datenverarbeitenden Prozesse aus, hat jedoch keinen Einfluss auf deren Verlauf. Das Gesamtsystem nimmt, entsprechend der Request-Response Verfahrensweise, eine Benutzeranfrage entgegen und liefert eine Antwort aus. Ist der Request eine Benutzerfrage,

wird die Antwort berechnet. Ist der Request ein Anweisung zum Starten eines datenverarbeitenden Prozesses wird dieser Ausgeführt.

Die Aufteilung des Systems auf zwei Subsysteme begünstigt die Implementierung als eine Client/Server Anwendung. Das Frontend könnte auf der Client-Seite ausgeführt werden, das Backend auf dem Server. Dies ist jedoch im Rahmen dieser PG, aus Zeitgründen nicht beabsichtigt.

Als generisches Architekturmodell verwenden wir das von Buxton eingeführte Repository Modell [J.80]. Ein Repository ist dem entsprechend eine passive Sammlung aus gemeinsam genutzten Datenstrukturen und Datenbanken, welche von Subsystemen umgeben sind. Wie bereits erwähnt Betrachten wir die Prozesse P00-P01 und P02 als eben diese Subsysteme. Jedes Subsystem, besteht aus mehreren Komponenten, die in ihrem gemeinsamen Wirken diesen Prozess implementieren. Die einzelnen Komponenten sind in dem Modell für die Kontrolle und Verarbeitung der Daten verantwortlich, es gibt keine zentrale verwaltende Instanz die sich um Konsistenz bemüht. Jede Komponente sorgt selber dafür das sie die Daten lesen und wieder speichern kann. Um diese Aufgabe zu erleichtern verwenden wir XML als Dateiformat für unsere Datenbanken. Einige Komponenten werden teilweise prozessübergreifend verwendet und gewährleisten damit größtmögliche Wiederverwertbarkeit des Quellcodes.

Die Projektgruppe hat sich bei der Implementierung des Prozesses P00 ganz bewusst für die Erstellung einer beinahe Abbildung der Inhalte des Dokumenten-Archivs entschieden. Die Entwicklung und Test der Software ist damit unabhängig von einer Internet Verbindung und wir erkaufen uns, auf Kosten des zusätzlichen Speicherplatzes wesentlich schnelleren Zugriff auf die Dokumente. Außerdem betrachten wir die im pdf-Format vorliegenden Dokumente des Bundestages als statische Dateien, die im laufe der Zeit vom Bundestag nicht geändert werden dürfen. Anders verhält es sich mit den dynamischen Inhalten der Webseite, diese müssen regelmäßig auf Änderungen überprüft werden. Diese Mechanismen werden jedoch nicht implementiert, obwohl eine Grundlage durch das Nachhalten der heruntergeladenen Dateien im Ansatz gegeben ist.

2.2.2 Dekomposition der Architektur

Die Subsysteme werden entsprechend der Abbildung 2.2 in einzelne Komponenten zerlegt. Jedes Subsystem interagiert, gemäß der Vorgabe des Architekturmodells mit dem Repository in dem es Daten liest und/oder speichert. Die durchgezogenen Pfeile markieren eben diese Interaktion. Die gestrichelten Pfeile drücken die „Komponente benutzt eine andere Komponente“ Beziehung aus. Im vorangegangenem Abschnitt haben wir die Aufgaben der Prozesse beschrieben, in diesem Abschnitt werden die beteiligten Komponenten kurz skizziert. Eine genaue Beschreibung einzelner Komponenten befindet sich im im Kapitel 2.2 dieser Ausarbeitung.

AES: [P00] - Datenakquisition

Die Komponenten der Datenakquisition interagieren miteinander um den Weg der Dokumente aus dem Internet in das Repository zu realisieren und können folgender weise aufgelistet werden:

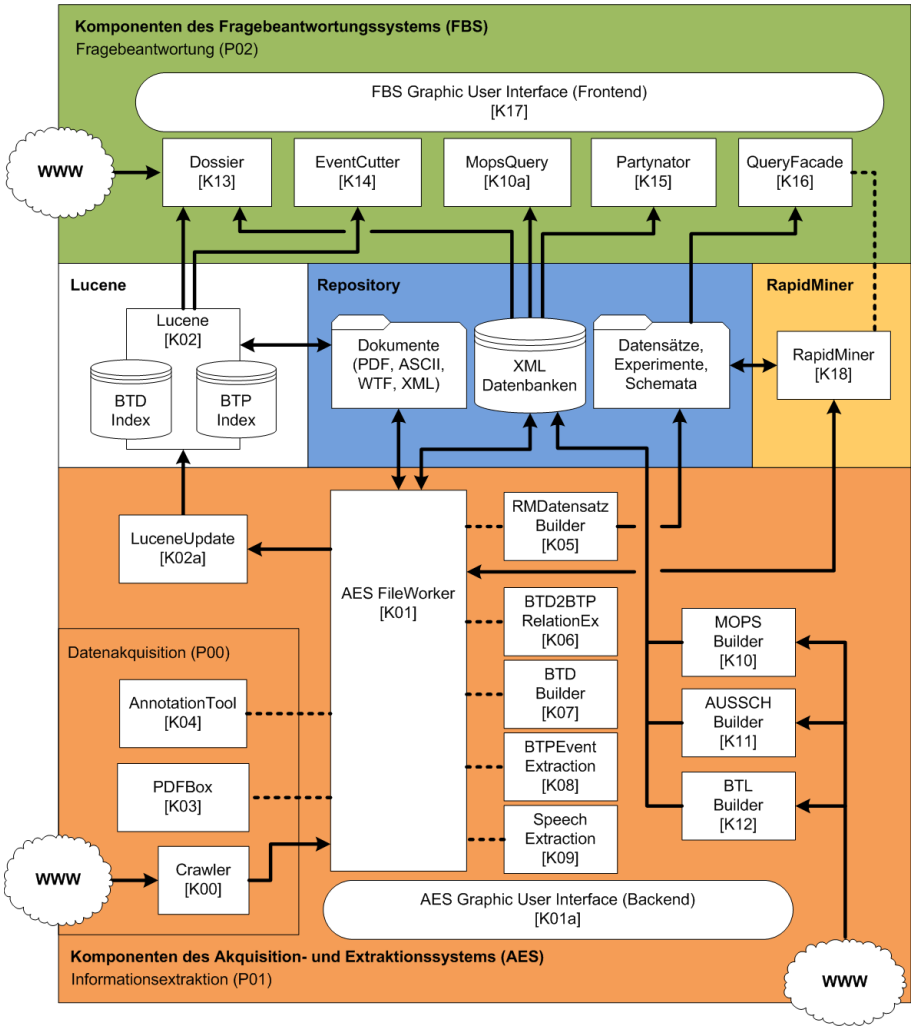


Abbildung 2.2: Komponentenmodell

- [K00]Crawler
  Lädt Dateien an Hand einer eingegebenen URL herunter und speichert sie im Dateisystem ab. Die Dateien sollen im pdf-Format abgespeichert werden und deren URL an das Subsystem übergeben werden.

- [K01]AES FileWorker
  Diese Komponente ist entsprechend der MVC Architektur der Controller in dieses Subsystems und steuert den weiteren Ablauf der Datenverarbeitung.

- [K01a]AES Graphic User Interface
  Die GUI des Backends mit der die Datenakquisition und Extraktionsprozesse gestartet werden.

- [K03]PDFbox
  Eine externe Open Source Komponente um PDF Dateien laden und verarbeiten zu können. Wird von uns hauptsächlich dazu benutzt Dokumente aus dem pdf-Format ins plain text zu konvertieren.

- [K04]AnnotationTool
  Das AnnotationTool ist eine standalone Anwendung mit einer eigenen GUI um Texte mit einem vordefinierten Set aus Named Entities zu taggen. An dieser Stelle benutzen wir sie um Dokumente aus dem plain text in das wortweise-tokenisierte Format (wt-Format) zu konvertieren. Das wt-Format wird für die Named-Entity-Recognition benötigt.

AES: [P01] - Informationsextraktion

Die Komponenten der Informationsextraktion bilden in ihrer Interaktion die Prozesse ab, mit denen Informationen aus Dokumenten, dem Internet und anderen Datenbanken in das Repository transferiert werden. Das Ziel der Informationsextraktion ist es für die Fragebeantwortung lesbare Datenstrukturen zu erzeugen.

- [K01]AES FileWorker
  Diese Komponente ist entsprechend der MVC Architektur der Controller in dieses Subsystems und steuert den weiteren Ablauf der Datenverarbeitung.

- [K01a]AES Graphic User Interface
  Die GUI des Backends mit der die Datenakquisition und Extraktionsprozesse gestartet werden.

- [K02a]LuceneUpdate
  Eine Interface Komponente zur Ansteuerung des Lucene Indexers.

- [K05] RM DatensatzBuilder
  Diese Komponente erstellt einen Eingabe-Datensatz für die RapidMiner Anwendung. Dieser Datensatz wurde aus den Meta-Daten der Dokumente und Abgeordneten erstellt und wird benötigt um mit Hilfe von Lernverfahren im RapidMiner die so genannten Warum-Fragen zu beantworten.

- [K06]BTD2BTP RelationExtraction
  Diese Komponente erstellt eine simple XML Datenbank aller Verweise zwischen den Protokollen und Drucksachen. Sie bildet also die Relationsrichtung: welche Drucksachen wird in einem Protokoll besprochen, bzw. zittiert. Die umgekehrte Relation: in welchen Protokollen wird eine Drucksache behandelt ist auf kosten von ein wenig Rechenzeit aus der Liste zu entnehmen.

- [K07]BTD Builder
  Diese Komponente ist eine Vorstufe vor dem Aufbau des BTD Index in Lucene. Sie extrahiert aus den Drucksachen des Bundestages mit Hilfe von Regular Expressions relevante Meta-Daten. Sie erzeugt also eine XML Datenbank mit diesen Daten, die später bei der Erzeugung des BTD Index in die Lucene frei-definierbaren Felder zu einem Dokument importiert wird.

  Diese, ursprünglich nur auf dem Inhaltsverzeichnis des Bundestages, also den IDX Dokumenten arbeitende Komponente wurde durch eine verbesserte Version ausgetauscht, die mit Hilfe der PDFbox Komponente die BTD pdf-Dokumente erneut öffnet und daraus Informationen extrahiert.

- [K08]BTP Event Extraction
  Die Event Extraction Komponente markiert Ereignisse in den Protokollen des Bundestages. Die komponente erkennt an Hand der im Event Schema definierten Triggerwörter im Text stattfindende Ereignisse und markiert sie mit xml Tags. Diese können später von der Maschine gelesen und interpretiert werden.

- [K09]SpeechExtraction
  Diese Komponente durchsucht Sitzungsprotokolle nach bestimmten Mustern zerlegt ihre Inhalte in Tagesordnungspunkte und damit auch in einzelne Redebeiträge. Zu jedem Tagesordnungspunkt / Rede wird ein Satz aus Meta-Daten erstellt und im Repository als XML Datei gespeichert. Die einzelnen Redebeiträge erhalten eine eindeutige ID und werden im Repository gespeichert. Die Redebeiträge und die gesammelten Meta-Daten werden in dem BTP Index von Lucene integriert.

- [K10]MOPSBuilder
  „Members Of Parlament Search“ Builder lädt die vorhandenen Biographien der Abgeordneten von der Webseite des Bundestages. Sie extrahiert aus den HTML-Webseite zu jedem Abgeordneten einen von der Wahlperiode abhängigen Satz an Abgeordneten-Daten. Aus diesen Informationen wird eine eine XML Datenbank mit der man Fragen zu den Abgeordneten beantworten kann.

- [K11]AUSSCHBuilder
  Diese Schwester-Komponente zu MOPSBuilder arbeitet auf den Informationswebseiten zu den verschiedenen Ausschüssen des Detuschen Bundestages. Diese Komponente erstellt eine vollständige Liste der Mitglieder eines Ausschusses. Diese ebenso aus HTML-Webseiten extrahierten Informationen werden in einer XML Datenbank gespeichert.

- [K12]BTLBuilder
  Auf den Webseiten des Bundestages befinden sich erstaunlich wenige Informationen zur der Zusammensetzung und Verteilung der Posten in der Bundesregierung. Zu dem

fehlen die als Allgemeinwissen vorausgesetzten Informationen. Also einfache statistische Daten über die Parteien, Sitzverteilung, die Parteivorsitzenden und Gründungsdaten. Diese Daten wurden in einer separaten Datenbank der BTLookup XML Datenbank abgelegt, die anfänglich manuell erzeugt worden ist. Der BTL Builder erzeugt diese Datenbank automatisch aus den Webseiten der Wikipedia.org.

FBS: [P02] - Fragebeantwortung

Dieser Abschnitt muss im Laufe des zweiten Projektsemesters detailliert beschrieben werden. Die Zusammenstellung des Subsystems steht zum heutigen Zeitpunkt nicht fest, daher wird hier erst mal auf die Beschreibung aller beteiligten Komponenten verzichtet.

- [K10a] MopsQuery
  Diese fragebeantwortende und aus dem Kürzel MOPS also:„Members Of Parlament Search“ stammende Komponente, benutzt die gleichnamige XML Datenbank um statistische und direkte Fragen zu Abgeordneten zu beantworten.

- [K13] Dossier
  Das Dossier ist eine informationsdarstellende Komponente die sowohl MOPS XML, wie auch beide Lucene Indices benutzt. Sie stellt in der GUI ein Dossier eines Abgeordneten. Dieses Dossier ist abhängig zu der Wahlperiode und beinhaltet nicht nur alle biographischen Daten eines Abgeordneten, sondern auch alle Drucksachen an den der Abgeordnete beteiligt war. Zusätzlich wird eine Liste der Reden des Abgeordneten aus den Protokollen geführt.

- [K14 EventCutter
  Der EventCutter gehört zu den fragebeantwortenden Komponenten, es benutzt Lucene um zu einer Stichwortmenge, eine Auswahl an Dokumenten zu laden und die Textausschnitte um diese Stichwörter herauszuschneiden und zurückzugeben.

- [K15]Partynator
  Diese ursprünglich zum Campus Fest erstellte Komponente vergleicht Benutzerangaben mit der MOPS Datenbank nach Übereinstimmungen. Als Ergebnis dieser Suche wird eine Liste der Abgeordneten zurückgegeben die dem Benutzer am ähnlichsten sind.

- [K16]QueryFacade
  Ist eine fragebeantwortende Komponente die ein Framework aus Fragebausteinen anbietet damit der Benutzer komplexe, Datenbankübergreifende Fragen an das System stellen kann. QueryFacade nutzt beinahe alle vorhandenen Datenbanken und kann die Beantwortung von den so genannten „warum-Fragen“ mit Hilfe der RapidMiner Anwendung anstossen.

- [K17]FBS Graphic User Interface
  Die GUI des Frontends integriert die P02 Komponenten in einer großen Anwendung.

Lucene (lucene.apache.org)

- [K02]Lucene
  Lucene ist eine Open Source Anwendung die in der Lage ist einen Index der Dokumente aufzubauen, also verantwortlich für die Zuordnung: wie oft kommt ein Wort in

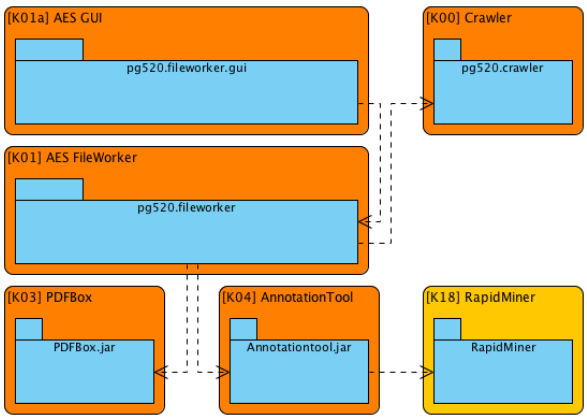


Abbildung 2.3: JAVA Packages der Datenakquisition

welchen und wie vielen Dokumenten vor. Lucene kann zusätzlich zu jedem Dokument frei definierbare Attribute speichern.

- BTD Index
  Der von Lucene Index über den Drucksachen des Bundestages, der zusätzlich alle extrahierten Meta-Daten eines Dokuments beinhaltet.
- BTP Index
  Im Zuge der Informationsextraktion wurden aus den Protokollen des Bundestages Redebeiträge der Abgeordneten extrahiert, diese wurden separat als einzelne Reden im plain-text Format gespeichert. Der BTP Index ist ein weiterer Lucene Index der genau auf diesen Reden und ihren Meta-Daten erstellt wurde.

RapidMiner (yale.sourceforge.net)

- [K06]RapidMiner
  Diese am LS8 entwickelte Anwendung stellt ebenso eine eigene, ausgefeilte GUI zur Verfügung mit der Data Mining und Information Extraction Prozesse erstellt werden können. Die Modell-Dateien eines Lernlaufes sollen im Repository gespeichert werden. Eine genaue Beschreibung dieser Anwendung befindet sich im Kapitel 2.2.

2.3 Statisches Package Modell

Wir haben nun alle Komponenten kurz vorgestellt, die hier aufgeführten UML Package Diagramme beinhalten die oberste Hierarchieebene der Java Packages beider Systeme. Die Diagramme orientieren sich, entsprechend unserer Konvention, an den Prozessen P00 bis P02. Wie bereits erwähnt benutzen wir einige Open Source Anwendungen als Komponenten die über public Methoden aufgerufen werden. Diese, von uns verwendeten 3rd-party Anwendungen, stehen alle unter der GPL Lizenz auf www.sourceforge.net frei zur Verfügung. Alle übrigen Komponenten wurden von den Mitgliedern der Projektgruppe selbständig entwickelt und können aus dem CVS (Concurrent Versioning System) der Projektwebseite bezogen werden. (pg520.sourceforge.net)

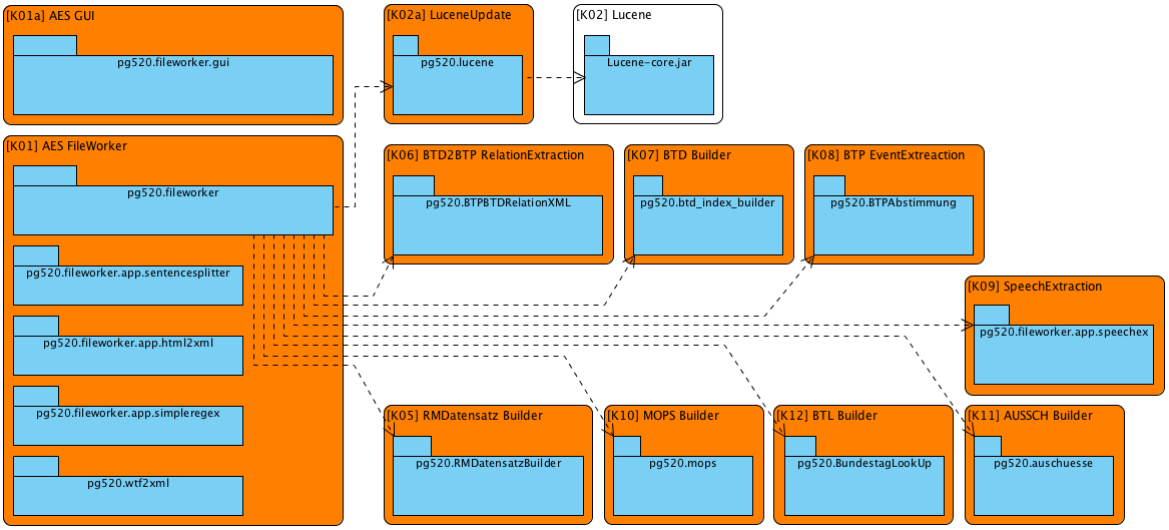


Abbildung 2.4: Packages der Informationsextraktion.

Die von den PG-Mitgliedern in Java (Version 6) programmierten Komponenten wurden mit der Open Source IDE Eclipse entwickelt und sind in Packages unterteilt. Gemäß der Java Naming Convention für Packages sollen diese eindeutig, in Kleinbuchstaben, ASCII benannt sein und mit den top-level Domains oder den Ländercodes des ISO Standard 3166 beginnen, gefolgt von Organisation, Abteilung, Projektname oder einem Eigennamen. Die Projektgruppe hat sich für den eindeutigen Namen:

- de.uni-dortmund.pg520

entschieden.

Während der Entwicklung wurden die einzelnen Komponenten als separate Projekte behandelt. Alle Komponenten besitzen eine gemeinsame Referenz auf das Repository und ein „Master Projekt“ in dem sich alle importierten Libraries der Komponenten befinden. Das Repository ist in der Entwicklungsphase ebenso ein leeres Projekt, die Daten und Datenbanken wurden, während der Entwicklung, händisch vom LS8 Fileserver in das Projekt umkopiert. Die Struktur des Reposiory wurde also im CVS gespeichert, die Daten auf dem LS8 Server. Die Größe des Repository und die Geschwindigkeit mit der das Repository ein- und ausgecheckt wird, waren die wichtigsten Argumente für dieses Vorgehen.

Die Abbildung 2.3 stellt die Packages der Datenakquisition dar. Die von dem Prozess verwendeten Komponenten werden von [K01] AES FileWorker angesteuert. Die Abbildung 2.4 ist ähnlich aufgebaut und zeigt alle beteiligten Komponenten. Zusätzlich wurden einige, meistens nur aus einer bis zwei Klassen bestehenden Komponenten direkt in das AES FileWorker Projekt implementiert. Sie sind dort als Sub-Packages enthalten.

Die verwendeten 3rd-party Anwendungen sind die beiden Open-Source-Java-Bibliotheken:

- Lucene - eine Bibliothek zum Erzeugen und Durchsuchen von Text-Indizes, mit der sich Voltextsuche auf beliebigen Texten implementieren lässt.

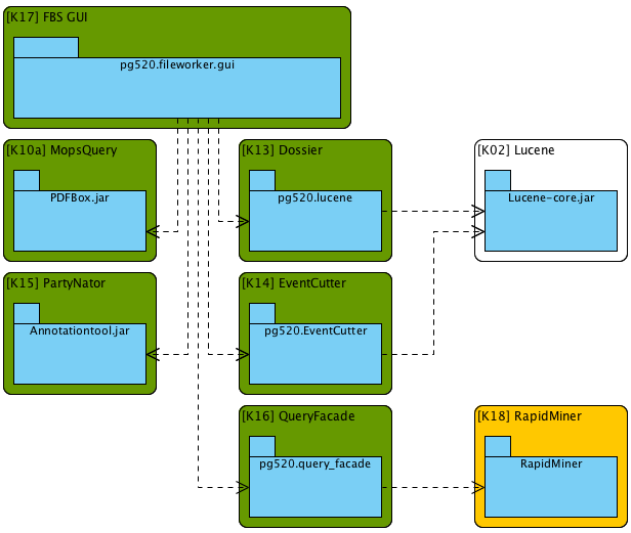


Abbildung 2.5: Java Packages im Prozess der Fragebeantwortung.

- PDFbox - eine Java PDF Bibliothek zum Erzeugen und Verarbeiten von pdf Dokumenten, welche gleichzeitig Extraktion von Inhalten erlaubt.

Zusätzlich wird die unter GPL lizensierte Version der Data-Minig Software: Rapid Miner (ehemals Yale) von Rapid-I (www.rapid-i.com) in Kombination mit dem Information Extraction Plug-In von Dipl.-Inf. Felix Jungermann verwendet.

Die Abbildung 2.5 zeigt die in UML modellierte Ansicht der Fragebeantwortung. Die gestrichelten Linien in allen Abbildungen drücken die „benutzt“ Beziehung zwischen den Packages aus.

2.4 Das Repository und seine Datenstrukturen

2.4.1 Das Repository

Unser System ver- und bearbeitet Dokumente und Datenbanken, es benötigt also eine physikalische Lokation, ein Depot, um diese Dateien zu holen und nach der Verarbeitung wieder abzulegen. Bei der Planung des Systems stand die Frage einer verteilten Datenhaltung im Raum, da eine größere Menge unabhängiger Komponenten vermutet wurde. Eine verteilte Datenhaltung hätte als Konsequenz die Implementierung von Synchronisierungs- und Kommunikationsmechanismen. Dies hätte unsere Aufwände vervielfacht und unsere Aufmerksamkeit viel zu sehr auf diese Mechanismen gelenkt, anstatt sie in auf die anwendungskritischen Problemen zu bündeln. Die Projektgruppe hat sich deshalb für ein simples, zentrales Datendepot entschieden, welches allen Komponenten bekannt ist.

Wir haben unser Datendepot frei als Repository benannt, wobei das Repository nicht viele Gemeinsamkeiten mit einer CVS oder SVN ähnlichen Struktur besitzt. Wir benutzen den Begriff in Anlehnung an das Repository Architekturmodell von Buxton, der eine zentrale, datenhaltende Instanz beschreibt, welche aus mehreren Datenbanken besteht und gemeinsame, von allen Komponenten nutzbare Datenstrukturen besitzt. Diese Form der Datenhaltung

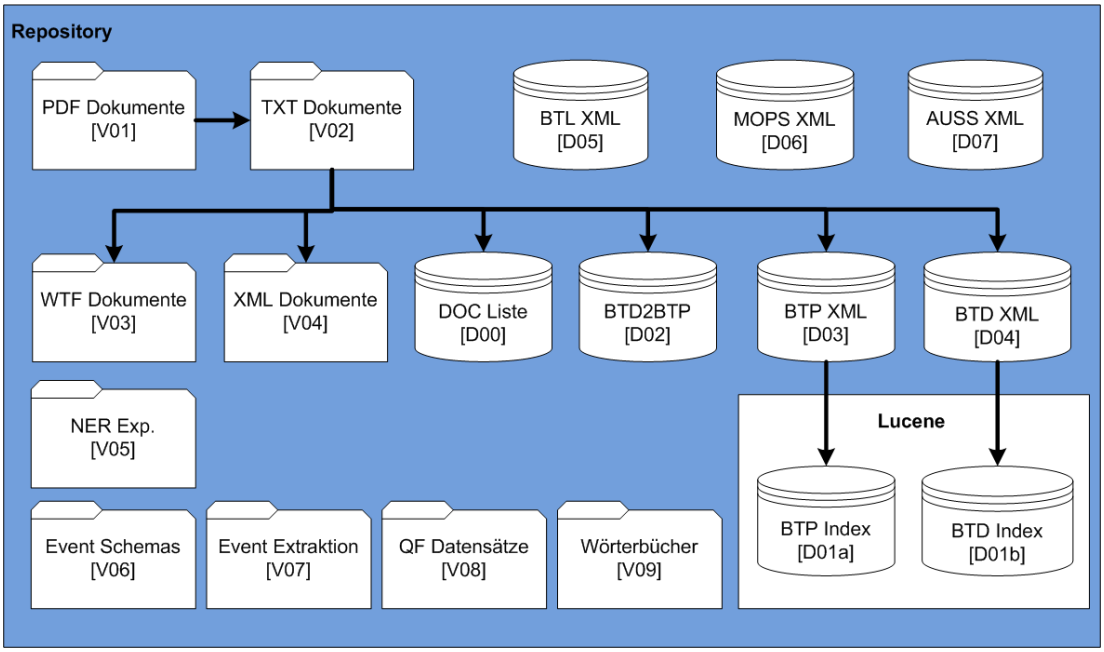


Abbildung 2.6: Datenstrukturen und Datenbanken im Repository

induziert gleichzeitig die Standardisierung der Verarbeitungsprozesse. Die Grundform eines jeden atomaren Prozesses entspricht etwa diesen drei Schritten:

- Daten aus dem Repository holen,
- Daten verarbeiten,
- Daten ins Repository ablegen.

Diese einfache Vorgehensweise reduziert den Verwaltungsaufwand und erlaubt die Modellierung längerer Prozessketten durch Hintereinander-Ausführung der Komponenten.

2.4.2 Datenbanken und Verzeichnisse

Das Repository besteht aus Verzeichnissen und Datenbanken. Die Verzeichnisse sind nichts anderes als Dateiordner, welche vom System erzeugte Dateien und aus dem Internet geladenen Dokumente beinhalten. Ein Verzeichnis hält in der Regel einen bestimmten Typ von Dateien vor. Die Entscheidung, welche Verzeichnisse angelegt wurden, hängt von der Struktur des Anwendungsgebietes und den zur Informationsextraktion notwendigen Verarbeitungsschritte.

Wie bereits erwähnt sammeln wir extrahierte Informationen in Datenbanken, es ist nicht weiter überraschend, dass sich dieses ebenfalls im Repository befinden. Die Abbildung 2.6 zeigt die Übersicht der vorhandenen Verzeichnisse und Datenbanken.

Die prozessorientierte Ausrichtung des Systems kann auch in der Struktur des Repository nachvollzogen werden. Die Richtungspfeile der Abbildung zeigen die Konvertierung der Dokumente aus einem Dateiformat ins Nächste. Die verbundenen Verzeichnisse zeigen eben diesen Weg der Konvertierung. Zusätzlich wird der Informationsfluss aus den Textdokumenten

zu den Datenbanken dargestellt. Wie man sieht, spielen die aus den PDF Quellen erzeugten Textdokumente eine zentrale Rolle im System. Sie sind der Ursprung des verzweigten Extraktionsprozesses, dementsprechend wirkt sich ihre Qualität auf die Güte der nachfolgenden Ergebnisse.

Wir wollen an dieser Stelle die an dem Konvertierungsweg beteiligten Verzeichnisse beschreiben:

- [V01] PDF Dokumente
  Dieses Verzeichnis enthält alle aus dem Internet geladenen PDF Dateien des Deutschen Bundestages. In seinen Unterverzeichnissen werden die drei Dokumenttypen gespeichert, die von dem IT-Dienstleister des dt. Bundestages im Internet bereitgestellt worden sind. Die drei Typen sind Drucksachen (BTD), Protokolle (BTP) und Drucksachenbänder (IDX).

- [V02] TXT Dokumente
  Die PDF Quellen können nicht ohne weiteres von einem Informationssystem verarbeitet werden. Sie wurden in einem Stapelverarbeitungslauf aus dem pdf-Format in ein gewöhnliches ASCII Text Format konvertiert. Diese konvertierten Dateien wurden in diesem Verzeichnis gespeichert, wobei die Struktur der Unterverzeichnisse erhalten blieb.

- [V03] WTF Dokumente
  Das „Wordwise Tagged“ Format (WTF) enthält pro Wort zusätzliche Informationen, nämlich die annotierte Named Entity des Wortes. Dieses Verzeichnis enthält alle WTF Dokumente die aus den Text Dokumenten erzeugt wurden.

- [V04]XML Dokumente
  Die Protokolle des Bundestages beinhalten Redebeiträge der Abgeordneten. Das XML Dokumente Verzeichnis enthält die aus den Text Dokumenten extrahierten Reden, die gleichzeitig in eine XML Dokument überführt worden sind. In diesen XML Dokumenten befinden sich alle Meta-Daten einer Rede, also Informationen über den Redner, Wortanzahl, Unterbrechungen usw.

Aus den PDF Dokumenten des Bundestages wurden zusätzlich einige Datenbanken erzeugt, die mit Meta-Informationen zu den Dokumenten gefüllt worden sind.

- [D00] DOC Liste
  Dies ist eine vom Crawler [K00] verwaltete Datenbank der bereits erfassten Dokumente.

- [D02] BTD2BTP
  Diese XML Datenbank enthält alle Relationen zwischen den Drucksachen und Protokollen. Sie listet alle Vorkommen einer Drucksache in den vorhandenen Protokollen.

- [D03] BTP XML
  Enthält alle Redner und Daten der Protokolle. Wird als Datenquelle für die Erzeugung der XML Dokumente [V04] sowie des Lucene Index über den Reden benötigt.

- [D04] BTD XML
  Wie die bereits beschriebene BTP XML ist diese Datenbank auch eine notwendige Voraussetzung zur Erzeugung einer nachfolgenden Datenbank. BTD XML wird vom BTD Builder [K07] erzeugt und enthält alle Meta-Daten zu einer Drucksache.

Unser System nutzt Lucene als zentralen Indexer. Die Zuordnung zwischen den Wörtern und Dokumenten wird also in zwei Lucene Instanzen gespeichert. Damit die Anfragen zu Dokumenten nicht über mehrere Datenbanken ausgeführt werden müssen, werden alle Meta-Daten ebenfalls als Attribute eines Dokuments in Lucene gespeichert. Die beiden folgenden Lucene Indizes werden also jeweils aus den Text und XML Dokumenten sowie den beiden Vorgänger Datenbanken erzeugt.

- [D01a] BTP Index
  Dieser Lucene Index wurde auf den Reden [V04] des Deutschen Bundestages aufgebaut, zusätzlich wurden die Meta-Daten aus der BTP XML [D03] genutzt.
- [D01b] BTD Index
  Der BTD Index wurde auf den Drucksachen des Bundestages erstellt. Zusätzlich wurde BTD Index [D04] benutzt um in Lucene zu jedem indexierten Dokument die entsprechenden Meta-Daten zu speichern.

Wir benötigen drei weitere Datenbanken, deren Inhalte aus HTML Webseiten erzeugt worden sind. Die erzeugenden Komponenten haben hierfür die entsprechenden Seiten abgerufen und mit Hilfe von regulären Ausdrücken und selbst geschriebenen Parsern verarbeitet. Unglücklicherweise ist die Lebensdauer der Inhalte einer Webseite in Zeiten von Content Management Systemen sehr kurz. Sobald sich also das Layout einer Webseite, oder ihre Inhalte ändern, werden auch die von uns geschriebenen Komponenten schlechter Arbeiten.

- [D05] BTL XML
  Die Bundestag-Lookup Datenbank enthält generelle Informationen zum Deutschen Bundestag, also Informationen zur Sitzverteilung, Parteinamen, Ministerposten und Regierungen. Diese Informationen sind nach Wahlperioden geordnet und in einer umfangreichen XML vorgehalten. Diese generellen Infos werden von vielen Komponenten genutzt, sie sind vor allem bei der Beantwortung statistischer Fragen sehr nützlich.
- [D06] MOPS XML
  Die Members of Parlament Datenbank ist eine aus den Biographien extrahierte Datenbank mit allen Informationen zu den Abgeordneten des Deutschen Bundestages. Zusätzlich wurden Informationen zu den Ministern und Regierungsmitgliedern aus der Wikipedia.org extrahiert und eingefügt.
- [D07] AUSS XML
  AUSS XML ist eine weitere Datenbank die aus den Seiten des Bundestages erzeugt worden ist. In ihr sind alle Daten zu den Ausschüssen des Deutschen Bundestages gespeichert. Diese Datenbank ist also eine nach Wahlperioden geordnetes XML Dokument der Ausschüsse und ihrer Mitglieder.

Einige Komponenten sind auf die Existenz weitere Verzeichnisse angewiesen:

- [V05] NER Experimente
  RapidMiner [K18]benötigt neben den Daten die verarbeitet werden sollen auch so genannte Experiment Dateien um die Verarbeitung zu Steuern. Dieses wird benutzt Verzeichnis um diese Prozessketten im XML Format abzulegen.

- [V06] Event Schemas
  Im Zuge der Projektgruppe wurde ein XML Schema erstellt um die verschiedenen Ereignisse zu identifizieren. Genauer gesagt wurden in diesen XML Schemas die Stichwörter und Relationen definiert an denen ein System die Ereignisse erkennen könnte. Leider ist die entsprechende Komponente, die diese Verarbeitung bewältigen sollte, nicht erstellt worden.

- [V07] Event Extraktion
  In diesem Verzeichnis sollen die extrahierten Ereignisse gespeichert werden, die Ergebnisse der Extraktion von Abstimmungen wurden hier bereits abgelegt.

- [V08] QueryFacade Datensätze
  Das System zur Fragebeantwortung ist in der Lage Fragen zu den Ursachen von bestimmten Abfragen zu beantworten. In der QueryFacade [K16] Komponente werden diese Fragen an RapidMiner [K18] weitergereicht. RapidMiner benötigt als Eingabe einen speziell formatierten Datensatz, der aus den Daten verschiedener Datenbanken konstruiert wird. In diesem Verzeichnis werden diese Datensätze von RMDatensatz Builder [K05] abgelegt.

- [V09] Wörterbücher
  Komponenten die natürlich sprachliche Texte verarbeiten sollen, benötigen Wörterbücher zur Auflösung von Ambiguitäten. In diesem Verzeichnis liegen alle von den Komponenten genutzten Wörterbücher.

2.4.3 Datenstrukturen und Dateiformate

Ein Repository setzt voraus, dass die Anzahl der verwendeten Dateiformate und Datentypen überschaubar bleibt. Deshalb haben wir uns, entsprechend der Vorgabe, an XML als zentrales Dateiformat gehalten.

- XML eXtended Markup Language
  Wir verwenden das XML Format, mit den ihm inheränten Bäumen als einen einfachen Weg um strukturiert Informationen zu speichern. Wir bezeichnen deshalb alle solche XML Dateien die innerhalb des Repository hauptsächlich diesem Zweck dienen als Datenbanken.

  Es folgt eine kurze Liste der verwendeten XML Datenbanken:

  - [D02] BTD2BTP
  - [D03] BTP XML
  - [D04] BTD XML
  - [D05] BTL XML
  - [D06] MOPS XML

– [D07] AUSS XML

Auf der anderen Seite reichern wir gewöhnliche unstrukturierte Textdokumente mit Hilfe von XML Tags an. Diese angereicherten Dokumente (a.k.a. enriched documents) können von den Komponenten einfacher verarbeitet werden, und bleiben selbst für das menschliche Auge lesbar.

– [V04] Aus den Protokollen extrahierte Redebeiträge.
– [V04] Vom RapidMiner NE annotierten Drucksachen.

Neben XML werden folgende Dateiformate als Dokumentformate gespeichert.

- PDF Portable Document Format
  Alle Dokumente des Bundestages sind in dem dem von Adobe entwickelten Dateiformat vorhanden. Die verschiedenen Typen der Bundestagsdokumente wurden bereits in der Beschreibung des Anwendungsgebiets in aller Ausführlichkeit behandelt, wir wollen hier die vorhanden Typen anreissen:

  – [V01] PDF Version der Drucksachen des Deutschen Bundestages
  – [V01] PDF Version der Protokolle des Deutschen Bundestages
  – [V01] PDF Version der Drucksachenbänder des Bundestages

- TXT Plain Text Format
  Diese im reinem Textformat vorliegenden Dokumente sind das Ergebnis der Dokumentakquisition [P00]. Wir konvertieren alle im PDF Dokumente Verzeichniss [V01] liegenden Dateien in dieses Format. Zusätzlich sind die Wörterbücher in diesem Format abgespeichert.

  – [V02] Drucksachen des Deutschen Bundestages
  – [V02] Protokolle des Deutschen Bundestages
  – [V09] Wörterbücher

- WTF Word Tagged Format
  Dieses Format wird benötigt als Eingabe für die Information Exraction Komponente des RapidMiner [K18]. Die charakteristische Eigenschaft dieses Formats ist die Tokenisierung der Texte. Jedes Zeile der Datei wird aus einem Tupel gebildet der aus einem einzelnen Wort, gefolgt von einem Tag der Named Entity Notation, besteht. Satzenden werden als Leerzeilen markiert.

  Wir nutzen folgende automatisch im AnnotationTool [K04] erzeugte Typen der WTF Dokumente:

  – [V03] Drucksachen des Deutschen Bundestages
  – [V03] Protokolle des Deutschen Bundestages
  – [V07] Abstimmungen aus den Protokollen des Dt. Bundestages.

Zusätzlich gibt es zwei Typen von Dokumenten die von einem Benutzer im AnnotationTool [K04] per Hand markiert worden sind. Diese sind ebenso als Eingabe für den RapidMiner [K18] gedacht, allerdings wird auf dieser Eingabe gelernt um im nächsten Durchlauf nicht-markierte Dokumente zu markieren.

  - [V03] Drucksachen und Protokolle für den NER Learner
  - [V07] Protokolle für den Learner zur Extraktion von Abstimmungen

- CSV Character Separated Values
  Der Crawler [K00] verwaltet diese Semikolon separierten Listen um Änderungen im Dokumentarchiv auf der Webseite des Bundestages nachzuhalten. Dem entsprechend werden nur dem System unbekannte, oder geänderte Dateien heruntergeladen.

  - [D00] Drucksachen des Deutschen Bundestages
  - [D00] Protokolle des Deutschen Bundestages
  - [D00] Drucksachenbänder des Bundestages

- AML RapidMiner Experimente
  RapidMiner [K18] ist in der Lage innerhalb seiner GUI Prozesse abzubilden. Durch Hintereinanderschaltung von Komponenten sind komplexe Abläufe möglich. Diese Abläufe können dauerhaft als AML Dateien gespeichert werden um zu einem späteren Zeitpunkt das selbe Experiment noch einmal, ohne Benutzerintervention auszuführen.

  - [V05] NER Experimente
  - [V07] Event Extraction Experimente
  - [V08] Experimente für die QueryFacade Datensätze

- DAT Semikolon separierte RapidMiner Datensätze
  RapidMiner [K18] benötigt als Eingabe eine Semikolon separierte Tabelle in der die Daten zeilenweise aufgeführt sind.

  - [V08] QueryFacade Datensätze zur Beantwortung der Warum-Fragen.

2.4.4 Physikalische Struktur

Das Repository besteht physikalisch aus einem Verzeichnis im Dateisystem des Rechners auf dem das System installiert wurde. Datenbanken und Dateisammlungen befinden sich in den Unterverzeichnissen. Wie üblich für Verzeichnisse im Dateisystem eines Rechners ist die Struktur unseres Repository ein Baum mit */data* als das Wurzelverzeichnis unseres Repository. Alle weiteren Bestandteile befinden sich in den Unterverzeichnissen von */data*.

Im */data* Verzeichnis befinden sich folgende Unterverzeichnisse

/ ascii - [V02] TXT Dokumente

  / btd - Drucksachen des Bundestages
  In den Unterverzeichnissen befinden sich die nach Wahlperioden geordneten Dokumente.

- / btp - Sitzungsprotokolle des Bundestages
  Genau so wie die Drucksachen sind die Protokolle auch nach Wahlperioden geordnet.

/ aus-lookup - [D07] AUSS XML
In diesem Verzeichnis liegt die Datenbank mit Informationen über Ausschüsse des Bundestages.

/ btd-index - [D04] BTD XML

/ btp-index - [D03] BTD XML

/ btpcontent-abstimmung - [V07] EventExtraktion

/ config - Konfigurationsdateien des Systems

/ dictionaries - [V09] Wörterbücher

/ events - [V06] Event Schemas

/ experiments - [V05] NER Experiments

/ import - [V01] PDF Dokumente

- / btd - Drucksachen des Bundestages
  In den Unterverzeichnissen befinden sich die nach Wahlperioden geordneten Dokumente.
- / btp - Sitzungsprotokolle des Bundestages
  Genau so wie die Drucksachen sind die Protokolle auch nach Wahlperioden geordnet.
- / idx - Drucksachenbänder des Bundestages

/ lucene-index - [D01b] BTD Index
In diesem Verzeichnis liegt der Lucene Index der Drucksachen und Protokolle.

/ lucene-speeches - [D01a] BTD Index
Hier befindet sich der Lucene Index für die Redebeiträge der Abgeordneten.

/ mops - [D06] MOPS XML

/ nerd - [V03] WTF Dokumente

/ btd - Drucksachen des Bundestages

/ btp - Sitzungsprotokolle des Bundestages

/ relationen - [D02] BTP2BTD

/ speech-extraction - [V04] XML Dokumente
Hier befinden sich nach Wahlperioden geordnete Redebeiträge als XML Dokumente, die von der SpeechExtraction [K09] Komponente erstellt wurden.

/ xml - [V04] XML Dokumente
Hier befinden sich die von WTF nach XML konvertierten, annotierten Drucksachen und Protokolle.

Diese Verzeichnisstruktur ist sie während der Entwicklung gewachsen und wir haben uns, da es sich um einen Prototypen handelt, nicht um durchgehende Konsistenz zwischen Verzeichnis und Komponentennamen bemüht. Es wurde jedoch stets dafür gesorgt, dass jede vorhandene Komponente die Pfade zu den von ihr benötigten Datenbanken und Verzeichnissen kennt.

2.5 Prozess Modell und Beziehungen zwischen den Subsystemen

Dieser Abschnitt beschreibt die Zusammenhänge zwischen den Komponenten und den Datenbanken und dient der allgemeinen Übersicht im System. Die dargestellten Zusammenhänge werden nur grob an Hand einfacher Prozessdiagramme erläutert. Detaillierte Beschreibungen der Funktionsweise vorhandener Komponenten befinden sich in den darauf folgenden Kapiteln. Als Konvention zur Prozessbeschreibung wird auf SDL zurückgegriffen.

2.5.1 Grundsätzliches zu atomaren Prozessen

Prozesse sind in Bezug auf die Verarbeitung der Datenstrukturen möglichst atomar und in einzelnen Komponenten gekapselt. Die Komponenten der Repository Architektur sind darauf ausgelegt Dateien aus dem Repository zu holen, zu verarbeiten und ggf. wieder im Repository abzulegen. Das vereinfacht die vorhanden Schnittstellen und macht das System robuster gegenüber Änderungen in der Programmstruktur. Änderungen der Datenstrukturen sind natürlich für alle Systeme mit vergleichbaren Aufwänden verbunden, unabhängig von der Architektur.

Da die Dateien nicht erst bei der Benutzeranfrage verarbeitet werden, sondern a priori, ist diese IO-lastige Art Daten zu verarbeiten nicht weiter schlimm. Zusätzlich ist sie gegenüber Programmabbrüchen robust, da man in dem Prozessschritt und in der Datei weitermachen kann, an der die Verarbeitung abgebrochen wurde.

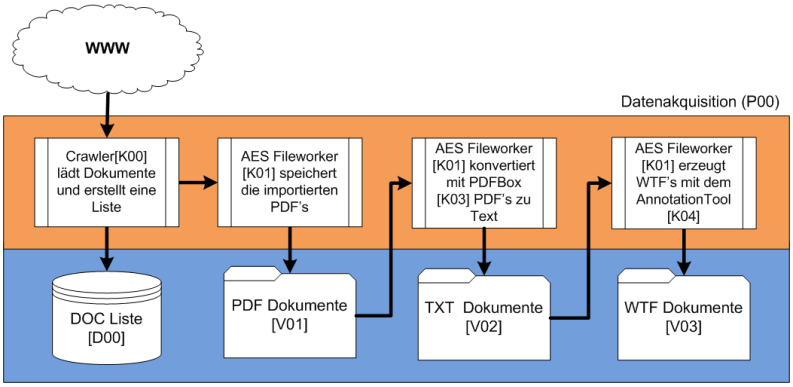


Abbildung 2.7: P00 - Datenakquisition

2.5.2 Datenakquisition

Die Abbildung 2.7 billdet den Prozess der Datenakquisition als Teil des AES Subsystems ab. Der Prozess wird hier nur kurz angerissen und selbstverständlich im späteren Kapitel detailiert beschrieben. Deshalb zeigen wir hier nur das Zusammenspiel des AES Fileworker [K01] mit dem Repository. Der Fileworker ist für die Ausführung der Prozessschritte und die Koordination der beteiligen Komponenten zuständig. Ähnlich einem Batch Lauf lädt der Crawler [K00] die noch nicht erfassten Dokumente, erzeugt die entsprechenden Einträge in seine Liste und übergibt sie an den Fileworker. Die Dokumente werden im Repository gespeichert. Im nächsten Schritt werden die Dokumente zuerst aus dem pdf-Format in Plain Text und schliesslich in das wtf-Format konvertiert. Nach jeder Konvertierung werden die Ergebnisse in Verzeichnissen gespeichert.

2.5.3 Prozesse der Informationsextraktion

Die Informationsextraktion ist eine ins gesamt komplexe Aufgabe, die wir auf einfachere, teilweise unabhängige Prozesse aufgeteilt haben. Sie besteht aus fünf Unterprozessen und kann zum teil parallel verarbeitet werden. Sollte das System auf verschiede Server aufgeteilt werden, so besteht dadurch die Möglichkeit zumindest paarweise die Prozesse zu verarbeiten. Parallele Verarbeitung ist logischerweise immer dann Möglich wenn die Quell und Ziel Verzeichnisse oder Datenbanken unterschiedlich sind.

Informationen über den Bundestag, Ausschüsse und die Abgeordneten werden in drei XML Datenbanken vorgehalten. Der Aufbau dieser Datenbanken ist in der Abbildung 2.12 dargestellt. Diese drei Prozessschritte sind zwar im Prinzip von einander Unabhängig, sollten aber in der dargestellten Reihenfolge ablaufen. Zumindest bei der Initialbefüllung der Datenbanken. MOPS XML [D06] ist hierbei die zentrale Datenbank, welche die Personen ID's verwaltet.

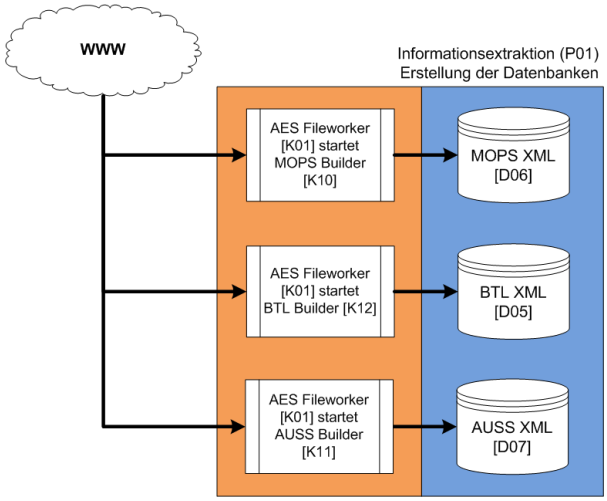


Abbildung 2.8: [P01] - Erstellung der Datenbanken

Abbildung 2.9 zeigt die Verarbeitung der Dokumente in der Informationsextraktion. Ausgehend von den in Plain Text konvertierten Dateien werden drei unterstützende Datenbanken aufgebaut aus denen im späteren Verlauf, und zwar beim Aufbau der Indizes Informationen eingeholt werden. BTD2BTP [D02] wird benötigt um in Lucene [K02] die Beziehungen zwischen den Drucksachen und Protokollen herzustellen. BTD XML [D04] und BTP XML [D03] werden mit Meta-Daten befüllt. Lucene [K02] erzeugt nicht nur den Index über den Dokumenten, es speichert auch diese Meta-Daten als Attribute zu jedem Dokument. Abbildung 2.11 verdeutlicht diese Beziehung. Schließlich extrahiert die SpeechExtraction [K09] Komponente die Reden aus den Protokollen und legt sie als XML annotierten Texte ab.

