Automatische Extraktion von Schlüsselwörtern aus Text

eine Algorithmen-Analyse

Freie wissenschaftliche Arbeit zur Erlangung des Grades eines
Diplom-Systemwissenschaftlers (Medien) an der Fakultät Medien
der Bauhaus-Universität Weimar

Eingereicht von: Karsten Klüger

wiss. Betreuer: Sven Meyer zu Eißen
Prof. Benno Stein

1. Gutachter: Prof. Benno Stein
2. Gutachter: Prof. Charles A. Wüthrich

Weimar, 22. Juni 2006

Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich, dass ich die vorliegende Arbeit selbständig angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Weimar, 22. Juni 2006

Abstract

Die fokussierte Suche nach relevanter Information im Internet ist in Anbetracht der stetig wachsenden Menge der online verfügbaren Dokumente eine zunehmende Herausforderung für den Benutzer. Dies zeigt sich, indem Suchanfragen immer präziser gestellt werden müssen, um eine überschaubare Menge von Ergebnissen zu erhalten. Die Auswahl und Kombination qualifizierter Suchbegriffe ist daher das entscheidende Kriterium für die erfolgreiche Benutzung der existierenden Suchmaschinen.

Diese Diplomarbeit befasst sich mit der automatischen Extraktion von Schlüsselwörtern aus Textdokumenten zur Ermittlung relevanter, stark diskriminierender Suchbegriffe. Dies