Der über eine eigene GUI verfügende AES Fileworker [K01] spielt bei der Verarbeitung der Dokumente wieder eine zentrale Rolle. Diese Komponente startet die datenbankerzeugenden Komponenten und überwacht die Ausführung.

Die Named Entity Recogition (NER) spielt bei der Informationsextraktion unseres Systems eine sehr wichtige Rolle. Auf diesem Prozess baut die gesamte, weitere Verarbeitung, welche später für die Extraktion der Relationen notwendig ist. In der NER werden die ins WTF Dateiformat konvertierten Dokumente in einem Lauf vom RapidMiner [K18] annotiert. Der in der Abbildung 2.10 dargestellte Prozess induziert die Verantwortung des AES Fileworker [K01] für die Schnittstelle zum RapidMiner [K18]. Diese Stapelverarbeitung überschreibt die WTF Dokumente und übergibt bei Erfolg die Kontrolle wieder an den Fileworker. Im letzen Schritt des Prozesses werden die Dokumente nochmals ins XML Format konvertiert. Dies erzeugt den Corpus für die Relation Extraction zwischen den Named Entities.

Die Grundlage jeder Stichwortsuche in den Dokumenten ist ein invertierter Index, also die Zuordnung zwischen den Wörtern und Dokumenten. Dessen Aufbau wird aus dem AES Fileworker [K01] angestossen und von Lucene [K02] durchgeführt. Abbildung 2.11 zeigt den Aufruf von Lucene und die Datenquellen die für den Aufbau benötigt werden.

Der dargestellte Prozess zeigt beide Lucene Instanzen die den jeweils eigenen Index aufbauen. Wir benutzen einen großen Index über alle Drucksachen und Protokolle, sowie in der zweiten Instanz einen kleineren Index über den Redebeiträgen der Abgeordneten. Gleichzeitig wer-

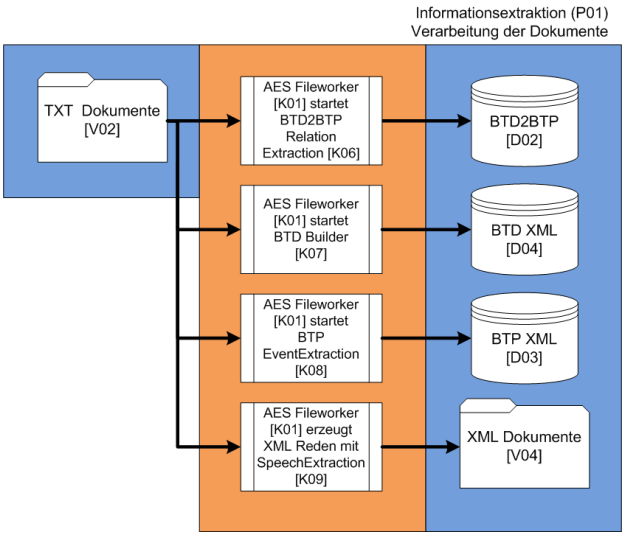


Abbildung 2.9: [P01] - Verarbeitung der Dokumente

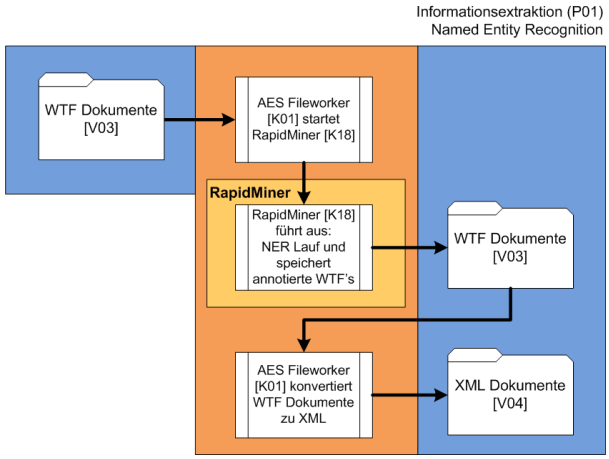


Abbildung 2.10: [P01] - Named Entity Recognition

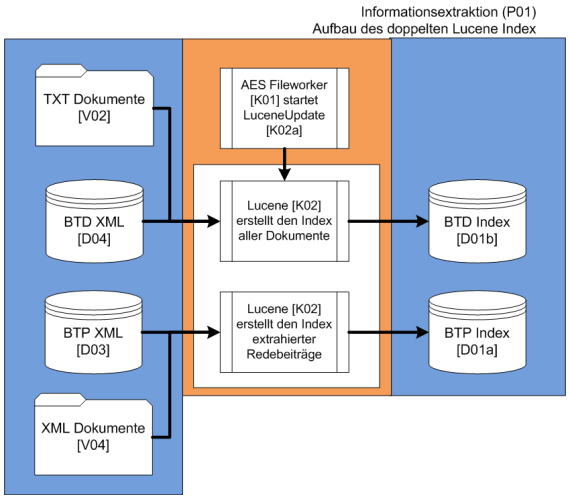


Abbildung 2.11: [P01] - Aufbau der Indizes

den zu jedem Dokument, welches gerade verarbeitet wird, die zuvor extrahierten Meta-Daten aus den Datenbanken abgerufen und gespeichert. Lucene bildet ein eigenes Objekt als Repräsentation eines indexierten Dokuments, diesem Objekt werden die Attribute hinzugefügt.

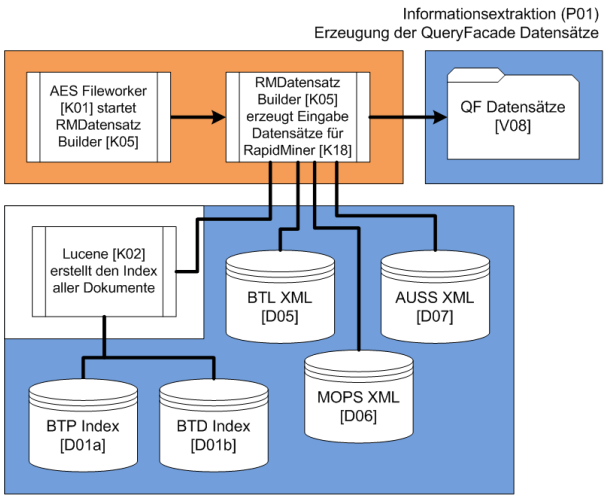


Abbildung 2.12: [P01] - QueryFacade Datensätze

Im letzten, finalen Prozess werden aus allen Datenbanken, Semikolon separierte Datensätze für die RapidMiner [K18] Eingabe generiert. Die zuständige Komponente: RMDatensatz Builder [K05] erstellt für jede Wahlperiode eine dat-Datei mit der die Beantwortung der so genannten „Warum-Fragen“ möglich ist.

2.5.4 Fragebeantwortung

Das Subsystem der Fragebeantwortung ist aus Ideen der Projektgruppe und der PG-Leitung gewachsen. Einige Komponenten wurden zum Campusfest der TU Dortmund entwickelt um die Attraktivität der Anwendung für das Publikum beurteilen zu können. Die Entwicklung des FBS und seiner Komponenten war stark an die Erstellung der Datenbanken gekoppelt. Da MOPS XML [D06] und der Index BTDIndex [D01b] unsere ersten Datenbanken waren, ist es nicht weiter verwunderlich, dass MopsQuery [K10a] und der EventCutter [K14] unsere zuerst entwickelten Komponenten waren.

Ursprünglich wurde der Index von einer Lexical-Tree Komponente verwaltet, diese fiel jedoch dem Java Performance- und Speicherhunger zum Opfer. Der alte Index wurde deshalb gegen den mächtigeren Lucene Index ausgetauscht, der neben RapidMiner, zentrale Rolle in der Fragebeantwortung spielt.

Auf den folgenden Seiten wollen wir die einzelnen, von einander unabhängigen Prozesse kurz skizzieren. In allen Abbildungen verbinden Pfeile die aufeinander folgenden Prozessschritte. Durchgezogene Linien symbolisieren die Verwendung einer Datenbank durch einen Prozess. Das FBS kann auf Grund seiner Prozesse als Frontend auf einem oder mehreren Servern installiert werden. Parallelität kann erreicht werden durch Auslagerung der Komponenten und der dazugehörenden Datenbanken auf separate Maschinen.

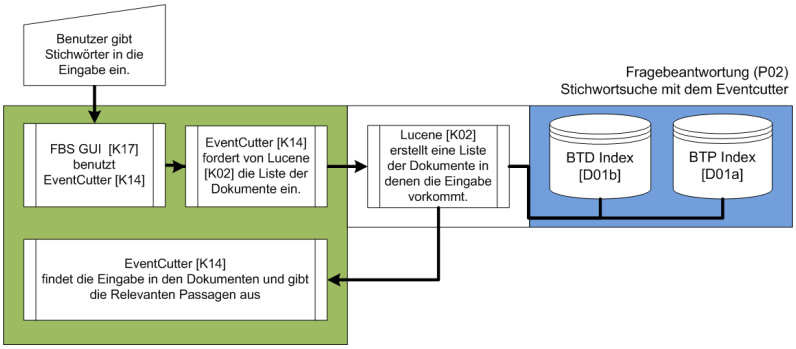


Abbildung 2.13: [P02] - Prozesse der Stichwortsuche

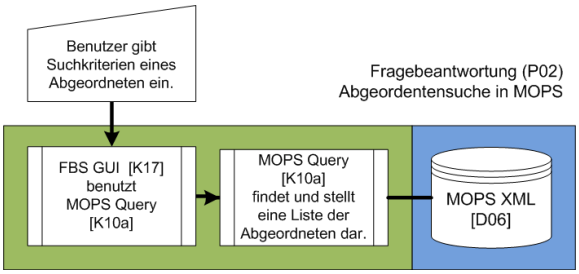


Abbildung 2.14: [P02] - Fragebeantwortung mit MOPS Query

Die Stichwortsuche wird über ein einfaches Formular in der GUI gestartet. Die Tokens der Benutzereingabe werden an den EventCutter [K14] weitergeleitet, der wiederum Lucene [K02] mit dieser Eingabe versorgt. Lucene liefert eine Liste aus Pfaden zu den relevanten PDF Dokumenten. Der EventCutter öffnet die Dokumente und schneidet die Passagen aus, in denen die Stichwörter vorkommen. Dieses Passagen und ein Verweis auf das PDF werden als Ausgabe für den Benutzer dargestellt. Abbildung 2.13 zeigt den groben Aufbau aus GUI [K17], EventCutter [K14], Lucene [K02] und die beteiligten Indizes.

Die Abbildung 2.14 zeigt die Funktionalität der MOPS Query [K10a]. Diese Komponente stellt die Algorithmen bereit um die MOPS XML [D06] Datenbank zu durchsuchen und einfache Fragen zu den Abgeordneten des Bundestages zu stellen.

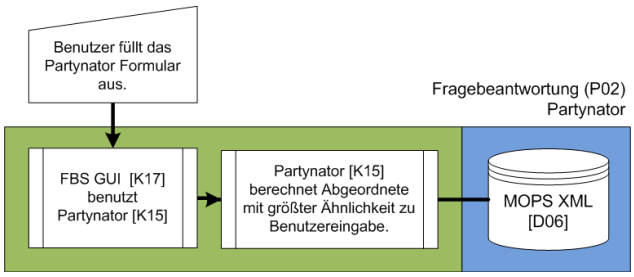


Abbildung 2.15: [P02] - Partynator

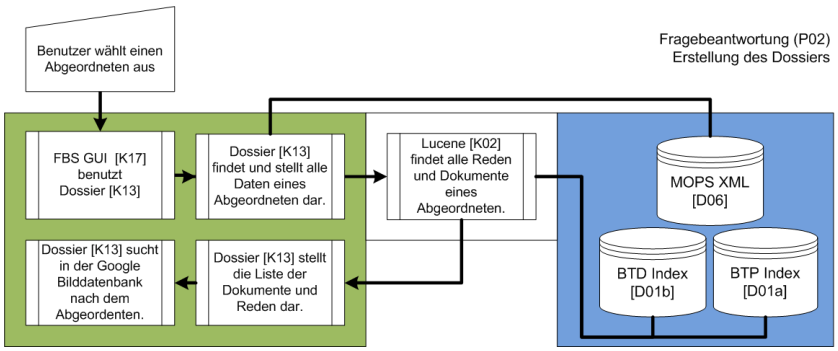


Abbildung 2.16: [P02] - Aufbau des Dossiers als Prozesse

Zum Campusfest der Technischen Universität Dortmund wurde eine Komponente geplant und implementiert, mit der das Publikum eigene Daten mit den Daten der Abgeordneten vergleichen konnte. Abbildung 2.15 zeigt diese Partynator [K15] genannte Komponente. Partynator stellt Formularfelder bereit, die den Feldern in der MOPS XML [D06] entsprechen. Der Benutzer ist angehalten diese mit eigenen Daten auszufüllen. Beim Absenden dieser Partynator Abfrage wird die größte Ähnlichkeit zu einem oder mehreren Abgeordneten errechnet und daraus die mögliche Bereitschaft der Wahl einer Bundestagspartei als Kreisdiagramm dargestellt. Natürlich werden die vom Benutzer eingegebenen Daten nicht gespeichert oder weiterverarbeitet.

Das Dossier ist eine Komponente die aus verschiedenen Datenquellen alle verfügbaren Informationen zu einem Abgeordneten sammelt und darstellt. In der Abbildung 2.16 wird dieser Prozess an Hand von wenigen Schritten kurz skizziert. Die GUI stellt eine alphabetisch geordnete Liste der Abgeordneten dar und bietet zusätzlich ein Suchformular mit dem der Benutzer direkt nach dem Abgeordneten suchen kann. Im ersten Schritt werden die allgemeinen Informationen über den Abgeordneten in der MOPS XML [D06] Datenbank nachgeschlagen. Worauf die Listen der Drucksachen und Reden des Abgeordneten bei Lucene angefragt werden. Im letzten Schritt, also direkt vor der Darstellung der Daten wird die Google Bilddatenbank nach einem Bild des Abgeordneten befragt. Die gesammelten Informationen werden abhängig von der Wahlperiode dargestellt.

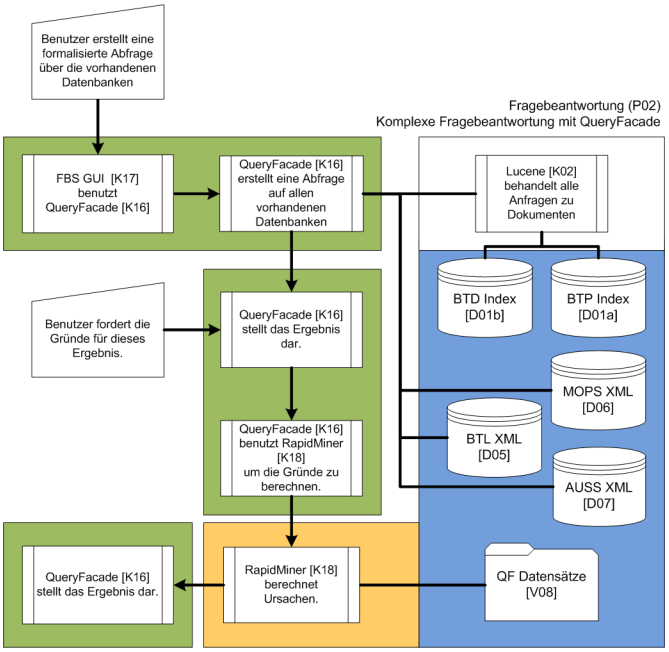


Abbildung 2.17: [P01] - QueryFacade und die „Warum“-Fragen

Abbildung 2.17 zeigt den Ablauf der komplexen, datenbankübergreifenden Anfragen die mit QueryFacade [K16] beantwortet werden. QueryFacade bildet eine Schicht zwischen Datenbanken und der eigentlichen Fragebeantwortung, welche die XML Datenbanken und die Lucene Ausgaben abstrahiert. Damit ist der Benutzer in der Lage strukturierte, SQL-ähnliche Abfragen auf allen vorhandenen Datenbanken. Der erste Teil des Prozesses beantwortet eben diese strukturierten Fragen. Zusätzlich besteht die Möglichkeit nach den Ursachen für ein Ergebnis zu fragen. Hierfür werden an Hand der vorhin zur Beantwortung genutzten Daten, entsprechende Datensätze aus dem QF Datensätze [V08] Verzeichnis geladen und mit einer Anfrage an RapidMiner [K18] übergeben. RapidMiner erstellt eine Antwort und diese wird von QueryFacade in der GUI dargestellt.

Wie bereits erwähnt ist dies nur eine grobe Einführung in die Komponenten und Abläufe innerhalb des Systems. Die Komponenten werden noch mal ausführlich in den folgenden Kapiteln beschrieben. Den eigen entwickelten Komponenten ist ein Kapitel am Ende des Projektberichts gewidmet.

3 Datenakquisition

In Kapitel 1.12.4 wurde bereits das Format der Dokumente vorgestellt, welche wir zur Weiterverarbeitung zur Verfügung stehen haben. Mit diesen PDF-Dateien können wir in den weiteren Schritten aber nicht weiterarbeiten, deshalb müssen diese in ein für unsere Weiterverarbeitung sinnvolles Format konvertiert werden.

3.1 Corpus-Erstellung

Zunächst müssen alle Daten, aus denen der Corpus erstellt werden soll, lokal vorliegen. In unserem Fall werden also alle Drucksachen des Deutschen Bundestages, sowie alle Plenarprotokolle des Deutschen Bundestages im PDF-Format benötigt. Aus diesen soll nun ein Corpus erstellt werden, mit dem das System weiterarbeiten kann.
Das Herunterladen der Dokumente soll automatisch geschehen, wie in Kapitel 3.1.3 dargestellt, dies soll über einen Crawler erzielt werden. Dies ist alleine schon sinnvoll, da auf diese Weise das automatische mitaufnehmen neuer Dokumente leichter zu realisieren ist.

3.1.1 Umwandlung von PDF zu regular ASCII

In Abbildung 3.1 ist zu sehen, wie die Umwandlung von PDF zu regular ASCII vorgenommen wird.
Zunächst wird der eigentliche Umwandlungsschritt von PDF zu Ascii mit Hilfe der PDF-Box vorgenommen. PDFBox ist eine Open-Source Java PDF-Bibliothek und hilft dabei, mit PDF-Dokumenten arbeiten zu können. Mit Hilfe der PDFBox ist es möglich den Inhalt der PDF-Dokumente zu extrahieren. Diese musste jedoch für uns angepasst werden, da ihre eigentliche Version zum Teil sehr schlechte Ergebnisse für die PDF-Dateien des Bundestages erzielt hat.
Nach diesem Hauptumwandlungsschritt werden im Anschluss diverse Textbereinigungen durch-

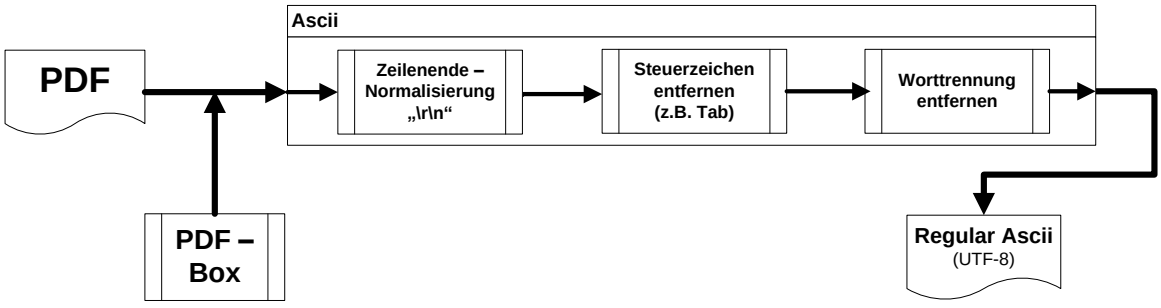
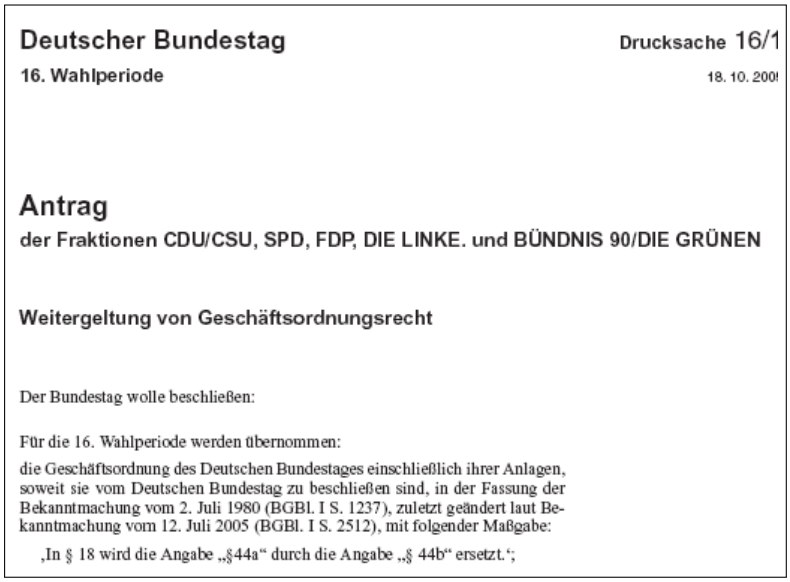


Abbildung 3.1: PDF nach regular Ascii



Deutscher Bundestag Drucksache 16/1

16. Wahlperiode 18. 10. 200!

Antrag

der Fraktionen CDU/CSU, SPD, FDP, DIE LINKE. und BÜNDNIS 90/DIE GRÜNEN

Weitergeltung von Geschäftsordnungsrecht

Der Bundestag wolle beschließen:

Für die 16. Wahlperiode werden übernommen:

die Geschäftsordnung des Deutschen Bundestages einschließlich ihrer Anlagen, soweit sie vom Deutschen Bundestag zu beschließen sind, in der Fassung der Bekanntmachung vom 2. Juli 1980 (BGBl. I S. 1237), zuletzt geändert laut Bekanntmachung vom 12. Juli 2005 (BGBl. I S. 2512), mit folgender Maßgabe:

‚In § 18 wird die Angabe „§44a" durch die Angabe „§ 44b" ersetzt.';

Abbildung 3.2: Ausschnitt einer Drucksache im PDF-Format

geführt. Es wird zum einen eine Zeilenenden-Normalisierung vorgenommen. Das bedeutet, dass alle Zeilenenden auf \r\n umgeändert werden, um eine einheitliche Formatierung zu bekommen.
Desweiteren werden sämtliche Steuerzeichen entfernt. Das können z.B. Tabulatoren oder mehrere Leerzeichen hintereinander sein. Diese sind für unsere weitere Verarbeitung unnütze Informationen, da sie nur die Formatierung im PDF-Dokument betrafen und sie für unsere weitere Verarbeitung eher hinderlich sind. Für diese wird lediglich eine reine Textversion benötigt.
Schließlich müssen alle Worttrennungen entfernt werden. Dies ist ein sehr wichtiger Schritt um später mit den Dokumenten arbeiten zu können. Zum einen würde ohne diesen Schritt ein Wort wie Bundestag nicht gefunden, wenn es mit Bundes- tag im Dokument enthalten wäre und zum anderen würden alle vorhandenen Trennungen als zwei eigene Wörter indexiert werden.
Nachdem alle diese Umwandlungsschritte vorgenommen wurden, soll die nun reguläre Ascii-Datei gespeichert und mit Lucene indexiert werden.

Beispielhafte Umwandlung einer PDF-Datei nach regular ASCII: In Abbildung 3.2 ist ein Auszug der, die PDF-Datei darstellt, die diesem Beispiel zu Grunde liegt.
Nachdem alle „PDF zu Ascii"-Transformationen vorgenommen wurden, ist der in Abbildung 3.3 zu sehende Text entstanden.
In diesem Dokument wurden einige Zeilenendennormalisierungen vorgenommen. Eine z.B. zwischen der zweiten und der dritten Zeile. Bereits in der ersten Zeile ist zu sehen, dass dort einige Steuerzeichen zwischen „Deutscher Bundestag" und „Drucksache 16/1" entfernt wurden.
Bei der Textpassage „zuletzt geändert laut Bekanntmachung vom 12. Juli 2005" ist eine Entfernung einer Worttrennung vorgenommen worden. So wurde das Wort „Bekanntmachung" im PDF-Dokument noch „Be- kanntmachung" geschrieben.

Deutscher Bundestag Drucksache 16/1
16. Wahlperiode 18. 10. 2005
Antrag
der Fraktionen CDU/CSU, SPD, FDP, DIE LINKE. und BÜNDNIS 90/DIE GRÜNEN
Weitergeltung von Geschäftsordnungsrecht
Der Bundestag wolle beschließen:
Für die 16. Wahlperiode werden übernommen:
die Geschäftsordnung des Deutschen Bundestages einschließlich ihrer Anlagen,
soweit sie vom Deutschen Bundestag zu beschließen sind, in der Fassung der
Bekanntmachung vom 2. Juli 1980 (BGBl. I S. 1237), zuletzt geändert laut
Bekanntmachung vom 12. Juli 2005 (BGBl. I S. 2512), mit folgender Maßgabe:
‚In § 18 wird die Angabe „§44a“ durch die Angabe „§ 44b“ ersetzt.‘;

Abbildung 3.3: Die Drucksache nach der Umwandlung in das regular ASCII Format

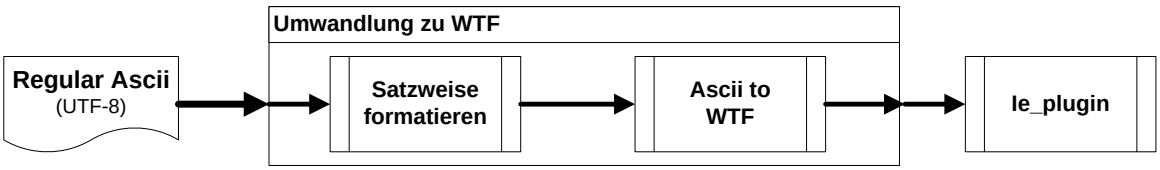


Abbildung 3.4: regular Ascii ins WTF-Format für das RapidMiner ie-Plugin

3.1.2 Umwandlung von regular ASCII zu WTF

In Abbildung 3.4 ist zu sehen wie die Umwandlung von regular ASCII zu WTF vorgenommen wird. Dieses Format wird als Eingabe für das RapidMiner ie-Plugin benötigt.
Als Eingabe dient in diesem Schritt das im vorangegangenem Abschnitt erläuterte regular Ascii Format. Dieses soll nun in das „Wort-Tag“-Format (WTF) konvertiert werden, um z.B. mit dem ie-Plugin NE's labeln zu können.
Dabei wird wie folgt verfahren: Zunächst wird das Dokument Satzweise formatiert, da später im WTF-Format zwei Sätze durch eine Leerzeile von einander getrennt werden. Für diese Aufgabe wird der *SentenceSplitter* benutzt. Dieser sucht die Satzenden in einem Dokument und markiert diese mit einem Tag (<end/>). Abkürzungen oder Organisationen die Punkte enthalten sollen nicht als Satzende identifiziert werden. Deshalb benutzt der *SentenceSplitter* unter anderem Abkürzungswörterbücher, für Abkürzungen wie Bsp. oder F.D.P., und reguläre Ausdrücke, um z.B. Datumsangaben zu erkennen. Satzzeichen die das Ende eines Satzes kennzeichnen, wie z.B. !,. oder ?, werden entfernt und durch ein Tag ersetzt, welches für die Weiterverarbeitung in das WTF-Format eindeutig kennzeichnet, dass hier ein Satzende vorliegt.
Nun wird dieses Zwischenformat mit den <end/>-Tags in das WTF-Format konvertiert. Jedes Wort bekommt hier eine eigene Zeile. In jeder Zeile gibt es zwei Spalten, welche einfach durch ein Leerzeichen separiert werden. In der ersten Spalte steht das jeweilige Wort und

```
Deutscher O
Bundestag O
Drucksache O
16 O
/ O
1 O
16 O
. O
Wahlperiode O
18 O
. O
10 O
. O
2005 O

Antrag O
der O
Fraktionen O
CDU/CSU O
```

Abbildung 3.5: Die Drucksache im WTF-Format

in der zweiten Spalte das Tag, das diesem Wort zugeordnet wurde. In diesem Fall wird ein unannotiertes Dokument im WTF-Format erstellt. Das bedeutet das jedes Wort nur das „leere Tag“ (O) zugewiesen bekommt. Über das *Annotation-Tool* kann ein solches Dokument auch manuell getaggt werden (siehe Kapitel 3.2). Außerdem kann über das ie-Plugin anhand eines mit Trainingsdaten gelernten Modells eine automatisierte Zuordnung von Tags gestartet werden (siehe Kapitel 4.2). Für jedes Wort wird also ein Zeilenumbruch eingefügt. Analog wird für jedes Satzende eine Leerzeile eingefügt. Nach diesen Schritten ist ein Dokument nun im WTF-Format vorliegend.

Beispielhafte Umwandlung einer regular ASCII Datei in das WTF-Format: In Abbildung 3.5 ist ein Auszug aus dem obigen Beispiel im WTF-Format zu sehen.
Hier ist z.B. zu erkennen, dass die Punkte im Datum, wie gewünscht, nicht als Satzendezeichen erkannt wurden.

3.1.3 Der FileWorker

Der *FileWorker* ist das Tool, welches die Transformation aus den Abschnitten 3.1.1 und 3.1.2 vornimmt.
In Abbildung 3.6 ist das Hauptmenü des *FileWorkers* dargestellt. Mit der Option „Quell-Verzeichnis “ muss ein Ordner ausgewählt werden. Auf diesen und alle seine Unterordner wird der ausgewählte Filter angewandt, sofern die Datei das passende Eingabeformat hat.
Unter „Filter wählen “ wird der gewünschte Verarbeitungsschritt eingestellt. Dabei stehen

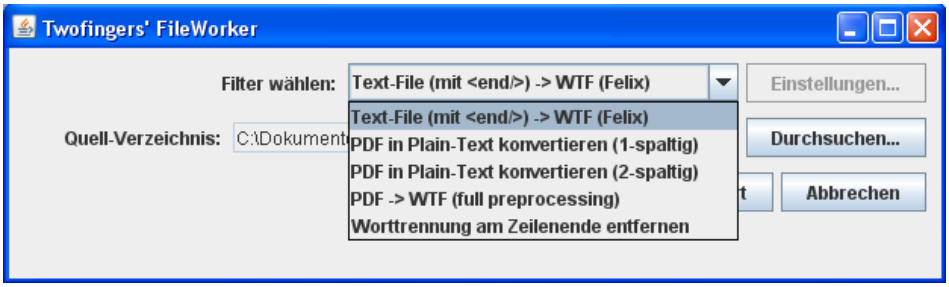


Abbildung 3.6: Das Menü des FileWorkers

die folgenden möglichen Umwandlungsschritte zur Wahl:

- *Text-File (mit <end/>) → WTF*
  Dieses war ein Umwandlungsschritt der benötigt wurde, bevor der SentenceSplitter fertig gestellt wurde. So konnten regular Ascii Dokumente per Hand mit <end/>-Tags versehen werden und dann automatisch in das WTF-Format konvertiert werden. Dieser Schritt ist mit Fertigstellung des SentenceSplitters überflüssig geworden.

- *PDF in Plain-Text konvertieren (1-spaltig)*
  Dieser Umwandlungsschritt beschreibt die in Abschnitt 3.1.1 vorgestellten Schritt, allerdings ohne dass Worttrennungen entfernt werden. Dieser Schritt ist auf 1-spaltiges Layout optimiert worden.

- *PDF in Plain-Text konvertieren (2-spaltig)*
  Dieser Konvertierungsschritt ist identisch zum vorangegangenen Schritt, mit der Ausnahme, dass dieser Schritt auf 2-spaltiges Layout optimiert worden ist.

- *PDF → WTF (full preprocessing)*
  Hierbei werden alle benötigten Umwandlungen in einer Ausführung durchgeführt. Das heißt, dass sowohl die Schritte aus Abschnitt 3.1.1, als auch die Schritte aus Abschnitt 3.1.2 hintereinander ausgeführt werden. Dieses ist die für uns wichtigste Umwandlung, da bei ihr sowohl die regular Ascii Dateien, als auch die WTF-Dateien in einem Schritt ausgegeben werden. Deshalb hat diese Transformation auch noch zusätzliche Einstellungsmöglichkeiten (siehe Abbildung 3.7), welche das Abspeichern der erstellten Dateien betreffen.

- *Worttrennung am Zeilenende entfernen*
  In diesem Konvertierungsschritt wird lediglich der Worttrennungsschritt aus 3.1.1 durchgeführt. Das heißt, dass die Eingabe eine Ascii-Datei sein muss und als Ausgabe, eine regular Ascii Datei entsteht.

Nachdem der FileWorker sämtliche Daten des Deutschen Bundestag bearbeitet hat, liegen uns nun für alle weiteren Schritte sowohl die regular Ascii Dateien vor, als auch die gleich ungetaggten Dokumente im WTF-Format.

BTD/BTP-Crawler

Am Anfang der Datenakquise steht die Beschaffung der PDF-Dokumente, um sie dann weiter verarbeiten zu können. Beschaffung heißt in diesem Fall, dass sie von der Webseite des Bun-

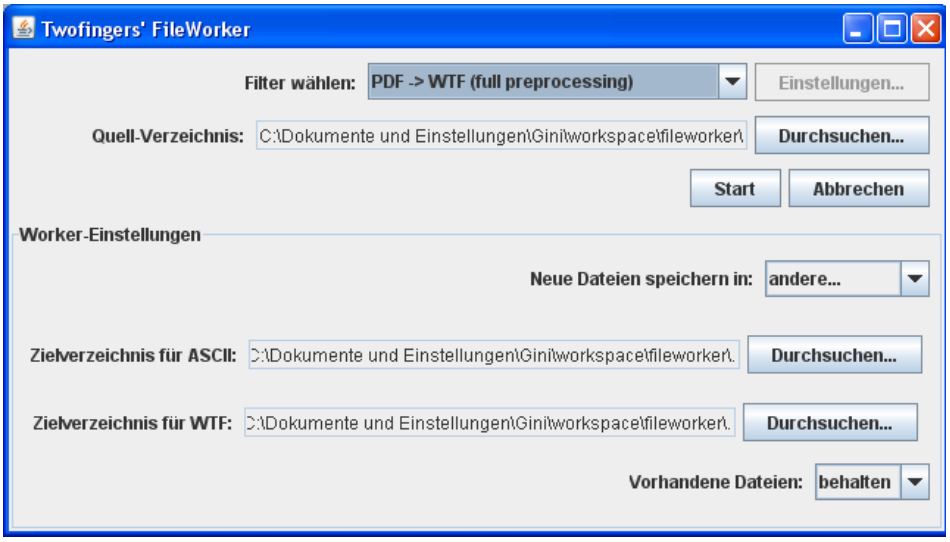


Abbildung 3.7: Das erweitere Menü des FileWorkers, beim Filter PDF → WTF

destages heruntergeladen werden müssen. Nur wenn dieser Schritt automatisiert ist, können unsere Datenbanken auf einfache Weise aktuell gehalten werden und neue Informationen integrieren. Diesen automatisierten Download erledigt der BTD/BTP-Crawler.
Der Crawler gliedert sich in zwei logische Bereiche, die durch die unterschiedlichen Speicherorte der PDFs bestimmt sind. Die Dokumente der Wahlperioden 14, 15 und die Anfänge der 16. Wahlperiode sind in das DIP (dip.bundestag.de) eingebunden. Alle aktuelleren Dokmente sind außschließlich in das neue Informationssystem DIP21 (dip21.bundestag.de) aufgenommen worden. Das sich DIP und DIP21 grundlegend unterscheiden, behandelt sie der Crawler auch unterschiedlich.
Der DIP-Webserver bietet die Möglichkeit, sich, ähnlich eines Dateimanagers, durch das Dateisystem der Dokumente zu hangeln. Es sind über- und untergeordnete Dateiordner und die darin enthaltenen Dateien anwählbar. Der Crawler lädt die Webseite mit der obersten Verzeichnisebene herunter und arbeitet sich anhand der Links der Webseite durch den Verzeichnisbaum. Die Links werden mit regulären Ausdrücken gefunden. Da auch die PDFs verlinkt sind, können sie direkt heruntergeladen werden.
Die Beschaffung der PDFs vom DIP21-Server gestaltet sich hingegen ungemein schwieriger, da die Struktur des Dateisystems nicht mehr in Form von verlinkten Webseiten verfügbar ist. Ein Link auf ein PDF-Dokument lässt sich nur noch über die Verwendung des DIP21-Suchformulars erreichen. Auch wenn sie nicht vollständig einsehbar ist, so ist im DIP21 aber glücklicherweise die Ordner-Struktur des DIP beibehalten worden. Kennt man also die Drucksachen- oder Plenarprotokollnummer, so lässt sich daraus der Pfad zur PDF-Datei rekonstruieren. Der DIP21 Crawler verfolgt demnach die Strategie, die Nummer des zuletzt veröffentlichten BTP bzw. BTD auszumachen und dann einfach die Dokumentennummer herunterzuzählen. Da die Dokumente alle fortlaufend nummeriert sind, schein dies die effizienteste Methode zu sein, auch wenn Lücken in der Nummerierung erst bei erfolgloser Anfrage an den Webserver erkannt werden können. Die Lücken kommen am häufigsten bei den aktuellsten Dokumenten vor, da es immer wieder solche gibt, die zwar ins DIP21 aufgenommen wurden, aber noch nicht im Volltext (als PDF) vorliegen.

Die komplexeste Aufgabe bei der Dokumentenbeschaffung aus dem DIP21 ist also die Bestimmung der Nummer der aktuellsten Drucksache bzw. des Plenarprotokolls. Ein einfaches Heraufzählen der bei eins beginnenden fortlaufenden Nummerieung von Dokumenten einer Wahlperiode kommt nicht in Frage, da so viel zu viele unnötige Anfragen an den Webserver gehen würden und ein Abbruchkriterium festgelegt werden müsste. Wenn die Anzahl der erfolglosen Anfragen gering gehalten wird, läuft man Gefahr zu früh abzubrechen. Daher zählt der Crawler die Dokumentennummer herunter. Die aktuell höchste Nummer wird vorher durch eine automatische Anfrage an das Suchformular des DIP21 gestellt. Angefragt werden alle Dokuemnte, die bis zum dem Tag, an dem die Anfrage ausgeführt wird, veröffentlicht wurden. Dies ist das Ergebnis der Tatsache, dass eine Suche ohne Einschränkungen nicht möglich ist. Durch weitere Parameter werden werden so die aktuellsten 100 Dokumente absteigend sortiert als HTML-Seite zurückgeliefert. Ist zu einem Dokument ein PDF vorhanden, so ist die Dokumentennummer in der Ergebnisliste direkt damit verlinkt. Der Crawler parst das HTML-Dokument nach der ersten verlinkten Dokumentennummer und bestimmt so die aktuelle Wahlperiode und die höchste Dokumentennummer. Danach folgt das Herunterzählen. Ist die aktuelle Wahlperiode abgearbeitet (Dokumentennummer = 1), so wird in die vorherige Wahlperiode gewechselt. Hier stellt sich, zur Vermeidung unnötiger Anfragen, wieder das gleiche Problem, wie zu Anfang: Was ist das letzte Dokument dieser Wahlperiode? Um dies zu bestimmt, wird die bekannte Anfrage an das Suchformular um eine Einschränkung auf die zur Zeit betrachtete Wahlperiode erweitert. Nun funkioniert die Bestimmung analog zur ersten.
Der die älteste Wahlperiode, die der DIP21-Crawler verarbeitet, ist die Periode 16, die zur Zeit der Verfassung dieses Dokuments die aktuelle ist. Durch die zuvor beschriebene Funktionsweise ist der Crawler auch für die Verarbeitung folgender Legislaturperioden geeignet.
Um den Datentransfer gering zu halten, führt der Crawler einen eigenen Index, in der zu jeder herunterladenen Datei das Datum der letzten Modifikation gespeichert wird. Der Index wird als einfache CSV-Datei in jedem Verzeichnis abgelegt. Bei der großen Anzahl an PDFs, wäre ein zentraler Index unhandlich und langsam. Das Änderungsdatum sendet der Webserver bei der Anfrage nach der Datei automatisch mit zurück. Es wird bei einem erneuten Crawling in die Anfrage an den Webserver eingebaut. Das HTTP/1.0 Protokoll [HTT] hält dafür den Header *IfModifiedSince* bereit. Erhält der Webserver eine Anfrage mit diesem Header, kann er neben den sonst üblichen Antworten (*200 - OK*, *404 - File not found*, ...) auch mit *304 - Not Modified* antworten, wenn die angefragte Datei nicht aktueller ist, als das übermittelte Datum. Bei der Antwort *200 - OK* sendet der Webserver die PDF-Datei im Datenbereich der Antwort mit. Die Not-Modified Nachricht reduziert diese Antwort auf die Antwort-Header. Der Datenteil, in dem sich das PDF befände, bleibt leer. So wird der Traffic erheblich reduziert.

Lucene Update

Im Fileworker ist auch eine Komponente enthalten, die die beiden Lucene-Indizes aktuell halten kann. Dazu wird im Lucene-Index zu jedem Lucene-Dokument in einem Feld das Datum der letzten Änderung der Quelldatei gespeichert (zum Aufbau der Lucene-Indizes siehe Kaptiel 5.2). Bei Dokumenten-Index ist dies zum Beispiel das Änderungsdatum einer BTD-ASCII-Datei. Der Updater durchläuft zunächst den gesamnten Lucene-Index und schaut, ob dort Dateien referenziert werden, die es mittlwerweile nicht mehr gibt. Die entsprechenden Einträge werden dann aus dem Index entfernt. Danach werden alle zu indexierenden Dateien durchlaufen. Ist die Datei im Dateisystem aktueller als ihr Zeitstempel im Index oder garnicht

im Index, wird die Datei (neu) indexiert, andernfalls übersprungen. Durch diese Vorgehensweise können die Lucene-Indizies effizient aktuell gehalten werden, ohne jedes Mal den ganzen Index neu zu erzeugen.
Da die Felder der Lucene-Dokumente aber auch aus externe Quellen (XML-Dateien) ihre Daten beziehen, wird auch für diese Quellen ein Zeitstempel gespeichert. Ändert sich eine externe Quelle, werden alle Lucene-Dokumente, die diese referenzieren, ungeachtet ihres eigenen Änderungsdatums neu erstellt.

BTD_Index erneuern

Im Werkzeuge-Menü des Fileworkers kann auch eine neue BTD_Index-Xml für die Dokumente erstellt werden. Hierfür werden alle Dokumente verwendet, welche im data-Verzeichnis unter ascii/btd/ zu finden sind.
Desweiteren wird für die Erstellung der neuen BTD_Index-Xml die Mops-Xml, welche die Daten über die Abgeordneten beeinhaltet, benötigt. Diese wird ebenfalls aus dem data-Verzeichnis, unter mops/results.xml, gelesen. Die Informationen über die Abgeordneten werden benötigt, um den Autoren eines Dokuments die eindeutigen Id's zuzuweisen.
Der über diese Aktion neu erstellte Index wird dann wie gewohnt im data-Verzeichnis unter btd_index/btd_index_update.xml gespeichert.

3.2 Erstellen der Trainingsdaten

Für die Erstellung der Trainingsdaten, wurden zuerst die Dokumente mit Hilfe der PDF-BOX in ASCII-Format konvertiert. Diese sind dann in das Annotion-Tool, welches von Daniel. S. nach unseren Vorstellungen entwickelt wurde, geladen worden. Der Vorteil des Tools für unsere Zwecke war, dass das taggen per Hand erleichtert wurde und es als Ausgabedatei automatisch das WT-Format nutzt. Bevor das taggen per Hand begonnen werden konnte, mussten nun passende NEs (Tags) ermittelt werden, welche in den Dokumenten am häufigsten vorkommen und für uns am nützlichsten erschienen. Es wurde beschlossen nach folgenden Merkmalen zu taggen:

- Name
- Person (zu einer Person gehören auch Namenstitel wie Dr., Prof., Herr, Frau sowie auch Kollege.)
- Ort
- Organisation/Institut/Behörde (wie z.B. Ministerien)
- Reaktionen (Positive oder Negative Reaktionen auf Reden)
- Abstimmungen (Pro, Contra, Unentschlossen und das Ergebnis)
- Amt/Rolle (wie z.B. Bundestagssprecher/in, Sekretär/in, Bundestagspräsident/in, Opposition)
- Partei (wobei für einige Parteien synonyme Bedeutungen auch mit getaggt wurden wie z.B. Die Union, die Grüne.)

Das Datum und die Drucksachennummern werden mit Hilfe von regulären Ausdrücken extrahiert. Es wurde beschlossen Fraktion und Koalition zunächst außer Betracht zu lassen.

Für das manuelle Taggen haben wir zwei Methoden angewendet, wobei die zweite Methode, die erfolgreichere und konsistentere war:

1. Es wurden zufällig zwei Dokumente ausgesucht und unter uns abschnittsweise aufgeteilt, wobei zwei Personen den gleichen Abschnitt getaggt haben. Die ganzen Abschnitte wurden dann zusammengefügt. Das Lernresultat ist inkonsistent gewesen, es wurden z.B. viele Orte und Parteien nicht richtig getaggt.

2. Es wurden dieselben Dokumente getaggt aber diesmal nur von zwei Personen und nach genauer Absprache über Tag-Kriterien. Dadurch konnte das Lernresultat beim anschließenden Lernlauf auf 80% richtig getaggter NEs verbessert werden.

Als Ausgabe liefert das Annotion-Tool, die getaggten Dokumente in WT-Format welche für die Weiterverarbeitung innerhalb des RapidMiner-Tools wichtig ist. Mit Hilfe von RapidMiner sehen wir wie gut gelernt wurde.

3.3 Sentence Splitter

3.3.1 Grundlegendes

Jede nicht-triviale computerlinguistische Verarbeitung von natürlichen Texten erfordert Vorverarbeitung, die im Allgemeinen folgende Teilaufgaben lösen soll:

- (1) Bereinigung der Eingabekette von Sonderzeichen,
- (2) Zerlegung der Eingabekette in einzelne Einheiten, genannt: Tokens,
- (3) Erkennung der Satzenden.

Der SentenceSplitter ist eine Java Komponente, die an Hand von Regeln Satzenden in einem Text markieren soll.

3.3.2 Problematik der Satzendeerkennung

Die Erkennung von Satzenden ist wegen der Ambiguität des Punktes eine schwierige Aufgabe. Der Punkt markiert oft nicht nur das tatsächliche Satzende, sondern auch eine Zahl oder eine Abkürzung. Die Verwendung von Punkten zu Markierung einer bestimmten Zahl innerhalb einer Aufzählung ist ebenso doppeldeutig wie die Markierung von Abkürzungen. Die folgende überspitzte Passage verdeutlicht die Problematik und zeigt wie kleine Formulierungsfehler oder Umgangssprache regelbasierte Erkennung erschweren:

> „Der 14. Tag war genau so sonnig wie der 13. Oktober ist im Allg. sehr verregnet.“

Selbst wenn die Maschine eine Abkürzung oder Aufzählung als solche erkennt, kann man nur mit Hilfe des semantischen Kontexts und der umliegenden Wörter entscheiden, ob es sich bei diesem Punkt um ein tatsächliches Satzende handelt.

3.3.3 Satzenden in den Dokumenten des Dt. Bundestags

Die Texte des Dt. Bundestags liegen im Allg. als PDF Dokumente vor. Für die sinvolle Verarbeitung werden sie von PDF zu Text umgewandelt und von Sonderzeichen bereinigt. Im Zuge dieser Konvertierung fließen einige, vom Typ der Dokumente abhängige Layoutinformationen mit in das Resultat. Die Ausgabe beinhaltet aus dem PDF konvertierte Zeilen, die nicht mit einem Punkt enden, aber dennoch logisch als abgeschlossene Sätze behandelt werden müssen. Dies ist besonders für Überschriften, Kapitel und Listen der Fall. Die ohne hin schwierige Teilaufgabe der Satzendeerkennung, speziell in den Dokumenten des Bundestags wird also durch die Ambiguität von Zeilenenden zusätzlich erschwert.

3.3.4 Markierung von Satzenden

Für die Lösung der oben genannten, dritten Teilaufgabe, ist eine bereinigte Zeichenkette notwendig. Der SentenceSplitter liest eine Eingabekette ein und gibt eine verarbeitete Zeichenkette aus. Die Markierung von Satzenden erfolgt durch das Einfügen eines abgeschlossenen XML Tags: <end/> an jedem erkannten Satzende.

Durch die besondere Form der Eingabekette muss die Verarbeitung Zeilenweise erfolgen, wobei jede Zeile, gemäß Teilaufgabe 2 in Wörter (Tokens) zerlegt wird. Die Trennung erfolgt an jedem Zeichen, dass kein Buchstabe oder Satzzeichen ist (whitespace characters). Die einzelnen Tokens werden in einem indizierten Array gespeichert. Für jede Zeile wird direkt entschieden, ob das letzte Wort der Zeile als abgeschlossener Satz markiert werden soll. Diese Heuristik basiert auf zwei einfachen Regeln, welche die Zeichenlänge und Wortanzahl der Zeile mit zwei gewählten Werten vergleichen. Beide Werte sollen nur die „offensichtlichen“ kurzen Zeilen markieren. Es wäre sinnvoll diese Regel so zu modifizieren, so dass sie abhängig vom Typ und der durchschnittlichen Zeilenlänge des Dokumentes optimale Werte berechnet/nutzt.

3.3.5 Betrachtung der Wörter im Sliding Window Verfahren

Im nächsten Verarbeitungsschritt wird das Array traversiert. Jedes im Array gespeicherte Wort wird auf ein mögliches Satzende von der im Diagramm 3.8 dargestellten Entscheidungspipeline überprüft. In vielen Fällen wird sowohl der Vorgänger, sowie der Nachfolger des Wortes betrachtet. Ein Wort wird um die <end/> Markierung erweitert und an die Zeichenkette der Ausgabe angehängt, falls es als Satzende erkannt worden ist. Die Verarbeitung ist beendet sobald das letzte Wort des Arrays betrachtet worden ist.

3.3.6 Satzendeerkennung mit Regular Expressions und Wörterbüchern

Die einzelnen Bausteine der Entscheidungspipeline sind linear angeordnet. Ein Wort wird von einer “wenn-dann“ Regel zur Nächsten weitergereicht, bis eine Regel zutrifft. Die trivialen Fälle werden direkt am Anfang der Pipeline identifiziert, ggf. Markiert und aus der Pipeline herausgeworfen. Die Erkennung von Satzenden erfolgt mit Hilfe von einfachen, ein

Wort überprüfenden Regular Expressions. Zusätzlich werden Aufzählungen und Abkürzungen gegen Wörterbücher geprüft. Es wurden im Vorfeld folgende Wörterbücher angelegt:

- Vorgänger einer Aufzählung (vom, am, dem, der)

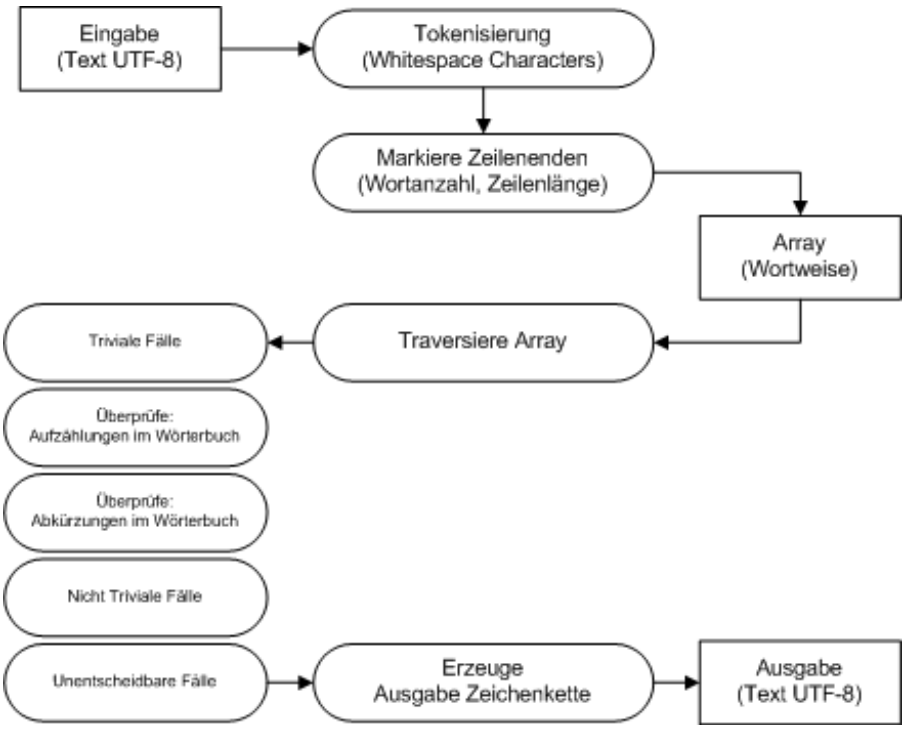


Abbildung 3.8: Aufbau der Komponente

- Nachfolger einer Aufzählung mit Monatsnamen und Wörtern die besonders häufig Vorkommen. (Oktober, Antrag, Sitzung)
- Liste deutscher Abkürzungen, extrahiert mit Hilfe einfacher Skripte aus den Webseiten:
  - www.abkuerzungen.org
  - abkuerzungen.woxikon.de

3.3.7 Ergebnisse

Der SentenceSplitter markiert alle eindeutigen Satzenden. Für die meisten Fälle, der nicht eindeutigen Satzenden existiert eine Regel in der Entscheidungspipeline. Fälle die nicht entschieden werden können, werden nicht als Satzende markiert. Das gilt besonders für Satzenden die nur durch semantischen Kontext erkennbar sind.

4 Informationsextraktion

4.1 Information automatische Extraction aus HTML

4.1.1 HTML Datei lesen

Um die Information zu extrahieren, müssen alle Daten zunächst aus denen HTML Datei gelesen werden. Eine Beispiel für HTML Format: Um die Information aus denen HTML Datei zu lesen, soll Str1 und Str2 gelesen werden. Wie kann man Str1 und Str2 lesen, ist es in folgend Automater verdeutlichet (sieh Abb.4.3). Das Programm sieht man in "Html2Xml.HTML-lesen.java". Die Daten Struktur ist in "HTML-Page.java"definiert.
pp-type ist hier Str1 und pp-daten ist hier Str2.

4.1.2 Umwandlung von HTML mit einigen Regeln zu XML

HtmltoXml

Architektur Grobentwurf sieht man in folgend Diagramm (Abb. 4.4 4.5 4.6 4.7).

4.1.3 Benutze zum Umwandeln für MOPs mit HtmltoXml

Mann kann MOPs durch das Allgemeine Programm "*HtmltoXml*" erstellen. Zunächst soll man Regeln-Datei schreiben. (siehe Regeln-MOPs.xml)
(i) XMLSchema für MOPs.xml
(ii) Regeln für MOPs.xml
In folgende werden einige Regeln für MOPs.xml vorgestellt.

```
1 <Element name ="wahlperiode" anzahl="1" eigenschaft="Element"
2 type="ordner" path="../p14">
```

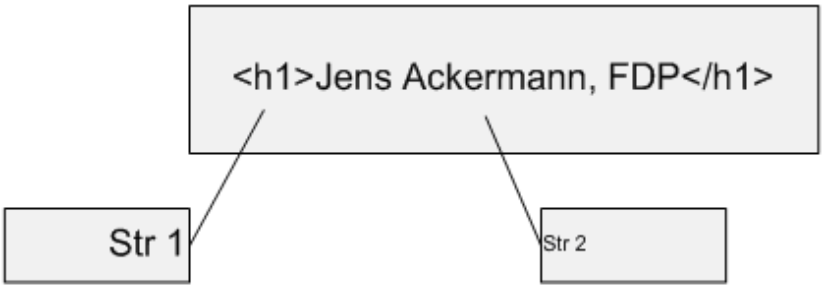


Abbildung 4.1: HTML Format

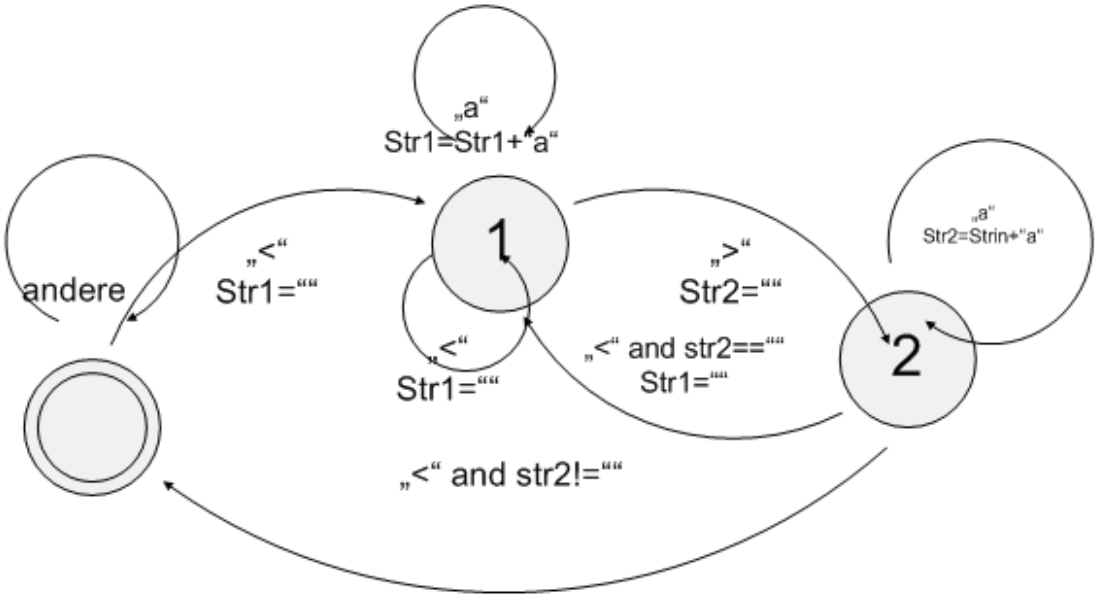


Abbildung 4.2: Automater um HTML Datei zu lesen

```
1 public class HTML-Page{
2    public  String pp-type;
3    public  String pp-daten;
4    }
```

Listing 4.1: HTML-Page.java

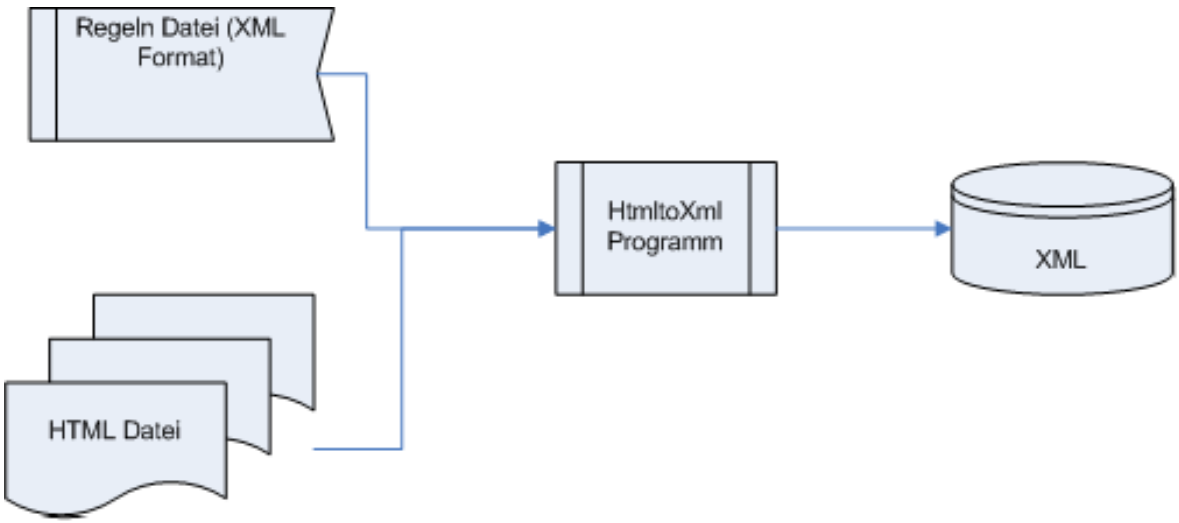


Abbildung 4.3: Umwandlung von HTML zu XML

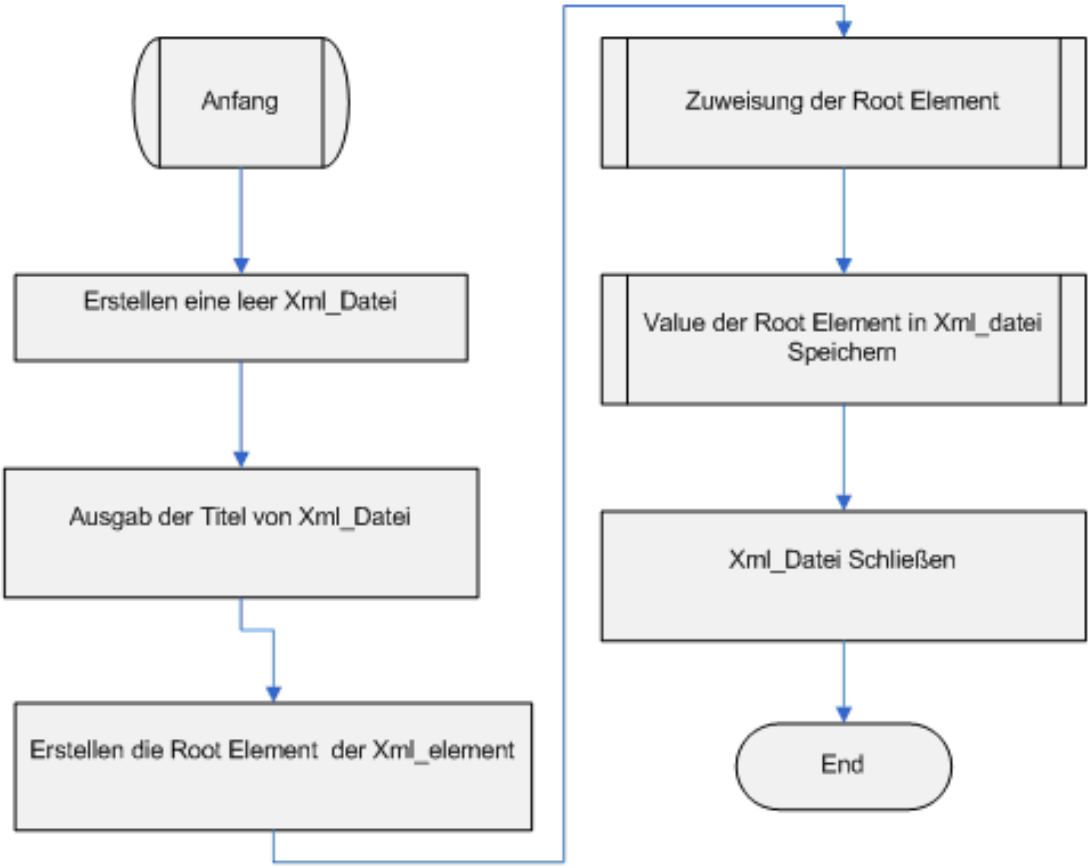


Abbildung 4.4: Umwandlung von HTML zu XML Level-1

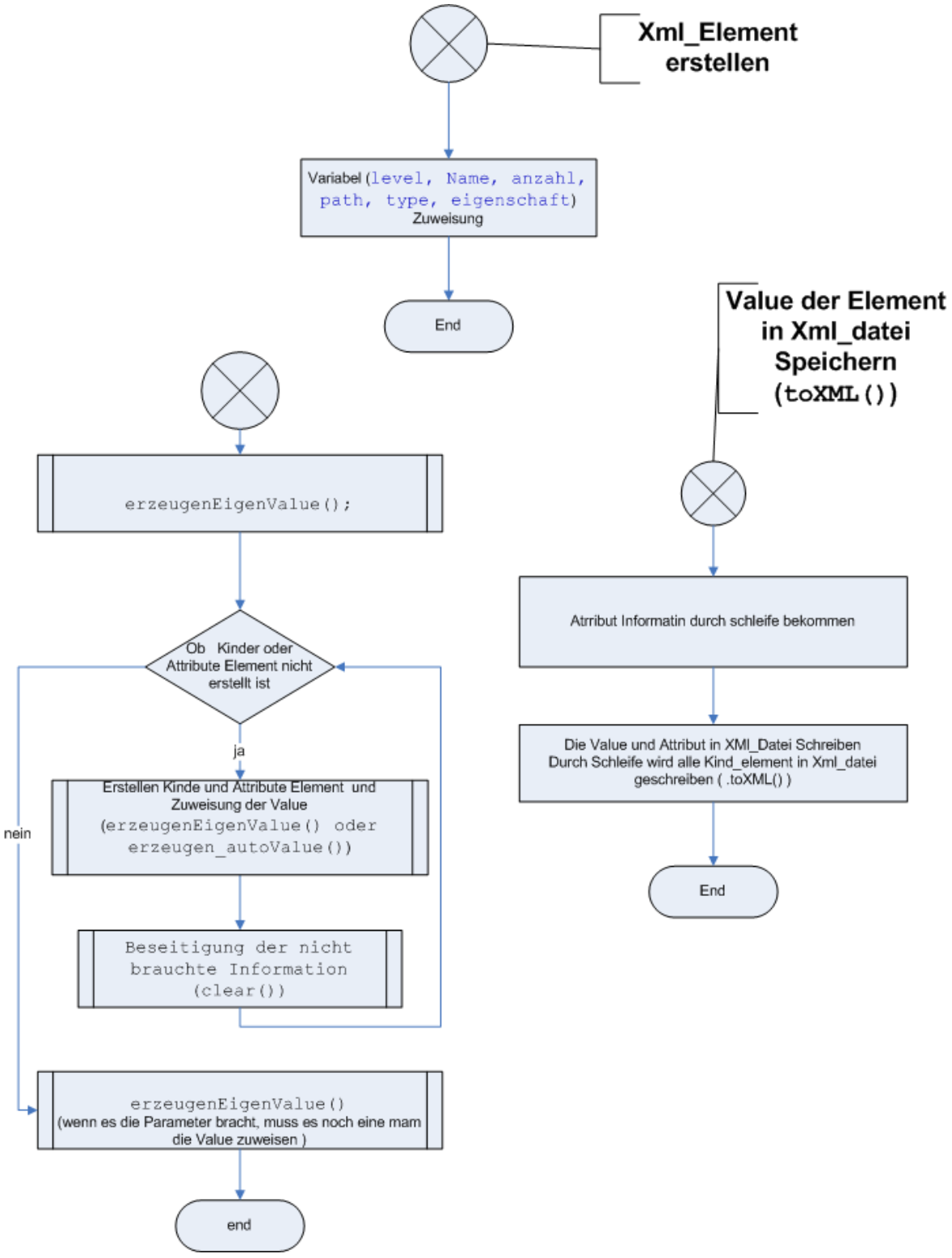


Abbildung 4.5: Umwandlung von HTML zu XML Level-2

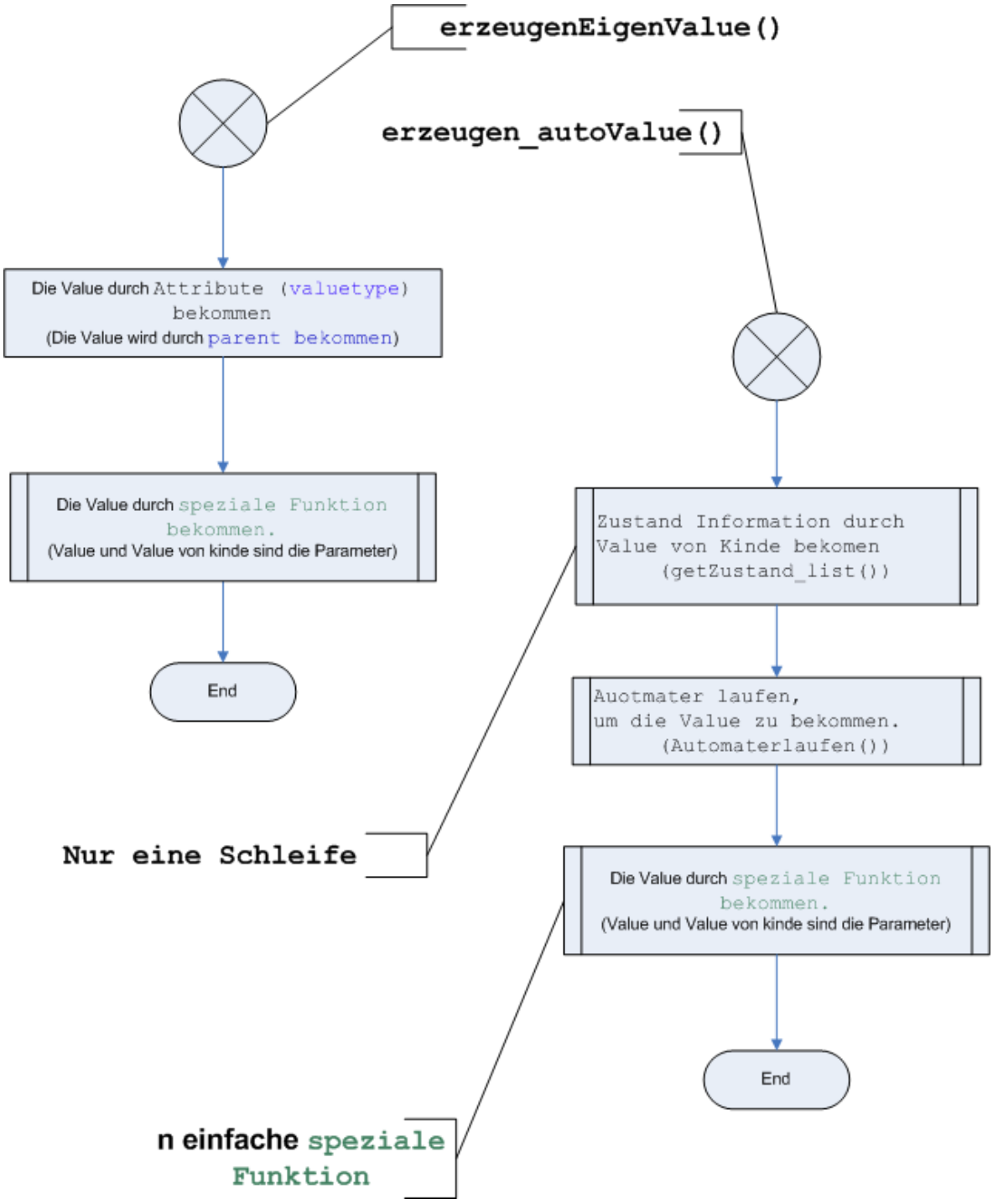


Abbildung 4.6: Umwandlung von HTML zu XML Level-3

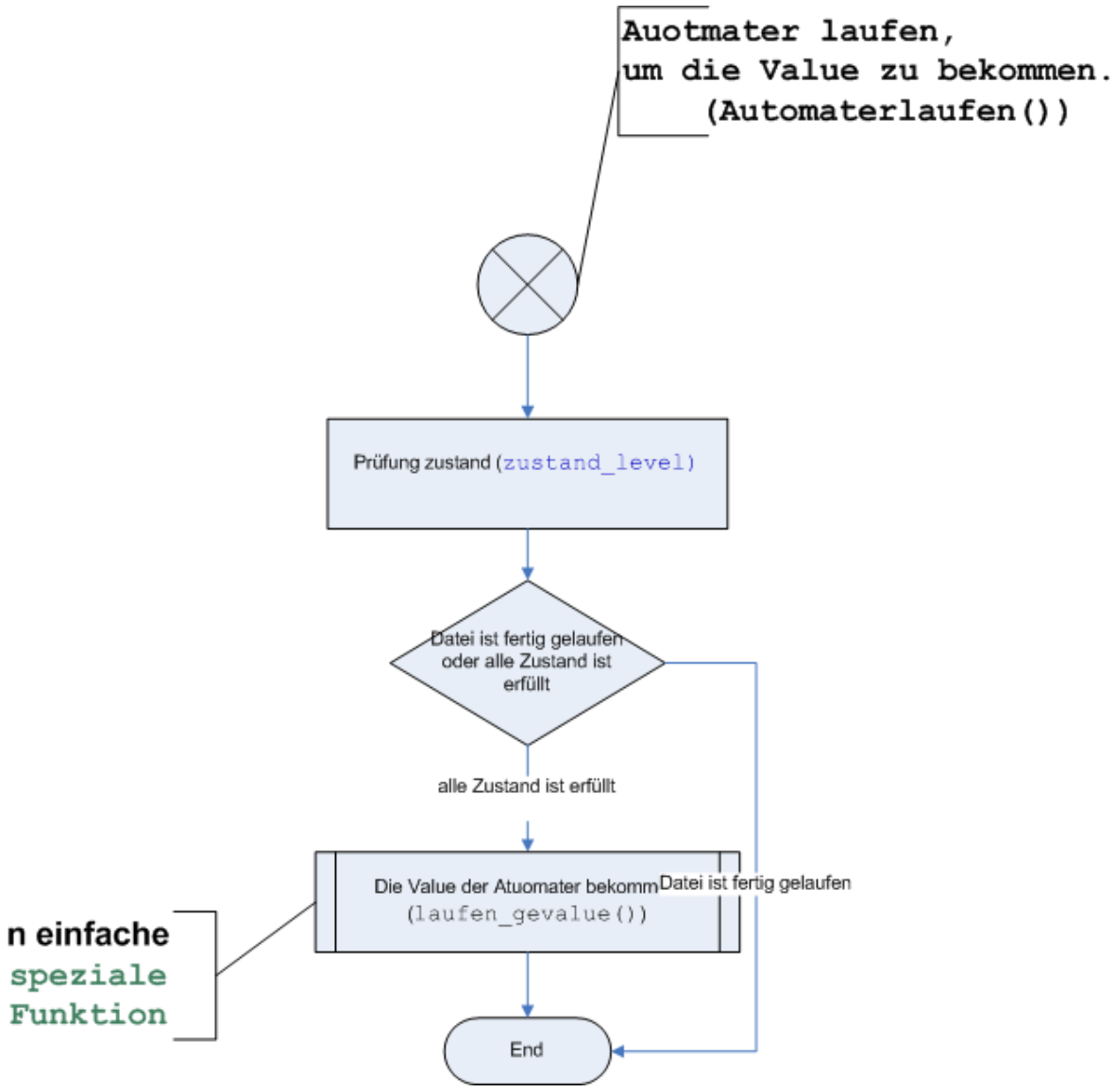


Abbildung 4.7: Umwandlung von HTML zu XML Level-4

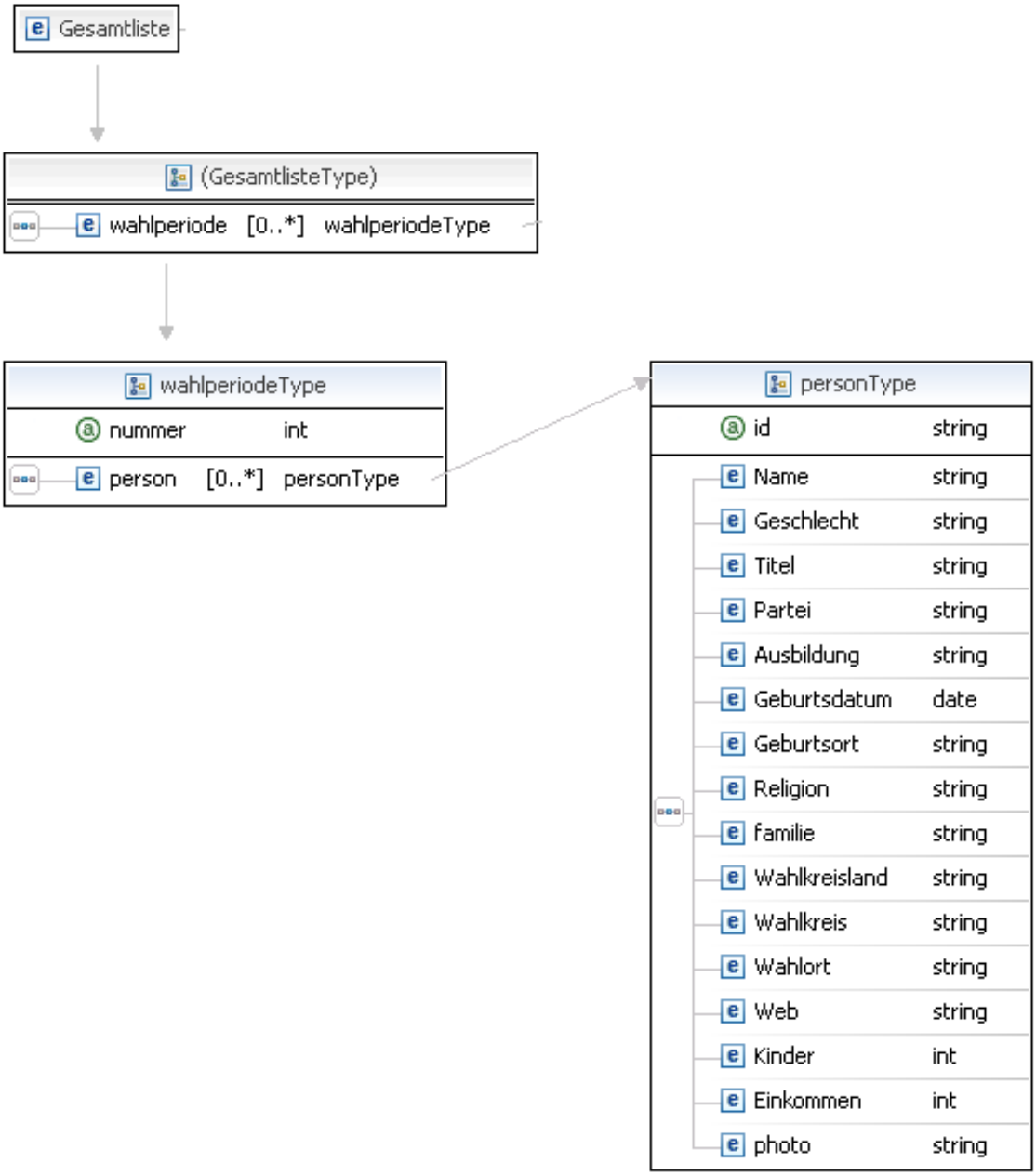


Abbildung 4.8: XMLSchema für MOPs.xml

eigschaft =”*Element*”:
Es gebt 3 Eigenschaften von Element.
(1.) Element : Es ist ein Element
(2.) Attribute: Es ist eine Attribute
(3.) Automater: Es ist ein Element, und der Wert wird durch Automater erhalten.

type =”*ordner*”:
Wenn das Element eine Type ”ordner” hat, muss es ein Wert für path haben.

path =”../*p*14”:
Die alle Datei soll in de Ordner ”../p14” gefunden werden, wenn die Kinder-Element eine Type ”datei” haben.

```
1 <Element name="nummer" anzahl="1" eigenschaft="Attribute" value=
  "14"
2 type="fest"> </Element>
```

Es ist ein Attribut mit Name ”*nummer*”. Und es hat fest Werte ”14”.

```
1 <Element name="person" anzahl="n" eigenschaft="Element"
2 type="datei">
```

Es ist ein Element. Und der Wert soll aus eine Datei gelesen werden.

```
1 <Element name="H1" anzahl="1" eigenschaft="Automater" type="main
  "
2 pruefung="pp_type" geAutomatervalue="getzustandvalue()"
3 zustandNr="0" ausgabe="F">
4     <Zustand eigenschaft="Automater">
5           <value eigenschaft="Automater"> H1 </value>
6           <value eigenschaft="Automater"> h1 </value>
7     </Zustand>
8 </Element>
```

Es ist eine Automater. Die Automater hat nur ein Zustand definiert. Die Automater soll den Zustand in pp-type prüfen. Wenn das Zustand erfüllt, wird der Wert aus erste Zustand erhalten (*geAutomatervalue* = ”*getzustandvalue*()” *zustandNr* = ”0”). Das Element soll nicht in XML Datei ausgeben, weil *ausgabe* = ”*F*” (F=false).

```
<Element name="Titel" anzahl="1" valuetype="parent.Element"
value="H1" eigenschaft="Element" gevalue="Name.getTitel()">
 </Element>
```

Der Wert ist durch spezielle Funktion "$Name.getTitel()$" erhalten. Der Wert von vordefinierte H1 ist die Parameter.

```
<Element name="Geburt" anzahl="1" eigenschaft="Automater"
type="main" pruefung="pp_daten" geAutomatervalue="getpagedaten()
"
pageindex="0" ausgabe="F">
     <Zustand eigenschaft="Automater">
          <value eigenschaft="Automater"> geboren </value>
          <value eigenschaft="Automater"> Geboren </value>
          <value eigenschaft="Automater"> gebort </value>
          <value eigenschaft="Automater"> Gebort </value>
     </Zustand>
     <Zustand eigenschaft="Automater">
          <value eigenschaft="Automater"> am </value>
          <value eigenschaft="Automater"> Am </value>
     </Zustand>
</Element>
```

Die Automater hat 2 Zustände definiert. Und sie soll die Zustände in pp-daten prüfen. Wenn alle Zustände erfüllt, der Wert ist genau den Wert von aktuelle pp-daten, weil $pageindex$ ="0". Wenn $pageindex = 1$ ist, soll die Automater nächste pp-daten lesen.

```
1  <Element name="Geburtsdatum_0" anzahl="1" valuetype="parent.
   Element"
2  value="Geburt" eigenschaft="Automater" type="main"  pruefung="
   value"
3  geAutomatervalue="getnextwort()" getwoerteanzahl="5" ausgabe="F"
   >
4        <Zustand eigenschaft="Automater">
5               <value eigenschaft="Automater"> geboren </value>
6               <value eigenschaft="Automater"> Geboren </value>
7               <value eigenschaft="Automater"> gebort </value>
8               <value eigenschaft="Automater"> Gebort </value>
9         </Zustand>
10        <Zustand eigenschaft="Automater">
11               <value eigenschaft="Automater"> am </value>
12               <value eigenschaft="Automater"> Am </value>
13        </Zustand>
14 </Element>
```

Die Automater lesen die Information aus vordefiniert Geburt. Wenn alle Zustände erfüllt, ist der Wer genau nächste Wort in den Wert von Element Geburt.
Analog für andere Regeln.

4.1.4 IdErzeugen (PersonIdErzeugen.java)

Die Funktion "$getPersonId(StringName)$" dientet dazu, für eine Person eine Id zu verteilen. Wenn man die Funktion benutzt hat, soll man nicht vergessen, die Funktion "$PersonIdSpeichern()$" aufzurufen, um die neue Name zu speichern.

4.1.5 MOPsZugriff

Der Klass diente dazu, die MOPs zu zugreifen.
"$Suchen()$": beliebigen Information in MOPs zu suchen
"$add(personp)$": Eine neue Person in MOPs einfügen.
"$toXml(Stringtoxml)$": MOPs als XML ausgeben.
In Folgend sieht man eine Sequenz Diagramm als Beispiel AusschuesseZugriff analog.

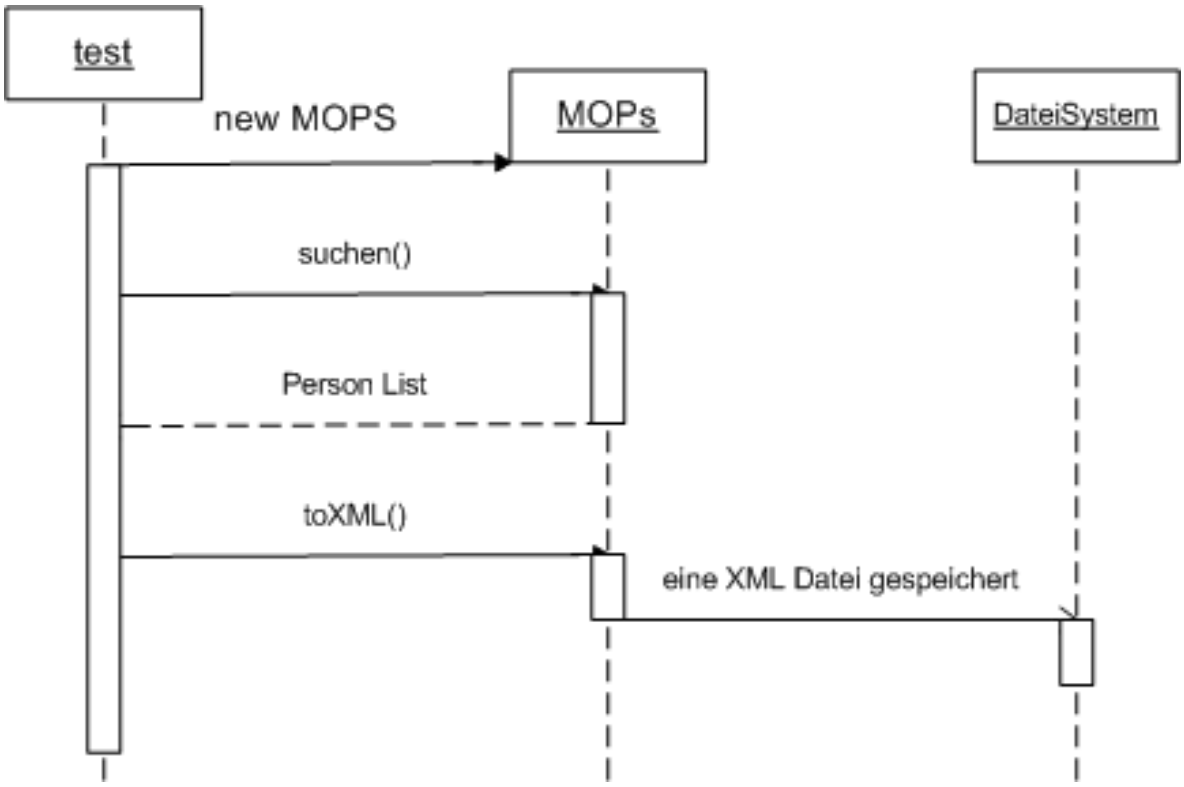


Abbildung 4.9: MOPsZugriff

Abbildung 4.10: Format von Link in HTML

4.1.6 Seitenbeschaffung

Seitenbeschaffung ist ein Programm (Crawler), der sich die HTML automatisch aus dem Internet zieht, um aus diesen HTML-Seiten dann XML-Instanzen entsprechend Schemas und Regeln zu erstellen. Zunächst muss die Link aus einer Webseite ausgelesen werden.
Eine Beispiel für Link in HTML Format: Analog zu HTML lesen, muss man eine Datentype für Link definieren.

```
1 public class web_link {
2       public String Name;
3       public String link;
4 }
```

Name ist hier die Information (z.B. Person).
Um die Linke auszuziehen, wird hier noch eine Automater geschrieben (sieh Abb. 4.11).

4.1.7 Automatischem Update

Um die Update-Funktion zu benutzen, brauchte man nur eine Instanz von $updateClass()$ zu erstellen.
Siehe eine Sequenz Diagramm als Beispiel

4.2 NER

4.2.1 Entitäten

Zur Beantwortung der Fragen ist es hilfreich, bestimmte Textstellen zu klassifizieren. Dazu benutzen wir die Named Entity Recognition (NER). In den Rohdaten müssen dazu im ersten Schritt einige Dokumente von Hand mit den NER-Entiätem markiert werden. Nachfolgend sind die Entitäten aufgeführt, die wir gewält haben.

- ORG - Organisation / Institution / Behörde
  Beispiel: Bundesregierung

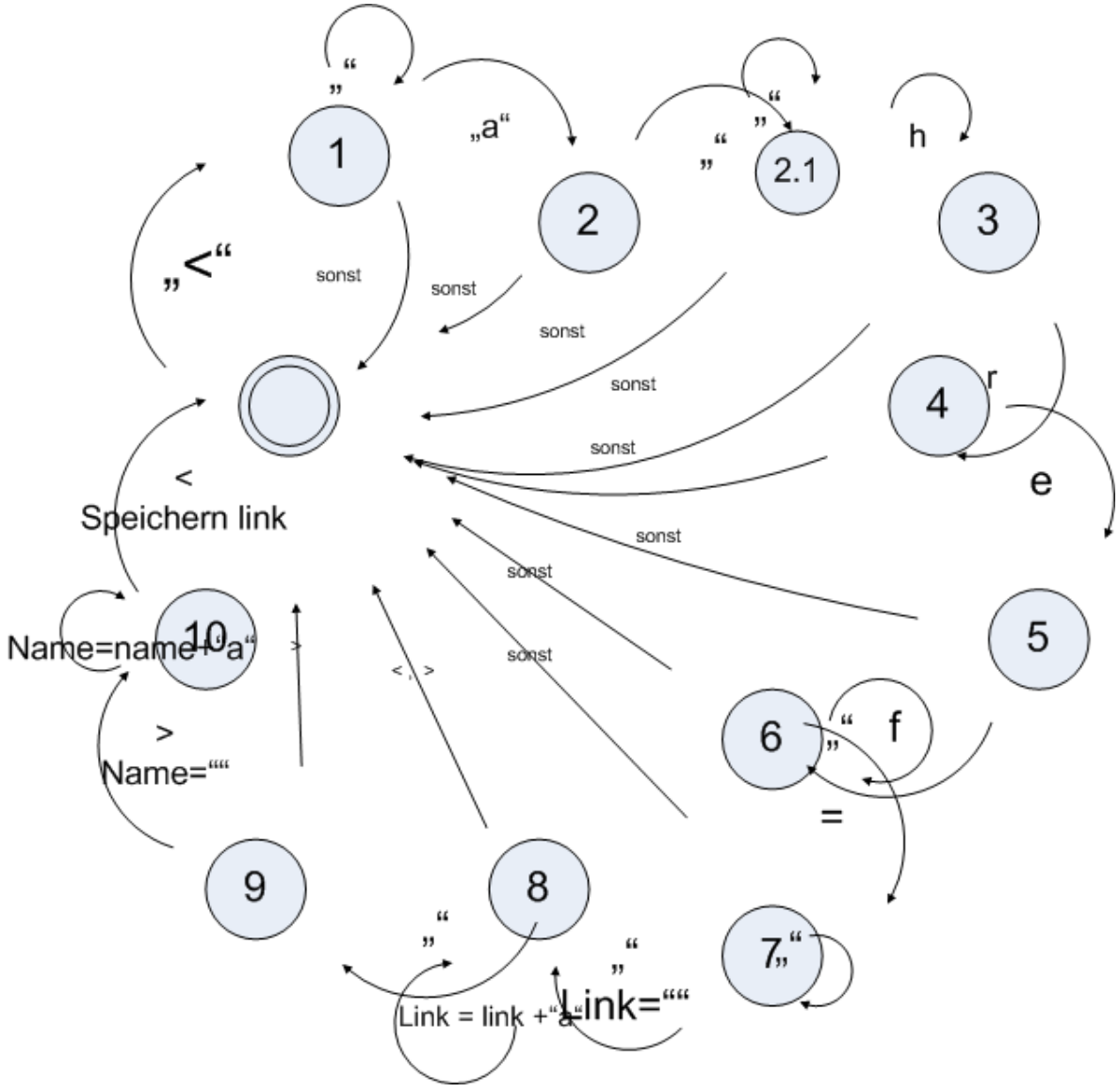


Abbildung 4.11: Automater, um die Linke auszuziehen

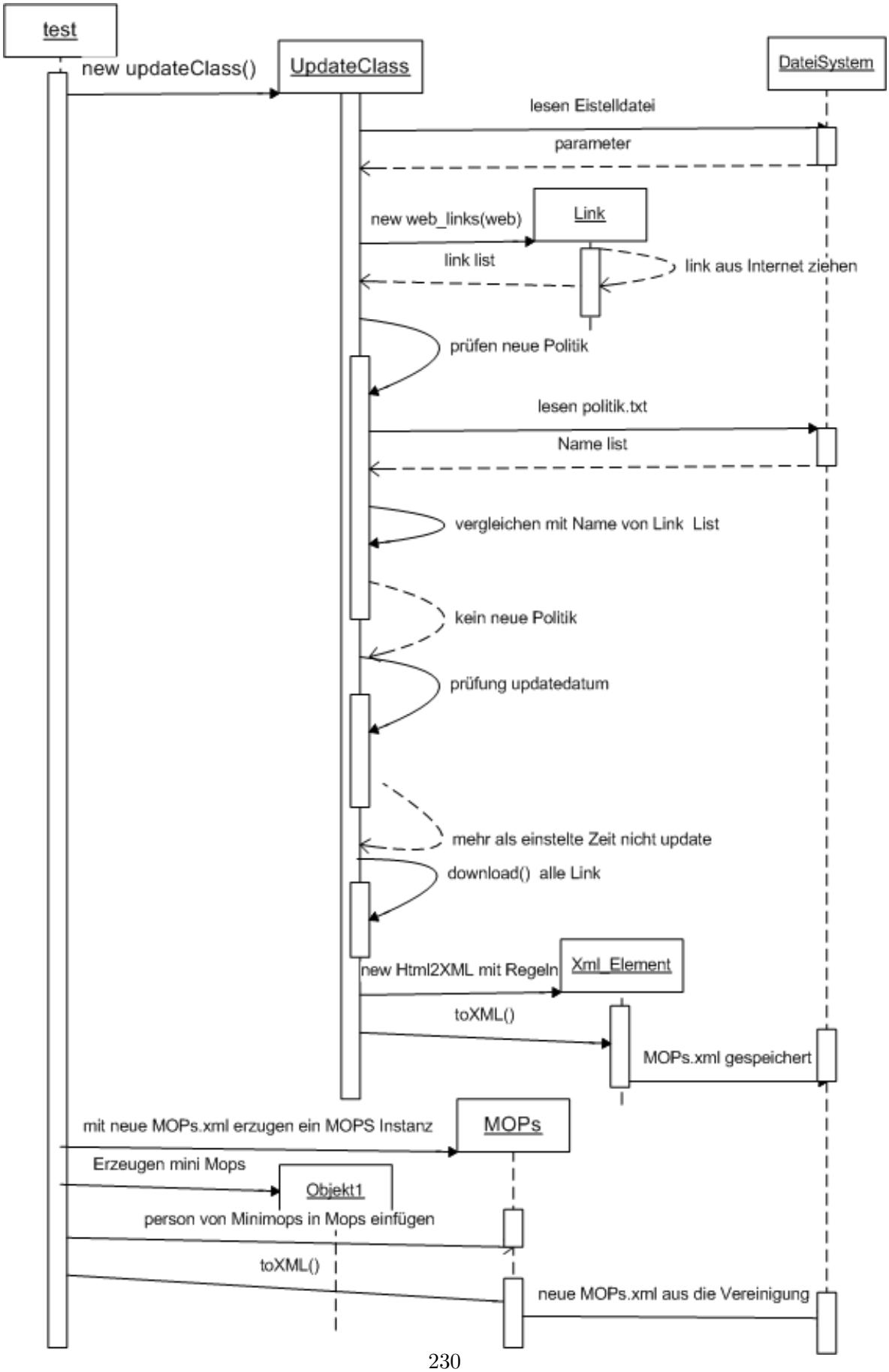


Abbildung 4.12: Sequenz Diagramm für Automatischem Update

- PERS - Person
  Beispiel: Abgeordnete
  Diese Entität kennzeichnet eine Person ohne Namen.

- REAC - Reaktion
  Beispiel: Applaus
  Zeigt die Einstellung einer oder mehrerer Personen und gibt somit Hinweise auf das Verhältnis gegenüber einer anderen Person oder die Haltung zu einem Ereignis oder Thema.

- LOC - Ort
  Beispiel: Berlin
  Hilfreich für Fragen, in denen geographische Informationen enthalten sind oder erfragt werden.

- PROJ - Projekt
  Beispiel: Aufklärungskampagne

- NAME - Name
  Beispiel: Angela Merkel
  Wird in einer Frage ein Name erwähnt, liefert diese Entität sofort die Sätze, in denen es um die benannte Person geht. Durch Extraktion eines Namens kann zusätzlich das Geschlecht bestimmt werden.

- PAR - Partei
  Beispiel: CDU/CSU
  Dies ist eine der wichtigsten Entitäten unserer Domäne.

- BUR - Amt (politisch) / Rolle
  Beispiel: Finanzminister
  An vielen Stellen werden Personen mit ihren Ämtern erwähnt. Sind diese getagt, lässt sich im zeitlichen Kontext die dazugehörige Person ermitteln. Diese Zuordnung erhöht die Qualität der Fragebeantwortung, da auch weitere Informationen zu der ermittelten Person mit einbezogen werden können.

- ADJ+ - Abstimmung positiv
  Beispiel: 15 Stimmen dafür
  Dies ist eine wichtige Entiät zu Extraktion von Ereignissen.

- ADJ+ - Abstimmung positiv
  Beispiel: 145 Stimmen dagegen.

- ADJ? - Abstimmung unentschlossen
  Beispiel: Enthaltungen

- ADJ! - Abstimmungsergebnis
  Beispiel: Antrag einstimmig angenommen

- Drucksache - Referenz auf Drucksache
  Beispiel: Drs 14/45

Tag	Vorkommen absolut	Vorkommen relativ
ORG	532	6,36%
PERS	430	5,14%
REAC	136	1,62%
LOC	475	5,68%
PROJ	0	0%
NAME	2540	30,35%
PAR	2078	24,83%
BUR	1632	19,50%
ADJ+	0	0%
ADJ-	0	0%
ADJ?	0	0%
ADJ!	0	0%
DRUCKSACHE	135	1,61%
PLENARPROTOKOLL	54	0,65%
DATUM	358	4,28%

Tabelle 4.1: Verteilung der NER Entitäten am Beispiel eines manuell getaggten Bundestagsprotokolls

Mit dieser Entität lassen sich wichtige Verweise finden. So wird zum Beispiel die Entwicklung eines Antrags zeitlich nachverfolgbar und es lassen sich Zusatzinformationen (Empfehlungen, Plenarprotokolle ...) finden.

- PLENARPROTOKOLL - Referenz auf Plenarprotokoll
  Beispiel: 15/47

- DATUM
  Beispiel: Dezember 2007
  Mit einem Datum kann ein zeitlicher Verlauf verfolgt oder der Zeitpunkt eines Ereignisses ermittelt werden.

Beim manuellen Tagging der Trainingsdaten bestand vor allem die Schwierigkeit, für jede Entität Daten zu finden. Man kann sich leicht vorstellen, dass die Vorkommen der einzelnen Tags sehr ungleich verteilt sind. Am seltensten sind Abstimmungen, die aber aus der Sicht der Informationsextraktion zu den wichtigsten Entitäten gehören. Wie zu erwarten, dominieren die Entitäten Person (PERS), Name (NAME) und Amt (BUR). Tabelle 4.1 zeigt diese Verteilung exemplarisch.

4.2.2 Rapid Miner, IE-Plugin

Zur Durchführung der NER wurde das Open-Source Tool RAPIDMINER (http://www.rapidminer.com) verwendet. Es ist vor allem unter seinem ehemaligen Namen YALE (Yet Another Learning Environment) bekannt und wird vom Lehrstuhl für künstilche Intelligenz an der Universität Dortmund entwickelt. Wie der Name schon sagt, handelt es sich dabei um ein Programm zur Analyse und Manipulation von Datensätzen. RAPIDMINER wird zusammen mit einem Plugin für Information Extraction (IE-PLUGIN) verwendet.

Das Plugin bereichert RapidMiner um spezielle Funktionen, die die NER unterstützen und vereinfachen. Die Datenverarbeitung geschieht in Form von Prozessketten, die sich aus verschiedensten Operatoren zusammensetzen. Jeder Operator nimmt Daten entgegen und gibt welche aus, so dass die zu Beginn eingelesenen Ausgangsdaten durch die Prozesskette transformiert werden. Eine weitere Arbeit, die sich mit NER in RapidMinder beschäftigt ist [Jun06]. Hier soll nun die Prozesskette erläutert werden, die für die NER benutzt wurde. Abbildung 4.13 zeigt den Prozess schematisch, der im Groben aus den gängigen drei Phasen Preprocessing, Trainingsphase, Testphase besteht.
Der NER-Prozess beginnt, wie jeder andere, mit dem Einlesen der Daten, die in diesem Fall im WTF-Format, beschrieben in Kapitel 3.1.2), vorliegen. Dies bewerkstelligt der SentenceExampleSource-Operator, der die Daten satzweise einliest. Ein Beispiel für die Rohdaten ist:

[...] 08.03.2006
Antwort der Bundesregierung auf die Kleine Anfrage der Abgeordneten
Patrick Meinhardt ,
Cornelia Pieper ,
Uwe Barth ,
weiterer Abgeordneter und der Fraktion der FDP [...]

Die Daten im WTF-Format, wie sie der SentenceExampleSource-Operator einliest, haben die Form:

`08.03.2006` **B-DATE**
`Antwort` **O**
`der` **O**
`Bundesregierung` **B-BUR**
`auf` **O**
`die` **O**
`Kleine` **O**
`Anfrage` **O**
`der` **O**
`Abgeordneten` **B-BUR**
`Patrick` **B-NAME**
`Meinhardt` **I-NAME**
`,` **O**
`Cornelia` **B-NAME**
`Pieper` **I-NAME**
`,` **O**
`Uwe` **B-NAME**
`Barth` **I-NAME**
`,` **O**
`weiterer` **O**
`Abgeordneter` **B-BUR**
`und` **O**
`der` **O**
`Fraktion` **O**
`der` **O**

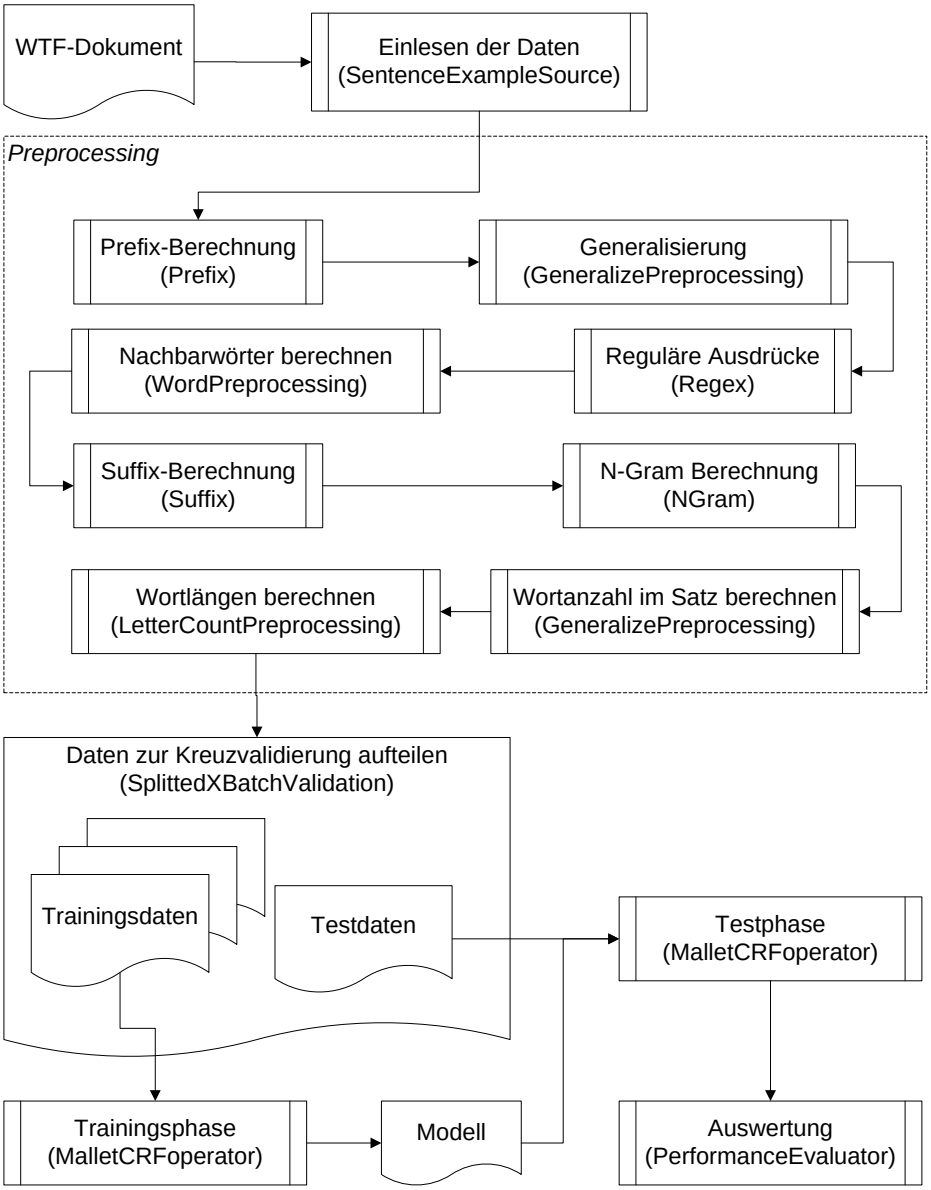


Abbildung 4.13: Die NER Prozesskette

In jeder Zeile kommt nur ein Wort vor, rechts daneben das „Named Entity Tag“. O heißt, dass das Wort keine Named Entity ist. „B-“ weist darauf hin, dass die Named Entity gerade beginnt, wo hingegen „I-“ anzeigt, dass die Named Entity fortgesetzt wird, wie es zum Beispiel bei Namen (Vorname und Nachname) der Fall ist.
Anschließend werden die Daten durch das Preprocessing angereichert. Dabei werden die Merkmale berechnet, die später die Basis für den Lernlauf bilden. Die einzelnen Preprocessingschritte werden im Folgenden beschrieben.

- Prefix-Berechnung
  Der Prefix gibt den Anfang eines Wortes wieder. In der Prozesskette werden Prefixe der Länge zwei, drei und vier verwendet. Zu jeder Länge wird der Prefix des aktuell betrachteten Wortes, sowie der des Vorvorgängers, Vorgängers, Nachfolgers und Nachnachfolgers berechnet. Jedes Wort wird also in einem Fenster der Größe fünf betrachtet.

- Generalisierung
  Der Generalisierungsoperator vereinheitlicht den Text, indem er alle Großbuchstaben durch „A“, alle Kleinbuchstaben durch „a“ und alle sonstigen Zeichen durch „x“ ersetzt. Auch hier werden zu jedem Wort die Generalisierungen der zwei vorherigen und nachfolgenden Wörter, sowie des aktuellen Wortes, erzeugt.

- Reguläre Ausdrücke
  Mit regulären Ausdrücken können komplexere Merkmale extrahiert werden.

- Nachbarwörter berechnen
  Zu jedem Wort werden die umgebenden Wörter mit einer Fenstergröße von fünf berechnet.

- Suffix-Berechnung
  Der Suffix-Operator funktioniert analog zum Prefix-Operator und wird ebenfalls mit den gleichen Längen und derselben Fenstergröße verwendet.

- N-Gram
  Ein N-Gram bezeichnet alle Subsequenzen der Länge N einer Sequenz. Wir verwenden ein zeichenbasiertes N-Gram von einzelnen Wörtern. Für das Wort „Gesetzentwurf“ wären die N-Gramme der Länge 6: „Gesetz“, „esetze“, „setzen“, „etzent “, „tzentw“, „zentwu“, „entwur“, „ntwurf“.
  Es werden N-Gramme der Längen zwei, drei und vier mit je einer Fenstergröße von fünf berechnet.

- Wortanzahl im Satz
  Zu jedem Wort wird die Anzahl der Wörter im Satz, in dem sich das Wort befindet, zugeordnet.

- Wortlängen
  Mit einem Fenster der Größe fünf werden die Wortlängen zu jeden Wort extrahiert.

Nach dem Preprocessing werden die angereicherten Daten in x Teile aufgeteilt, um eine Kreuzvalidierung vorzunehmen. Einer der x Teile wird für die Testphase benutzt, während anhand der übrigen trainiert wird. Dieser Vorgang wird x-mal wiederholt, so dass auf jeden Teil der Daten einmal das auf den anderen Teilen berechnete Modell angewendet wird. Als Bewertungsmaßstab der Prozesskette werden die Durschnittswerte von Precision, Recall und F-Measure über alle x Iterationen ermittelt. Die Kreuzvalidierung ist ein wichtiges Element zur Bewertung der Qualität der Prozesskette. Die Ergebnisse können durch die Eigenheiten der für den Prozess verwendeten Daten beeinflusst werden. So ist gut vorstellbar, dass die vorgestellten Features mit bestimmten Dokumenten besser funktionieren. Die zufällige Aufteilung der Kreuzvalidierung versucht eine Art Unabhängigkeit von der verwendeten Datenbasis zu erzeugen. Je größer die Anzahl der Teile (x) gewählt wird, desto realistischer sollten die Ergebnisse für die Verwendung mit unbekannten Daten sein. Tabelle 4.2 zeigt einige exemplarische Ergebnisse für verschiedene Werte von x bei gleicher Konfiguration der übrigen Komponenten unter Verwendung des selben WTF-Dokuments. Die Laufzeiten wurden auf einem 2,8Ghz PC mit 2048 MB Arbeitspeicher unter Windows XP gemessen.

x **Laufzeit**	2 75 min			4 300 min			6 530 min		
Tag	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
ORG	0.493 ± 0.046	0.210 ± 0.018	0.292 ± 0.010	0.644 ± 0.103	0.470 ± 0.258	0.511 ± 0.189	0.656 ± 0.142	0.452 ± 0.286	0.501 ± 0.221
PERS	0.558 ± 0.297	0.497 ± 0.030	0.476 ± 0.128	0.764 ± 0.164	0.531 ± 0.086	0.615 ± 0.090	0.724 ± 0.315	0.567 ± 0.104	0.595 ± 0.234
REAC	0.744 ± 0.244	0.618 ± 0.118	0.672 ± 0.172	0.688 ± 0.410	0.584 ± 0.366	0.628 ± 0.382	0.720 ± 0.403	0.738 ± 0.342	0.696 ± 0.350
LOC	0.766 ± 0.093	0.338 ± 0.102	0.452 ± 0.081	0.872 ± 0.056	0.511 ± 0.104	0.638 ± 0.083	0.878 ± 0.059	0.515 ± 0.080	0.645 ± 0.071
PROJ	-	-	-	-	-	-	-	-	-
NAME	0.760 ± 0.027	0.640 ± 0.022	0.695 ± 0.024	0.877 ± 0.052	0.825 ± 0.072	0.849 ± 0.059	0.880 ± 0.117	0.787 ± 0.167	0.829 ± 0.144
PAR	0.692 ± 0.111	0.515 ± 0.057	0.580 ± 0.003	0.945 ± 0.049	0.880 ± 0.087	0.910 ± 0.068	0.881 ± 0.162	0.813 ± 0.227	0.841 ± 0.202
BUR	0.783 ± 0.042	0.585 ± 0.147	0.655 ± 0.082	0.860 ± 0.102	0.787 ± 0.125	0.820 ± 0.109	0.845 ± 0.083	0.736 ± 0.218	0.768 ± 0.172
ADJ+	-	-	-	-	-	-	-	-	-
ADJ-	-	-	-	-	-	-	-	-	-
ADJ?	-	-	-	-	-	-	-	-	-
ADJ!	-	-	-	-	-	-	-	-	-
Drucksache	0.500 ± 0.500	0.500 ± 0.500	0.500 ± 0.500	0.750 ± 0.433	0.571 ± 0.404	0.626 ± 0.400	0.395 ± 0.447	0.484 ± 0.485	0.421 ± 0.450
Plenarprotokoll	0	0	0	0.500 ± 0.500	0.417 ± 0.433	0.450 ± 0.456	0.333 ± 0.471	0.312 ± 0.443	0.322 ± 0.456
Datum	1.000 ± 0.000	0.422 ± 0.347	0.505 ± 0.364	0.705 ± 0.410	0.495 ± 0.381	0.557 ± 0.377	0.665 ± 0.388	0.494 ± 0.391	0.505 ± 0.367
Gesamt	0.703 ± 0.010	0.518 ± 0.063	0.595 ± 0.046	0.852 ± 0.067	0.733 ± 0.023	0.787 ± 0.034	0.823 ± 0.132	0.705 ± 0.115	0.755 ± 0.107

Tabelle 4.2: Ergebnisse von NER-Testläufen auf einem WTF-Dokument

4.3 Events

4.3.1 Allgemeine Definition

Die Bestimmung von Events erleichert die Beantwortung bestimmter Fragestellungen und hilft eine Struktur in die gesammelten Informationen zu bekommen. Ziel ist es, die Events möglichst atomar zu gestalten, so dass die Informationsextraktion der Dokumente möglichst vollständig geschehen kann. Bestimmte Fragestellungen, wie z.B. „Wieviele Anträge von weiblichen Abgeordneten wurden abgelehnt?“ kann man nachfolgend ausschließlich anhand der Event-Informationen beantworten. Jedes Event wird als XML gespeichert, so dass Anfragen an eine Menge von XML-Dateien via XPATH gestellt werden können. Hier ein kleines Beispiel für eine solche XML-Datei:
Am Beispiel „Antrag“ erkennt man die Struktur der Events. Grundsätzlich kann jedem Event eine *eventId*, ein *datum* sowie eine *drucksache* zugewiesen werden. Darüberhinaus gibt es eine Menge von speziellen Slots, die üblicherweise ein Named Entity darstellen. Im obigen Beispiel wären dies der Named Entity *Name*, *Partei* und *Topic*.

4.3.2 Event-Schemata

In diesem Abschnitt folgt eine Auflistung der vorhandenen Events mit Triggern und Slots. Jedes Event verfügt über die folgenden Standard-Slots, die nachfolgend nicht mehr explizit erwähnt werden: eventId (Identifikator des Events), Datum und DokId (Identifikator des zugehörigen Dokuments).

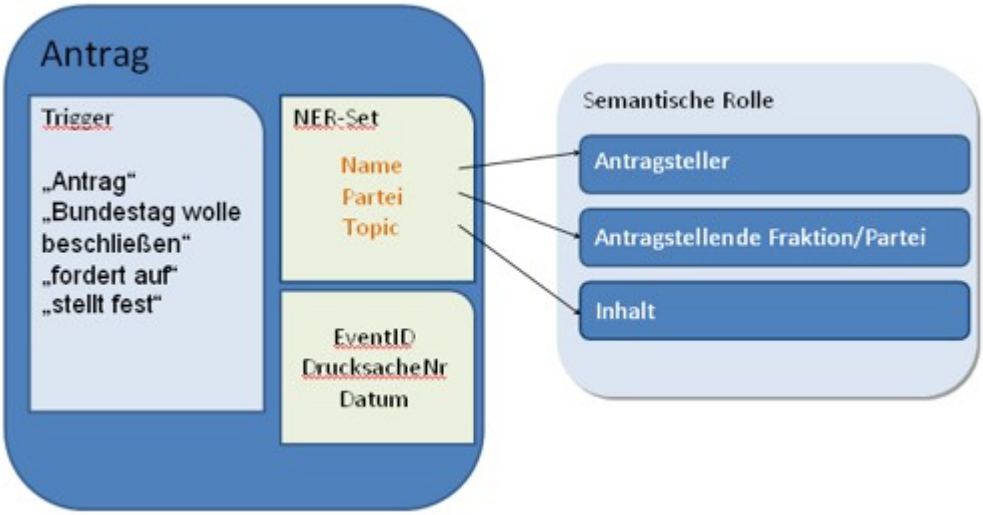
Antrag

- Triggerworte
  - Antrag
    In jeder Drucksache wird Antrag trivialerweise als Überschrift genutzt.
  - fordert auf
    „fordert auf“ ist ein Schlüsselsatz, der in vielen Anträgen an die Regierung gerichtet ist.
  - stellt fest
    „stellt fest“ ist ein Schlüsselsatz, der in vielen Anträgen die Beobachtung der Antragsteller einleitet.
  - Bundestag wolle beschließen
    Konkrete Aufforderung der Antragsteller über den zu fassenden Beschluss im Bundestag.

Daraus ergibt sich nachfolgendes Event-Schema:
Zum Event gehören „Antrag“ die wichtigen Slots Name, Partei und Topic. Bei den letztgenannten handelt es sich um Named Entities, denen wir in einem zweiten Schritt eine Semantische Relation geben. So kann, im Regelfall, allen Personen in einem Dokument „Antrag“ die Semantische Rolle „Antragsteller“ zugewiesen werden. Korrespondierend können Parteien als „AntragstellendePartei“ und das Topic als „Inhalt“ des Antrags bezeichnet werden. Durch die Rollen werden Relationen zwischen den Events möglich.

```
1  <event typ="Antrag">
2          <named_entity typ="Name" semantische_rolle="Beantragt">
           Rainer Brüderle</named_entity>
3          <named_entity typ="Name" semantische_rolle="Beantragt">
           Birgit Homburger</named_entity>
4          <named_entity typ="Name" semantische_rolle="Beantragt">
           Dr. Karl Addicks</named_entity>
5          <named_entity typ="Name" semantische_rolle="Beantragt">
           Christian Ahrendt</named_entity>
6          <named_entity typ="Partei" semantische_rolle="Beantragt"
           >FDP</named_entity>
7          <named_entity typ="Topic" semantische_rolle="Forderung">
           Mehr Wettbewerb im Schornsteinfegerwesen</named_entity>
8          <drucksache>1603344</drucksache>
9          <datum>08.11.2006</datum>
10         <event_id>324234</event_id>
11     </event>
```

Listing 4.2: XML-Event am Beispiel „Antrag“



Beschlussempfehlung

- Triggerworte

  - Beschlussempfehlung
    In jeder Drucksache wird „Beschlussempfehlung“ als Überschrift genutzt.
  - Annahme, Ablehnung, Ablehnen, Annehmen
    „Annahme“, „Ablehnung“, „Ablehnen“, „Annehmen“ sind Schlüsselwörter, die alle im Zusammenhang mit der Empfehlung für oder gegen einen Antrag genannt werden können.
  - Zustimmung
    Bei jeder Empfehlung wird die Zustimmung verschiedener Fraktionen explizit ausgedrückt.
  - Stimmenthaltung
    Auch Stimmenthaltungen werden nach Fraktionen geordnet in der Drucksache angegeben.
  - Lösung
    Lösung ist ein spezieller Teilbereich jeder Beschlussempfehlung.
  - Alternativen
    Alternativen kann ein spezieller Teilbereich einer Beschlussempfehlung sein.
  - Bundestag wolle beschließen
    Konkrete Aufforderung zur Empfehlung für oder gegen einen Antrag im Bundestag.

Jede Beschlussempfehlung referenziert den vorlaufenden Antrag, also haben wir einen Slot vom Typ „DrucksachenNr“ mit einer Antragsrelation. Ein weiterer Slot speichert das Ergebnis, ist daher vom Typ „Ergebnis“. Daneben werden auch die beteiligten Institutionen in einem Slot gespeichert. Institutionen werden bereits im NER-Lauf erkannt und sind daher einfach zu erkennen.

Abstimmung

- Triggerworte

  - Enthaltungen?,Wer stimmt dagegen,Wer stimmt dafür
    „Enthaltungen?“, „Wer stimmt dagegen“, „Wer stimmt dafür“ sind Schlüsselsätze mit der der Bundestagspräsident jede Abstimmung einleitet.
  - Stimmen
    Bei abgezählten Abstimmung werden auch oft die Anzahl der abgegeben Stimmen bestimmt.
  - Abstimmung
    Das Wort „Abstimmung“ kommt sinnigerweise auch als Triggerwort in Frage.

Das Event „Abstimmung“ beinhaltet zwei Slots des Typs „DrucksachenNr“ mit einer Semantischen Relation „Beschlussempfehlung“ sowie einer Relation zum Antrag. Die Antragsrelation

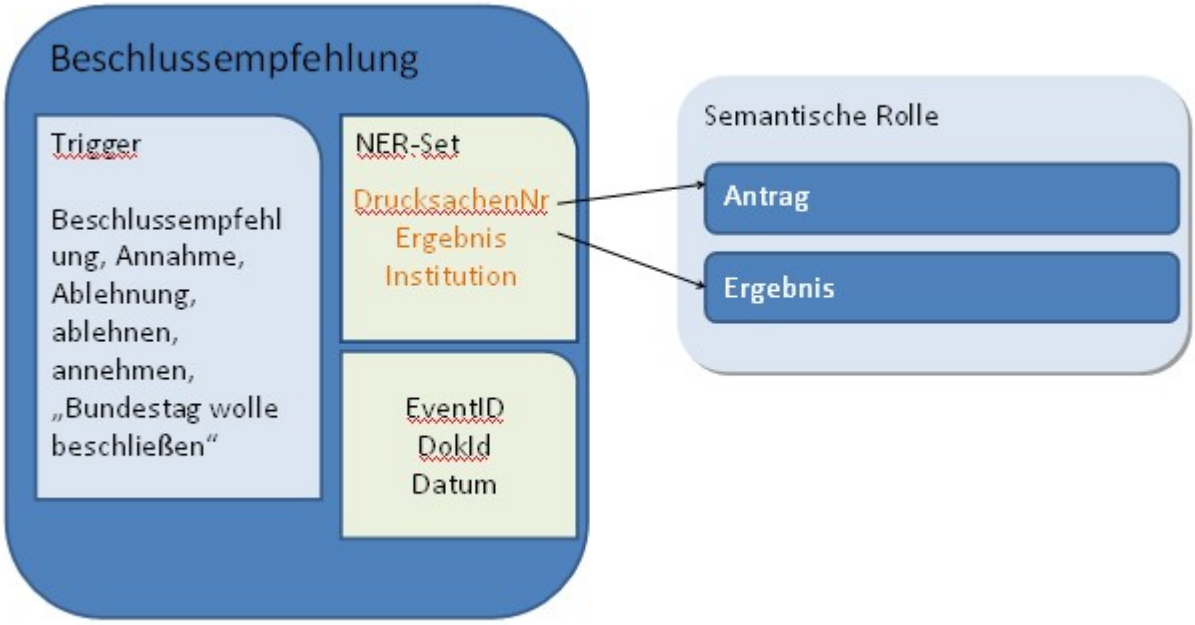


Abbildung 4.14: Schematische Darstellung des Events Beschlussempfehlung

ist eine 1 zu 1 Beziehung. Es gibt also zu jeder Abstimmung auch einen Antrag, allerdings gibt es nicht zu jedem Antrag eine Beschlussempfehlung, folglich auch nicht zu jeder Abstimmung eine Relation zu einer Beschlussempfehlung.
Ein weiterer Slot speichert das Ergebnis der Abstimmung, das bereits bei der NER bestimmt wurde. Anhand dieses Ergebnisses werden die Semantischen Relationen der Parteien bestimmt. Einer Partei ist es möglich einer Abstimmung zuzustimmen, abzulehnen oder sich zu enthalten. Dies wird mit den Semantischen Relationen „Zustimmend“, „Ablehnend“ oder „Enthaltend“ abgebildet. Zur Bestimmung dieser Relationen wird eine Analyse des Satzes benötigt, in dem die Partei erwähnt wird.

Zwischenruf

- Triggerworte

  - ( )
    Jeder Zwischenruf in einem Bundestagsprotokoll ist in Klammern gesetzt.
  - Heiterkeit, Lachen, Beifall, Zuruf
    Werden als Reaktion der Zwischenrufer genannt und kommen dabei sehr häufig vor.

Jedes Zwischenruf-Ereignis enthält zwei Slots vom Typ „Name“. Einerseits um den Zwischenrufer, der dem aktuellen Redner entweder Zustimmend, Ablehnend oder Neutral gesinnt ist, zu bestimmen. Andererseits gibt es auch einen Slot für den Redner. Um diesen auszufüllen muss man den Redner des Absatzes vor dem Zwischenruf bestimmen.

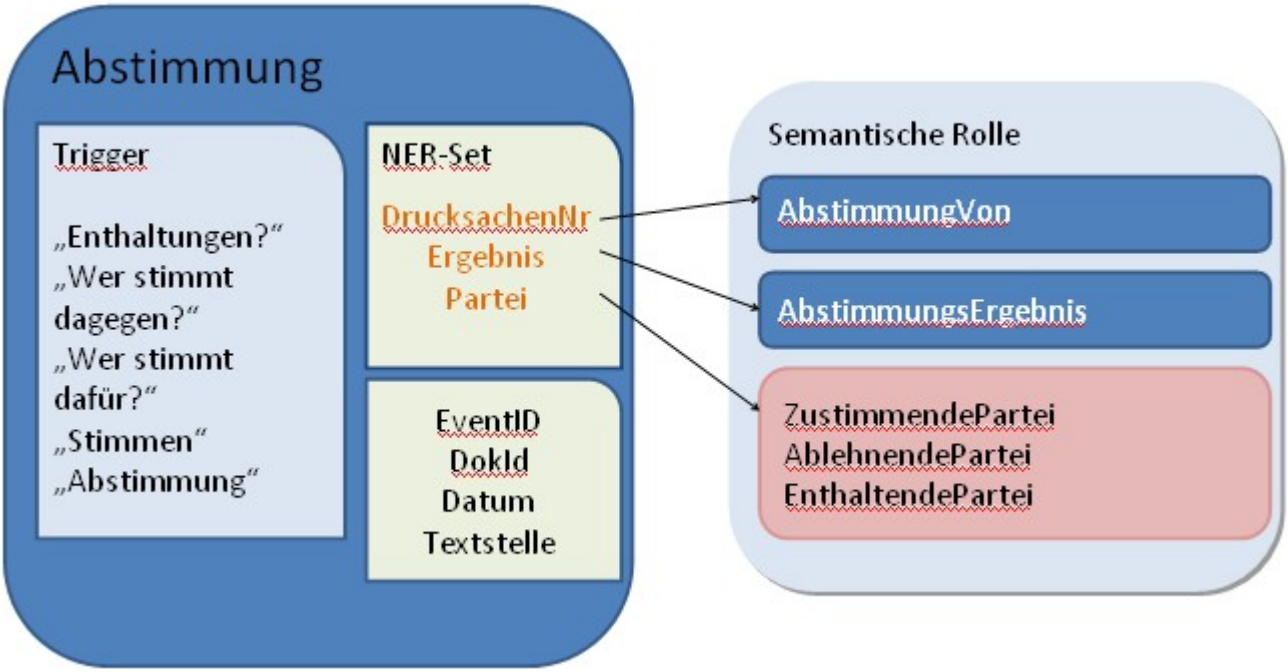


Abbildung 4.15: Schematische Darstellung des Events „Abstimmung“

Jeder Zwischenrufer wird mit einer Semantischen Relation „Zustimmend“, „Ablehnend“ und „Neutral“ abgebildet. Diese Relationen können wieder über zuvor bestimmte Trigger ausgelöst werden. Hierzu vergleicht man die Reaktion des Zwischenrufers, die ebenfalls ein Slot des Events ist, mit den Triggern und findet die passende Relation. Eine Liste dieser speziellen Trigger befindet sich im Anhang.
Für ganz einfach Zwischenrufe wie „Beifall bei der SPD“ gibt es zusätzlich einen Slot vom Typ „Partei“, der ebenfalls ein semantisches Attribut „Zustimmend“, „Ablehnend“ oder „Enthaltend“ bekommt.

In der Praxis wurden im Programm die Semantischen Relationen nicht eingesetzt, da dies zu einer erheblich komplexeren Extraktion geführt hätte. Eine Erweiterung diesbezüglich, würde allerdings einige neue Funktionen in das Programm einbringen. So wäre beispielsweise auch die Darstellung der Meinung eines bestimmten Politikers zu einem bestimmten Thema möglich.

4.3.3 Relationen zwischen Events

Zwischen den Events können Relationen vorhanden sein. So befindet sich in einem Event „Abstimmung“ immer eine Referenz zur Drucksache des Antrags, sowie in den meisten Fällen auch eine Referenz zu relevanten Beschlussempfehlungen. Hier kann also anhand der Drucksachen-Nr nach Events vom Typ Antrag gesucht werden und man erhält das passende Event „Antrag“. Eine derartige Verknüpfung besteht zwischen fast jedem Event. Ein Zwischenruf-Event enthält immer das Quellprotokoll in dem man nach Abstimmungen oder Reden suchen könnte. Andersherum kann man über ein Rede-Event und der passenden Text-Stelle die zugehörigen

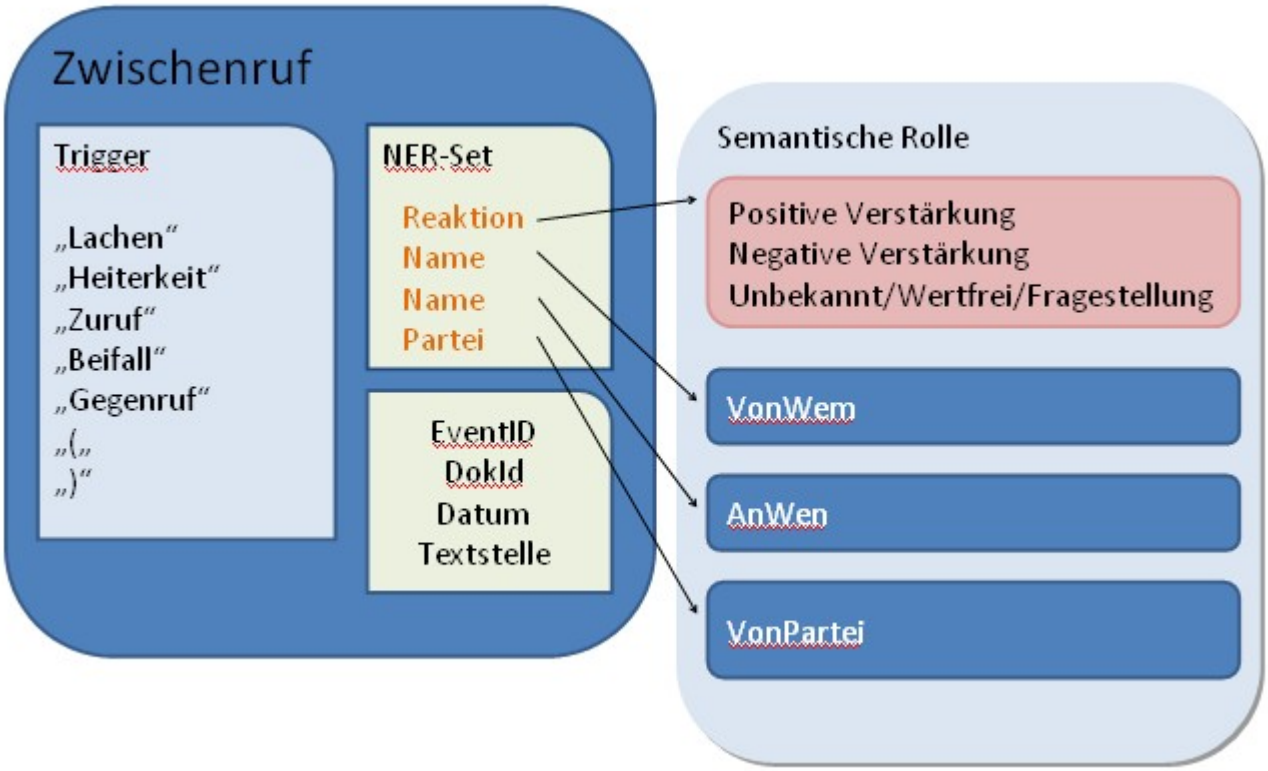


Abbildung 4.16: Schematische Darstellung des Events „Zwischenruf“

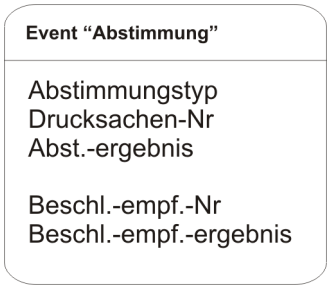


Abbildung 4.17: Zu füllendes Template für Event „Abstimmung“

Zwischenrufe definieren.

4.3.4 Relation Extraction am Beispiel von Abstimmungen

Events können einerseits mit den in Abschnitt [4.3.2] beschriebenen Event-Triggern gefunden werden, andererseits aber auch über Relation Extraction [1.5] aus den NE-getaggten Dokumenten extrahiert werden. Nachfolgend wird in diesem Abschnitt gezeigt, wie die Relation Extraction aufgebaut ist, wie wir Rapid Miner als automatischen Triggerfinder einsetzen und welche Ergebnisse wir damit letztlich erzielen konnten.

Formale Notation der Relation Extraction

Da wir die Relation Extraction am Beispiel von Abstimmungen beschreiben wollen, hier noch einmal ein kurzer Überblick über das zu füllende Template (siehe Abbildung 4.17). Die einzelnen Slots des Events Abstimmungen sind: Abstimmungstyp (entspricht einem Drucksachentyp), Drucksachen-Nr und Abstimmungsergebnis. Die Slots Beschlussempfehlungs-Nr bzw. Beschlussempfehlungsergebnis sind optional und werden nur dann gefüllt, wenn eine solche Referenz existiert.

Bei der Relation Extraction [1.5] geht es darum eine Beziehung zwischen einer Menge von Named Entities oder anderen Datenobjekten herzustellen. In diesem Fall interessiert uns nur die Relation zwischen Named Entities. Diese Relationen sollen nur innerhalb eines Satzes hergestellt werden, d.h. sowohl die Named Entities, als auch die Relationen befinden sich in einem Satz. Nehmen wir als Beispiel folgenden Text aus einem Plenarprotokoll:

S1: Die **Beschlussempfehlung** ist mit den Stimmen der Koalitionsfraktionen und der **FDP**-Fraktion bei Enthaltung von **Bündnis 90/Die Grünen angenommen**.

Die fettgedruckten Wörter markieren die vorhandenen Named Entities mit folgender Bedeutung: „Beschlussempfehlung“ entspricht einem Drucksachentyp (DrsTyp), „FDP“ / „Bündnis 90/Die Grünen“ sind Parteien (Par) und „angenommen“ ist vom Typ Ergebnis (Erg).

Zwischen diesen Named Entities bestehen zwei verschiedene Relationen R_1 und R_2, wobei eine Relation immer aus zwei Entitäten e_1, e_2 besteht.

- R_1 entspricht einer „ist“ Relationen zwischen DrsTyp als Entität e_1 und Erg als Entität e_2. Umgangssprachlich beschreibt $R_1(DrsTyp, Erg)$ also, dass ein bestimmter Drucksachentyp entweder angenommen oder abgelehnt worden ist.

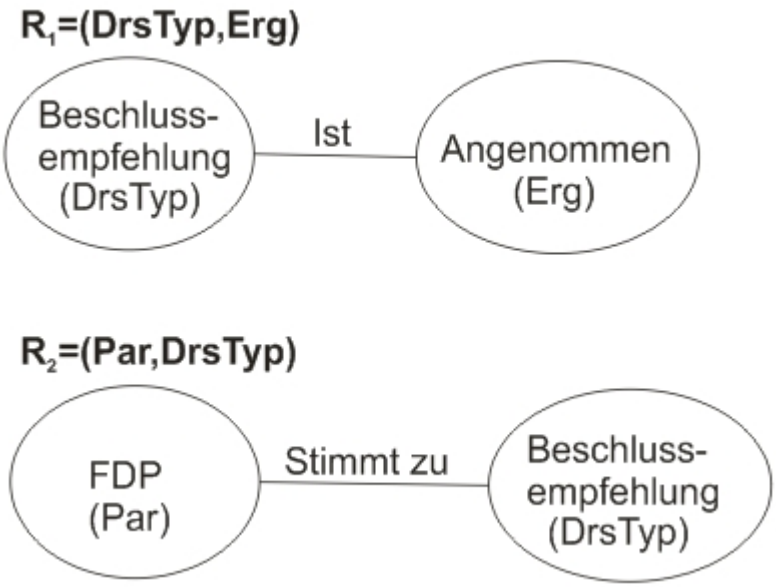


Abbildung 4.18: Schematische der Relationen im Event „Abstimmung“

- Relation R_2 besteht zwischen Par und DrsTyp und ist eine „stimmt zu“-Relation. Umgangssprachlich beschreibt die Relation $R_2(Par, DrsTyp)$ alle Parteien, die einer Drucksache in einer Abstimmung zustimmen.

Die Notation erfolgte nach [BM05]. Beide Relationen kann man auch in einem Graphen verdeutlichen.

Aufbau des Rapid Miner Experiments

Üblicherweise wird Relation Extraction über Dependency Tree Kernels [CS03] oder verschiedenen Optimierungen wie Shortest Path Dependency Kernels [BM05] realisiert. Die Grundidee dabei ist es, zu jeder Entität einen Feature Vector aufzubauen, über den die Bedeutung innerhalb eines Satzes gewonnen wird. Dies kann eine POS-Bestimmung, aber auch die Suche nach Synonymen oder Oberbegriffen sein.

Da Abstimmungen in den Plenarprotokollen relativ homogen aufgebaut sind und von der Struktur immer gleich ablaufen, werden wir nachfolgend die Relation Extraction mit Hilfe des Rapid Miners beschreiben, wo wir ein ähnliches Verfahren wie bei der Named Entity Recognition anwenden und dabei als Feature das Preprocessing sowie die erkannten Named Entities nutzen. Da die weiter oben im Abschnitt 4.2 beschriebenen Named Entities für dieses Verfahren nicht ausreichen würden, wurde die Menge der Named Entities um das Ergebnis (Erg) von Abstimmungen sowie den Drucksachentypen (DrsTyp) erweitert. Dies ermöglicht in einem nachfolgenden Schritt die Erkennung von Abstimmungen.

Für die Erweiterung der NER wurden zwei Protokolle per Hand annotiert und dieses Modell mit dem ebenfalls in 4.2 beschriebenen Verfahren trainiert. Auf Basis dieser Protokolle konnten bei 10-facher Kreuzvalidierung die in der Tabelle beschriebenen Ergebnisse für die einzelnen Named Entities erzielt werden. Die Laufzeit wurde auf einem 3,0Ghz PC mit 4096 MB Arbeitspeicher unter Windows Vista gemessen.

Während die Ergebnisse in fast allen Bereichen sehr gut ausfallen und auch die Gesamtperformance mit einem F-Measure von $89,1\%$ gut ist, fällt gerade das Ergebnis der Entität „Erg“ ins Auge. Hier konnte nur ein F-Measure von $13,5\%$ erzielt werden. Grund dafür könnte eine unzureichende Tokenmenge in den handmarkierten Protokollen sein. Während Entitäten wie „DrsTyp“, „Par“ oder „Name“ in allen Protokollen sehr häufig vorhanden sind, kommt „Erg“ jeweils nur bei den Abstimmungen vor.

In einem zweiten Schritt soll dann mit Rapid Miner ein **Trainingsmodell für die oben genannten Trigger-Sätze** erstellt werden. Bevor Rapid Miner aus den per Hand markierten Sätzen lernen kann, werden, wie bei der NER [4.2], die Attribute der Preprocessing-Schritte als Feature genutzt. Auch wird das NER-Modell auf den Datensatz angewendet, so dass wir im Trainingsschritt für die Abstimmungssätze Named Entities zur Erkennung der Relationen in einem Satz nutzen können. Als Lernverfahren werden dabei Conditional Random Fields benutzt 1.2.4.
Insgesamt wurden in 10 Plenarprotokollen die relevanten Sätze (vgl. S1) am Ende einer Abstimmung markiert. Leider konnten jedoch nicht alle Plenarprotokolle für den Trainingslauf genutzt werden. Sowohl am Lehrstuhl als auch auf dem heimischen Rechner war der Arbeitsspeicher dafür zu knapp bemessen. Insofern kann man schon an dieser Stelle einen Schwachpunkt der Ergebnisse festmachen, denn dieses beruht nicht auf den kompletten zur Verfügung stehenden Trainingsdaten, sondern lediglich auf zwei jeweils einzelnen Protokollen. Hier wäre ein inkrementelles Lernverfahren wünschenswert.

Nachdem wir ein Modell für die NER und ein Modell für die Erkennung der Abstimmungen vorliegen haben, können wir in einem dritten Schritt komplett unmarkierte Dokumente mit Marken für Named Entities und Marken für Abstimmungen versehen. Zur Messung der Performance wurden dabei verschiedene Protokolle unter unterschiedlichen Bedingungen automatisch markiert. Eine genauere Auswertung der Ergebnisse erfolgt im nächsten Abschnitt dieses Kapitels.

Ergebnisse der automatischen Trigger-Suche

Zur Messung der Performance wurden zwei Trainingsmodelle für die Suche nach Abstimmungen gelernt. Diese Trainingsmodelle basieren auf markierten Abstimmungen mit den weiter oben genannten Relationen $R_1(DrsTyp, Erg)$ sowie $R_2(Par, DrsTyp)$. Als Protokolle dienten hierfür Plenarprotokoll 16/85 sowie 16/102. Protokoll 16/85 bietet 12 markierte Abstimmungen, Protokoll 16/102 sogar 41, also mehr als dreimal so viele Abstimmungen, was sich auch auf die Ergebnisse auswirken kann.
Anhand dieser Lernmodelle wurden jeweils fünf unterschiedliche Protokolle automatisch markiert und das Ergebnis mit den per Hand markierten Dokumenten verglichen. Zur Überprüfung der Ergebnisse wurde dasselbe Verfahren nocheinmal ohne ein Preprocessing, also nur mit Named Entities, getestet und die Performance geprüft. Ein Vergleichswert liefert ebenso ein Durchlauf, der die Abstimmungen nur anhand des Preprocessing erkennt.
Insgesamt erkennt man in Tabelle 4.4 eine relativ hohe Schwankungsbreite der Ergebnisse. Während gerade bei kleinen Protokollen (P16/50) der F-Measure relativ gut ist, geht dieser Wert bei sehr langen Protokollen (P16/102) doch merklich zurück. Interessant ist auch die Tatsache, dass bei Protokoll 102, welches sowohl als Lernmodell als auch als Testprotokoll benutzt wurde, trotzdem keine perfekten Ergebnisse zustande kamen. Natürlich ist diese eine Zeile trotzdem nicht ganz repräsentativ. Insgesamt liegt der F-Measure gemittelt über alle

Testprotokolle bei 73 bzw. 72 Prozent.
In Tabelle 4.5 sind die Lernergebnisse für den Fall, dass kein Preprocessing stattfindet, eingetragen. Hier erkennt man einen deutlichen Unterschied zwischen den beiden Lernmodellen. Während Modell P16/85, das deutlich weniger Abstimmungen besitzt, mit einem F-Measure von 42% schlecht abschneidet, kann das Lernmodell P16/102 mit 64% deutlich besser performen.
Dabei ist zu erwähnen, dass die hohe Fehlerrate besonders durch den Header in den Protokollen verursacht wird. Fast alle Fehler sind so zustande gekommen. Die Relationen zwischen den Entitäten werden zumeist sehr gut erkannt und präzise abgegrenzt, ein deutlicher Unterschied zu der folgenden Methode, wo nur das Preprocessing zur Erkennung genutzt wurde.
In Tabelle 4.6 sind die Ergebnisse der Relation Detection auf einem Lernmodell, das nur auf Preprocessing beruht, eingetragen. Auch hier schneidet das Modell mit der höheren Anzahl an markierten Abstimmungen besser ab (P16/102) und liefert gleichzeitig den besten Wert mit 88% F-Measure. Die Ergebnisse waren in diesem Fall allerdings deutlich unschärfer. Markierungen für Abstimmungen wurden teilweise schon im Satz davor begonnen oder endeten erst einen Satz später. Solang die Relationen exakt enthalten waren, wurde dies als korrekt gewertet. Aus diesem Grund wurde die Performance der Relation Detection (mit ausschließlichem Preprocessing) nochmals mit exakter Abgrenzung der Abstimmungen gemessen. Wie bereits erwähnt, wurde in Tabelle 4.6 auch eine unscharfe Abgrenzung zugelassen, solang das Extraktionsergebnis davon nicht beeinflusst wäre, es also trotzdem die in Abschnitt 4.3.4 beschriebenen Relationen, und nur diese, liefert.
In Tabelle 4.7 erkennt man einen deutlichen Einbruch der Werte, wenn man nur noch exakte Markierungen für die Abstimmungen zulässt. So sinkt der F-Measure des Lernmodells auf Basis von P16/85 von 74% auf 22%. Hier zeigt sich im Vergleich mit Tabelle 4.4, dass die Extraktion mit ausschließlichem Preprocessing Probleme hat, den Bereich abzugrenzen. Mit Named Entities als Feature ist dieses Problem hingegen nicht vorhanden.

Die abschließende Bewertung der Ergebnisse fällt relativ schwierig aus. Es zeigt sich, dass ein Preprocessing wichtig ist um Streufehler zu vermeiden. Allerdings ist auch die reine Erkennung mittels Named Entities erfolgreich und vor allem hinsichtlich der Extraktionsgenauigkeit wichtig. Wie in Tabelle 4.7 gezeigt, ist dies ein großes Problem, wenn man keine Named Entities benutzt.
Wenn man das Lernmodell erweitern könnte, so dass mehr Abstimmungen in den Operator einfließen, könnte das Ergebnis deutlich besser ausfallen. Die besten Ergebnisse bezüglich der Erkennung sind darüberhinaus bei einem etwas verringerten Preprocessing und in Verbindung mit der NER zu erwarten.

4.4 Datenextraktion

4.4.1 Extraktion von Relationen zwischen Protokollen und Drucksachen

Der Relation-Builder erstellt einen einfachen Index, der alle referenzierten Drucksachen eines Plenarprotokolls auflistet. Wichtig ist dies, da wir in den verschiedenen Komponenten unseres Systems wissen müssen, wo wir nach Inhalten suchen können. Beispielsweise ist es wichtig zu wissen, wann ein bestimmter Antrag in einem Plenarprotokoll besprochen bzw. über diesen abgestimmt wurde, da wir diese Informationen nicht aus der Drucksache geliefert bekommen. Ebenso wir der Index bei der Erstellung des Lucene-Index benötigt.

x	10		
Laufzeit	1590 min		
Tag	**P**	**R**	**F**
ORG	0.821 ± 0.100	0.700 ± 0.070	0.751 ± 0.062
PERS	0.954 ± 0.056	0.930 ± 0.102	0.941 ± 0.081
REAC	0.994 ± 0.018	0.994 ± 0.009	0.994 ± 0.012
LOC	0.838 ± 0.261	0.505 ± 0.226	0.597 ± 0.221
NAME	0.879 ± 0.209	0.812 ± 0.099	0.824 ± 0.159
PAR	0.978 ± 0.021	0.978 ± 0.015	0.978 ± 0.013
BUR	0.944 ± 0.038	0.904 ± 0.063	0.922 ± 0.041
DRSTYP	0.844 ± 0.161	0.835 ± 0.162	0.828 ± 0.146
ERG	0.233 ± 0.396	0.100 ± 0.166	0.135 ± 0.224
DRUCKSACHE	0.463 ± 0.466	0.467 ± 0.476	0.464 ± 0.469
Gesamt	0.916 ± 0.139	0.879 ± 0.044	0.891 ± 0.091

Tabelle 4.3: Ergebnisse der erweiterten NER-Testläufe auf zwei markierten Protokollen

Lernmodell	P16/85			P16/102		
Testprotokoll	**P**	**R**	**F**	**P**	**R**	**F**
P16/41	100%	66,00%	72,00%	72,73%	66,67%	69,57%
P16/50	100%	100%	100%	100%	100%	100%
P16/102	100%	46,67%	63,64%	100%	80,00%	88,89%
P16/103	87,50%	51,22%	64,62%	59,09%	31,71%	41,27%
P16/109	83,33%	55,56%	66,67%	71,43%	55,56%	62,50%
GESAMT	94,17%	63,89%	73,38%	80,65%	66,79%	72,44%

Tabelle 4.4: Performance der automatischen Triggersuche (mit Preprocessing und NER)

Lernmodell	P16/85			P16/102		
Testprotokoll	**P**	**R**	**F**	**P**	**R**	**F**
P16/41	66,00%	33,00%	44,00%	87,50%	58,33%	70,00%
P16/50	14,29%	40,00%	21,05%	100%	100%	100%
P16/102	93,75%	100%	96,77%	26,67%	80,00%	40,00%
P16/103	30,77%	29,27%	30,00%	50,00%	43,90%	46,75%
P16/109	15,38%	22,22%	18,18%	48,65%	100%	65,45%
GESAMT	44,04%	44,90%	42,00%	62,56%	76,45%	64,44%

Tabelle 4.5: Performance der automatischen Triggersuche (nur NER)

Lernmodell	P16/85			P16/102		
Testprotokoll	**P**	**R**	**F**	**P**	**R**	**F**
P16/41	100%	66,00%	72,00%	83,33%	83,33%	83,33%
P16/50	100%	80,00%	100%	100%	100%	100%
P16/102	80,00%	53,33%	64,00%	91,67%	73,33%	81,48%
P16/103	80,00%	58,54%	67,61%	89,19%	80,49%	84,62%
P16/109	87,50%	77,78%	82,35%	89,47%	94,44%	91,89%
GESAMT	89,50%	67,13%	74,97%	90,73%	86,32%	88,26%

Tabelle 4.6: Performance der automatischen Triggersuche (nur Preprocessing)

Der Relation-Builder ist eine Java Komponente (im Java-Projekt BTPBTDRelationXML genannt), die mit Hilfe von Regular Expressions nach Drucksachen-Referenzierungen in Plenarprotokollen sucht und diese geordnet in XML-Dateien ablegt.

4.4.2 Vorgehensweise der Referenz-Extraktion

Die Erstellung der XML Datei erfolgt in drei Schritten:

1. Finden von Drucksachennummern,
2. Vermeidung doppelter Vorkommen,
3. Ablage in XML-Datei.

Das Finden der Drucksachennummern ist mit Hilfe von Regular Expression realisierbar. Zunächst wird nach dem Ausdruck „$(Drucksache|Drucksachen|Drs)1[4,5,6]$ $/[0-9]1,5$“ gesucht, was sicherstellt, dass alle extrahierten Nummern wirklich den Drucksachen zugehörig sind. Möglich wären auch Plenarprotokoll-Nummern, da diese ebenfalls in den Reden genannt werden. Haben wir einen solchen String gefunden, trennen wir die einleitenden „Drucksachen“ ab und leiten den restlichen String zur Speicherung weiter. Zur Speicherung der XML nutzen wir den XMLOutputter der jdom-Klasse. Als root-Knoten haben wir „BTP2BTD“ gewählt, danach folgt direkt die Aufteilung nach Wahlperiode und Plenarprotokoll. Beide tragen als Parameter die jeweilige Nummer. Alle Drucksachen sind unterhalb des BTP-Knotens in einem BTD-Knoten eingetragen. Ein doppelter Eintrag wird durch die vorherige Prüfung auf ein vorhandenes Vorkommen vermieden. Die daraus entstehende XML sieht dann wie in Abbildung 4.19 aus.

4.4.3 Der umgedrehte Fall: BTD2BTP

Wie man in Abbildung 4.19 sieht, ist durch eine einfache XML-Anfrage nur der Weg von den Plenarprotokollen nach den Drucksachen zu beantworten. Für den umgedrehten Fall, dass wir zu einer Drucksache alle Vorkommen in den Plenarprotokollen haben möchten, wurde eine Methode *getProtokolleByDrs(String Drs)* in der Klasse BTPBTDQuery implementiert. Diese fragt die Elternknoten der BTD-Knoten ab und gibt als Ausgabe ein String-Array der betreffenden Protokolle aus.

4.4.4 Extraktion von Reden

Für die Dossiererstellung ist es notwendig, dass wir Zugriff auf jede im Bundestag gehaltene Rede bekommen. Alle Reden werden in den Bundestagsprotokollen, die im PDF-Format auf http://dip.bundestag.de/ vorliegen, festgehalten. Da in einem Protokoll eine Vielzahl von gehaltenen Reden dokumentiert sind, und diese Dokumente einige 1000 Zeilen lang sein können, ist ein schneller Zugriff auf eine einzelne Rede innerhalb des Dokuments nicht möglich. Daher ist die organisierte Ablage der Reden extrem wichtig. Nachdem wir die Reden einzeln extrahiert und einem Politiker zugeordnet haben, kommen diese in einen eigenen Lucene-Index, der alle Reden enthält. Folgende Aufgaben müssen dabei bewältigt werden:

- (1) Zerlegung des gesamten Bundestagprotokolls in einzelne Sätze,

Lernmodell	P16/85			P16/102		
Testprotokoll	**P**	**R**	**F**	**P**	**R**	**F**
P16/41	87,50%	58,33%	72,00%	75,00%	75,00%	75,00%
P16/50	0,00%	0,00%	0,00%	60,00%	60,00%	60,00%
P16/102	25,00%	13,33%	17,39%	91,67%	73,33%	81,48%
P16/103	22,22%	14,63%	17,65%	59,46%	53,66%	56,41%
P16/109	6,67%	5,56%	6,06%	66,67%	66,67%	66,67%
Gesamt	28,28%	18,37%	22,62%	70,65%	65,73%	67,91%

Tabelle 4.7: Performance der automatischen Triggersuche mit scharfer Abrenzung (nur Preprocessing)


```
- <BTP2BTD>
  - <LEGISLATIVE_PERIOD number="14">
    - <BTP number="14/65">
        <BTD>14/1836</BTD>
        <BTD>14/1898</BTD>
        <BTD>14/5074</BTD>
        <BTD>14/5531</BTD>
        <BTD>14/5544</BTD>
        <BTD>14/5479</BTD>
        <BTD>14/2518</BTD>
        <BTD>14/2363</BTD>
        <BTD>14/5137</BTD>
        <BTD>14/5769</BTD>
        <BTD>14/5457</BTD>
        <BTD>14/5066</BTD>
        <BTD>14/4929</BTD>
        <BTD>14/4329</BTD>
        <BTD>14/850</BTD>
        <BTD>14/5584</BTD>
```


Abbildung 4.19: Der Inhalt der XML-Index-Datei

- (2) Suchen nach dem Beginn der Reden,
- (3) Erkennung von einzelnen Redeabschnitten,
- (4) Zuordnung der Politiker,
- (5) Erkennung von Zwischenrufen innerhalb dieser Reden,
- (6) Split der Zwischenrufe nach Politiker.

Der so genannte SpeechExtractor ist eine Java Komponente, die mit Hilfe von regular expressions einzelne Reden in XML-Dateien ablegt. Die Klasse SpeechExtractor ist dabei Teil des speechex Pakets im repository-Builder.

Problematik der Reden-Extraktion

Bei der Erfüllung der oben gestellten Aufgabe wurde schnell klar, dass durch die Konvertierung der PDFs in das ASCII-Format einige Schwierigkeiten auftauchten. So kann es vorkommen, dass in den Protokollen ganze Ketten von Zeichen an der falschen Stelle stehen oder auseinandergezogen werden und an anderer Stelle wieder auftauchen. Eine weitere Schwierigkeit, die bei der Extraktion umgangen werden musste, war die Einleitung der Politiker-Reden. Normalerweise geschieht diese in der Form **Vorname, Name** [**Parteizugehörigkeit**]. In vielen Dokumenten wurde bei der Konvertierung die Reihenfolge vertauscht zu [**Parteizugehörigkeit**] **Vorname, Name** was ebenfalls einiges an Mehraufwand beim Finden der Reden erfordert. Grund für diese Fehler ist das PDF-Format der Bundestagsprotokolle. Hier kann es passieren, dass der Text einspaltig, zweispaltig oder dreispaltig geschrieben ist, was bei der Konvertierung in das ascii-Format zu Problemen führt. Auch werden die im vorigen Abschnitt beschriebenen Einleitungen teilweise vertauscht.

4.4.5 Der Prozess der Reden-Extraktion

Jedes Protokoll wird vom SpeechExtractor zunächst eingelesen und mit Hilfe des Sentence Splitters in einzelne Sätze unterteilt. Da jedem Protokoll eine Inhaltsangabe und eine Auflistung der teilnehmenden Abgeordneten vorausgeht, wird zunächst nach dem Beginn der Redeabschnitte gesucht. Sobald dieser gefunden ist, markieren wir die Stelle mit einem <content/>-Tag. Alle Reden müssen sich nachfolgend innerhalb dieses Elements befinden.

Die Suche nach dem Beginn der einzelnen Reden ist schwieriger. Hier können wir uns zur Hilfe machen, dass üblicherweise der Bundestagspräsident jeden einzelnen Redner aufruft. Wir suchen also in einem Satz nach dem Vorhandensein von „präsident“ und der Beendigung des Satzes mit einem Doppelpunkt, da hiermit der Redner eingeleitet wird. Abbildung 4.20 zeigt ein Beispiel anhand eines PDF-Dokuments.

Alles was zwischen dem nächsten Vorkommen des oben genannten Triggers steht, wird in der XML-Datei als Rede erfasst und mit einer eindeutigen ID sowie der Anzahl von Wörtern und Sätzen , als Parameter, abgelegt. Da wir allerdings als Rede nicht das Preludium des Präsidenten haben möchten, unterteilt sich jede Rede in ein Preludium, das wiederum einem bestimmten Akteur, nämlich den Präsidenten, enthält. Um das Preludium vom restlichen

Vizepräsident Dr. h. c. Rudolf Seiters: Es gibt doch noch eine Frage zu diesem Punkt. Bitte schön, Frau Fischer.

Andrea Fischer (Berlin) (BÜNDNIS 90/DIE GRÜNEN): Herr Minister, gerade haben Sie in der Antwort auf die Frage des Kollegen Dautzenberg gesagt, dass es Regeln der ordnungsgemäßen Preisfeststellung gibt. Vorhin haben Sie darauf hingewiesen, dass 90 Prozent des Handels elektronisch stattfinden. Würden Sie so weit gehen, zu sagen, dass der Beruf des Kursmaklers ein Anachronismus ist?

Abbildung 4.20: Beispiel für den Beginn einer Rede

<speech speech_id="16023_18" sentences="95" words="1218"></speech>

Abbildung 4.21: Repräsentation der Rede in der XML

Redetext abzutrennen, suchen wir in der erfassten Rede, innerhalb der nächsten Sätze nach einem neuen Redner. Wird ein Redner gefunden, können wir sicher sein, dass es sich um einen neuen Redner handelt und markieren alle vorangeganen Sätze als Preludium, alle Nachfolger erhalten hingegen die Kennzeichnung des Akteuers mit einem <actor> Element und den Parametern Rolle, Name, Partei und mopsid. Die Rolle beschreibt, dass es sich bei diesem Akteur um eine Person handelt und nicht etwa um eine Partei, wie es beispielsweise bei Zwischenrufen der Fall sein kann. Der Name, Partei und mopsid werden aus der Members of Parliament-Datenbank gewonnen, die eine Liste aller Bundestagsabgeordneter mit einer zugehörigen ID enthält.

Damit wir dieses Programm allerdings auch ansprechen können, müssen wir den Namen des Abgeordneten ohne Titel in unserem Redetext finden. Auch hier werden eine Reihe von Regular Expressions benutzt um den Namen in der Redeeinleitung zu finden. Unterscheiden muss man den weiter oben geschilderten Fall, dass die Reihenfolge der Namensaufrufe variieren kann, den Aufruf von Bundesministern, den Aufruf von Präsidenten sowie der/des Bundeskanzler/in. Damit hat die bis jetzt entstanden XML-Datei folgendes Aussehen:

Extraktion von Zwischenrufen innerhalb von Reden

In jeder Rede gibt es eine Menge an Zwischenrufen. Diese können entweder von einer ganzen Fraktion oder auch von einzelnen Abgeordneten ausgehen. Beifall oder Heiterkeit wären Reaktionen einer ganzen Partei, während es bei einzelnen Abgeordneten auch möglich ist Fragen zu stellen oder zu widersprechen. Da die erkannten Reden auch nur Reden des Abgeordneten enthalten sollten und keine Zwischenrufe, werden diese markiert und für eine Analyse ebenfalls einer Partei oder einem Abgeordneten zugeordnet. Das Erkennen dieser Zwischenrufe kann mittels Regular Expressions realisiert werden, da jeder Zwischenruf in Klammern steht. Es

```
− <speech speech_id="16023_18" sentences="95" words="1218">
  − <prelude>
    − <actor role="person" name="Susanne Kastner" party="SPD" mopsid="000177">
        Vizepräsidentin Dr. h. c. Susanne Kastner: Das Wort hat der Kollege Axel Schäfer, SPD-Fraktion.
      </actor>
    </prelude>
  + <actor role="person" name="Axel Schäfer" party="SPD" mopsid="000864"></actor>
  </speech>
```

Abbildung 4.22: Repräsentation der Rede mit Preludium und Aktuer

> (Beifall bei der SPD – Dr. Uwe Küster [SPD]: Stellen Sie sich das einmal vor: So viel Frage und so wenig Antwort! Ist das nicht schön? – Gegenruf des Abg. Dirk Niebel [FDP]: Das ist auch nichts Neues!)

Abbildung 4.23: Zwischenrufe im PDF-Dokument

muss als nur geprüft werden, ob der Inhalt innerhalb der Klammern auch einem Zwischenruf entspricht, was durch den Aufruf eines Abgeordneten oder auch einer Reaktion einer Partei passieren kann. Diese Reaktionen sind relativ eng begrenzt und im Programm als Trigger-Liste gegeben. Sobald ein Zwischenruf erkannt wurde, markieren wir dieses mit dem <interjection>

Element und splitten innerhalb des Zwischenrufs nach verschiedenen Reaktionen. In einem Zwischenruf können mehrere Reaktionen festgehalten sein, so dass diese Prozedur nötig ist. Im <interjection> Tag ist daher auch noch einmal der komplette Zwischenruf vorhanden, während innerhalb des <actor> Elements nur der Beitrag des Akteurs enthalten ist. Genau wie bei der Annotation der Reden gibt es zu jedem Zwischenruf eine eindeutig ID bestehen aus Protokollnummer, Redennummer und einer Laufnummer des Zwischenrufs. In Abbildung 4.24 sehen wir ein Beispiel für die Repräsentationd er Zwischenrufe.

Ergebnisse

Der SpeechExtractor extrahiert Reden aus den Bundestagsprotokollen und annotiert dazu ein mögliches Preludium sowie alle Zwischenrufe innerhalb der Reden. Die Extraktion der Zwischenrufe ist durch die gute Markierung im Text kein Problem. Probleme entstehen bei der Extraktion der Reden sowie der Erfassung des Akteurs. Hintergrund ist hier die Problematik der Umwandlung von PDF nach ASCII.

```
− <interjection interjection_id="16023_18_2">
    <actor role="person" name="Klaus Uwe Benneter" party="SPD" mopsid="000715"> (Klaus Uwe Benneter [SPD]: Hört! Hört!) </actor>
    (Klaus Uwe Benneter [SPD]: Hört! Hört!)
  </interjection>
```

Abbildung 4.24: Repräsentation der Zwischenrufe in Reden

Wir kommen zur Abstimmung über den Entwurf eines Gesetzes der Fraktion Die Linke zur Änderung des Gesetzes zur Regelung der erwerbsmäßigen Arbeitnehmerüberlassung. Der Ausschuss für Arbeit und Soziales empfiehlt in seiner Beschlussempfehlung auf Drucksache 16/7513, den Gesetzentwurf der Fraktion Die Linke auf Drucksache 16/4805 abzulehnen. Ich bitte diejenigen, die dem Gesetzentwurf zustimmen wollen, um das Handzeichen. – Wer stimmt dagegen? – Enthaltungen? – Der Gesetzentwurf ist damit in zweiter Beratung mit den Stimmen von SPD, Bündnis 90/Die Grünen, CDU/CSU, FDP bei Gegenstimmen der Fraktion Die Linke abgelehnt. Damit entfällt nach unserer Geschäftsordnung die weitere Beratung.

Abbildung 4.25: Beispiel für eine typische Abstimmung innerhalb des Plenarprotokolls

4.4.6 Extraktion von Abstimmungen

Damit wir Aussagen über das Abstimmungsverhalten der verschiedenen Parteien oder Personen mit Hilfe von Rapid Miner machen können, müssen wir die Abstimmungen zunächst aus den Bundestagsprotokollen extrahieren und in einer XML abspeichern. Bevor wir die Komponente näher beschreiben, sehen wir in Abbildung 4.25 ein Beispiel für eine typische Abstimmung innerhalb eines Plenarprotokolls.

Das Programm BTPExtract ist eine Java Komponente (im Java-Projekt BTPContent), die mit Hilfe von Regular Expressions nach Abstimmungen innerhalb der Plenarprotokolle sucht und mit Hilfe eines Sliding Window über den Stringdaten, relevante Informationen zu den Abstimmungen extrahiert.

Vorgehensweise der Abstimmungs-Extraktion

Die Erstellung der XML Datei erfolgt in fünf Schritten:

1. Finden von Abstimmungstextstellen in den Protokollen anhand von Triggern,
2. Extraktion des Abstimmungsergebnis,
3. Extraktion der Drucksachen,
4. Extraktion der Empfehlung.

Finden von Abstimmungstextstellen Sobald wir den Text des Plenarprotokolls eingelesen haben, gehen wir Satz für Satz durch diesen Text und suchen nach Triggern für Abstimmungen im Bundestag. Das satzweise Durchlaufen des Textes wird durch den Sentence

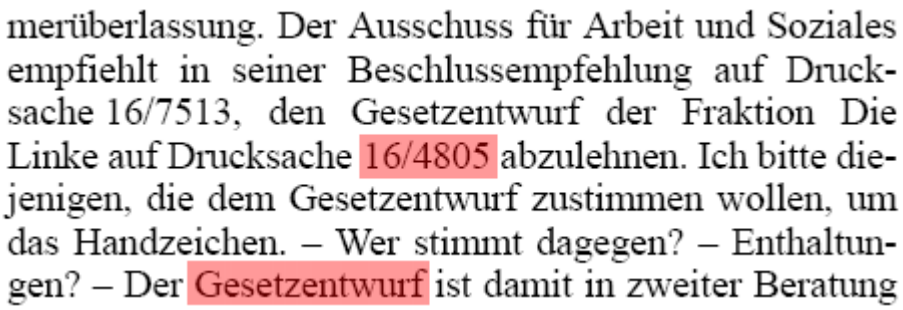

merüberlassung. Der Ausschuss für Arbeit und Soziales empfiehlt in seiner Beschlussempfehlung auf Drucksache 16/7513, den Gesetzentwurf der Fraktion Die Linke auf Drucksache 16/4805 abzulehnen. Ich bitte diejenigen, die dem Gesetzentwurf zustimmen wollen, um das Handzeichen. – Wer stimmt dagegen? – Enthaltungen? – Der Gesetzentwurf ist damit in zweiter Beratung

Abbildung 4.26: Beispiel für die Relation zwischen Drucksache und Abstimmung

Splitter 3.3 ermöglicht, der Markierungen für jedes Satzenden innerhalb des Textes anlegt. In nahezu allen Fällen wird einer der Trigger „Enthaltungen?“, „enthält sich?“ oder „enthalten?“ genannt, so dass wir unseren Text nach Fundstellen dieser Trigger durchsuchen. Wichtig ist für die weitere Extraktion der Abstimmung, dass wir ein festes Extraktionsfenster wählen, da wir sicherstellen müssen, das nur Daten zur gefundenen Triggerstelle bzw. zur Abstimmungstelle gefunden werden.
In diesem Fall beträgt das Fenster 7 Zeilen nach hinten bzw. 1 Zeile nach vorn, ausgehend von der Triggerstelle.

Extraktion des Abstimmungsergebnis Das Abstimmungsergebnis wird in jedem Plenarprotokoll direkt nach der Frage zur Abstimmung genannt. Wir suchen also direkt in der nächsten Zeile nach Vorkommen von „angenommen“ oder „abgelehnt“. Falls wir ein solches Vorkommen finden, endet an dieser Stelle die Extraktion des Abstimunngsergebnis, andernfalls suchen wir auch noch in den nächsten 4 Zeilen nach einem Ergebnis und stoppen, sobald ein solches gefunden wurde.

Extraktion der Drucksachen In jeder Abstimmung kommen eine Vielzahl von Drucksachen vor. So gibt es zu einem Antrag auch eine oder mehrere Beschlussempfehlungen und Berichte oder es können Änderungsanträge referenziert werden. Daher ist es wichtig, zu jeder extrahierten Drucksache zu prüfen, um welchen Typ es sich handelt. Hierzu benutzen wir den BTD-Index, der zu jeder Drucksachennummer den zugehörigen Typ abspeichert. Falls nun dieser zurückgegebene Typ, mit dem Typ des zuvor extrahierten Abstimmungstyps in der Ergebniszeile, übereinstimmt, übernehmen wir die Drucksachennummer als Drucksache der Abstimmung.
Alle anderen vorkommenden Drucksachen werden als Related Documents zu dieser Abstimmung abgespeichert. So können Hinweise auf vorhandene Beschlussempfehlungen ebenfalls nützlich sein und werden in der XML-Datei abgelegt.

Extraktion der Empfehlung Im Zusammenhang mit einer Abstimmung wird oft auch das Ergebnis der Beschlussempfehlung genannt. Damit wir direkt anhand der Daten unserer XML-Datei entscheiden können, ob bei einer Zustimmung, die Zustimmung zur Beschlussempfehlung oder aber die Zustimmung zum Antrag gemeint ist, speichern wir das Ergebnis dieser Beschlussempfehlung mit ab. Daher ist es auch wichtig zu wissen, auf welche Drucksache sich die Annahme oder Ablehnung bezieht. Nur so können wir die Abstimmung sinnvoll nachvollziehen.

```
- <BTP>
    <NR>16/35</NR>
    <DATE>11.05.2006</DATE>
    <MAXTO>54</MAXTO>
  - <VOTE>
      <TYPE drs="16/1413">Entschließung</TYPE>
      <RESULT>abgelehnt</RESULT>
      <TO>Tagesordnungspunkte 3</TO>
      <TONR>26</TONR>
    </VOTE>
  - <VOTE>
      <TYPE drs="16/738" empfehlung="anzunehmen">Gesetzentwurf</TYPE>
      <RELATED typ="Beschlussempfehlung und Bericht">16/1419</RELATED>
      <RELATED typ="Beschlussempfehlung und Bericht">16/1419</RELATED>
      <RESULT>angenommen</RESULT>
      <TO>Tagesordnungspunkt 23</TO>
      <TONR>31</TONR>
    </VOTE>
```


Abbildung 4.27: XML-Repräsentation der Abstimmungen

Ablage in XML-Datei

Die extrahierten Daten werden in einer XML, nach Legislaturperiode und Protokollnummer sortiert, abgespeichert. In jedem Protokollknoten gibt es eine Datumsknoten sowie eine Reihe von „VOTE“-Knoten, die das Ergebnis der Abstimmungen speichern. Jeder „VOTE“-Knoten hat als Kinder Knoten vom Typ „TYPE“, „RESULT“, „TO“, „TONR“ und kann „RELATED“-Knoten enthalten. Im Knoten „TYPE“ wird als Attribut die Drucksachennummer („drs“), auf die sich die Abstimmung bezieht sowie, falls vorhanden, die Empfehlung („empfehlung“) aus einer vorhandenen Beschlussempfehlung gespeichert. Im Element selbst steht der Abstimmungstyp, also eine Information, die angibt, über welchen Drucksachentyp hier abgestimmt wurde.
Der „RESULT“-Knoten speichert lediglich das Ergebnis der Abstimmung, so wie es dort vorkommt. In der Regel also „abgelehnt“ oder „angenommen“. Im „RELATED“-Knoten befinden sich weitere zu dieser Abstimmung zugehörige Drucksachen mit Typ und Nummer.
Die letzten beiden Knoten „TO“ und „TONR“ speichern einserseits den echten Tagesordnungspunkt innerhalb der Protokollsitzung, andererseits die laufende Tagesordnungspunktnummer der Sitzung. Hintergrund ist hier, dass nicht alle Tagesordnungspunkte mit der Numerierung übereinstimmen. Es kann also vorkommen, dass eine Sitzung bereits bei Punkt 2 oder 3 anfängt. Nützlich können diese Informationen sein, wenn wir Gründe für ein bestimmtes Abstimmungsverhalten suchen.

Deutscher Bundestag Drucksache 16/128
16. Wahlperiode 01. 12. 2005
Antrag
der Abgeordneten Gisela Piltz , Sibylle Laurischk , Sabine Leutheusser-Schnarrenberger , Jens Ackermann , Dr. Karl Addicks , Christian Ahrendt , Rainer Brüderle , Angelika Brunkhorst , Ernst Burgbacher , Patrick Döring , Mechthild Dyckmans , Jörg van Essen , Otto Fricke , Paul K. Friedhoff , Horst Friedrich (Bayreuth), Dr. Edmund Peter Geisen , Hans-Michael Goldmann , Miriam Gruß , Joachim Günther (Plauen), Dr. Christel Happach-Kasan , Heinz-Peter Haustein , Birgit Homburger , Dr. Werner Hoyer , Michael Kauch , Dr. Heinrich L. Kolb , Hellmut Königshaus , Gudrun Kopp , Jürgen Koppelin , Heinz Lanfermann , Ina Lenke , Horst Meierhofer , Patrick Meinhardt , Jan Mücke , Burkhardt Müller-Sönksen , Dirk Niebel , Hans-Joachim Otto (Frankfurt), Detlef Parr , Jörg Rohde , Dr. Konrad Schily , Marina Schuster , Dr. Hermann Otto Solms , Dr. Max Stadler , Dr. Rainer Stinner , Carl-Ludwig Thiele , Florian Toncar , Christoph Waitz , Dr. Claudia Winterstein , Dr. Volker Wissing , Hartfrid Wolff (Rems-Murr), Martin Zeil , Dr. Wolfgang Gerhardt und der Fraktion der FDP
Gegen eine europaweit verpflichtende Vorratsdatenspeicherung
Der Bundestag wolle beschließen:
I. Der Deutsche Bundestag stellt fest:
1. Der Deutsche Bundestag bekräftigt angesichts des nunmehr vorliegenden
...

Abbildung 4.28: Eine Beispieldrucksache mit Makierungen der Attribute

4.4.7 Extraktion der Attribute aus Drucksachen

Um Referenzen zwischen verschiedenen Dokumenten, Personen und sonstigen Entitäten herstellen zu können, müssen auch für Drucksachen festgelegte Attribute aus diesen extrahiert werden. Dies wurde bereits für andere Entitäten erklärt, z.B. für Personen in Kapitel 4.1, für Protokolle in 4.4.6 und für Reden in 4.4.4.

Drucksachenattribute

Für eine Drucksache haben wir uns darauf geeinigt, dass folgende Attribute extrahiert werden müssen (in kursiv ist immer ein explizites Beispiel angegeben, welches aus dem Beispiel aus Abbilung 4.4.7 entnommen wurde):

- Nummer - *16/128*
- Legislaturperiode - *16*
- Datum - *01.12.2005*
- Typ - *Antrag*

```
⟨BTD_INDEX⟩
  ⟨LEGISLATIVE_PERIOD number="14"⟩
    ⟨DOC id="14348"⟩
      ⟨NR⟩16/128⟨/NR⟩
      ⟨DATE⟩01.12.2005⟨/DATE⟩
      ⟨TYPE⟩Antrag⟨/TYPE⟩
      ⟨TITLE⟩Gegen eine europaweit verpflichtende Vorratsdatenspeicherung⟨/TITLE⟩
      ⟨AUTHORS⟩
        ⟨AUTHOR id="000849"⟩Gisela Piltz⟨/AUTHOR⟩
        ⟨AUTHOR id="000813"⟩Sibylle Laurischk⟨/AUTHOR⟩
        ⟨AUTHOR id="000220"⟩Sabine Leutheusser-Schnarrenberger⟨/AUTHOR⟩
        ⟨AUTHOR id="001070"⟩Jens Ackermann⟨/AUTHOR⟩
        ⟨AUTHOR id="000702"⟩Karl Addicks⟨/AUTHOR⟩
        ...
        ⟨AUTHOR id="001064"⟩Hartfrid Wolff⟨/AUTHOR⟩
        ⟨AUTHOR id="001067"⟩Martin Zeil⟨/AUTHOR⟩
        ⟨AUTHOR id="000103"⟩Wolfgang Gerhardt⟨/AUTHOR⟩
        ⟨AUTHOR id="−1"⟩FDP⟨/AUTHOR⟩
      ⟨/AUTHORS⟩
    ⟨/DOC⟩
  ⟨/LEGISLATIVE_PERIOD⟩
⟨/BTD_INDEX⟩
```

Abbildung 4.29: Die Beispieldrucksache im XML-Format des Btd-Indexes

- Titel - *Gegen eine europaweit verpflichtende Vorratsdatenspeicherung*
- Authoren - *Gisela Piltz, Sibylle Laurischk, Sabine Leutheusser-Schnarrenberger, Jens Ackermann, Karl Addicks, . . . , Hartfrid Wolff, Martin Zeil, Wolfgang Gerhardt, FDP*

Desweiteren wird zu jedem Author, sofern dieser eine Person ist und nicht z.B. eine Partei, die jeweilige eindeutige *Mops-Id* (siehe Kapitel 4.1.4) benötigt. Außerdem wird eine eindeutige Drucksachen-Id angelegt.

Erstellung einer Drucksachen-Xml

Das Programm **btd_index_builder**, welches die Drucksachenattribute automatisch extrahiert, benötigt als Eingabe einen Pfad zu den Plain-Ascii-Dokumenten (siehe Kapitel 3.1.1). Von diesem Pfad aus crawlt sich das Programm durch alle Unterverzeichnisse und arbeitet alle Dokumente in diesen ab. Dokumente die nicht sauber genug umgewandelt wurden, werden nicht mit in den Index aufgenommen.
Es werden nun über reguläre Ausdrücke und Reihenfolgeberücksichtigung die in Kapitel 4.4.7 aufgelisteten Attribute extrahiert. Über einen kleinen selbst angelegten Thesaurus, werden die Parteinamen normalisiert. Für die Authoren *Ausschuss* und *Partei* wird die eindeutige Personen-Id auf „-1“ gesetzt. Für alle anderen Authoren wird eine Anfrage an eine einmalig erzeugte *Name-zu-Id-Hashmap* gestellt. Wird ein Author nicht in dieser gefunden, so erhält auch er die Id -1. Jede Drucksache wird mit seinen Attributen in einer Xml-Datei abgelegt.

⟨ bundestag⟩
  ⟨ wahlperiode **n**ummer = „“ **a**nfangsdatum = „“ **e**nddatum = „“ ⟩
    ⟨ fraktion ⟩
      ⟨ partei **n**ame = „“ **r**egiert = „“ **s**ize = „“ ⟩
        ⟨ person **n**ame = „“ **v**on = „“ **b**is = „“ ⟩
        ⟨ amt **v**on = „“ **b**is = „“ ⟩ Name des Amtes⟨/amt⟩
        ⟨/ person⟩
      ⟨/ partei⟩
    ⟨/ fraktion⟩
  ⟨/ wahlperiode⟩
⟨/ bundestag⟩

Abbildung 4.30: Die XML für den BundestagLookUp

Unsere Beispieldrucksache aus Abbildung 4.4.7 hat in der Xml-Datei nun die Form, welche in Abbildung 4.4.7 zu sehen ist.

Anfragen an den Drucksachen-Index

Es ist ein leichtes Anfragen an den Drucksachen-Index zu stellen. Beispielsweise kann über eine Drucksachennummer ein Objekt vom Typ *Dokument* mit allen Attributen erhalten werden. Da dies allerdings nur für die Erstellung des Lucene-Index (siehe Kapitel 5.2) benötigt wird und von da an alle Informationen im Lucene-Index enthalten sind und auch von dort abgefragt werden können, werde ich hier nicht weiter auf die Anfragen eingehen.

4.4.8 BundestagLookUp

Die XML des Bundestags (Parteien, Minister, Regierung) ist per Hand erstellt worden. Dies ist nicht automatisiert worden, weil die Informationen zu Beginn schnell verfügbar sein sollten. BundestagsLookUp liefert Information über die Abgeordneten, die innerhalb der jeweiligen Wahlperioden ein Ministerposten besitzen oder besaßen.

Aufbau des Strukturbaumes:

Die Baumstruktur zeigt auf wie die XML aufgebaut ist.
1) gibt an in welcher WP wir uns befinden und wie lange diese dauerte 2) gibt die Fraktionen an, dieser folgt 3)Partei, welche an Fraktion gebunden ist. Hier erfahren wir um welche Partei es sich handelt und ob diese die Regierung bilden. Mit Hilfe von 4) und 5)bekommen wir die Namen der Abgeordneten heraus, die Ämter besitzen oder im Falle der nicht regierenden Parteien nur die Parteivorsitzenden. Zu diesen Informationen kommt immer die Dauer von wann bis wann die Ämter/Parteivorsitz von den Abgeordneten bezogen worden sind und ein Vermerk für den Eintritt in die Partei. Partei, Person und Amt sind unter Fraktion verschachtelt.
Falls die Partei nicht regiert, wurde nur der Parteivorsitzender eingetragen.
Um die ganzen Minister der Legislaturperioden (14-16) und die Dauer dieser Ämter zu finden wurden folgende Seiten verwendet.

1) für die Minister der Regierung Schröder:
http://www.muenster.de/ urcom/Bundestagswahl/Minister_Schroeder.htm
2) für die Minister der Regierung Merkel:
http://www.muenster.de/ urcom/Bundestagswahl/Minister_Merkel.htm

Um die Daten, für die Parteizugehörigkeit zu finden, wurde dafür zum einen Wikipedia und zum anderen die Bundestagsseite genutzt. Die Anzahl der Sitze im Bundestag, erhält man mit der Hilfe der Seiten:
http://www.bundeswahlleiter.de/wahlen/btw98/btwahl_btw98.htm (Für WP 14,15) und
http://www.bundeswahlleiter.de/bundestagswahl2005/ergebnisse (Für WP 16).

BTLQueryAnfragen:

Mit der Hilfe dieser Query, welche auf BTL zugreift, können nun einfache Fragen über die Minister und Parteien im Bundestag beantwortet werden. Diese wird innerhalb der Query Facade verwendet.
Automatisierter BundestagLookUp:

Dieses automatisierte Verfahren wurde erstellt, um aufwändiges und erneutes per Hand suchen von Informationen über die Abgeordneten, die ein Amt in der jeweiligen Wahlperiode besaßen oder besitzen zu vermeiden. Die fehlende Update-Funktion über Änderungen innerhalb der Informationsquelle, wie z.B. Minister von einem Amt ändert sich usw., wurde nun auch ergänzt. In dem Fileworker startet die Updatefunktion und zwar wird einfach auf das Programm zugegriffen und die automatisch erstellte XML durch eine neue XML ersetzt.
Dies ist ohne weiteres möglich, denn die Menge der vorhandenen Einträge ist ziemlich gering. Ein anderer Ansatz wäre gewesen die vorhandene XML und die Einträge auf der Seite zu Vergleichen. Da dies im Endeffekt die gleiche Zeit braucht, lassen wir einfach die vorhandene XML ersetzen. Die Query-Anfragen laufen wie schon beim normalen BTL über die Query Facade. Die ganzen Informationen sollen von der Wikipedia-Seite extrahiert werden.
Mit Hilfe dieses Programms kann man nun automatisch Informationen über die Minister und Kanzler der jeweiligen Legislaturperioden bekommen.
Es wird zwischen den Wahlperioden (auf der „Wikipedia“ Seite Kabinett genannt) unterschieden und angegeben welche Partei welche Minister stellt/e. Änderungen innerhalb der Seite wie z.B. Amtsinhabers ändert sich oder neue Wahlperiode, werden mit Hilfe der Updatefunktion angezeigt.
Aufbau des Strukturbaumes:
Die Struktur hat sich ein wenig geändert und manche Attribute und Werte sind auch nicht mehr vorhanden(siehe vorherige XML).

⟨ bundestag⟩
  ⟨ wahlperiode **n**ummer = „“ **a**nfangsdatum = „“ **e**nddatum = „“ ⟩
    ⟨ amt ⟩ Name des Amtes
      ⟨ partei ⟩ Name der Partei
        ⟨ person **n**ame = „“/⟩
      ⟨/ partei⟩
    ⟨/ amt⟩
  ⟨/ wahlperiode⟩
⟨/ bundestag⟩

Abbildung 4.31: Die XML für den Auto-BundestagLookUp

5 Systemkomponenten

5.1 Lexical Tree (L-Tree)

Im Rahmen der Aufgaben der PG ist an vielen Stellen eine wortbasierte Suche nach Dokumenten erforderlich. Aufgrund der Größe des Dokumentenarchivs (ca. 40.000) ist dazu ein besonders effizienter Wort-Index nötig.

Die Datenstruktur Der L-Tree ist eine Datenstruktur, die diesen Anforderungen gerecht wird. Im Kern ist sie ein B-Baum, wobei jeder Knoten einen Buchstaben repräsentiert. Ein Wort $w = (b_1, ..., b_n)$, bestehend aus n Buchstaben, entsteht dann durch einen von der Wurzel ausgehenden Pfad. Der Knoten des letzten Buchstaben speichert eine Liste mit Verweisen auf die Dokumente, in denen das Wort vorkommt. Das Vorhandensein der Dokumentenliste in einem Knoten kennzeichnet damit ein gültiges Wort(-ende). Abbildung 5.1 verdeutlicht die Datenstruktur. D stellt dabei die Menge aller Dokumente dar. Die Dokumentenliste enthält aber nicht nur einen Verweis auf die entsprechenden Dokumente, sondern speichert auch die Anzahl der Vorkommen des Wortes im jeweiligen Dokument selbst. Dadurch ist es ein Leichtes, ein Ranking der Dokumente zu einem Wort vorzunehmen. Hierzu wurde die TF-IDF Gewichtung anhand der nachfolgenden Formel verwendet. Sei dafür $L(w) = \{d \in D | w \in d\} \subseteq D$ die Dokumentenliste für ein Wort w und $freq(w, d_j)$ die Häufigkeit, mit der w in d_j vorkommt.

$$tfidf(w, d_j) = \frac{freq(w, d_j)}{\max\limits_j \ freq(w, d_j)} * log\left(\frac{\|D\|}{\|L(w)\|}\right) \tag{5.1}$$

Wie man sich leicht klarmacht, sind alle zur Berechnung erforderlichen Angaben in der beschriebenen L-Tree Datenstruktur verfügbar. Die Dokumentenliste wurde mit dem Java-Datentyp *Hashmap* implementiert. Die Schlüssel der Hashmap bilden die Dateinamen, denen

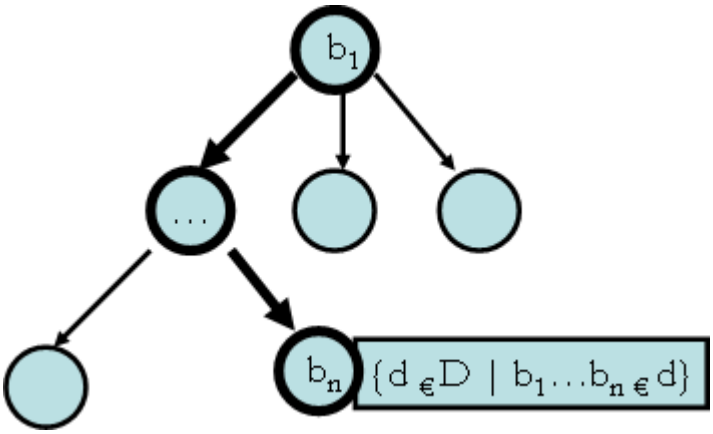


Abbildung 5.1: L-Tree Datenstruktur

dann die Häufigkeiten der Wörter im jeweiligen Dokument zugeordnet werden. Dies bildet die Funktion $freq(w, d_j)$ ab. Die Anzahl der verschiedenen Dokumente $\|L(w)\|$, in denen das betrachtete Wort vorkommt, kann durch die Eigenschaft *size()* abgerufen werden. Die maximale Häufigkeit $\max_j freq(w, d_j)$ lässt sich leicht in der Liste finden, die Anzahl aller Dokument $|D|$ wird beim Erstellen des Index mitgezählt und dann gespeichert.

Effizienz Was diese Datenstruktur als Index so effizient macht ist die Tatsache, dass die Wörter in sich ergänzender Weise gespeichert werden. Teilt sich ein Wort mit einem anderen die ersten k Buchstaben, so werden für beide Worte die selben k Anfangsknoten benutzt. Erst wenn sich die Buchstaben unterscheiden, also für den jeweils $(k+1)$-ten Buchstaben, teilt sich der Pfad. Im besten Fall ist das kürzere Wort vollständig in dem längeren enthalten, so dass für das kürzere Wort keinerlei eigene Knoten angelegt werden müssen. Lediglich der letzte Knoten beider Worte enthält dann die Liste mit den Dokumentenverweisen. Dieser Aspekt sorgt für eine sehr gute Kompaktierung und spart viel Speicherplatz.
Ein weitere wichtige Anforderung ist das effiziente Auffinden der Dokumentenliste zu einem gegebenen Wort. Durch die Baumstruktur kann die Dokumentenliste zu einem Wort mit n Buchstaben auch in $O(n)$ gefunden werden, da nur der Pfad, der das gegebene Wort bildet, verfolgt werden muss.

Größenabschätzung Der Index bildet eine zentrale Komponente des Systems und muss daher an vielen Stellen schnell verfügbar sein. Am Besten ist dies durch eine vollständige und dauerhafte Residenz im Hauptspeicher zu erreichen. Eine grobe Abschätzung soll zeigen, ob so der Index bei der vorliegenden Datenmenge in einen handelsüblichen Hauptspeicher passt. Dabei darf der L-Tree natürlich auch den Speicher nicht zu großen Teilen für sich beanspruchen, da das System noch viele weitere Komponenten hat. Die grobe Abschätzung soll dann empirisch belegt werden.
Sei l_max die Länge des längsten Worts, l_avg der Längendurchschnitt. l_max ist auch die Länge des längsten Pfades und kennzeichnet die Tiefe des L-Trees. Da die Knoten Buchstaben repräsentieren ist auch die Menge der möglichen Knotentypen auf die zugelassenen Buchstaben begrenzt. Dies sei b_max. Ein Knoten kann also maximal b_max Kinder haben. Sei b_avg die durchschnittliche Anzahl der Kinder. Jedes Kind wird am Elter als 32-Bit-Adresse referenziert. Die Speicherung des Buchstaben als Java Datentyp *char* benötigt 16 Bit. Ein L-Tree mit k Wörtern hat maximal $k * l_max$ Knoten mit je $(b_max * 4 + 2)$ Byte. Die Größe der Dokumentenliste wird zunächst nicht betrachtet. Aus den vorhandenen Dokumenten lassen sich für die Domäne Politik $l_max = 67$ und $k = 470.000$ ermitteln. Als Buchstaben werden a-z, A-Z, Ä, Ö, Ü , ä, ö, ü und ß zugelassen, was $b_max = 59$ festlegt. Demnach wäre die Größe des L-Tree auf ca. 6,8 GB begrenzt. Offensichtlich eine zu grobe Abschätzung, da die reine Speicherung aller 470.000 Wörter als Textdatei nur 10 MB benötigt und unsere Datenstruktur keinen derart großen Overhead produziert. Um die Größenabschätzung realistischer zu gestalten werden nachfolgend Durchschnittswerte für die Parameter verwendet. Aus der Verteilung der Wortlängen, die in Abbildung 5.2 dargestellt ist, lässt sich $l_avg = 15$ bestimmen. Da fast nie ein Großbuchstabe auf einen Kleinbuchstaben folgt, fallen 30 Buchstaben als Kinder heraus. Zudem wird der Baum mit jeder Ebene immer spärlicher besetzt, so dass für $b_avg = 15$ angenommen wird. Als durchschnittliche Größe ergibt sich so ca. 450 MB. Ein Testlauf (ohne Dokumentenlisten) ließ den L-Tree 150 MB groß werden.
Mit Hinzunahme der Dokumentenlisten wurde die Indexierung problematisch. Nach gut ei-

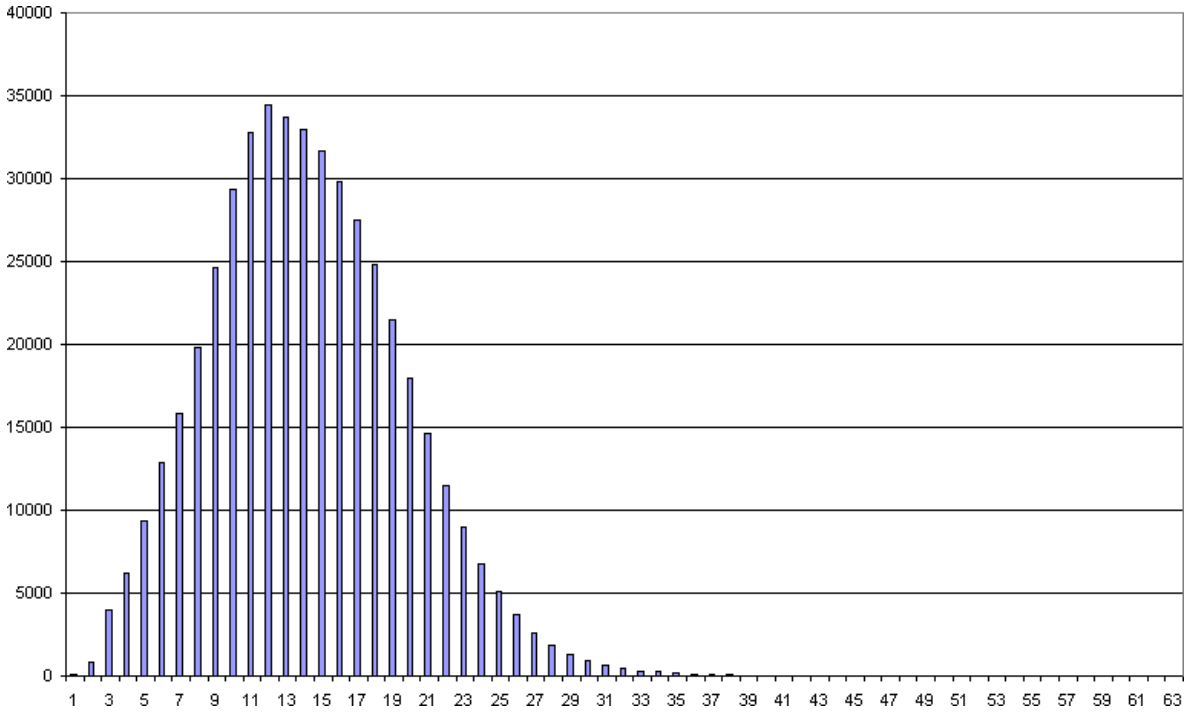


Abbildung 5.2: Häufigkeiten (y-Achse) der Wortlängen (x-Achse) in den Dokumenten des Systems

nem Drittel der Dokumente brach das Programm, ausgeführt auf einem Windows-Computer mit 2 GB Arbeitsspeicher, aus Mangel an Hauptspeicher ab. Die Heap-Größe der JVM wurde zuvor auf den maximal möglichen Wert von ca. 1,6 GB vergrößert. Das Problem konnte auch nach optimierender Umprogrammierung nicht behoben werden. Erst durch die Einführung von 250 Stopwörtern, die bei der Indexierung ignoriert werden, ließ sich der Index erstellen. Die Stopwörter wurden hautsächlich durch eine Häufigkeitsanalyse ermittelt. Paradoxerweise ist der L-Tree mit Verwendung der Stopwort-Liste nur ca. 400 MB groß. Die Gründe dafür sehen wir einerseits in eventuellen Unzulänglichkeiten in der Programmierung, andererseits aber auch in der Speicherverwaltung der JVM, vor allem in Bezug auf die Verwendung des Datentyps *Hashmap* mit sehr vielen Einträgen. Unsere Recherchen haben allerdings ergeben, dass dieser Datentyp der effizienteste Standard-Datentyp in Java ist. Diese Umstände sorgen bei der Verwendung des L-Trees im System für Probleme.

Anwendungsprobleme Wie bereits erwähnt soll der L-Tree vollständig im Hauptspeicher residieren. Einmal geladen zeigt der L-Tree die erwartete Schnelligkeit im Abruf der Dokumentenlisten zu einem Suchwort. Das Laden des 400 MB Indexes dauert allerdings 10 Minuten - eine Zeitspanne, die einen zu großen Mangel des fertigen Systems darstellen würde. Daher wurde, parallel zu den Verbesserungsversuchen am L-Tree, nach fertigen Open-Source Lösungen gesucht, die ähnliches leisten können. Das Lucene-Projekt (http://lucene.apache.org/) stellte sich dabei als Alternative heraus. Die Architektur unterscheidet sich dahingehend, dass Lucene einen dokumentbasierten Index anlegt, nicht einen wortbasierten, wie der L-Tree. Nach Testläufen sprachen aber folgende Aspekte für die Verwendung von Lucene an Stelle unserer

L-Tree Implementierung:

- Problemlose Indexerstellung mit allen Dokumenten (auch ohne Stopwörter)
- Sehr geringe Ladezeit im Bereich von Sekunden
- Gute Kontrolle über die indexierten Daten, mit der Möglichkeit beliebige Zusatzdaten zu einem indexierten Dokument zu speichern
- Parser für Suchanfragen, der Abfragen nach dem Prinzip gängiger Suchmaschinen erlaubt
- Gesucht wird optional auch unter Einbeziehung der Zusatzdaten
- Integriertes Ranking
- Größe der Indexdatei beträgt insgesamt nur ca. 150 MB. Das ist die Größe, die der L-Tree Index ohne Dokumentenlisten hat.

Aufgrund der genannten Vorteile haben wir uns zur Benutzung von Lucene entschlossen und die Arbeiten am L-Tree eingestellt. Die Implementierung von Lucene wird im Folgenden noch genauer beschrieben.

5.2 Lucene

Lucene (http://lucene.apache.org/) ist eine in Java realisierte Open-Source Suchmaschine. Da sie schon seit vielen Jahren besteht, gilt sie als sehr ausgereift und effizient. Im vorherigen Kapitel wurde bereits erläutert, in wie weit Lucene unserem L-Tree überlegen ist. Ein weiteres westenliches Feature ist die nahezu unbegrenzte Anpassbarkeit des Systems. Die zentrale Komponente ist der dokumentenbasierte Index. Ein Dokument (Lucene-Dokument) besteht dabei aus Name-Value-Paaren, die wahlweise indexiert werden können. Abbildung 5.3 verdeutlicht dies. Wir pflegen mit Lucene zwei Indizes. Der erste Index ist der *Dokumenten-Index*, der alle BTD und BTP enthält. Dieser wird mitunter auch als der „Lucene-Index“ bezeichnet, da er lange Zeit der einzige Lucene-Index war. Ein Lucene-Dokument entspricht in disem Fall den einzelnen BTP-/BTD-Dokumenten in Form der ASCII-Dateien und hat die folgenden Felder:

- Dokumenttext
  Der Dokumenttext wird indexiert. Dadurch ist das Dokument komplett durchsuchbar.
- Autorenliste
  Diese Liste wird einmal als Liste mit den Namen der Autoren im Klartext und einmal als Liste mit den Identifikationsnummern der MOPS-Datenbank (eine Personen-Datenbank) geführt. Die erste Liste vereinfacht die Suche nach Teilen des Namenes. So lassen sich schnell alle Dokumente finden, die z.B. von einem „Helmut“ verfasst wurden. Würden die Namen nicht im Klartext indexiert, so müsste für das Beispiel erst eine Anfrage an die MOPS-Datenbank ausgeführt werden, um die Identifikationsnummern aller „Helmuts“ zu erhalten. Die zweite Liste schafft die direkte Verbindung zur MOPS-Datenbank.

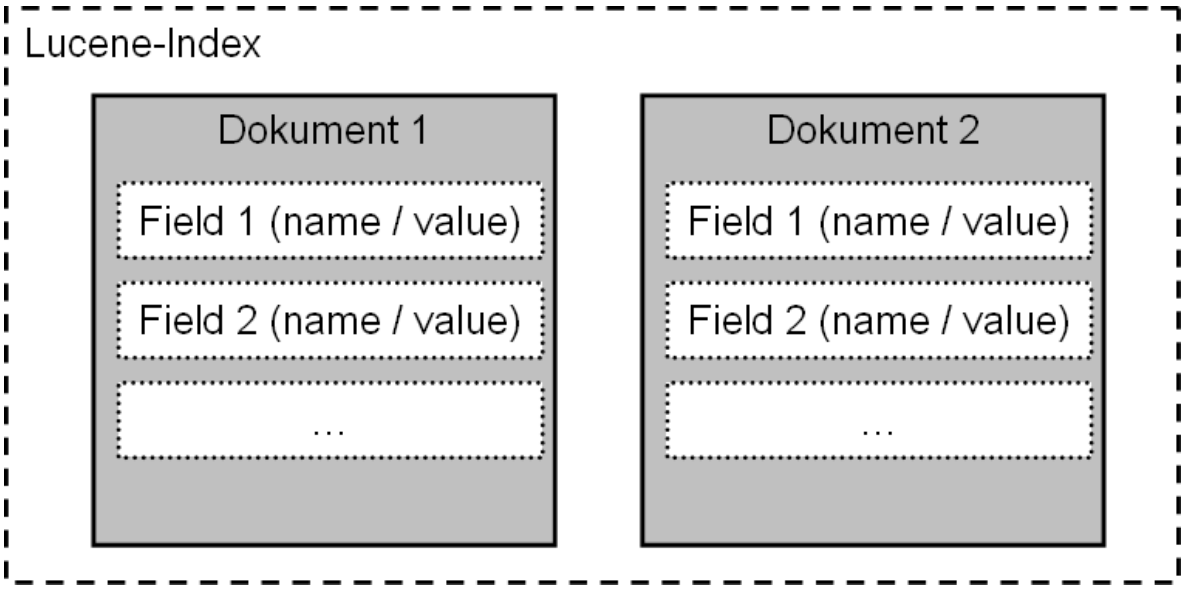


Abbildung 5.3: Aufbau des Lucene Index

- Dokumenttyp
  Der Typ des Dokument nach den Kategorien des Bundestages (Kleine Anfrage, große Anfrage, Beschlussempfehlung, ...)
- Titel
  Der Titel des Dokuments soll den Inhalt zusammenfassen und wird ebenfalls indexiert.
- Pfad
  Der Pfad des ASCII-Dokuments im Dateisystem.
- URL
  Der Pfad des PDF-Dokuments auf der Bundestags-Webseite. Dieses Feld wird genutzt, damit die GUI bei Bedarf eine aktuelle Version des Dokuments für den Benutzer aus dem Internet laden kann. So ist es nicht nötig, alle PDFs lokal vorzuhalten.
- Datum
  Datum der Veröffentlichung (BTD) bzw. der Plenarsitzung (BTP).
- Nummer
  Drucksachen- oder Plenarprotokollnummer.
- Drucksache oder Protokoll
  Dieses Feld gibt an, um welchen der beiden Typen es sich handelt.
- Wahlperiode

Die Informationen in den Feldern werden bei der Erstellung des Dokumenten-Index aus drei Quellen bezogen:

1. aus den ASCII-Dokuemten selbst (Inhalt, Pfad).

2. aus dem BTD-Index
   Dieser Index extrahiert zuvor mittels regulärer Ausdrücke alle relevanten Informationen zu den BTD in ein XML. Bei der Erstellung des Lucene-Index wird anhand der Drucksachennummer der entsprechende XML-Knoten gefunden.

3. BTP-Index
   Dieser Index funktioniert analog zum BTD-Index.

Der zweite Index, der neben dem Dokumenten-Index mit Lucene gepflegt wird, ist der *Reden-Index*. In Kapitel 4.4.4 wird die Extraktion der Reden in das XML-Format beschrieben. Der Lucene-Reden-Index setzt auf die XML-Daten auf. Ein Lucene-Dokument entspricht hier einer Rede, die folgende Felder enthält:

- Inhalt
  Der reine Redetext wird hier indexiert.
- Redner
  Der Redner wird im Klartext und als ID aus der MOPS-Datenbank gespeichert.
- Datum
  Das Datum der Rede.
- Wahlperiode
- Anzahl der Zwischenrufe
- Zwischenrufer
  Auch diese Personen werden auf die bekannte, duale Weise in den Index aufgenommen.
- Protokollnummer Nummer des Plenarprotokolls, in dem die Rede festgehalten ist. Damit ist auch die Plenarsitzung bestimmt, in der diese Rede gehalten wurde.

Lucene unterstützt auch Funktionen wie Stemming und Stopworte. Dazu wird eine sogennannte Analyser-Klasse verwendet, von der jeder Text, der indiziert werden soll, vorverarbeitet wird. Lucene hat bereits solcherlei Klassen für die deutsche Sprache integriert, so dass wir in einfacher Weise auf diese Funktionaliäten zurückgreifen können.
Die Suchfunktion von Lucene ermöglicht die Suche in einem oder allen Feldern. Sie ist in einer einfachen Abfragesprache realisiert [Luc]. Damit können ganz gezielt Informationen abgefragt werden. Ebenfalls möglich ist die Pharsen- und Platzhaltersuche, wie z.B. nach „Helmut Kohl“ oder „Abgeordnete?“. Die Suchergebnisse werden von Lucence nach einem TF/IDF basierten Score geordnet.

5.3 WT2XML

Nach dem die Dateien den RapidMiner passieren haben, müssen die als XML-Dokumente gespeichert werden. Eine Funktion, die eine Menge von WT-Dokumenten in die Menge von XML-Dokumente abbildet, ist eine Grundlage für das RapidMiner-Plugin. Die Implementierung in Java und dazugehörige Erläuterungen sind im Folgenden dargestellt:

```
1  import java.io.File;
2  import java.io.FileReader;
3  import java.io.BufferedReader;
4  import java.io.BufferedWriter;
5  import java.io.FileOutputStream;
6  import java.io.OutputStreamWriter;
7  import java.nio.charset.Charset;
8  import java.nio.charset.CharsetEncoder;
9  import java.util.LinkedList;
10 import java.util.List;
```

Listing 5.1: WT2XML Klassen 1

```
1  import de.unidortmund.pg520.annotation.module.document.
   SimpleAnnotationDocument;
2  import de.unidortmund.pg520.annotation.module.document.tag.
   SimpleEntityTag;
3  import de.unidortmund.pg520.annotation.module.document.
   SimpleEntity;
4  import de.unidortmund.pg520.annotation.module.util.XmlUtil;
```

Listing 5.2: WT2XML Klassen 2

Die Klassen in Listing 5.1 sind notwendig um alle Funktion zu realisieren. Ersten drei Zeilen sorgen dafür, dass die Datei richtig ausgelesen wird. Die nächsten drei Zeilen sind notwendig um eine Datei zu speichern. Die siebte und achte Zeile sorgen, dass auch die Umlauten richtig gespeichert werden. Und endlich die letzten beiden Zielen laden die genannten Datenstrukturen runter.
Die Kassen in Listing **??** sind dafür da, dass die erzeugten XML-Dokumenten mit dem selben Schema erzeugt werden, das auch bei der Erzeugung der Annotations-Dokumente genutzt wird.
Die Methode in Listing 5.3 gibt die NE, die in der Zeile eines WT-Dokumenets ausgelesen wird, zurück. Falls die Zeile nicht markiert ist, das heißt mit einem „O“ versehen ist, dann wird „null“ zurückgegeben.
Die Methode in Listing 5.4 sagt uns, ob die jeweilige Zeile ein neuer oder fortgesetzter Tag ist.
Im Codeteil in Listing 5.5 wird das Verzeichnis und alle dort befindende WT-Dokumente ausgelesene und ein nach dem anderen bearbeitet. Nach der Entfernung aus der Datei und separaten Speicherung von Tags, wird die Datei zwischengespeichert.
In dem Codabschnitt in Listing **??** wird eine Zeile nach der anderen gescannt. Wenn die

```
1 public class WT2XML {
2 private static String getEntityName(String w2){
3 if(w2.length()<2)
4 return null;
5 else{
6 return w2.substring(2, w2.length());
7 }
8 }
```

Listing 5.3: WT2XML 3

```
1 private static boolean isStartEntity(String w2){
2 return(w2.length()<2) || w2.charAt(0) == 'B';}
```

Listing 5.4: WT2XML 4

```
1  public static void main(String[] args) throws Exception{
2  File file = new File (args[0]);
3  File[] fileArray = file.listFiles();
4  for (int k=0;k<fileArray.length;k++){
5  file = fileArray[k];
6  String fileName = file.getName().substring(0, file.getName().
   length()-4);
7  BufferedReader in = new BufferedReader(new FileReader(file));
8  FileOutputStream fos = new FileOutputStream(new File ("C:/temp.
   wtf"));
9  CharsetEncoder encoder = Charset.forName("UTF-8").newEncoder();
10 OutputStreamWriter osw = new OutputStreamWriter(fos, encoder);
11 BufferedWriter bw = new BufferedWriter(osw);
```

Listing 5.5: WT2XML 5

```
1  String zeile;
2  StringBuffer buf = new StringBuffer();
3  List<SimpleEntityTag<SimpleEntity>> list = new LinkedList<
   SimpleEntityTag<SimpleEntity>>();
4  int i = 0;
5  int begin = 0;
6  String prevEntity = null;
7  while((zeile = in.readLine()) != null ){
8  String[] s = zeile.split(" ");
9  if(s.length<2)
10 continue;
11 String entity = getEntityName(s[1]);
12 if((entity == null || isStartEntity(s[1])) && prevEntity!=null)
13 list.add(new SimpleEntityTag<SimpleEntity>(begin, i-1, new
   SimpleEntity(prevEntity)));
14 buf.append(s[0]+" ");
15 if(isStartEntity(s[1]))
16 begin = i;
17 prevEntity = entity;
18 i =i + s[0].length()+1;
19 }
```

Listing 5.6: WT2XML 6

Zeile mit einem Tag markiert ist, wir der entsprechender SimpleEntity-Objekt erzeugt. Aus dem WT-Dokument werden nun alle Tags entfernt, so dass weiter das Dokument ohne Tags zwischengespeichert wird.
Im Codesegment in Listing 5.7 wird die Datei zwischengespeichert und dann in das SimpleAnnotationDocument eingelesen. Auf den Objekten der SimpleAnnotationDocument-Klasse wird ein XML-Schema definiert, womit dann die XML-Dateien aus WT - Dokumenten erzeugt werden.

5.4 Graphical User Interface

Einleitung

Die Grafische Benutzer-Schnittstelle (kurz: GUI) des Intelligence Service (kurz: IS) orientiert sich stark an den Eigenschaften der benutzten Domäne. Bei der Domäne handelt es sich um den Deutschen Bundestag bzw. die Bundestags-Seite. Also erwartet der IS erfahrungsgemäß einen relativ hohen Altersdurchschnitt. Infolgedessen war von vorneherein klar, dass das GUI leicht zu bedienen sein sollte. Obwohl die GBS grafisch nicht überladen wirkt, navigiert sich

```
1  bw.write(buf.toString(),0,buf.toString().length());
2  bw.close();
3  file = new File ("C:/temp.wtf");
4  SimpleAnnotationDocument annotationDocument = new
   SimpleAnnotationDocument(file,Charset.forName("UTF-8").
   newDecoder());
5  for (int j=0; j<list.size();j++)
6  annotationDocument.insertTag(list.get(j));
7  XmlUtil.initXmlParser();
8  annotationDocument.saveXmlFile(new File(args[1],fileName+".xml")
   );
9  }
10 }
11 }
```

Listing 5.7: WT2XML 7

der Benutzer durch grafische Schaltflächen bis er zum gewünschten Service gelangt. Sofort am Anfang hat der Benutzer die Möglichkeit, den gewünschten Service, in Form von relativ großen Schaltflächen, zu wählen oder genauer zu spezifizieren. Dies sind folgende:

- Volltextsuche und andere Services
- PartyNator
- Fragebeantwortungssystem

Je nachdem welche Schaltfläche der Benutzer drückt, gelangt er direkt zum Service und kann Ihn benutzen oder bekommt weitere, kleinere Schaltflächen womit er zu den Services gelangt. Durch Klick auf “PartyNator” oder “Fragebeantwortungssystem” wird der Service direkt geöffnet. Der Service “Volltextsuche und andere Services” beinhaltet folgende Services:

- Abgeordneten Informationen
- Dokumentensuche
- Stichwortsuche
- Dossier

Zu jedem Service, ob nun direkt aufgerufen oder durch eine zweite Schaltfläche, gibt es eine Beschreibung. Die Beschreibung ist entweder in unmittelbarer Nähe der Schaltfläche angebracht oder erscheint, wenn man mit dem Mauszeiger drüberfährt. Sollte der Benutzer trotzdem mal nicht wissen wo er sich gerade befindet, oder z.B. seine Sitzung unterbrochen hat und sie später wieder fortsetzt, steht das sogenannte “InfoPanel” zur Verfügung. Das “InfoPanel”

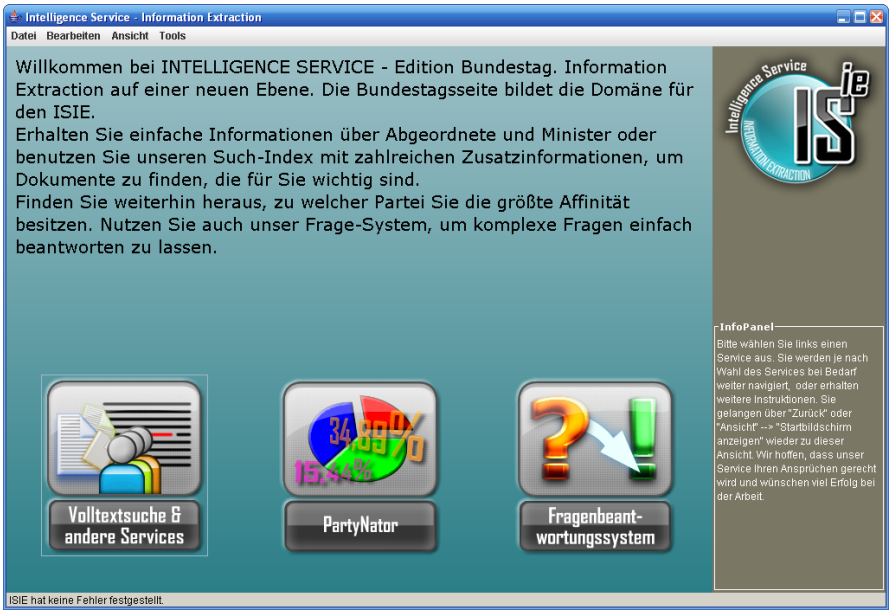


Abbildung 5.4: Startseite der Grafischen Benutzer-Schnittstelle

Abbildung 5.5: Serviceseite

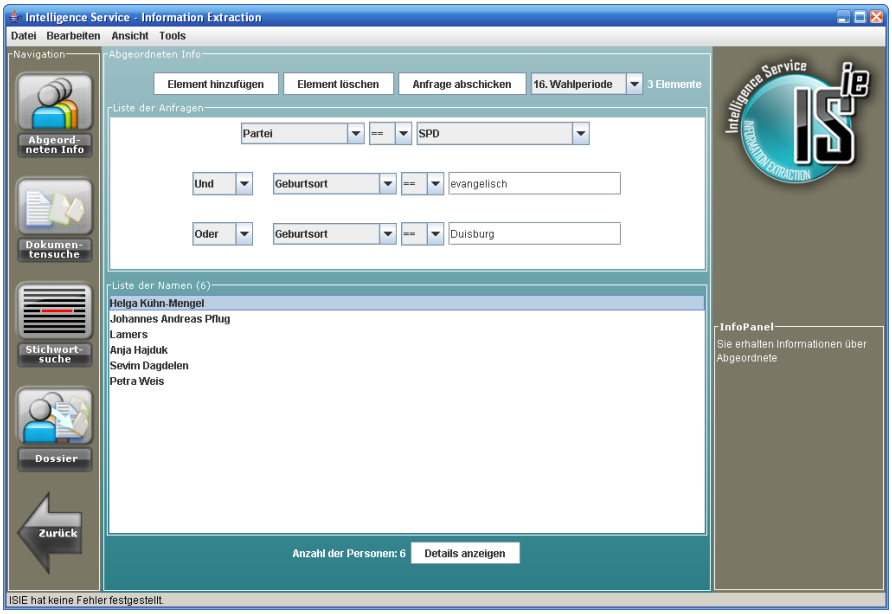


Abbildung 5.6: Abgeordneten Information

zeigt zu jedem Zeitpunkt an, was der Benutzer tun kann oder muss, um sein Ziel zu erreichen. Zusätzlich fängt das "InfoPanel" unvermeidbare Benutzerfehler ab, und gibt Anweisungen wie weiter fortgefahren werden kann. Der Benutzer hat zu jedem Zeitpunkt die Möglichkeit wieder zum Startbildschirm zu gelangen. Die Anfragen und Ergebnisse bleiben dennoch erhalten wenn er zum Service später zurückkehrt. Auch ein Wechsel zwischen den Services vor dem Ausführen der Anfrage führt nicht zum Ergebnisverlust.

Services im GUI

Abgeordneten Informationen Das Anfrage-Schema bei den Abgeordneten Informationen basiert auf einzelnen Elementen. Durch "Element Hinzufügen" kann der Nutzer Anfrage-Elemente beifügen. Die einzelnen Anfragen-Elemente können dann disjunkt oder konjugiert abgeschickt werden. Falls Abgeordnete existieren, die der Anfrage entsprechen, werden Sie unter "Liste der Namen" angezeigt. Der Nutzer hat nun die Möglichkeit eine Person anzuwählen und sich alle vorhandenen Informationen über die Person anzeigen zu lassen. Die letzte Information zeigt immer die Webadresse der Person und ist durch einen Klick abrufbar.

Dokumentensuche Die Anfrage-Maske der Dokumentensuche ist relativ einfach aufgebaut. Der Nutzer hat die Möglichkeit ein oder mehrere Stichworte anzugeben, und somit die Dokumente der Bundestagsseite nach Schlagwörtern zu durchsuchen. Weiterhin besteht die Möglichkeit nicht nur alle Wahlperioden zu durchsuchen sondern eine bestimmte Wahlperiode zu spezifizieren. Die Reihenfolge der gefundenen Dokumente ist durch den Lucene-Wert (Score) festgelegt und wird unter der Liste für das aktuelle Dokument angezeigt. Dementsprechend erscheinen Dokumente, die relevanter sind, auch weiter oben in der Liste. Wenn ein Dokument angewählt ist, werden zudem weitere Informationen angezeigt wie z.B. Titel, Typ,

Abbildung 5.7: Serviceseite

Autor oder die Webadresse des Dokuments womit das Quelldokument auch geöffnet werden kann.

Stichwortsuche Die Anfrage-Maske der Stichwortsuche ist identisch mit der Anfrage-Maske der Dokumentensuche. Allerdings ist die Ausgabe sehr unterschiedlich. Man erhält hier nach einer Anfrage nicht nur die Namen der relevanten Dokumente, sondern Events in Form von kleinen Textschnipseln. Um ein Stichwort herum ist immer ein gewisser Kontext vorhanden, der hier dem Benutzer präsentiert wird. Obwohl die Textschnipsel auch hier bzgl. der Lucene-Werte sortiert sind, ist durch den gegebenen Kontext um ein Stichwort herum eine gezieltere Suche möglich. Wenn dem Nutzer ein Textschnipsel wichtig erscheint, kann er sich das Original-Dokument durch einen Klick anzeigen lassen.

Dossier im GUI Beim Dossier handelt es sich um eine Sammlung von Abgeordneten über Wahlperioden hinweg. Zu einer bestimmten Person kann auf schnellste Weise angezeigt werden, in welchen Dokumenten die Person vorkommt und in welchen Plenarprotokollen die Person eine Rede hält. Näheres dazu in Abschnitt 3.7.

PartyNator im GUI Der PartyNator in der Grafischen Benutzer-Schnittstelle stellt Textfelder zur Verfügung, die der Benutzer aktivieren und ausfüllen kann. Den verschiedenen Attributen können Attribute zugewiesen werden. Nachdem die Anfrage abgeschickt wurde und ein Ergebnis berechnet worden ist, wird dieses übersichtlich in einem animierten Tortendiagramm angezeigt. Die Höhe der Parteiaffinität hängt nicht nur von der Anzahl der Personen ab, die auch den Werten der angegebenen Attributen entsprechen. Näheres dazu in Abschnitt 3.8.

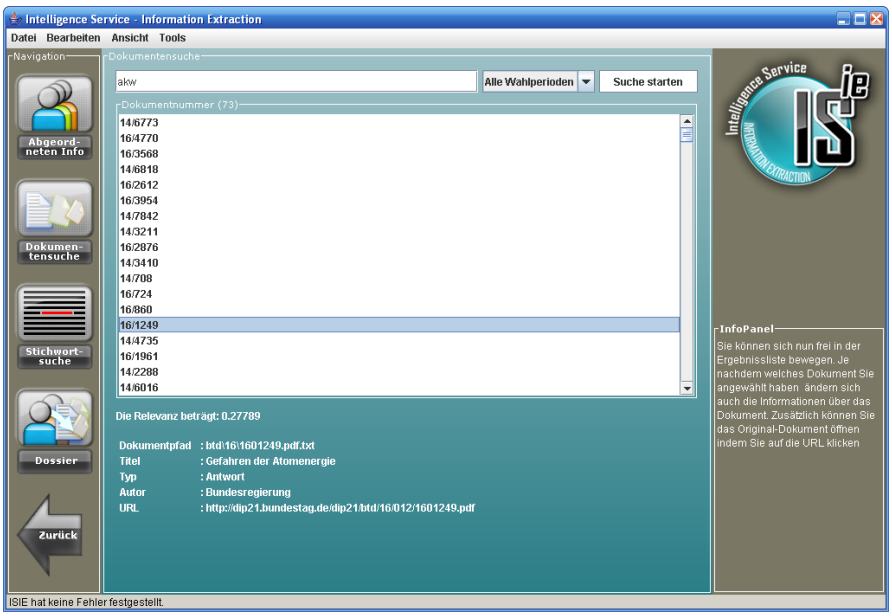


Abbildung 5.8: Dokumentensuche

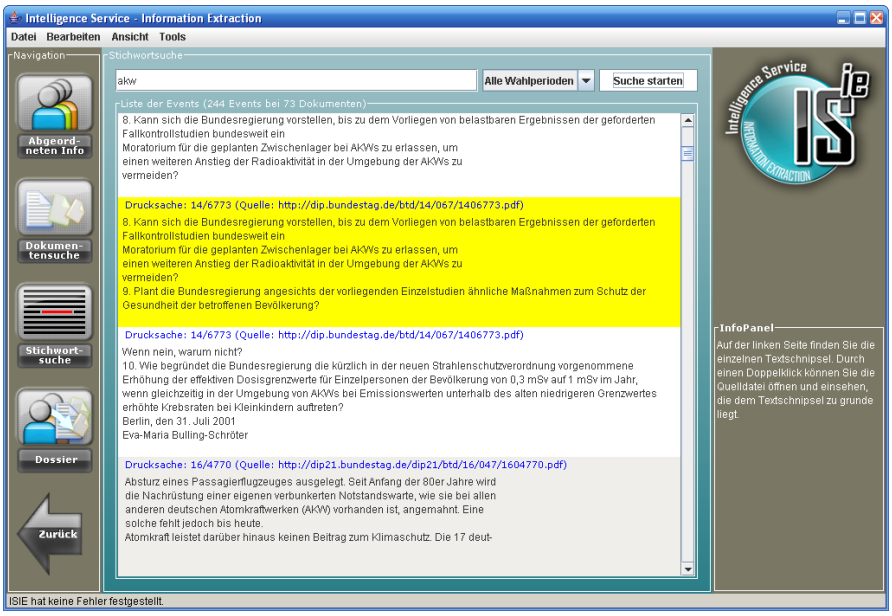


Abbildung 5.9: Stichwortsuche

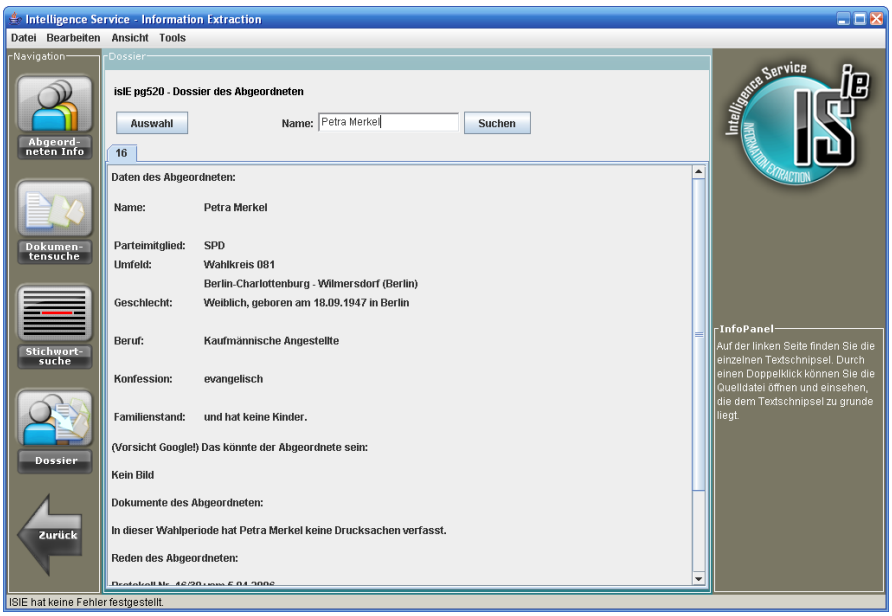


Abbildung 5.10: Dossier

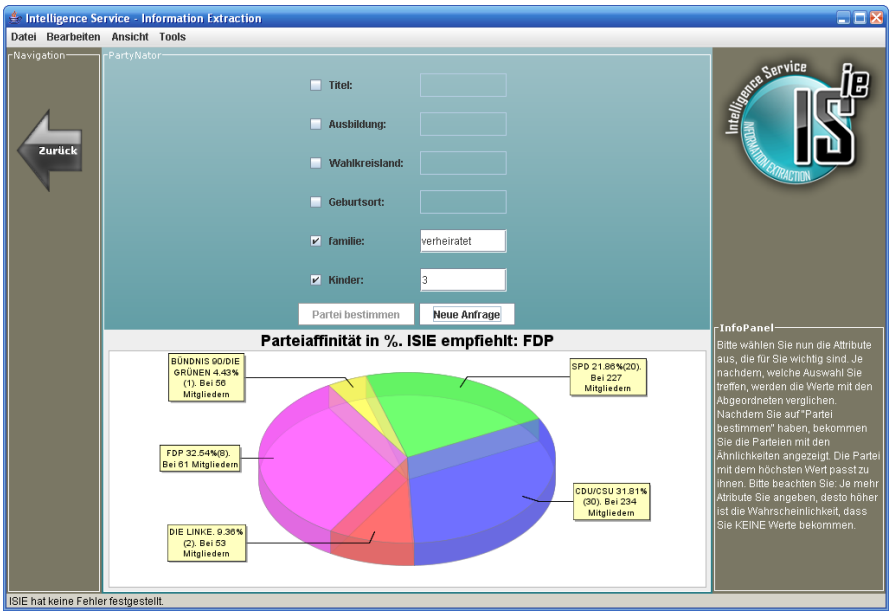


Abbildung 5.11: PartyNator

GUI-Vergleich mit der EN ISO 9241 - 110

Das GUI der IS ist natürlich sehr an die benutzte Domäne angelehnt. Um dennoch die Vor- und Nachteile zu untersuchen, eignet sich ein Vergleich mit einigen Grundsätzen der Dialoggestaltung, welches im Teil 110 der EN ISO 9241 - 110 normiert ist. Teil 110 ersetzt dabei den schon veralteten 10. Teil der Norm. Auch wenn die Norm eigentlich für Webanwendungen zugeschnitten ist, sind die Grundsätze teilweise auf die IS als Offline-Anwendung übertragbar.

Aufgabenangemessenheit Unter dem Aspekt der Aufgabenangemessenheit ist die Einfachheit und Zielorientierung des Interfaces gemeint. Wie in der Einführung schon erläutert, ist die Navigation bis zu einem bestimmten Service Schaltflächenorientiert. Die Schaltflächen im Startbildschirm sind relativ groß bemessen und helfen dem Nutzer zu verstehen, was sich hinter der Schaltfläche verbirgt. Die Schaltflächen haben zwar einen großen Farbraum, sind aber im Durschnitt nicht größer als 30 KB. Da IS in Java geschrieben, und es sich um eine Offline-Anwendung handelt, ist die Größe der Grafiken im Allgemeinen eher unkritisch. Die Anfragen, die die Benutzer stellen können sind so individuell, dass auf Speicherung von Zwischenergebnissen über Programmaufrufe hinweg, verzichtet wurde.

Selbstbeschreibungsfähigkeit Die Selbstbeschreibungsfähigkeit deutet auf gelungene Rückmeldungen und die allgemeine Verständlichkeit des Programms. Wenn im GUI eine Liste oder der gleichen zu sehen ist, wird am Kopf immer der Umfang angezeigt, was der Verständlichkeit enorm entgegen kommt.

Steuerbarkeit Die Steuerbarkeit hebt die Änderbarkeit der Parameter hervor, die bei einer Anfrage an das System wichtig sind. Beim IS sind die Parameter sehr spezifisch bzgl. der Domäne. Überall dort wo es Sinn macht, kann z.B. die Wahlperiode spezifiziert oder die Anzahl der Ergebnisse angegeben werden.

Erwartungskonformität Unter der Erwartungskonformität versteht sich die intuitive Bedienung des Programms. Wird z.B. der (größte Teil) der Nutzer bei der Nutzung intuitiv unterstützt oder sorgt das System für eine, nicht erwünschte und vermeidbare, Verwirrung. Hier empfiehlt die Norm die Platzierung des Logos auf der linken Seite. Im IS-System ist das Logo auf der rechten Seite. Die bedeutsamere linke Seite wird für wichtigere Navigation zwischen den Services benutzt. Links, die zu einer Seite ins Internet führt, sind immer blau und somit leicht unterscheidbar von herkömmlichem Text. Weiterhin gelangt man mit der Tabulator-Taste zum nächsten GUI-Objekt, welches einstellbar ist. Der Focus ist automatisch auf dem Objekt.

Fehlertoleranz Fehlermeldungen sollen laut der Norm nicht technisch oder durch kryptische Nummer angezeigt werden. Beim IS wird im Ansatz auf solche Fehlermeldungen verzichtet. Trotzdem werden unvermeidbare Fehler abgefangen und werden dem Benutzer entweder deutlich angezeigt (z.B. wenn keine Daten angezeigt werden können weil die Anfrage keine Ergebnisse liefert) oder werden natürlich-sprachlich umschrieben und im InfoPanel angezeigt.

Individualisierbarkeit Die Individualisierbarkeit soll ein System bzgl. der individuellen Vorlieben eines Nutzers parametrisieren. Dabei geht es nicht um Anfragespezifische Parameter sondern um reine GUI-Objekte. Diese Möglichkeit ist im System des IS nicht integriert, da die Gefahr einer Verwirrung größer erscheint als der Nutzen für ein Individuelles System.
Wie man erkennen kann ist die Norm nicht Eins zu Eins auf das GUI übertragbar. Dennoch decken sich die Grundzüge der Norm mit dem GUI des IS-Systems.

5.5 Queryfacade

5.5.1 Konzept

Als Queryfacade wird die Abstraktionsschicht bezeichnet, die es erlaubt SQL-ähnliche Anfragen an die stark heterogenen Daten zu stellen, die als Teilergebnisse der Projektgruppe erarbeitet wurden. Der Name leitet sich aus „Query“ für Anfrage und „Facade“ für die Umsetzung anhand des gleichnamigen Programmierpatterns ab. Um heterogene Daten als relationale Daten verarbeiten zu können, muss zuerst die Information festgehalten werden, wie die einzelnen Daten strukturiert sind und wie sie miteinander in Relation stehen. Jede Datenquelle wird also darauf untersucht welche „Entitäten“ sie liefern kann, also z.B. Person incl. aller persönlichen Daten, oder Partei incl. des Namens und der Mitglieder. Diese Eigenschaften von Entitäten werden im Folgenden Attribute genannt. Sind die verfügbaren Entitäten erfasst, muss festgelegt werden wie die Entitäten miteinander in Beziehung stehen. Analog zu einem relationalen Datenbank-Schema muss dabei auch angegeben werden, welches Attribut der verknüpften Entitäten als Fremdschlüssel fungiert. Bei einer Abfrage, die Kriterien an mehrere Entitäten stellt, werden dann anhand dieser Fremdschlüsselattribute Joins ausgeführt, also Wertevergleiche, um festzustellen welche konkreten Instanzen zweier Entitäten miteinander in Beziehung stehen. Daraus ergibt sich dann ein relationales Schema, das in XML gespeichert wird. Die Abstraktionsschicht besteht aus drei Hauptbestandteilen, zum einen ist dies der Parser, der das relationale Schema liest und in ein Objektmodell überführt, zum zweiten ist dies die Oberfläche, die das zusammenklicken von Fragen erlaubt, die dem eingelesenen Schema entsprechen und zuletzt aus einer Frageverarbeitungskomponente, die die zusammengeklickten Fragen annimmt und auswertet, sowie das Ergebnis zur Anzeige an die Oberfläche zurückgibt. Die letzte Schicht gliedert sich wiederrum in sogenannte Adapter. Dabei handelt es sich um Klassen, die einem bestimmten Interface genügen müssen und den strukturierten Zugriff auf jeweils eine Datenquelle erlauben. Im Grunde handelt es sich dabei um Rohdaten-zu-Objekt Konverter.

5.5.2 Architektur

Im Folgenden werden die oben genannten einzelnen Schichten näher erläutert.

Relationales Schema und Objektmodell

Im relationalen Schema müssen zu jeder Entität bestimmte Informationen hinterlegt werden. Für die Entität selbst muss angegeben werden, welche Java-Klasse als Container für Instanzen der Entität verwendet wird, welche Java-Klasse als Adapter für diese Entität fungiert und welchen eindeutigen Namen diese Entität tragen soll. Als Kinder-Knoten der Entität werden

```
<ENTITY name="Amt" queryClass="de.unidortmund.pg520.queryfacade.adapter.BTLAdapter"
        class="de.unidortmund.pg520.queryfacade.result.Amt">
    <ATTRIBUTE field="bezeichnung" name="Bezeichnung" type="String" role="mit der Bezeichnung %value%"/>
    <ATTRIBUTE field="name" name="Name" type="String" role="besetzt durch %value%">
        <RELATION name="besetzt durch" toEntity="Person" keyAttribute="Name" role=" ist besetzt durch eine Person %value%" />
    </ATTRIBUTE>
    <ATTRIBUTE field="wahlperiode" name="Wahlperiode" type="Select" values="14,15,16" role="in Wahlperiode %value%">
        <RELATION name="in Wahlperiode" toEntity="Wahlperiode" keyAttribute="Name" role="in Wahlperiode %value%" />
    </ATTRIBUTE>
    <ATTRIBUTE field="beginn" name="Beginn" role="mit Amtsbeginn am %value%" type="Date" />
    <ATTRIBUTE field="ende" name="Ende" role="mit Amtsende am %value%" type="Date" />
</ENTITY>
```

Abbildung 5.12: Beispiel einer Entitätendefinition

ihre Attribute festgehalten, zu denen die Informationen: eindeutiger Name, Name des Attributs in der Containerklasse, der Datentyp, sowie ein Textfragment zur Einbettung in einen natürlichsprachlichen Satz gehören. Erlaubte Typen für Attribute sind „String“, „Integer“, „Date“, „Select“ und „List“. Bei Angabe von Select muss zusätzlich eine Liste von Auswahlmöglichkeiten für die Selectbox angegeben werden. Bei der Angabe von „List“ muss der Rückgabetyp in der Containerklasse ein String-Array oder eine java.util.List mit Elementen vom Typ String sein. Mit diesen Mechanismen ist es möglich Beziehungen der Typen 1-zu-1, 1-zu-n, sowie m-zu-n zu realisieren. Attribute können als Kinder-Knoten nun Relationen enthalten, da eine Relation ja stets von einem Fremdschlüssel abhängig ist. Zu einer Relation müssen die Zielentität, ein Name für die Relation, der Attributname des Fremdschlüssels in der Zielentität, sowie ein Textfragment zur natürlichsprachlichen Einbettung. Da an jedes Attribut auch Bedingungen geknüpft werden könne, wird in diesem Textfragment durch die Zeichenfolge %value% markiert an welcher Stelle im Text das Eingabefeld für den Benutzer angezeigt werden soll.
Der oben genannte Parser liest dieses Schema nun ein und übersetzt es in ein Gefüge von Objekten, das durch direkte Referenzen miteinander in Beziehung steht. Dies erlaubt ein „Durchhangeln“ anhand dieses Graphen-ähnlichen Gebildes zur Halb-dynamischen Fragestellung. Dieses sogenannte QOM oder QueryObjectModel wird als zum einen von der Oberfläche als Schema benutzt, das der Fragestellung zu Grunde liegt, zum anderen wird es in der Frageverarbeitung dazu verwendet die Joins auszuführen, da dort alle dafür notwendigen Informationen vorliegen.

Verarbeitung von Anfragen

Die grafische Oberfläche der QueryFacade ist in der Lage die vom Benutzer erstellte Frage in XML Form auszudrücken. Dabei wird eine verschachtelte Struktur verwendet, die eine beliebige Tiefe von Joins zwischen Entitäten zulässt.
Im Wurzelelement „QUERY“ wird mit dem Attribut „targetEntity“ festgelegt nach welcher Entität gesucht wird. Darauf folgt ein „AND“, das theoretisch erlaubt aus mehreren per Join entstandene Ergebnismengen den Durchschnitt zu bilden (dies wird bislang nicht unterstützt). Danach kann über „CONSTRAINT“ eine Bedingung an die Zielentität gestellt werden und eine beliebig tiefe Folge von „JOIN“ Elementen folgen, wobei jedes „JOIN“ seinerseits „CONSTRAINT“ Elemente enthalten kann, die dann Bedingungen auf die über „toEntity“ angegebene Entität realisieren. Jedes „CONSTRAINT“ enthält dabei die Informationen auf welches Attribut es sich bezieht, von welchem Typ dieses Attribut ist und auf welchen Wert geprüft werden soll.

```
<QUERY targetEntity="Ausschuss">
    <AND>
        <CONSTRAINT type="String" value="" name="Name" />
        <JOIN toEntity="Wahlperiode">
            <CONSTRAINT type="String" value="14" name="Name" />
        </JOIN>
    </AND>
</QUERY>
```

Abbildung 5.13: Beispiel einer QueryFacade Anfrage

Die Verarbeitung der Anfrage verläuft dann rekursiv von innen nach aussen. Es wird also zuerst auf die Entität im innersten „JOIN“ Element zugegriffen, dann auf die Entität eine Ebene höher. Dann wird ein Join ausgeführt, der aus der äußeren Ergebnismenge alle herausfiltert, bei denen keine Fremdschlüsselbeziehung zu einem Elemenent der inneren Ergebnismenge besteht. Dieser Vorgang wird so lange wiederholt, bis das Wurzelelement der XML-Struktur und somit die gesuchte Entität erreicht ist.
Das JAVA Interface, das von den Adaptern implementiert werden muss, erzwingt für die Datenrückgabe den Datentyp „QueryResult“. Daher müssen alle Entitäten auf Containerklassen abgebildet werden, die von diesem Typ erben. In der Oberfläche wird dann per Fallunterscheidung der tatsächliche Datentyp geprüft und ein zur Entität passendes Ausgabepanel gewählt.

5.5.3 Fragevorschau

Die Fragevorschau wird innerhalb der GUI beim Fragebeantwortungssystem verwendet, um die vom User zusammen geklickten Fragen natürlich sprachlich anzeigen zu können. Hierbei gibt es unterschiedliche Verknüpfungsmöglichkeiten, die bei der Fragestellung durch „und“ verbunden werden. Entweder wählt man ein Attribut und macht eine Eintragung per Hand und schickt die Frage ab oder man wählt eine weiterführende Verknüpfung, diese sind mit einem ” → ” gekennzeichnet. Die Fragestellungsmöglichkeiten wurden von uns eingeschränkt damit unsinnige oder nicht auswertbare Fragen nicht gestellt werden können. Der User folgt einem vordefinierten Schema und hat nur die Möglichkeit, die von uns voreingestellten Entitäten und Attribute auszuwählen. Er kann nur in den Textfeldern frei Eingaben tätigen, um seine Frage weiter zu spezialisieren.

5.5.4 Fragebeantwortungssystem

Zu Begin hat der Nutzer die Möglichkeit eine Zielentität auszuwählen. Mit Hilfe dieser Entität wird die Frage spezialisiert und es können z.B. Fragen über eine Person(Abgeordneten), ein Amt, Protokolle, Ausschüsse, Drucksachen, Wahlperioden und Dokumente gestellt werden. Nachdem nun die Zielentität ausgewählt wurde gibt es für jede Entität eigene Bedingungen die angeklickt und bestimmt werden können. Als erstes folgt hier ein Attribut welches gewählt werden kann, danach können weitere weiterführende Verknüpfungen gewählt werden, welche innerhalb der Fragevorschau mit „und“ verbunden werden. Bei jeder Frage hat man zudem

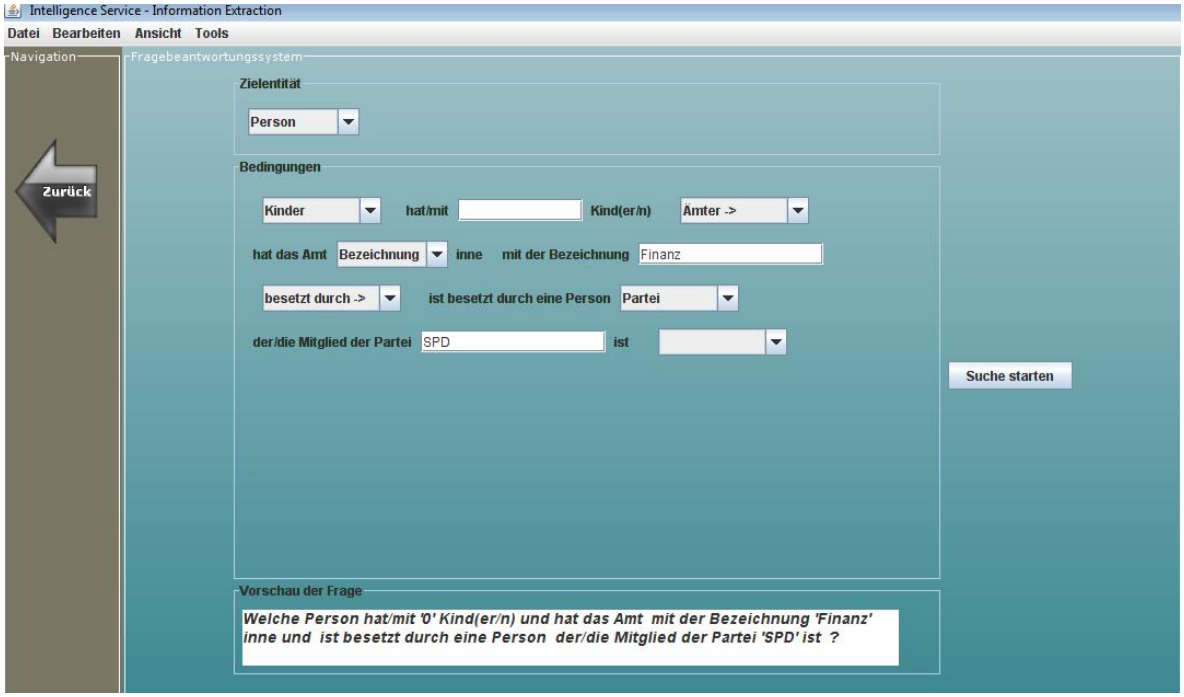


Abbildung 5.14: Eine beispielhafte Frage

die Möglichkeit selbst per Hand Daten/Informationen anzugeben und somit die Möglichkeit die Frage weiter einzuschränken, um die gewünschte Antwort zu erhalten.

Beispielhafter Ablauf

Nachfolgend betrachten wir beispielhaft ene Zielentität Person mit den folgenden Bedingungen:

1. Kinder (Angabe Anzahl der Kinder)
2. Ämter ” → ” (weiterführende Verknüpfung der Frage gekennzeichnet mit ” → ”)
3. Bezeichnung (Angabe eines Amtes)
4. Besetzt durch ” → ”(weiterführende Verknüpfung der Frage gekennzeichnet mit ” → ”)
5. Partei (Angabe einer Partei)

Danach klickt man auf den Button „Suche starten“ und es wird die Antwort für die gestellte Frage geliefert.

Vorschau

1. Wie es gut läuft: Welche Person hat/mit ’2’ Kind(er/n) und hat das Amt mit der Bezeichnung ’Finanzminister’ inne und ist besetzt durch eine Person der/die Mitglied der Partei ’SPD’ ist ?

2. Wie es nicht so gut läuft: In welchem Ausschuss und der/die Mitglied der Partei 'SPD' ist und tätig in Wahlperiode '14' ?

Wir sehen hier, dass dieses nicht immer optimal abläuft, denn es ist eine sehr komplexe Aufgabe eine Struktur in eine natürlich sprachliche umzuformen.
Dies ist uns auch anhand unserer Seminare und den dort besprochenen Themen klar geworden. Es wäre gut gewesen, wenn die Vorschau, wie unten gezeigt, aufgebaut worden wäre. Dies ist uns leider nicht so gut gelungen. Wieso dies nicht geklappt hat, wurde weiter oben kurz erläutert.

Beispiele für eine gute Fragevorschau

Amt: Welches Amt hatte die (Person) inne?
Welches Amt wurde von (Person) ausgeführt?// Welches Amt hatte die (Person) in der WP(Nr.) inne?
welche Ämter gab/gibt es in der Wahlperiode(Nr.)?
welches Amt begann am (Datum)?
welches Amt endete am (Datum)?
welches Amt wurde durch einen (MOPS Fragen einbringen) besetzt?
Welche Ämter wurden durch Abgeordnete der (Partei) besetzt?
Welche Ministerposten wurden durch Abgeordnete der (Partei) ausgefüllt?
Welche Ämter in der (Nr.)Wahlperiode wurden von welchem Abgeordneten ausgeführt?
Welches Amt wurde in Wahlperiode(Nr.) durch einen (MOPS Fragen)?

Person: Welche/r Abgeordneter hat welches Amt in der Wahlperiode (Nr.) ?
Welche/er Abgeordnete/er besetzt das Amt (Name Amt)?
Welcher Abgeordnete/er besitzt in der Wahlperiode (Nr.) ein Amt/Ausschuss mit der Bezeichnung(Bez.)?
Welcher Abgeordnete/er ist (MOPS fragen, verheiratet usw.) + Ämter/Ausschuss/Protokoll..?
Welcher abgeordneter besitzt ein Amt und ist ledig?
Welche Abgeordneten sind in der Wahlperiode(Nr.) Mitglied der Partei(Name)?
Welcher Abgeordneter war/ist vorsitzender eines Ausschusses?
Welcher Abgeordnete/er ist Mitglied der (Parteiname)?
Welcher Abgeordnete/er verfasste die Drucksache(Protokoll/Rede) mit der (Nr., ID, am Datum, in WP, mit Titel)?
Liefere/Suche alle Mitglieder des Ausschusse (Name) aus WP(Nr.)!
Liefere/Suche alle Mitglieder des (Name)Ausschusses!
Liefere/Suche alle Abgeordneten der Partei (Name) die in einem Ausschuss vertreten sind!
Liefere/Suche alle Abgeordneten der Partei (Name) die in der Wahlperiode(Nr.) in einem Ausschuss vertreten sind!

Protokoll: Liefere/Suche das Protokoll/Anträge/Reden mit der Nummer (Nr.)!
Welches/e Protokoll/Anträge/Reden wurde am (Datum)verfasst vom Autor(Name)?
Liefere/Suche alle Protokolle/Anträge/Reden von WP(NR) aus vom Autor(Name)!
Liefere/Suche alle Protokolle/Anträge/Reden von (Personname)!

Liefere/Suche alle Protokolle/Anträge/Reden aus die von Minister (Personname) verfasst wurden?
Liefere/Suche alle Protokolle/Anträge/Reden die in Wahlperiode(Nr.) durch Abgeordnete der (Parteiname) verfasst wurden!
Liefere/Suche alle Protokolle/Anträge/Reden die von Ausschlussmitgliedern/Vorsitzenden verfasst wurden!
Liefere/Suche alle Protokolle/Anträge/Reden die von Ausschlussmitgliedern/Vorsitzenden in der Wahlperiode (Nr.) verfasst wurden!
Liefere/Suche alle Protokolle/Anträge/Reden von Ministern!
Liefere/Suche alle Protokolle/Anträge/Reden von (MOPS anfrage)
Liefere/Suche alle Protokolle/Anträge/Reden von Ministern von der Partei(Name) die (in Wahlperiode(Nr.))ein Amt bezogen haben/hatten!
Liefere/Suche alle Protokolle/Anträge/Reden von Ministern die ein Amt bezogen haben/hatten!

Drucksache: Liefere/Suche alle Drucksachen vom Typ (Name)!
Liefere/Suche die Drucksache mit der Nummer (Nr.) aus!
Liefere/Suche die Drucksache die am (Datum) verfasst wurde!
Liefere/Suche alle Drucksachen aus Wahlperiode(Nr.)!
Liefere/Suche alle Drucksachen mit dem Schlagwort (Thema) aus!
Liefere/Suche alle Drucksachen in der (Personname) vorkommt!
Liefere/Suche alle Drucksachen die von (Personname)verfasst wurde!
Liefere/Suche alle Drucksachen die von (Personname)in Wahlperiode(Nr.)verfasst wurden und die in einem Ausschuss/Amt (Name) tätig sind/waren!
Liefere/Suche alle Drucksachen die von Abgeordneten (Name), welche in der WP(Name) tätig waren/sind!
Liefere/Suche alle Drucksachen die vom Abgeordneten mit der ID (Nr.) verfasst wurden!
Liefere/Suche alle Drucksachen die von (Personname)+(Mops Anfragen) verfasst wurden!

Partei: Welcher Partei gehört (Name) an?
In welcher Partei sind die meisten Abgeordneten verheiratet/ledig?
In welcher Partei sind die meisten Doktoren?
Zu welcher Partei gehört der Abgeordnete mit den meisten Kindern?
Welche Parteiabgeordneten kandidierten in Wahlkreis(Nr.)?
Welcher Partei gehört der Ausschussvorsitzende (Personname)von (Amtname) an?
Welche Mitglieder der (Parteiname) haben ein Ministerposten in der Wahlperiode (Nr.)?
Welche Mitglieder der (Parteiname) haben ein Ministerposten?
Welche Mitglieder der (Parteiname) haben ein Ministerposten der am (Datum) begann oder endete?

Wahlperiode(Wp): In welcher Wp wurde das Amt des Bundeskanzlers durch Schröder besetzt?
Welche Wp begann/ endete am (Datum)?

Welche Abgeordneten waren in der Wp(Nr.) im Bundestag vertreten?
In welcher Wp war (Person) Mitglied/Vorsitzender/stellv. eines Ausschusses?
Liefere/Suche aus der Wp (Nr.) alle Mitglieder des Ausschusses(Name)!
In welcher Wp(NR) wurde der Ministerposten(Name) von Person(Name) besetzt?
Liefere/Suche alle in der Wp(Nr.) verfassten Protokolle/Anträge/Reden!
Liefere/Suche alle in der Wp(Nr.) verfassten Protokolle/Anträge/Reden vom Autor!
Liefere/Suche alle in der Wp(Nr.) vorkommenden Ausschüsse/Ämter
Liefere/Suche aus Wp(Nr.) den Ausschuss Vorsitzenden/ stellv.Vorsitzenden!
Liefere/Suche aus Wp(Nr.) alle Drucksachen/Protokolle (vom Autor(Name))!
Suche aus WP(Nr.) das Amt von Person (Name)!

Ausschluss: In welchem Ausschuss sind Abgeordnete der (Parteiname)?
In welchem Ausschuss aus der Wp(Nr.) sind Abgeordnete der (Parteiname)?
Wer ist Ausschluss Vorsitzender/stellv.Vorsitzender für den Ausschuss (Name)?
Wer ist Ausschluss Vorsitzender/stellv. Vorsitzender für den Ausschuss (Name) in der Wp(Nr.)?
Liefere/Suche alle Ausschlüsse der Wp(Nr.)!
In welchem Ausschuss heißt der stellv. Vorsitzende (Personname)?
Liefere/suche Ausschussmitglieder der Wp(Nr.) die zur (Partei)!
Liefere/Suche Ausschussmitglieder der Wp(Nr.) die Mops!

Dokument (Dok): Gib/Liefere das Dok mit der Nummer(Nr.)!
Gib/Liefere das Dok was/welche am (Datum) verfasst wurde!
Gib/liefere das Dokument mit dem Thema (Themaname)!
Suche alle Dokumente der/des Abgeordneten (Name)!

Das Enity-Relation Schema:

5.6 Dossier

5.6.1 Aufgabe

Ein Dossier, französisch für Aktendeckel, ist eine Sammlung von Dokumenten zu einem bestimmten Thema und bildet einen substantiellen Teil der Projektaufgabe. Da in den Datenbanken des Bundestages die gemeinsame Darstellung von Daten zu einem Abgeordneten, inklusive einer Auflistung aller seiner Reden und Dokumente nicht vorhanden ist, hat die Projektgruppe beschlossen diese aus den extrahierten Daten zu erstellen.

Zu diesem Zweck wurde eine Anwendung erstellt die entweder „standalone“, oder als Komponente der Fragebeantwortung eingesetzt werden kann. Die Anwendung stellt in einer Listenansicht alle Abgeordneten des Bundestages dar. Die Abgeordneten Elemente werden im relevanten Zeitraum, hier Wahlperioden 14 bis 16, angezeigt. In der Listenansicht kann der Benutzer einen Abgeordneten auswählen um sein Dossier einzusehen. Das Dossier wird on-demand erstellt, es enthält alle extrahierten Informationen zu dem Abgeordneten in der jeweiligen Wahlperiode. Zusätzlich wird eine Liste aller Dokumente und aller Reden dargestellt, die in

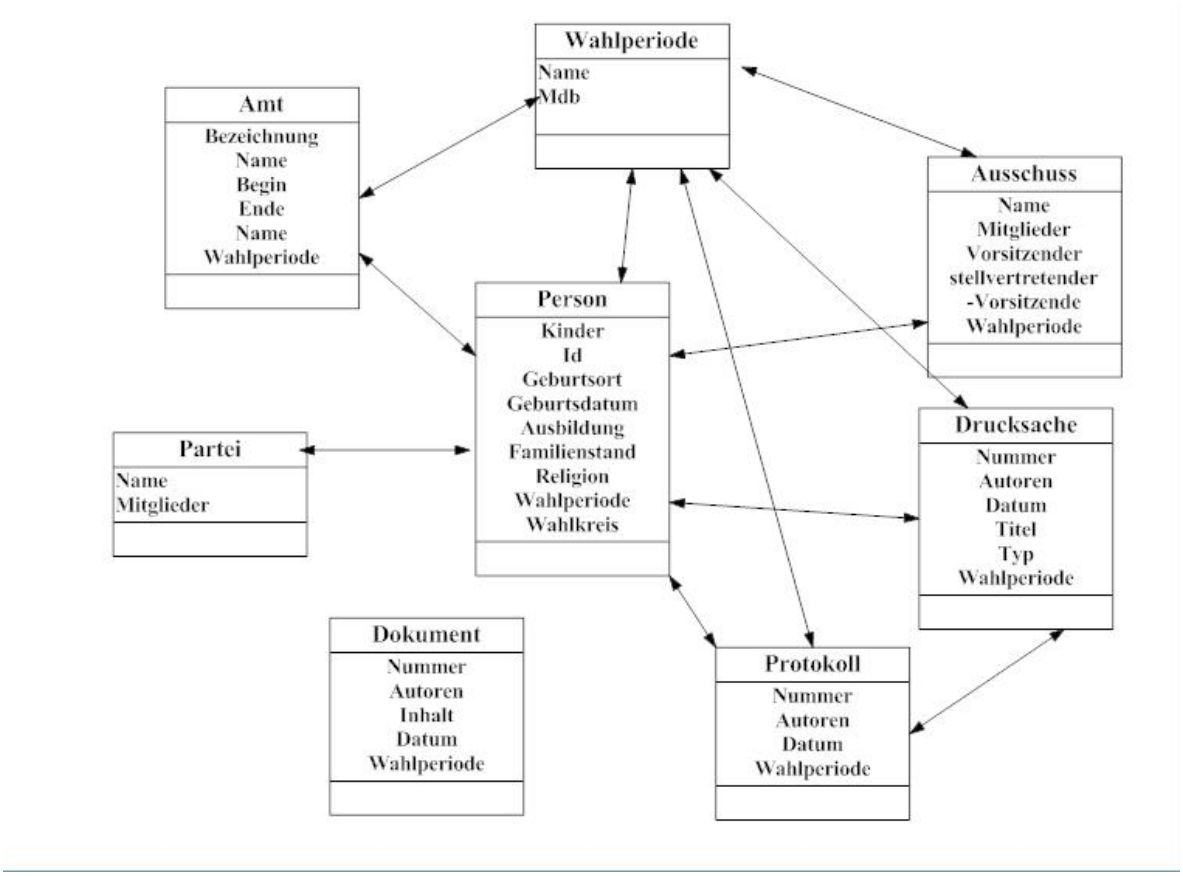


Abbildung 5.15: Vorschau des Schemas

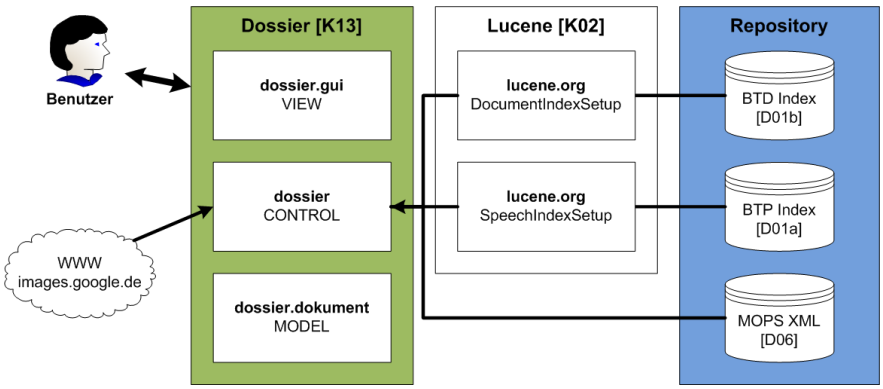


Abbildung 5.16: Dossier im Überblick

der Wahlperiode von dem Abgeordneten verfasst oder gehalten wurden. Für die Präsentation auf dem Campusfest der TU Dortmund wurde ein weiteres Feature eingebaut, nämlich die Query an die Google Bildersuche, die ein zufälliges Bild des/der Abgeordneten herunterlädt und zeigt.

5.6.2 Realisierung

Die Software wurde in der Programmiersprache Java geschrieben. Die aus der Softwaretechnik bekannte MVC-Architektur wurde durchgehend realisiert und gliedert dementsprechend die Software in drei Schichten. Die Model Schicht bildet alle vorhandenen Entities ab und ist nur für die Datenhaltung zuständig. In der Control Schicht befindet sich die gesamte Ausführungslogik. Diese Schicht ist für die Erzeugung und Verwaltung von Entitäten zuständig. Die GUI ist nur für die Darstellung verantwortlich und beinhaltet keine Ausführungslogik. Die GUI wurde im Hinblick auf die Integration in dem Gesamtsystem als Container implementiert, damit sie ohne weiteres in eine bereits vorhandene GUI des Fragebeantwortung Systems [P02] integriert werden kann.

Die Abbildung 5.16 zeigt die mehrschichtige MVC Architektur der Komponente und die Interaktion der Control Schicht mit Lucene [K02] und der MOPS XML [D06] Datenbank.

Die Komponente benutzt Lucene um auf die Indexdatenbanken der Dokumente und Reden zuzugreifen. Wird vom Benutzer ein Dossier eines Abgeordneten angefordert, so wird zuerst der Abgeordnete mit allen seinen Daten aus MOPS XML instanziert. Danach wird jeweils die passende Instanz von Lucene angesprochen und ihre Rückgabe, also eine Liste der Dokumente oder Reden als Links dargestellt. Zuletzt wird die Google Bilddatenbank nach einem Bild des Abgeordneten durchsucht.

5.6.3 Packages

Das Dossier besteht aus einem zentralen Package : *de.unidortmund.pg520.dossier* in dem die Klassen der Control Schicht und drei weitere Packages untergebracht sind:

- *dossier.gui* - in dem sich alle Klassen der GUI Schicht befinden,

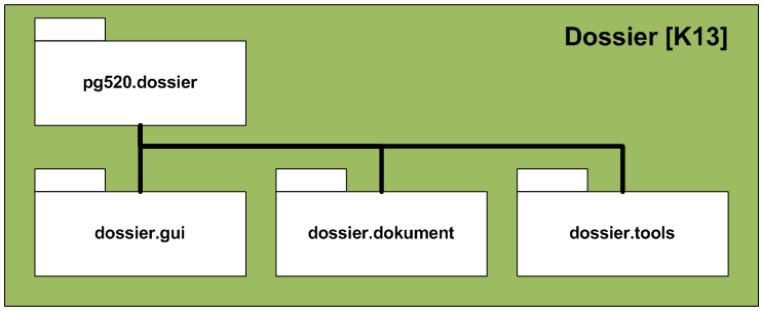


Abbildung 5.17: Java Packages der Komponente Dossier

- *dossier.dokument* - mit allen Entity Klassen, also der Model Schicht,
- und schließlich *dossier.tools* mit sogenannten Helfer Klassen.

Die GUI Schicht ist entsprechend dem Java GUI Container-Component Prinzip aufgebaut. Sie bildet mit Containern die Control Klassen ab und ist für die Darstellung ihrer Ausgabe verantwortlich. Zusätzlich werden alle Entities als GUI Components ausgegeben und referenziert. Die Zuordnung zwischen den GUI und den Control und Entity Klassen ist folgender weise zu verstehen: eine Control Klasse wird durch einen GUI Container dargestellt also in der Regel durch ein JPanel. Eine Entity wird meistens durch ein JLabel dargestellt. Ein Abgeordneten Objekt ist eine Entity, enthält aber Referenzen auf die Entities der Reden und Dokumente, sowie das geladene und skalierte Google Bild. Mit dieser einfachen Hierarchie kann die GUI in einem Aufruf top-down dargestellt werden.

Das in der Abbildung 5.17 abgebildete *dossier.dokument* Package enthält alle Entity Klassen der Komponente. Wir unterscheiden zwischen zwei Arten von Entities: einerseits Elemente der Abgeordnetenliste, hier Instanzen der Klasse DossierAbgeordneter. Andererseits ist das Dossier mit den Daten des einzelnen Abgeordneten und seine referenzierten Dokumente und Reden.

- dossier.dokument.DossierAbgeordneter
  Element der Abgeordneten Auswahlliste, besteht aus der MOPS ID, dem Namen und Vornamen des Abgeordneten.
- dossier.dokument.Dossier
  Das wahlperioden-abhängige Dossier eines Abgeordneten mit allen seinen Daten aus MOPS XML. Hält Referenzen auf alle seine Dokumente und Reden.
- dossier.dokument.DossierDokument
  Ein Objekt das ein Dokument abbildet, mit ID, Link und Titel.
- dossier.dokument.DossierRede
  Ein Objekt das eine Rede abbildet, mit ID, Link und Protokoll Nummer.

Das *dossier.tools* Package enthält Helfer Klassen die kleinere Funktionen erfüllen.

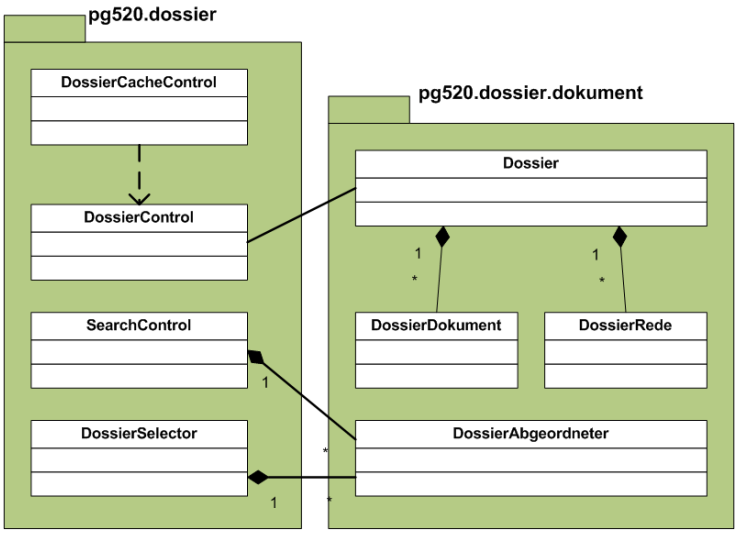


Abbildung 5.18: Control und Entity Klassen des Dossiers

- dossier.tools.GooglePicDownloader
  Diese Klasse ist für die Query der Google Bilddatenbank zuständig und lädt das erste verfügbare Bild des Abgeordneten runter. Zusätzlich wird das Bild passend skaliert, so dass es die Breite des Fensters nicht überschreitet.

- dossier.tools.JdomXMLReader
  Diese Klasse vereinfacht den Zugriff auf XML Dokumente in dem sie die Dokumente passend als Objekt zurückgibt. Diese ist für den Zugriff auf die MOPS XML Datenbank notwendig.

- dossier.tools.JdomXMLWriter
  Diese Klasse ist für das schreiben von XML Dateien zuständig.

Das *pg520.dossier* Package enthält die Control Klassen der Anwendung.

- dossier.DossierCacheControl
  Diese Klasse bildet transparent die Methoden der DossierControl Klasse ab und ist für die Verwaltung des Cache zuständig. Ein Aufruf aus der GUI wird zunächst in dieser Klasse verarbeitet bevor es eine Ebene tiefer, zu der eigentlichen Dossier Erstellung weitergeleitet wird. Diese Klasse prüft also ob im Dateisystem ein generiertes Dossier bereits vorhanden ist, sollte dies nicht der Fall sein, wird ein neues Dossier in der DossierControl Klasse erstellt. Diese Klasse ist noch nicht vollständig implementiert.

- dossier.DossierControl
  Die wichtigste Control Klasse der Komponente. Sie ist für die Erzeugung eines Dossiers zuständig und führt alle hierfür notwendigen Schritte aus. Über eine MOPS ID wird ein Abgeordneter in MOPS XML nachgeschlagen und die Listen seiner Dokumente und Reden aus der Ausgabe von Lucene erzeugt.

- dossier.SearchControl
  SearchControl implementiert die Funktionalität der Abgeordneten Suche. Sie stellt an

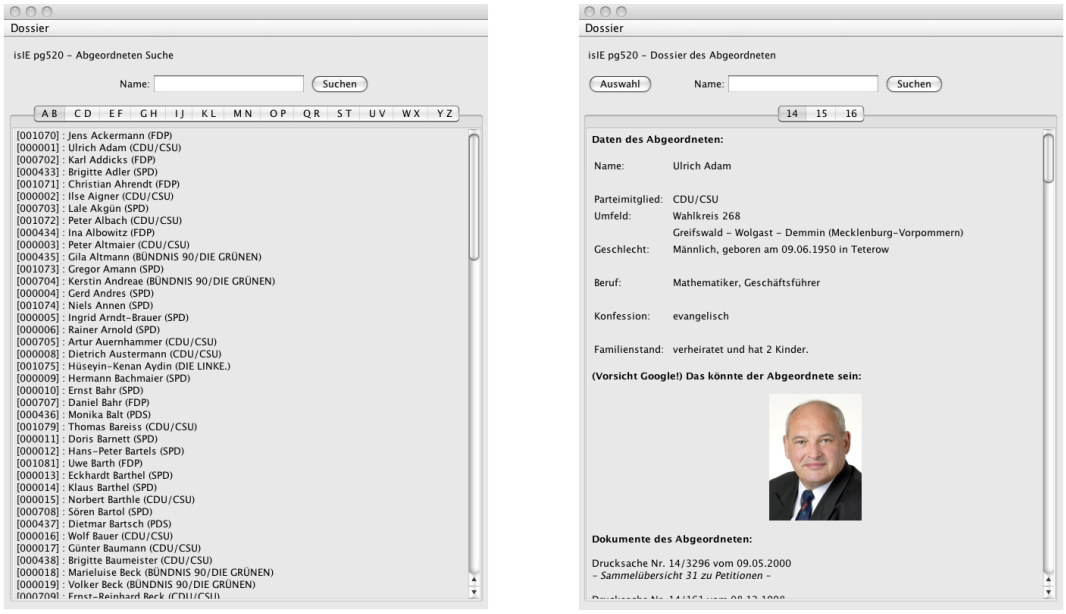


Abbildung 5.19: Bildschirmfoto der Abgeordnetenauswahl und eines Dossiers

Hand ein paar Stichwörter eine Liste der Abgeordneten dar, deren Vor- oder Nachname den Suchkriterien entsprechen.

- dossier.DossierSelector
  Eine Klasse mit der die Anwendung gestartet wird, sie erstellt eine, nach Anfangsbuchstaben geordnete Gesamtliste aller Abgeordneten. Aus dieser Liste kann der Anwender ein Dossier auswählen, welches dann von DossierCacheControl erstellt wird.

5.6.4 GUI Design und Anwendungsfälle

In dieser einfachen Anwendung werden alle Funktionen aus der GUI aufgerufen. Deshalb ist die GUI an dieser Stelle ganz eng mit den Anwendungsfällen verknüpft. Eine aktuelle Ansicht der GUI befindet sich in der Abbildung 5.19. Der Benutzer navigiert und ruft Funktionen mit der Maus auf. Das Suchformular erfordert selbstverständlich eine Tastatureingabe.

Die GUI Ansicht besteht aus zwei Teilen, in dem Oberen befindet sich die Suchfunktion und die Navigationsschaltflächen. Dieser Teil wird immer Dargestellt, es wird allerdings der Titel entsprechend dem Modus der Anwendung geändert und die Auswahl Schaltfläche ein- oder ausgeschaltet. Der untere Teil der Ansicht ist variabel und verändert seinen Inhalt abhängig zum Darstellungsmodus. Dem Benutzer stehen zwei Modi innerhalb der GUI zur Verfügung:

- Abgeordneten Suche
  wird von der *dossier.gui.SelectionListPanel* dargestellt.
  Zeigt eine nach Anfangsbuchstaben des Nachnamens gruppierte und sortierte Liste aller Abgeordneten. Jeder Eintrag besteht aus der aktuellen MOPS ID, dem Vornamen, Nachnamen und der Partei des Abgeordneten. Die Liste ist unabhängig von der Wahlperiode und wird aus dem gesamten Datensatz erstellt.

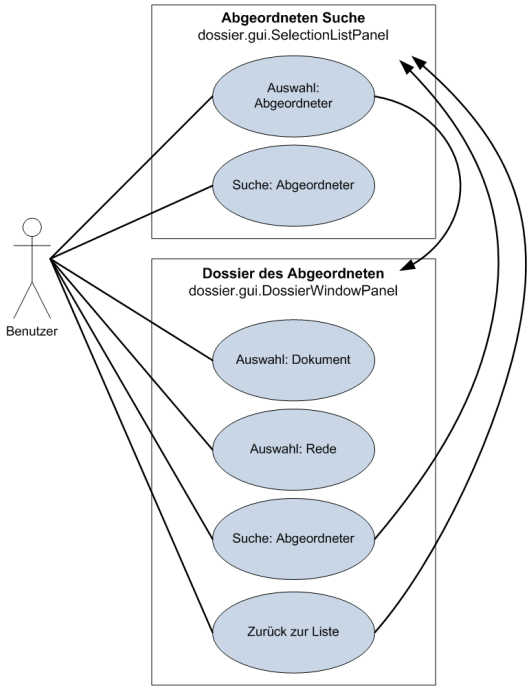


Abbildung 5.20: Anwendungsfall Diagramm

- Dossier des Abgeordneten
  wird von der *dossier.gui.DossierWindowPanel* dargestellt.
  Stellt einen Abgeordneten mit allen in MOPS verfügbaren Informationen dar. Zusätzlich wird ein Bild des Abgeordneten angezeigt. Darauf folgen die Listen der von ihm Verfassen Dokumente und der von ihm gehaltenen Reden. Das Bildschirmfoto 5.19 zeigt ein exemplarisches Dossier des Abgeordneten Ulrich Adam aus der 14-ten Wahlperiode.

Die Anwendung schaltet abhängig vom Zustand in einen der Modi, welche in dem Anwendungsfall Diagramm, Abbildung 5.20, gezeigt werden. Das Diagramm zeigt die Interaktion des Benutzers mit der GUI und die Funktionen, die in dem jeweiligen Modus auswählt werden können. Folgende Optionen stehen dem Benutzer in der Abgeordneten Suche zur Verfügung:

- Auswahl: Abgeordneter
  Diese Funktion schaltet in den Dossier des Abgeordneten Modus und stellt die Informationen des Abgeordneten dar.

- Suche: Abgeordneter
  Hiermit werden die Abgeordneten Elemente der hier dargestellten Liste auf die vom Benutzer in das Formularfeld eingegebenen Namen eingeschränkt. Die Suche funktioniert auch auf einzelnen Buchstaben oder Teilen der Vor oder Nachnamen. Diese Funktion wird durch das Klicken auf den Suchen Button gestartet.

Der zweite Modus der Darstellung ist das Dossier des Abgeordneten. Es wird in diesen Modus umgeschaltet, wenn ein Abgeordneter aus der Liste ausgewählt wird. In diesem Panel hat der Benutzer folgende Funktionen zur Auswahl:

- Auswahl: Dokument
  Diese Funktion öffnet das PDF Dokument der ausgewählten Drucksache in einem vom Betriebsystem ausgewählten Darstellungstool. Diese Funktion ist abhängig vom Betriebsystem des Host-Systems. Zur zeit funktioniert sie nur in Windows Umgebungen.

- Auswahl: Rede
  Diese Funktion öffnet das XML Dokument der ausgewählten Rede in einem vom Betriebsystem ausgewählten Darstellungstool. Diese Funktion ist abhängig vom Betriebsystem des Host-Systems. Zur zeit funktioniert sie nur in Windows Umgebungen. Diese Funktionalität ist nur testweise Eingebaut worden, da ein in Java geschriebenes Darstellungstool für diese annotierten Reden, geplant aber noch nicht entwickelt worden ist.

- Suche: Abgeordneter
  Diese Funktion schaltet die Anwendung in den Modus der Abgeordneten Suche. Es wird eine neue, an Hand der Formularfeld Benutzereingabe generierte, Liste der Abgeordneten Elemente erstellt. Diese Funktion wird durch das Klicken auf den Suchen Button gestartet.

- Zurück zur Liste
  Die Anwendung wird in den Modus der Abgeordneten Suche geschaltet, allerdings wird wieder die gesamte Liste der Abgeordneten dargestellt. Diese Funktion wird durch das Klicken auf den Auswahl Button gestartet.

5.6.5 Anmerkungen zur Performanz und Anekdoten

Die Geschwindigkeit mit der die Anwendung Benutzer Anfragen beantwortet, hängt in erster Linie von drei Faktoren ab. Lucene ist in den Antworten sehr stabil, es gab keinen merkbaren Unterschied zwischen der Rechenzeit einer 30-stelligen oder einer 5-stelligen Dokumentenliste. Das Antwortverhalten von Lucene beeinflusst zwar die Performance der Anwendung, spielt jedoch eine untergeordnete Rolle. Die Größe der MOPS XML Datenbank ist hingegen für den Zugriff auf die Abgeordneten entscheidend. Da es sich um eine XML Datei handelt, muss sie vollständig in den Speicher geladen und durchsucht werden. Der zweite Faktor ist auf den ersten Blick recht unscheinbar, spielt aber eine sehr wichtige Rolle. Die Google Bildersuche war nicht nur schwierig zu implementieren, sie liefert äußerst unterschiedliche Antwortzeiten. Eine Anfrage bei Google liefert bekannter maßen einen Link zurück. Das dahinter liegende Bild muss zuerst heruntergeladen und dann skaliert werden. Da dieses Bild eine zufällige Größe haben kann, ist die Zeit, welche die Anwendung dafür benötigt sehr schwer einzuschätzen. In den Testläufen dauerte eine Anfrage im durchschnitt 5 Sekunden. Die längste Zeit, 75 Sekunden, brauchte die Anwendung für ein Bild der Abgeordneten Petra Pau: es gibt tatsächlich Menschen die 5 Megapixel Bilder im Internet publizieren.

Wie man aus dem Klassendiagramm der Control Schicht in der Abbildung 5.18 entnehmen Kann wurde ein Cache-Mechanismus geplant der die bereits erstellten Dossiers und Bilder als XML Dateien ablegt. Bei Bedarf wird in der DossierCacheControl die Ausführung unterbrochen und dieses XML geladen und dargestellt. Dieser Mechanismus wurde leider nicht vollständig implementiert - es fehlte die Zeit diese Funktionalität zu programmieren.

Die Helfer Klasse, welche Google parst war eine Herausforderung in sich. Erstens musste Google vorgegaukelt werden, dass nicht eine Java Anwendung, sondern ein echter Internetbrowser die Anfrage stellt. Zweitens bestehen Webseiten von Google nur aus JavaSkript Funktionen die erst auf dem Browser gerendert werden. Damit mussten die Verweise auf die Bilder aus den Parametern der Funktionen mühsam herausgeschnitten werden. Die Google Bildersuche ist im Großen und Ganzen nur mit viel Aufwand und sehr vielen regulären Ausdrücken parsbar. Ironischer weise verlangt Google von allen anderen Internet Seiten, gerade den Einsatz solcher Verschleierungspraktiken zu unterlassen.

5.7 PartyNator

Der PartyNator ist eine Java Komponente, die speziell für das Campusfest entwickelt wurde, und dient zur Unterhaltung der Besucher. Sie bestimmt die Parteizugehörigkeit des Benutzers anhand von Attributen, die ausgewählt und ausgefüllt werden können. Zur Auswahl der Attribute stehen folgende zur Verfügung:

- Ausbildung
- Geburtsort
- Geburtsjahr
- Familie
- Kinder

Der PartyNator arbeitet eng mit den Datensätzen der Abgeordneten Information, dem MOPS, zusammen. Es werden Personen Objekte erstellt, die bestimmte Attribute haben. Zum einen ist es die User Person mit den Attributen, die der Benutzer eingibt und zum anderen die Personen aus dem MOPS, deren Attribute aus der MOPS-XML geholt wird.
Je nachdem, welche Attribute ausgewählt werden, werden diese mit den Feldern der Einträge im MOPS verglichen. Je nach Anzahl der Attribute wird eine spezielle Anfrage an MOPS gestellt, die eine Liste von Namen der Abgeordneten liefert. Diese Namen erfüllen stets die ausgewählten und konjugierten Attribute. Für alle Attribute wurden similarity-Werte festgelegt und für die Attribute "Geburtsjahr", "Kinder", "Beruf"wurde eine Ähnlichkeitsfunktion definiert. Nachdem die Attribute zwischen der User Person und der Attribute der MOPS-Person verglichen wurden, werden den MOPS-Personen globale similarity Werte zugeordnet. Die Personen liegen mit den similarity Werten in einem 1-dimensionalen Raum.
Durch das parametrisierbare Ähnlichkeitsmaß a, bestimmt man die Clustergröße zwischen der User Person und der MOPS-Person. Die Clustergröße korrespondiert zu der euklidischen Distanz zwischen zwei Personen. Wenn man das Ähnlichkeitsmaß a beispielsweise auf 1 setzt(enstpricht 100 % Ähnlichkeit) setzt, ist die Clustergröße auf Minimum. Es werden nur die Personen betrachtet, die eine euklidische Distanz von 0 besitzen zur User Person. Setzt man das Ähnlichkeitsmaß niedriger, also beispielsweise auf 70%-80% Ähnlichkeit, werden viel mehr Personen in das Cluster aufgenommen. Das Cluster wächst somit in seiner Größe, und die euklidische Distanz ist größer als 0.
Für das Attribut "Beruf"wurde zusätzlich eine XML-Datei erstellt, welches die Berufe in einer Hierarchie abbildet. Diese Baumstruktur ist die Grundlage für die Ähnlichkeit von Berufen.

Für zwei Berufe werden alle Knoten des jeweiligen Berufspfades in eine Liste überführt. Zum Schluß werden die similarity Werte bestimmt, indem die Listenelemente miteinander verglichen werden.
Weiterhin wird ein Parteizähler hochgezählt um zu bestimmen, wieviele Personen jeweils zu welcher Partei angehören. Wichtig ist dabei auch, alle Abgeordneten des Bundestages zu bestimmen. Dies ist notwendig, um die Personenanzahl zu normalisieren. Durch die recht simple Formel wird dann ein Parteiaffinitätswert bestimmt:

Beispiel

$$p_i = \frac{z_i}{a_i}$$

p_i ist dabei der Wert für die Partei i
z_i ist die Anzahl der Personen der jeweiligen Partei i auf die die Attribute zutreffen
a_i sind alle Personen die der Partei i angehören
Die Normalisierung führt dazu, dass Parteien, die im Bundestag mit einer relativ niedrigen Personenanzahl vertreten sind, auch grundsätzlich stärker bevorzugt werden. Die Gewichtung ist pro Person hier ausschlaggebend. Als Ausgabe liefert der Partynator schließlich die Affinitäten in Prozent. Diese werden weiter oben im System visualisiert.

5.8 Datensätze

5.8.1 Aufbau

Mit dem *RMDatensatzBuilder* wird ein in der Hauptgui eingebettetes Programm bezeichnet, das zwei Dateien für die Weiterverarbeitung mit dem RM erstellt. Diese zwei Dateien werden im folgenden *Datensatz* genannt. Ein Administrator ist in der Lage die benötigen Datensätze mit einem Mausklick zu erstellen. Der Vorgang besteht aus drei Komponenten, die weiter nähe erläutert werden. Zu beachten an dieser Stelle ist, dass die Laufzeit des Algorithmus je nach dem bis zu 20 Minuten dauern kann. Bei den Datensätzen ist es zwischen *Standard-* und *Personenattribute* zu unterscheiden. *Standardattribute* sind Attributen, die zwingend in den Datensatz kommen. *Personenattribute* sind Attributen aus MOPS z.B. Wahlkreis, die der Administrator durch die Auswahl dazufügen kann.

5.8.2 Grafische Oberfläche

Die erste Komponente ist die grafische Oberfläche, die die Mensch-Maschinen Interaktion erlaubt. Sie wird in die Hauptgui integriert, so dass der Administrator den gewünschten Datensatz auswählt, der er erstellen möchte. Folgende Datensätze ist der Administrator in der Lage zu erstellen:

- **Abgeordnete**
- **Minister**
- **Ausschus** mit den Standardattributen:
  - Mitglieder,
  - Name,

Abbildung 5.21: RMDatensatzBuilder

  - Wahlperiode,
  - Vorsitzender

- **Abstimmung** mit den Standardattributen:
  - Mitglieder,
  - Drucksache,
  - Wahlperiode,
  - Datum,
  - Name,
  - Ergebnis

- **Drucksache** mit den Standardattributen:
  - Autoren,
  - Wahlperiode,
  - Datum,
  - Titel,
  - Nummer,
  - Typ

- **Amt** mit den Standardattributen:
  - Begin,
  - Ende,
  - Wahlperiode,
  - Bezeichnung,

  - Typ
- **Protokoll** mit Standardattributen:
  - Autoren,
  - Datum,
  - Wahlperiode,
  - Nummer
- **Dokument** mit Standardattributen:
  - Autoren,
  - Datum,
  - Wahlperiode,
  - Nummer

Zu jedem Datensatz kann man auch wahlweise durch den entsprechenden Checkbox weitere Personenattribute (z.B. Geburtsort) bestimmen. Den Namen für die Datei gibt man auch ein. Mit dem Drücken auf den Button wird die Anfrage an die unteren beiden Komponenten weitergeleitet.

5.8.3 Anfrage

Die zweite Komponente stellt einen Parser da, der entsprechende XMLs ausließt und daraus Objekte der entsprechenden Klasse erzeugt. Weiterhin werden weitere Objekte je nach dem ausgewählten Datensatz erzeugt oder die Attributen zu den existierenden Objekten gesetzt. Z.B. wählt man der Administrator das Ausschuss als der Datensatz aus, so wird das Ausschuss.xml und das MOPs.xml gelesen und daraus Ausschuss- und Person-Objekte mit den entsprechenden Standardattributen erzeugt. Danach werden die Objekte verknüpft (hier an der Stelle PersonenId) und zu den Ausschuss-Objekten die Personenattribute aus Person-Objekten gesetzt. Die resultierende Objekte haben die gewünschte Eigenschaften, nämlich, sie sind die Instanzen der Klasse Ausschuss und haben alle Personenattributen, die der Administrator ausgewählt hat. Für die Datensätze Abgeordnete und Person sind keine Verknüpfungen mit den Person-Objekten nötig.

5.8.4 .dat und .aml Dateien

Nach dem von oberen zwei Komponenten alle Objekte vorbereitet sind, kommt die dritte Komponente ins Spiel, die aus den resultierenden Objekten zwei Dateien erstellt. Genauer, eine Liste von Objekten, z.B. Ausschuss-Objekten in eine .dat Datei schreibt und dazu eine beschreibende .aml Datei erstellt. Jedes Objekt mit seinen Attributen wird in eine neue Zeile der .dat Datei geschrieben. In den ersten n(Anzahl der Standardattributen) Spalten stehen die Werte von den Standardattributen. In den letzten Spalten wird, wenn der Administrator die Personenattribute ausgewählt hat, für jede Person und jede ihre Attribute, die beispielsweise in einem Ausschuss sind, eine Spalte investiert. Die Spalten werden wie folgt Personenname, Personenname-Wahlort, ... genannt. Die .aml Datei ist in der XML-Form geschrieben und beschreibt, wie die einzelnen Spalten in der .dat Datei auszusehen sind und welche Werte(nicht bei integer) sie annehmen können. In der Spalte Personenname kann nur true, wenn die

Person dabei ist oder false, wenn nicht, stehen. In der Spalte Personenname-Wahlort steht der tatsächliche Wert, wenn die Person dabei ist, oder null, wenn nicht. Weiter sind beispielhafte .dat und .aml Dateien dargestellt.

```
<?xml version="1.0" encoding="UTF-8" ?>
<attributeset default_source="Aussschuss.dat">
<attribute name="Wp" sourcecol="1" valuetype="integer">
</attribute>
<attribute name="Name" sourcecol="2" valuetype="nominal">
<value>Arbeit</value>
<value>Angelegenheiten</value>
...
<value>null</value>
</attribute>
<attribute name="Vorsitz." sourcecol="3" valuetype="nominal">
<value>Doriss Barnet</value>
<value>Werner Kuhn</value>
...
```

Tabelle 5.1: Allgemeine Form der .dat Datei

Standartattribute	Personenattributen (für jeden Mitglied, der je einen Antrag gestellt hat, eine Spalte)
...	...
...	...

Tabelle 5.2: Ausschnitt aus Ausschuss.dat

Wp	Name	Vorsitz.	Klaus Brand-ner	Klaus Brandner-Wahlort	...	Rainer Fornahl	Reiner Fornal-Wahlort	...
14	Arbeit	Doriss Barnet	true	Gütersloh	...	false	null	...
14	Angelegenheiten	Werner Kuhn	false	null	...	true	Leipzig I	...
...	...	...	...	..	...	...	...	...

5.9 Eigenentwicklungen

5.9.1 Event Cutter

Einleitung

Der Event Cutter ist eine Java Komponente, die als Eingabe ein Text Dokument x und ein Stichwort y erhält. Als Ausgabe bekommt man die Zeile i, in der das Stichwort y vorkommt und die Zeilen i-1, i-2, i+1, i+2.

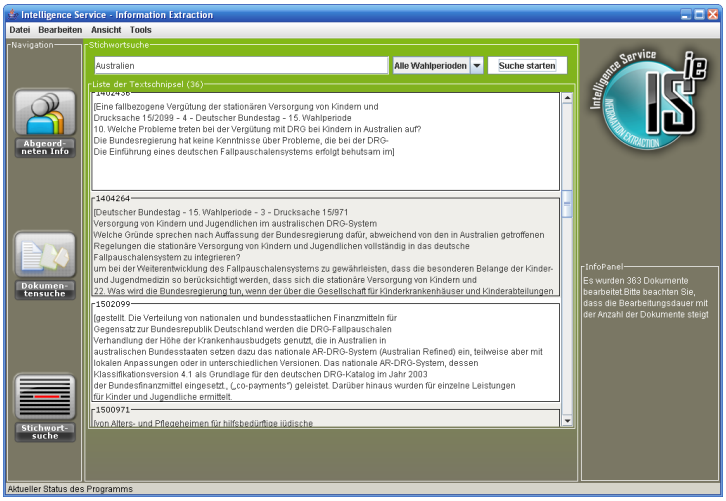


Abbildung 5.22: Ausschnitt des Event Cutters

Motivation

Da fast alle Text Dokumente des Deutschen Bundestags mehrere Seiten einnehmen, ist eine Suche nach einem bestimmten Suchbegriff meist zeitaufwändig, da der Nutzer wirklich Seite für Seite nach dem Suchbegriff suchen muss. Wie oben erwähnt erhält der Event Cutter ein Dokument bzw. Schritt für Schritt die zeilen eines Dokuments als Eingabe. Natürlich ist es sinnvoll dem Event Cutter nur die Dokumente zu übergeben, die auch das Stichwort enthalten. Dazu wird der LUCENE-Index verwendet, der dem Event Cutter n Dokumente liefert, die auch das Stichwort enthalten. Danach wird der Event Cutter n-mal aufgerufen und liefert genau n*i Schnipsel, wobei i die Anzahl der Schnipsel pro Dokument bezeichnet. Die Weiterverarbeitung der Schnipsel läuft über die grafische Benutzeroberfläche.

Aufbau der Komponente

Der Event Cutter arbeitet wahlweise mit den pdf Dateien und den ascii Dateien der Dokumente des Deutschen Bundestags. Zu Beginn werden die Dateien, welches das gesuchte Stichwort enthalten, mithilfe des Lucene-Index gesucht. Die Lucene-Anfrage gibt sowohl die Anzahl, als auch die Namen der relevanten ascii Dateien an, welche das gesuchte Stichwort enthalten. Mit dem Finden der Dateien, werden diese nun iterativ nach dem gesuchten Stichwort zeilenweise durchlaufen. Mit dem Vorkommen des Stichworts werden die ersten beiden und die letzten beiden Zeilen abgetrennt, so dass das Stichwort genau zwischen vier Zeilen liegt und einen Abschnitt bildet. Dieser Vorgang wird für jedes Wort, welches dem gefundenen Stichwort entspricht, durchlaufen. Die gefundenen Abschnitte können in einzelne ascii Dateien in einem separaten Ordner abgespeichert werden oder anderen Objekten als Strings übergeben werden.

Resultat

Der Event Cutter gibt als Endprodukt nur die Zeilen der Text Dokumente aus, die das gesuchte Stichwort enthalten. Somit ermöglicht der Event Cutter eine Kürzung von Text Dokumen-

ten in Abschnitte, die eine effiziente Suche nach dem Stichwort anbieten, ohne vollständige Dokumente durchzulesen.

6 Fragebeantwortung

6.1 Manuelle Fragebeantwortung

6.1.1 Einleitung

Die Webseite des Deutschen Bundestags bietet eine interne Suchmaschine an. Das Dokumentations- und Informationssystem für parlamentarische Vorgänge, kurz DIP. Das DIP ist das gemeinsame Informationssystem von Bundestag und Bundesrat. Wenn Nutzer der Bundestagswebseite an speziellen Ereignissen oder geschichtlichen Daten Interesse haben, können sie nach diversen Themen recherchieren.

Im DIP wird das parlamentarische Geschehen im Bundestag und Bundesrat dokumentiert und in Drucksachen und stenografischen Berichten festgehalten. Es verfügt über eine große Auswahl an Dokumenten und bietet den Nutzern eine umfangreiche Volltextsuche. Die Informationen zu einem Thema stehen meist verstreut in verschiedenen Dokumenten. Jedoch muss der Nutzer heutzutage nicht mehrere tausende Dokumente Seite für Seite durchlesen.
Man bekommt mithilfe des DIP's alle relevanten Dokumente geliefert, die auf eine vom Nutzer eingegebene Suchanfrage zutreffen. Spezielle Fragen in Satzform kann der Nutzer dem DIP nicht stellen. Es kann dafür natürlich keine Ergebnisse liefern, da die Suchanfrage in Satzform für das System viel zu komplex ist. Eine Frage in Satzform besteht aus mehreren Stichwörtern, so dass sich die Suchkategorien über mehrere Suchbegriffe erstrecken. Das DIP ist somit nicht in der Lage eine erfolgreiche Suche mit Treffern auszuführen. Deswegen bietet das DIP Hilfsoperationen, mit denen sich die Suchanfrage beschränken lässt. Ermöglicht wird dies durch 'u' für die UND-Verknüpfung und 'o' für eine ODER-Verküpfung von Stichworten. Verläuft die Suchanfrage erfolgreich, wird eine Trefferliste mit allen relevanten Dokumenten ausgegeben. Nun liegt es in der Hand des Benutzers alle Dokumente durchzugehen, das seiner Meinung nach am relevantesten erscheinende Dokument anzuwählen und im Dokument nachzulesen, ob er auf seine gestellte Suchanfrage eine Antwort erhält.

6.1.2 Konkretes Beispiel

Als Beispiel für einen solchen Vorgang, versuchen wir auf die folgende Frage eine Antwort zu finden:
„Wie ist die Haltung des deutschen Parlaments gegenüber dem EU-Beitritt der Türkei?"
Da diese Frage aus einem Satz mit mehreren Wörtern besteht, ist es hier nicht möglich, eine konkrete Auswahl an Dokumenten, als Ausgabe geliefert zu bekommen. Um diese Frage erfolgreich beantworten zu können, ist es somit sinnvoll, die Suchanfrage auf 2 Stichwörter zu beschränken.

Im oben genannten Beispiel sind die Schlagwörter der Frage „Türkei" und „EU". Der Nutzer erhält auf diese Suchanfrage eine Trefferliste mit relevanten Dokumenten. Auch hier muss der

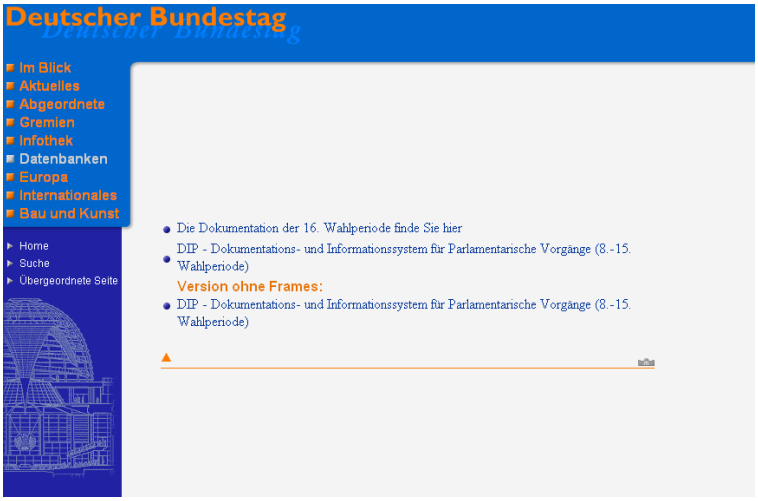


Abbildung 6.1: Startseite des DIP

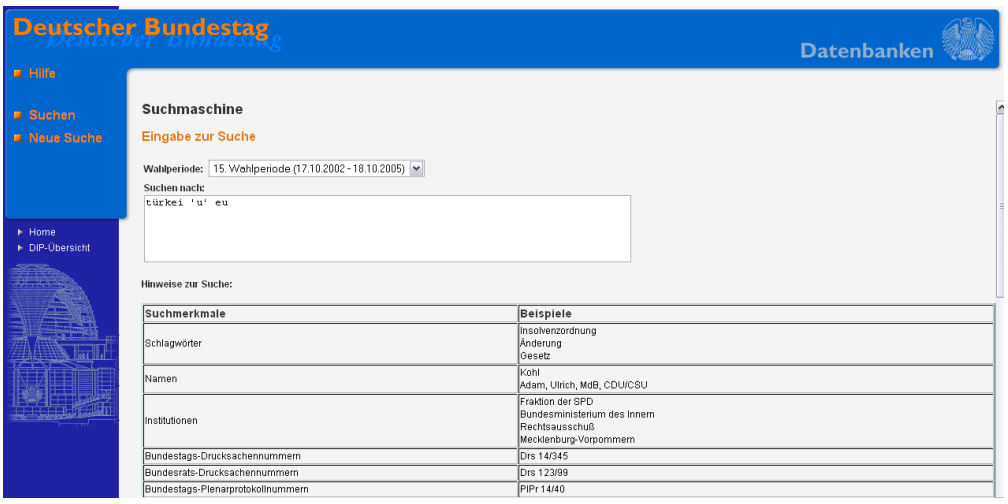


Abbildung 6.2: Sucheingabe im DIP

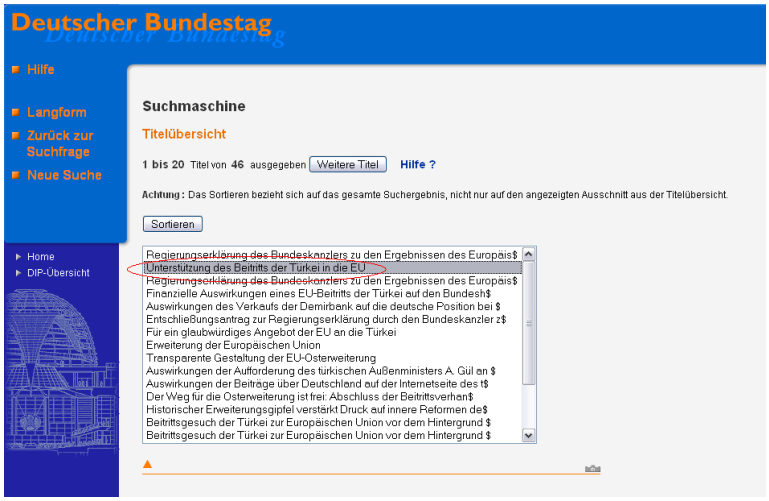


Abbildung 6.3: Titelübersicht

Abbildung 6.4: Übersicht eines Dokuments

Nutzer nicht jedes angezeigte Dokument durchgehen. Positiv ist auch die Anzeige der relevanten Dokumente. Man erhält als Rückmeldung eine kurze Inhaltszusammenfassung. Dies ermöglicht dem Leser gezielt unter den angezeigten relevanten Dokumenten zu suchen. Im Beispiel mit der Türkei und der EU gibt es eine breite Palette an parlamentarischen Sitzungen, die sich neben dem EU-Beitritt der Türkei auch mit der Jugendpolitik, Außenpolitik, Agrarpolitik und Wirtschaft beschäftigt haben. Ein Hinweis auf die Antwort der oben gestellten Frage könnte das hervorgehobene Dokument in Abbildung 6.3 liefern.
Der Titel des Dokuments enthält genau die Schlagwörter, die auch in der Frage auftauchen. Somit liegt es dem Nutzer nahe, dieses Dokument anzuwählen. Unter dem Menü Langform kann man sich das Dokument als Volltext im Pdf oder Ascii Format anzeigen lassen. Dies geschieht durch einen Mausklick auf den makierten Bereich in Abbildung 6.4.

Nach dem Öffnen des Dokuments kann man nun, mit Hilfe der Pdf-Suche (siehe Abbildung

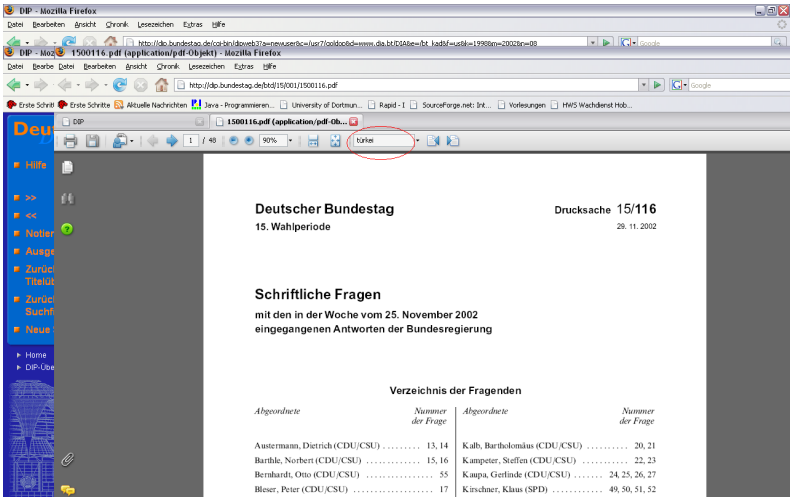


Abbildung 6.5: Stichwortsuche im Pdf Dokument

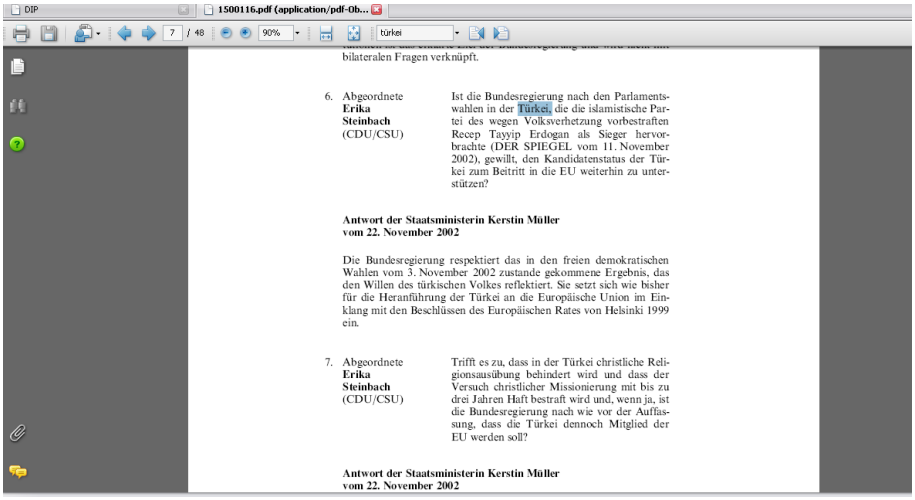
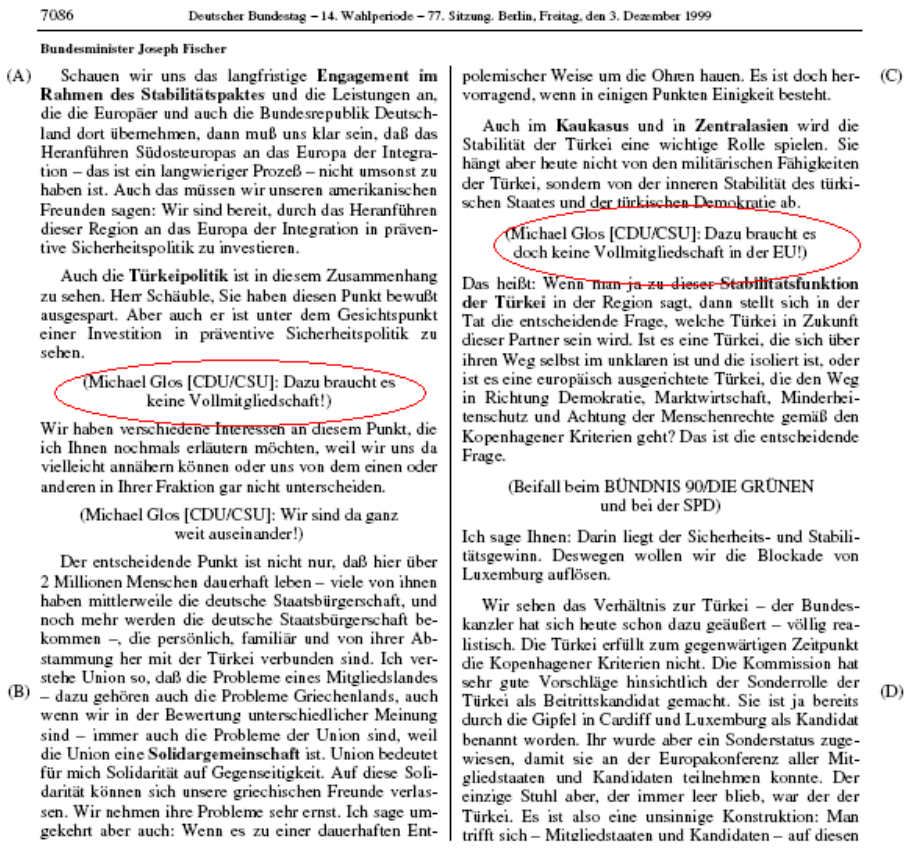


Abbildung 6.6: Ergebnis der Stichwortsuche im Pdf Dokument

6.5), nach dem Stichwort suchen. Mit der Eingabe des Stichwortes „Türkei“ liefert die Suche dem Nutzer, die Passagen in dem das Stichwort auftaucht (siehe Abblidung 6.6). Nun liest der Nutzer den gefundenen Abschnitt im Dokument, um seine Antwort auf die gestellte Frage zu erhalten.
Dieses Dokument hat allerdings keine wirkliche Antwort auf die Frage, trotz des zutreffenden Titels in der Trefferliste. Im Abschnitt treten zwar die Stichwörter auf, jedoch muss der Leser sich die Antwort selbst erarbeiten. Dabei muss man beim Lesen auch auf sprechende Personen und Parteien achten.
Wichtig bei der Suche nach der Antwort auf die Frage ist auch die Art des Dokumentes, in dem man sucht. In vielen Dokumenten wie Beschlussempfehlungen oder schriftlichen Fragen mit Antwort, die als Drucksache vorliegen, ist die Haltung der einzelnen Parteien nicht so deutlich wie in den Plenarprotokollen von stenografischen Berichten.
In diesen Berichten kommt jede Partei zu Wort und in vielen Dokumenten wurde über den



7086 Deutscher Bundestag – 14. Wahlperiode – 77. Sitzung. Berlin, Freitag, den 3. Dezember 1999

Bundesminister Joseph Fischer

(A) Schauen wir uns das langfristige **Engagement im Rahmen des Stabilitätspaktes** und die Leistungen an, die die Europäer und auch die Bundesrepublik Deutschland dort übernehmen, dann muß uns klar sein, daß das Heranführen Südosteuropas an das Europa der Integration – das ist ein langwieriger Prozeß – nicht umsonst zu haben ist. Auch das müssen wir unseren amerikanischen Freunden sagen: Wir sind bereit, durch das Heranführen dieser Region an das Europa der Integration in präventive Sicherheitspolitik zu investieren.

Auch die **Türkeipolitik** ist in diesem Zusammenhang zu sehen. Herr Schäuble, Sie haben diesen Punkt bewußt ausgespart. Aber auch er ist unter dem Gesichtspunkt einer Investition in präventive Sicherheitspolitik zu sehen.

(Michael Glos [CDU/CSU]: Dazu braucht es keine Vollmitgliedschaft!)

Wir haben verschiedene Interessen an diesem Punkt, die ich Ihnen nochmals erläutern möchten, weil wir uns da vielleicht annähern können oder uns von dem einen oder anderen in Ihrer Fraktion gar nicht unterscheiden.

(Michael Glos [CDU/CSU]: Wir sind da ganz weit auseinander!)

Der entscheidende Punkt ist nicht nur, daß hier über 2 Millionen Menschen dauerhaft leben – viele von ihnen haben mittlerweile die deutsche Staatsbürgerschaft, und noch mehr werden die deutsche Staatsbürgerschaft bekommen –, die persönlich, familiär und von ihrer Abstammung her mit der Türkei verbunden sind. Ich verstehe Union so, daß die Probleme eines Mitgliedslandes (B) – dazu gehören auch die Probleme Griechenlands, auch wenn wir in der Bewertung unterschiedlicher Meinung sind – immer auch die Probleme der Union sind, weil die Union eine **Solidargemeinschaft** ist. Union bedeutet für mich Solidarität auf Gegenseitigkeit. Auf diese Solidarität können sich unsere griechischen Freunde verlassen. Wir nehmen ihre Probleme sehr ernst. Ich sage umgekehrt aber auch: Wenn es zu einer dauerhaften Entpolemischer Weise um die Ohren hauen. Es ist doch hervorragend, wenn in einigen Punkten Einigkeit besteht. (C)

Auch im **Kaukasus** und in **Zentralasien** wird die Stabilität der Türkei eine wichtige Rolle spielen. Sie hängt aber heute nicht von den militärischen Fähigkeiten der Türkei, sondern von der inneren Stabilität des türkischen Staates und der türkischen Demokratie ab.

(Michael Glos [CDU/CSU]: Dazu braucht es doch keine Vollmitgliedschaft in der EU!)

Das heißt: Wenn man ja zu dieser Stabilitätsfunktion **der Türkei** in der Region sagt, dann stellt sich in der Tat die entscheidende Frage, welche Türkei in Zukunft dieser Partner sein wird. Ist es eine Türkei, die sich über ihren Weg selbst im unklaren ist und die isoliert ist, oder ist es eine europäisch ausgerichtete Türkei, die den Weg in Richtung Demokratie, Marktwirtschaft, Minderheitenschutz und Achtung der Menschenrechte gemäß den Kopenhagener Kriterien geht? Das ist die entscheidende Frage.

(Beifall beim BÜNDNIS 90/DIE GRÜNEN und bei der SPD)

Ich sage Ihnen: Darin liegt der Sicherheits- und Stabilitätsgewinn. Deswegen wollen wir die Blockade von Luxemburg auflösen.

Wir sehen das Verhältnis zur Türkei – der Bundeskanzler hat sich heute schon dazu geäußert – völlig realistisch. Die Türkei erfüllt zum gegenwärtigen Zeitpunkt die Kopenhagener Kriterien nicht. Die Kommission hat sehr gute Vorschläge hinsichtlich der Sonderrolle der Türkei als Beitrittskandidat gemacht. Sie ist ja bereits (D) durch die Gipfel in Cardiff und Luxemburg als Kandidat benannt worden. Ihr wurde aber ein Sonderstatus zugewiesen, damit sie an der Europakonferenz aller Mitgliedstaaten und Kandidaten teilnehmen konnte. Der einzige Stuhl aber, der immer leer blieb, war der der Türkei. Es ist also eine unsinnige Konstruktion: Man trifft sich – Mitgliedstaaten und Kandidaten – auf diesen

Abbildung 6.7: Triggerwörter in Plenarprotokollen

EU-Beitritt der Türkei ausführlich diskutiert. In den Plenarprotokollen treten die meisten Triggerwörter auf. Aufgrund von diversen Zurufen, wie Widersprüche, Beifall und Beschimpfungen, die mitten im Vortrag auftreten können, ist die Haltung der Parteien ersichtlich. Wie dies ein Beispiel in Abbildung 6.7 verdeutlicht.
Selbstverständlich ist dieses genannte Beispiel nur für ein relevantes Dokument beschrieben worden. Tatsächlich erstreckt sich die Suche nach der Antwort auf die Frage, wie die Haltung der deutschen Parteien gegenüber dem EU-Beitritt der Türkei ist, über mehrere hundert Dokumente. Dafür wurden 291 Dokumente gelesen und nach Triggerwörtern untersucht, darunter als Dokumententypen: Anträge, Beschlussempfehlungen und Berichte, schriftliche Fragen mit Antworten, Plenarprotokolle, kleine Anfragen mit Antwort.

6.2 Frage Eingabe

6.2.1 Freie Frageeingabe

Bei einer freien Frageeingabe des Benutzters, z.B. über ein Textfeld, müssen die Fragen analysiert werden. Unser erster Ansatz zielt darauf ab, die Fragewörter als Trigger für den Fragetyp,

und damit für das Beantwortungsverfahren zu benutzen:

Frage	Antwort	Beispiel
wann	am[Datum]	wann war letzte Misstrauensvotum?
wieviele	[Anzahl]	wieviele Kinder hat...?
wieviel	[Anzahl]	wieviel Kinder hat der Abgeordnete?
wie	[String]	Wie ist die Reaktion...?/ Wie oft Zwischenrufe bei Rede von ...
was	am[Datum],[String]	Was hat Schröder im Bundestag am...gesagt?
wer	am[Datum],[String]	wer ist der Parteivorsitzender der ..?
welche	[String]	Welche Parteien sind im Bundestag?
welcher	[String]	Welcher Partei gehört .. an?
werden	[String]	werden Anträge von ..abgelehnt?
hat	[Anzahl]	hat der Abgeordnete .. Kinder?
wo	[String]	wo wird die BW zurzeit eingesetzt?
wieso	[String]	[Event Extraction] wieso ist Deutschland gegen Türkei Beitritt
warum	[String]	[Event Extraction] warum BDR gegen TR Beitritt
weshalb	[String]	[Event Extraction] weshalb...

Dieser Ansatz wurde allerdings verworfen, da es dem Nutzer zu viele Möglichkeiten bei der Fragestellung gab, wodurch die Analyse und Auswertung sehr komplex wird. Auch die Möglichkeit dem Benutzer vorgefertige Fragen zur Auswahl zu geben ist wenig sinnvoll, da entweder nur sehr wenige Fragen zur Verfügung stehen, für die man dann auch kein aufwendiges System benötigt, sondern die nötigen Antworten einmal bestimmt und dann abspeichert. Oder man hat so viele verschiedene Fragen, dass der Benutzer so viel Zeit in das Suchen der Richtigen Frage investieren muss, dass es für ihn ähnlich effizient wäre die Frage selbst manuell zu beantworten. Es muss also ein Weg gefunden werden um dem benutzer sowohl genügend Freiraum zu lassen, als ihm auch eine strukturierte Form der Fragestellung vorzugeben.

BNF von Alex(ist noch auf seinem defekten Laptop)

6.2.2 Strukturierte Frageeingabe

Sinnvoll erscheint hier also der Mittelweg, dem Benutzer die Fragestellung in bestimmten Grenzen frei zu überlassen. Dazu ist es sinnvoll zunächst zu erfassen welche Datentypen als Datenbasis dienen und wie die einzelnen Teile dieser Daten zusammenhängen. Solche Datentypen können zum Beispiel sein: Person, Protokoll, Drucksache, Partei. Im folgenden werden diese als Entities bezeichnet. Um stukturierte Fragestellung zu erlauben müssen nun diese Entities in Ihren Attributen vollständig beschrieben werden, sowie die Relationen zwischen den einzelnen Entities erfasst werden. Analog zu relationalen Datenbanken wird für eine Relation zwischen zwei Entities immer ein Fremdschlüssel benötigt, also ein Attribut, bei dessen Gleichheit in zwei Entity Instanzen angenommen werden kann, dass diese Instanzen miteinander in Relation stehen. Ist dieses Schema einmal erfasst kann eine Oberfläche eine Semi-Freie Frageerstellung unterstützen, indem sie nach Auswahl einer Zielentität für die Frage, die aktuell abfragbaren Attribute und die Relationen zu anderen Entitäten anzeigt. Wird ein Attribut gewählt, kann man eine Bedingung an dieses Attribut stellen, die im folgenden Constraint genannt wird. Wählt man wiederrum eine Relation aus, springt man zur Zielentität dieser

Relation und kann von dort aus wieder wählen, ob man eine Bedingung an ein Attribut dieser Entität stellen möchte, oder eine Relation dieser Entität wählen. So ergibt sich eine Frage, die aus relationalen Verknüpfungen mehrerer Entitäten besteht und gleichzeitig Bedingungen an die Entitäten stellt.
Dieses Vorgehen erschien als praktikabel und passte gut zu der sehr heterogenen Datenlandschaft, da es Funktionen ähnlich zur Anfragesprache SQL erlaubt, aber nicht an ein Datenbank-System gebunden ist.

6.3 Warum Fragen

Vergleichsdatensatz muss vorhanden sein RapidMiner wird mit passendem Experiment gestartet Items des Ergebnissets werden positive Labels im Datensatz Entscheidungsbaum und Regellerner Anzeige der Ergebnisse

6.3.1 Die Basis

Die Basis der Warum-Fragen besteht aus zwei Teilen. Zum einen aus einer vom Benutzer über die QueryFacade eingegebene Frage, bzw deren Ergebnisset, und zum Anderen aus einem Vergelichsdatensatz, der wie im Kapitel Datensätze beschrieben erstellt wurde.

6.3.2 Die Bedingung

Damit eine Warum-Frage bearbeitet werden kann, muss das Ergebisset der Frage aus einem bestimmten Typ von Einträgen bestehen, der einem der Vegleichsdatensätze entspricht, da sonst keine Berechnung möglich ist. Diese Typen sind z.B. nur Personen der 16 WP oder nur Anträge, oder nur Drucksachen....

6.3.3 Der Ablauf

Der Benutzer giebt eine Frage ein und lässt sich das Ergebnisset anzeigen, wenn die Typen der Ergebnissets mit einem der Datensätze übereinstimmt, wird der „Warum“-Button eingeblendet, so das der Benutzer die Möglichkeit bekommt die Berechnung zu starten.
Durch das drücken des Buttons wird zuerst eine Daten geschrieben, in der die IDs der Einträge des Ergebnissets gespeichert werden.
Als nächstes wird RapidMiner gestartet.
Die Experimente sehen wie folgt aus:

- Der entsprechende Vergleichsdatensatz wird geladen
- Es wird ein Binäres Label geneiert indem die IDs der Daten im Vergleichsdatensatz mit denen in der zuvor gespeicherten Datei verglichen werden
- Es wird der „DecisionTree“ Operator gestartet

- Jetzt wird der „RuleLearner“ gestartet

Nach abschluss der Berechnung wird das Ergebnis in der aus RapidMiner bekannten Oberfläche, in einem neuen Fenster angezeigt.

6.3.4 Operatodetails

DecisionTree

Parameter	Wert
criterion	information gain
minimal leaf size	2
minimal gain	0.0
maximal depth	10
confidence	0.25

RuleLearner

Parameter	Wert
criterion	information gain
sample ration	0.9
pureness	0.9
minimal prune benefit	0.25

7 Evaluation

7.1 Bewertung auf Grundlage der Aufgabenstellung

In der Aufgabenstellung (Kapitel 1.1) sind einige Eckpunkte von ISIE aufgelistet. Anhand dieser Punkte wollen wir nachfolgend den Erfolg der Projektarbeit betrachten. Zur Aufgabe gehörte die Verwendung der Methoden des Information Retrieval sowie des Data Mining. Informationen mussten gesammelt und extrahiert werden. Dabei wurden unterschiedlichste Quellen verwendet wie Drucksachen-PDFs, die Bundestags-Website, Google und Wikipedia. Da diese Daten so nicht genutzt werden konnten, entstanden als Ergebnis verschiedenste Datenbanken, z.B. MOPS (Abgeordneten-Datenbank) oder ein Drucksachen-Index. Relevante Daten wurden durch reguläre Ausdrücke, XPath Anfragen auf XML-Dateien oder durch unsere QueryFacade-Komponente extrahiert.
Eine zentrale Aufgabe war die Strukturierung der Basisdaten zu einem personen-individuellen Pressespiegel. Die Dossier-Komponente geht über diese Anforderungen hinaus und stellt neben den Redebeiträgen und Drucksachen zusätzlich auch ein Foto und personen-bezogene Informationen des Abgeordneten zur Verfügung.
Ein weiteres Ziel war die Verbesserung der üblichen Stichwortsuche von Suchmaschinen, die ganze Dokumente zurückliefern. ISIE bietet mit der EventCutter-Komponente die Möglichkeit die Dokumente auf die Umgebung der relevanten Stellen zu reduzieren. Eine noch genauere Möglichkeit Antworten zu erhalten, ist durch das Fragebeantwortungssytem in ISIE realisiert. Eine strukturierte Fragestellung liefert nur Ergebnisse des gewünschten Typs (QueryFacade-Komponente). Eigentlich war hier eine freie Formulierung der Frage angedacht. Dies wurde aber aus Komplexitätsgründen auf die strukturierten Fragen mit natürlich-sprachlicher Fragevorschau reduziert. Die in der Aufgabenstellung gestellte Frage liefert beispielsweise folgende Ergebnisse:

Erika Steinbach (CDU/CSU)
Monika Brüning (CDU/CSU)
Bärbel Kofler (SPD)

Neben den Namen werden dabei auch die im Dossier vorhandenen personen-bezogenen Informationen angezeigt. Außerdem wird dem Benutzer die Möglichkeit geboten, grundlegende Aussagen über die Ergebnisse seiner Anfrage zu erhalten. Dies geschieht unter Zuhilfenahme von RapidMiner, der einen passenden Entscheidungsbaum generiert.
In den Vorverarbeitungsschritten wurde die NER benutzt und verschiedene Tags wie „Person“, „Partei“, „Amt“, etc. zunächst manuell getaggt und später automatisch per RapidMiner annotiert. Dies war im späteren Verlauf der PG hilfreich um Abstimmungen anhand der Tags zu erkennen und somit unabhängig von regulären Ausdrücken zu sein.

Abbildung 7.1: Möglichkeiten der Anzeige von Abgeordneten auf bundestag.de

7.2 Bewertung auf Grundlage der Website bundestag.de

Nachfolgend wird das System mit den bestehenden Funktionen auf der Bundestags-Website www.bundestag.de verglichen. Hauptaugenmerk liegt dabei auf den zusätzlichen Funktionen, die das System bietet sowie Verbesserungen im Funktionsumfang.

7.2.1 Suche nach Textausschnitten

Die Bundestags-Website bietet über den Suchdienst DIP21 zwar eine Funktion nach Stichwörtern innerhalb von Drucksachen und Plenarprotokollen zu suchen, allerdings werden hier nur ganze Dokumente zurückgeliefert. Die Anzeige der Fundstelle des Stichworts wird hier dem Benutzer verwehrt und muss wieder im Dokument gesucht werden. Man weiss also nur, dass es dort vorkommt.
Unser System bietet im Unterpunkt „Stichwortsuche“ die Möglichkeit ein Stichwort einzugeben. Hier wird als Antwort eine Menge von Abschnitten eines Dokuments geliefert, in dem das gesuchte Stichwort vorkommt. Damit bekommt der Benutzer direkt einen Eindruck über das Thema und muss nicht mühsam nach den Fundstellen suchen.

7.2.2 Informationen über Abgeordnete

Auf der Bundestags-Website ist es möglich Abgeordnete nach Fraktionen, Wahlkreis oder Bundesland anzeigen zu lassen. Diese Informationen werden statisch über eine Auswahl in der linken Navigationsleiste angezeigt (siehe Abbildung 7.1). Hier gibt es die eben erwähnten Möglichkeiten der Filterung von Abgeordneten nach Fraktion, Wahlkreis oder Bundesland. Daneben ist auch eine Gesamtliste anwählbar.
In isie geht diese Suche nach Abgeordneten sehr viel weiter. Verschiedenste Filtermöglichkeiten sind hier denkbar, beispielsweise nach Anzahl der Kinder, Religionszugehörigkeit, Geschlecht oder Geburtsort. Diese Funktion ist unter dem Punkt „Abgeordneten-Info“ realisiert. Insgesamt 16 Filterkriterien stehen zur Auswahl, wobei jeweils über Vergleichsoperatoren $<$

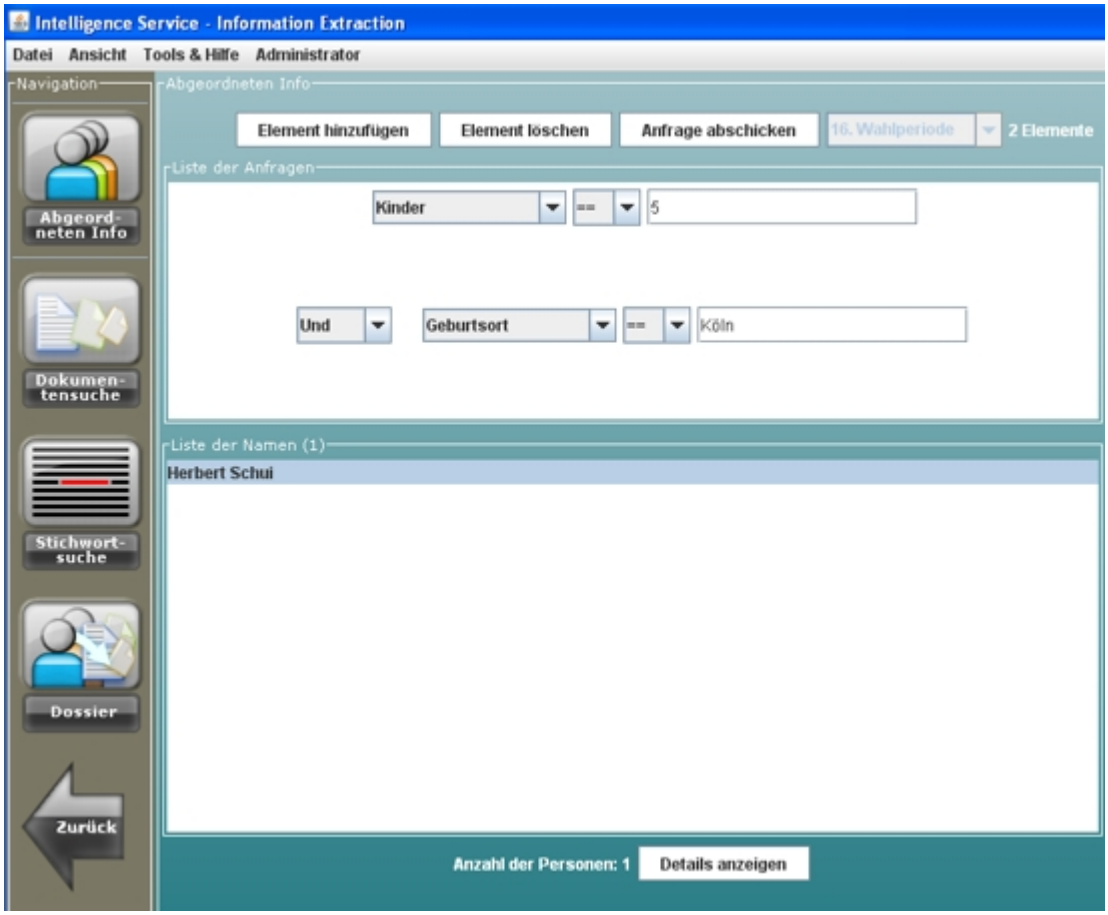


Abbildung 7.2: Filterung von Abgeordneten mit isie

$, ==, ! =, ...$ und einer freien Auswahl der Schranke, die Abgeordneten ausgewählt werden. Darüberhinaus sind beliebig viele Filterkombinationen denkbar. So gibt es nur einen Abgeordneten der genau 5 Kinder (Filterkriterium 1) hat und in Köln (Filterkriterium 2) wohnt (Stand vom 06.08.08). Ein Bildschirmfoto dieser Funktion ist in Abbildung 7.2 zu sehen.

7.2.3 Abgeordneten-Dossier

Das Abgeordneten-Dossier 5.6 eine Funktion, die man so nicht auf der Bundestags-Website findet. Hier gibt es zwar die Möglichkeit weitergehende Informationen zu den Abgeordneten anzuzeigen, allerdings sind diese Informationen statisch und bieten keine Angaben über Beteiligungen an Drucksachen oder Reden im Bundestag.
Genau das leistet das Abgeordneten-Dossier, dass im Menüpunkt „Dossier“ zu finden ist. Beim Aufruf dieser Funktion wird direkt eine alphabetisch sortierte Liste der Abgeordneten angezeigt. Alternativ kann auch nach Abgeordneten gesucht werden. Klickt man auf einen Namen, so öffnet sich ein Fenster mit den allgmeinen Daten zur Person, sowie ein Foto des Abgeordneten. Das Highlight in diesem Bereich sind die Angaben der Drucksachenbeteiligungen sowie die einzelnen Reden, die im Bundestag gehalten wurden. Beide Funktionen sind neu und so nicht auf der Website implementiert.

7.2.4 Fragebeantwortung

Isie bietet einen weiteren Mehrwert gegenüber der Website des Bundestags. Der Bereich Fragebeantwortung bietet die Möglichkeit, unter Angabe einer Zielentität, eine Frage zu konstruieren, deren natürlichsprachliche Form in einem Vorschaufenster angezeigt wird. Hier ist es also möglich genau eine Antwort auf eine gestellte Frage zu bekommen. Voraussetzung ist, dass man eine ungefähre Vorstellung von der Antwort hat, denn dies ist nötig um die Zielentität, also ein Amt, eine Person, usw., zu bestimmen. Danach erscheinen eine Reihe von Filtermöglichkeiten, die die gesuchte Antwort einschränken. Der Clou ist dabei die Darstellung der natürlichsprachlichen Frage am unteren Rand des Fensters.

Zwar existiert auf der Bundestags-Website der „virtuelle Berater“ in Form eines Bundestagsadlers, allerdings liefert dieser auf eine Frage nur vorgefertigte Sätze und Satzbausteine. Die Frage „Welche Abgeordneten sind in Wahlperiode 16 Mitglied der SPD?“ ergab hier folgende Antwort durch den virtuellen Berater: „Alle Informationen zu den Abgeordneten des deutschen Bundestages finden Sie hier: http://www.bundestag.de/mdb/index.html“. Somit ist dieser Service kein echtes Fragebeantwortungssystem, sondern lediglich eine erweiterte Stichwortsuche, da einzelne Stichwörter offensichtlich zur Beantwortung genutzt werden.

7.2.5 PartyNator

Ein kleines Gimmick von isie ist der so genannte PartyNator 5.7. Diese Komponente bestimmt die Parteizugehörigkeit bzw. die Parteiaffinität des Benutzers anhand von vorbestimmten Attributen, die ausgewählt werden können. Als Ergebnis wird eine dreidimensionale Tortengrafik generiert, die unter bestimmten Parteiattributen die Parteizugehörigkeit anzeigt. Ebenfalls eine Anwendung, die auf der Bundestags-Website nicht vorhanden ist.

7.3 Optimierungsvarianten

Um die Bearbeitungszeit der Fragebeantwortung zu verkürzen und die Datenhaltung zu zentralisieren, wäre ein Vorschlag, eine relationale Datenbank anzulegen, in der alle Daten enthalten sind. In Abbildung 7.3 ist ein erster grober Entwurf der Datenbank. Um die Datenbank nutzen zu können, müssten allerdings die kompletten Daten aus der bisherigen Datenhaltung übertragen werden. Dies ist mit etlichen Konvertierungsschritten verbunden. Allerdings hätte man danach eine einheitliche Datenhaltung, außerdem liefert eine SQL Datenbank auch die Antworten die bisher von der QueryFacade erstellt werden. Es wäre auch möglich die Datenbank zentral auf einem Server laufen zu lassen und via Webdienst auf diese zuzugreifen, dadurch würde der benötigte Speicherplatz bei jedem User deutlich sinken. Updates der Daten könnten dann auch einfacher von einem Servicetechniker zentral durchgeführt werden und müssten nicht mehr von jedem User selbst dezentral gemacht werden.

Eine weitere Verbesserung wäre die Einbeziehung der älteren Wahlperioden, dadurch könnte man noch umfassendere Aussagen über Regelmäßigkeiten oder Veränderungen über die Zeit machen. Hierzu ist es allerdings notwendig, dass man die BILD-PDFs analysiert und zum Beispiel mittels einer OCR-Software diese in ASCII Dokumente konvertiert, ohne wichtige Strukturmerkmale zu verlieren.

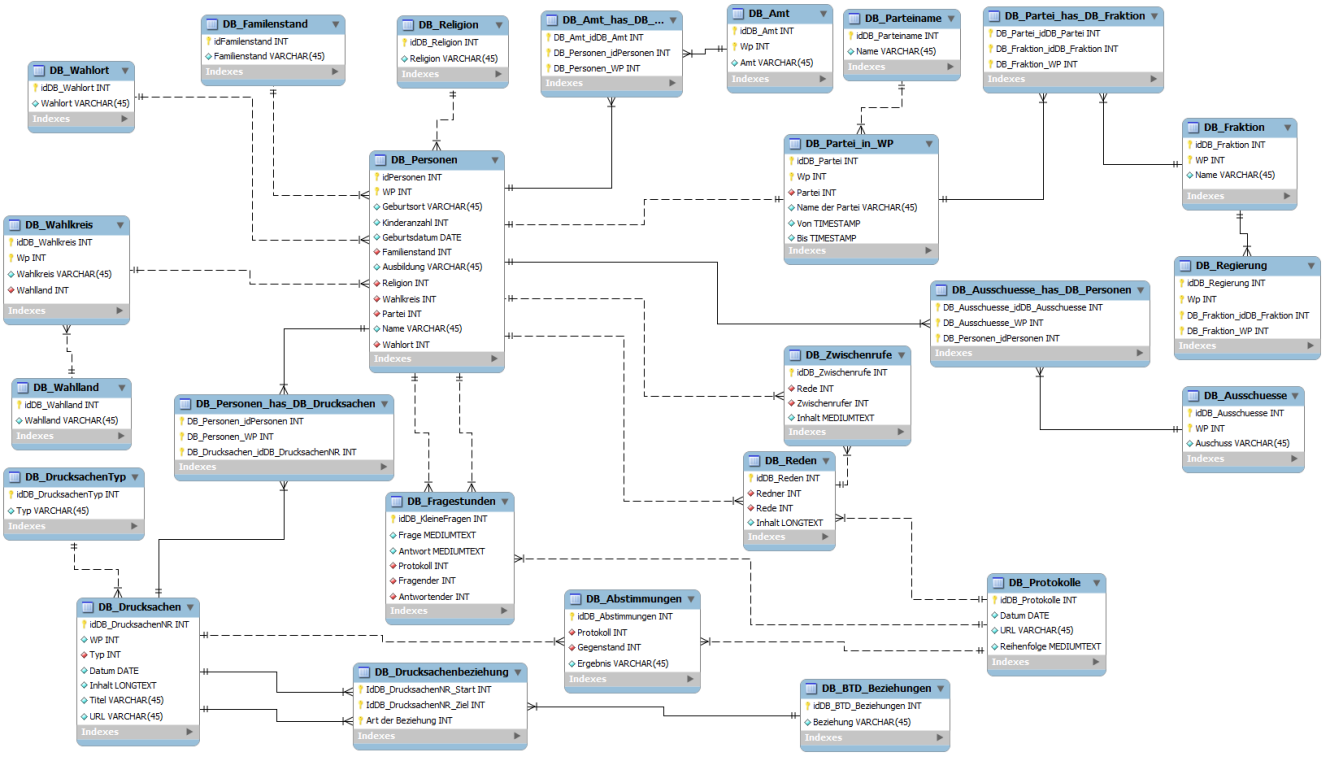


Abbildung 7.3: SQL Datenbank

Außerdem wäre ein Ansatzpunkt für Verbesserungen, die Fragevorschau bzw. die Frageeingabe, bisher haben wir die Fragevorschau aus fertigen Bausteinen generiert. Zur Kontrolle könnte man noch eine Grammatikprüfung einbauen, um die Korrektheit der Sätze zu gewährleisten. Noch besser wäre natürlich eine freie Frageeingabe des Benutzers, hierzu müsste allerdings die Eingabe aufwendig analysiert werden, und in eine, vom Computer verständliche Sprache gebracht werden, zum Beispiel SQL.

Da das gesamte System über 30GB Festplattenspeicher in Anspruch nimmt, wäre zu überlegen, ob man nicht einen Server für das System nutzt und dem User zum Beispiel über eine Browseroberfläche Zugriff auf das System verschafft. Neben dem Vorteil, dass man das System auch von jedem Internetzugangspunkt benutzen kann, hat man auch als Administrator nur ein System zu pflegen und kann hier für Updates und Aktualität sorgen. Anstelle eines Webservices könnte man das System auch als Server-Client System aufbauen, wodurch auch fast alle Vorteile gegeben wären.

Literaturverzeichnis

[AI99] Appelt, Douglas E. ; Israel, David J.: Introduction to Information Extraction Technology, 1999

[al.04] al., M.Diligenti M.: A Unified Probabilistic Framework for Web Page Scoring Systems. In: *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING* 16 (2004)

[al.07] al., Buntine et: ALVIS - Superpeer Semantic Search Engine - Project Overview. (2007)

[AN06] A. Nazarenko, et.al.: Deliverable D5.4 Complete document processing prototype. (2006)

[AT00] Ann TAYLOR, Beatrice S. Mitchell MARCUS M. Mitchell MARCUS: The Penn Treebank: *An Overview.* (2000)

[BM05] Bunescu, Razvan C. ; Mooney, Raymond J.: A Shortes Path Dependency Kernel for Relation Extraction / Dept. of Computer Sciences University Texas at Austin. 2005. – Forschungsbericht

[Boy] Boyle, Roger: *Hidden-Markov-Model online Tutorium.* HMM,

[BP98] Brin, Sergey ; Page, Lawrence: The Anatomy of a Large-Scale Hypertextual Web Search Engine. In: *Proceedings of the Seventh World Wide Web Conf. WWW7*, 1998, S. 107–117

[BR04] B.Young, A. Meyers R.Reeves C.Macleod R.Szekely V. ; R.Grishman: The NomBank Project: *An Interim Report.* In: Meyers, A. (Hrsg.): *HLT-NAACL 2004 Workshop: Frontiers in Corpus Annotation.* Boston, Massachusetts, USA : Association for Computational Linguistics, May 2 - May 7 2004, S. 24–31

[Bra99] Brants, Thorsten: Cascaded Markov Models. Bergen, 1999, S. 118 – 125

[Bra00] Brants, Thorsten: TnT – A Statistical Part-of-Speech Tagger. Seattle, 2000, S. 224 –331

[Bri92] Brill, Eric: A simple rule-based part-ofspeech tagger. In: *ANLP*, 1992, S. 152–155

[Bur98] Burges, Christopher J. C.: A Tutorial on Support Vector Machines for Pattern Recognition. In: *Data Min. Knowl. Discov.* 2 (1998), Nr. 2, S. 121–167. `http://dx.doi.org/http://dx.doi.org/10.1023/A:1009715923555`. – DOI http://dx.doi.org/10.1023/A:1009715923555. – ISSN 1384–5810

[CB03] Cunningham, Hamish ; Bontcheva, Kalina: Named Entity Recognition Tutorial, 2003

[Cha00] Charniak, Eugene: A maximum-entropy-inspired parser. In: *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2000, S. 132–139

[CM04] Carreras, Xavier ; Màrquez, Lluís: Introduction to the CoNLL-2004 Shared Task: Semantic Role Labeling. In: *Proceedings of CoNLL: Conference on Natural Language Learning*, 2004

[CM05a] Carreras, Xavier ; Màrquez, Lluís: Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In: *Proceedings of CoNLL: Conference on Natural Language Learning*, 2005

[CM05b] Carreras, Xavier ; Màrquez, Lluís: *Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling*. 2005

[Col99] Collins, Michael J.: *Head-driven Statistical Models for Natural Language Parsing*. Philadelphia, University of Pennsylvania, Diss., 1999

[CS03] Culotta, Aron ; Sorensen, Jeffrey: Dependency Tree Kernels for Relation Extraction / University of Massachusetts and IBM T.J. Watson Research Center. 2003. – Forschungsbericht

[DA04] D.Weld, O.Etzioni M.Cafarella D.Downey S.Kok A.Popescu T.Shaked S. ; A.Yates: Web-Scale Information Extraction in KnowItAll. (2004), S. 100–110

[DC03] David Cohn, Andrew M. Rich Caruana C. Rich Caruana: Semi-supervised Clustering with User Feedback / Cornell University. 2003. – Forschungsbericht

[Ebe06] Eberhardt, Stefan: *Semisupervised K-Means Clustering*. http://www.informatik.uni-ulm.de/ni/Lehre/SS06/SeminarNI/ausarbeitungen/Eberhardt.pdf, 2006

[EK03] Esther König, Holger V. Wolfgang Lezius L. Wolfgang Lezius: *TIGERSearch 2.1 User's Manual*. IMS, University of Stuttgart, 2003

[EV06] Engels, Eva ; Vikner, Sten: Satzglieder, Kasus und semantische Rollen: eine Einführung. In: *Tidsskrift for Sprogforskning* 4 (2006), Nr. 1, S. 17 – 37

[Fel03] Feldman, Ronen: Information Extraction - Theory and Practice, 2003

[GH03] Gildea, Daniel ; Hockenmaier, Julia: Identifying semantic roles using combinatory categorial grammar. Sapporo, Japan, 2003

[GJ00] Gildea, Daniel ; Jurafsky, Daniel: Automatic labeling of semantic roles. In: *ACL '00: Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. Morristown, NJ, USA : Association for Computational Linguistics, 2000, S. 512–520

[GJ02a] Gildea, Daniel ; Jurafsky, Daniel: Automatic labeling of semantic roles. In: *Comput. Linguist.* 28 (2002), Nr. 3, S. 245–288

[GJ02b] Gildea, Daniel ; Jurafsky, Daniel: Automatic labeling of semantic roles. (2002), S. 245?88

[Gri97] Grishman, Ralph: Information Extraction: Techniques and Challenges. In: *SCIE*, 1997, S. 10–27

[Hea99] Hearst, M.: Automatic acquisition of hyponyms from large text corpora. In: *In Proceedings of the 14th International Conference on Computational Lingustics*, 1999, S. 539–545

[HTT] *HTTP/1.0 specification.* http://www.w3.org/Protocols/HTTP/1.0/spec.html,

[J.80] J., Buxton: *STONEMAN-Requirements for Ada Programming Support Environments.* 1980

[JA02] J.Lin, S.Dumas M.Banko E. ; A.Ng: Web question answering, Is more always better? In: *In Proceeedings of SIGIR 2002*, 2002, S. 291–298

[JN06] Jiang, Zheng P. ; Ng, Hwee T.: Semantic Role Labeling of NomBank: A Maximum Entropy Approach, 2006

[Joa98] Joachims, T.: Making large-scale support vector machine learning practical. Version: 1998. `citeseer.nj.nec.com/joachims98making.html`. In: B. Schölkopf, A. S. C. Burges B. C. Burges (Hrsg.): *Advances in Kernel Methods: Support Vector Machines.* MIT Press, Cambridge, MA, 1998

[Joa02] Joachims, T.: Optimizing Search Engines using Clickthrough Data. In: *Proceedings of Knowledge Discovery in Databases*, 2002

[Jos06] Jost, Michael: *Document Clustering for Information Organization and Retrieval.* http://wwwi2.informatik.uni-wuerzburg.de/lehre/se0506/ausarbeitungen/jost.pdf., 2006

[Jun06] Jungermann, Felix: *Named Entity Recognition mit Conditional Random Fields*, Computer Science, University of Dortmund, Diplomarbeit, 2006

[Kas08] *Kasusgrammatik.* http://de.wikipedia.org/wiki/Kasusgrammatik, 2008

[Kin02] Kingsbury, Marcus M. und Palmer M. P.: Adding Predicate Argument Structure to the Penn TreeBank. San Diego, 2002

[KM00] Kudoh, Taku ; Matsumoto, Yuji: Use of support vector learning for chunk identification. In: *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning.* Morristown, NJ, USA : Association for Computational Linguistics, 2000, S. 142–144

[Kri04] Krifka, Manfred: *Argumentenstruktur und Verbsemantik.* 2004

[Lez02] Lezius, Wolfgang: Ein Werkzeug zur Suche auf syntaktisch annotierten Textkorpora. (2002)

[LR89] Lawrence R., Rabiner: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, Institute of Electrical and Electronics Engineers, 1989, S. 257–286

[Luc] *Lucene Query Parser Syntax*. http://lucene.apache.org/java/docs/queryparsersyntax.html,

[Mey02] Meyer, Paul: *Synchronic English Linguistics. An Introduction*. Tübingen : Narr, 2002

[Mit98] Mitkov, Ruslan: Comparing a statistical and a rule-based tagger for German. (1998), S. 125–137

[Mit03] Mitkov, Ruslan: The Oxford Handbook of Computational Linguistics. (2003), S. 545–560

[MRM+04] Meyers, A. ; Reeves, R. ; Macleod, Catherine ; Szekeley, Rachel ; Zielinska, Veronkia ; Young, Brian: The Cross-Breeding of Dictionaries. In: *Proceedings of LREC-2004*. Lisbon, Portugal, 2004

[PH01] Pasca, Marius ; Harabagiu, Sanda M.: Answer Mining from On-Line Documents / Department of Computer Science and Engineering Southern Methodist University. 2001. – Forschungsbericht

[PHK+05] Pradhan, Sameer ; Hacioglu, Kadri ; Krugler, Valerie ; Ward, Wayne ; Martin, James H. ; Jurafsky, Daniel: Support Vector Learning for Semantic Argument Classification. In: *Mach. Learn.* 60 (2005), Nr. 1-3, S. 11–39. http://dx.doi.org/http://dx.doi.org/10.1007/s10994-005-0912-2. – DOI http://dx.doi.org/10.1007/s10994–005–0912–2. – ISSN 0885–6125

[Pla99] Platt, John C.: Fast training of support vector machines using sequential minimal optimization. (1999), S. 185–208. ISBN 0–262–19416–3

[Rat97] Ratnaparkhi, Adwait: A simple introduction to maximum entropy models for natural language processing / Institute for Research in Cognitive Science, University of Pennsylvania. 1997. – Forschungsbericht

[Rij79] Rijsbergen, C. J. V.: *Information Retrieval*. 1979

[Sem08] *Semantische Rolle*. http://de.wikipedia.org/wiki/Semantische_Rolle, 2008

[SG02] Strehl, Alexander ; Ghosh, Joydeep: Cluster Ensembles – A Knowledge Reuse Framework for Combining Partitionings. In: *Proceedings of AAAI 2002, Edmonton, Canada*, 2002

[Som04] Sommerville, Ian: *Software Engineering 7th. Ed.* Pearson Studium, 2004

[SS04] Swier, Robert S. ; Stevenson, Suzanne: Unsupervised semantic role labelling. In: *Proceedings of the 2004 conference on EMNLP*, 2004, S. 95–102

[SS05] Swier, Robert S. ; Stevenson, Suzanne: Exploiting a verb lexicon in automatic semantic role labelling. In: *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*.

Morristown, NJ, USA : Association for Computational Linguistics, 2005, S. 883–890

[THJA04] Tsochantaridis, Ioannis ; Hofmann, Thomas ; Joachims, Thorsten ; Altun, Yasemin: Support vector machine learning for interdependent and structured output spaces. In: *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, 2004

[TJP04] Topchy, A. ; Jain, A ; Punch, W: A mixture model for clustering ensembles. In: *Proc. of SIAM Conference on Data Mining*, 2004

[WLB05] Wray L. Buntine, Michael P. T. Kimmo Valtonen V. Kimmo Valtonen: The ALVIS Document Model for a Semantic Search Engine. In: *Demos and Posters of the 2nd European Semantic Web Conference ESWC2005*, 2005

[Yam] *YamCha, Yet another multipurpose Chunk Annotator.* http://www.chasen.org/ taku/software/yamcha/,

[Zho06] Zhou, Zhixiong: *Semi-supervised Hierarchical Clustering.* http://www.informatik.uni-ulm.de/ni/Lehre/SS06/SeminarNI/ausarbeitungen/Zhou.pdf, 2006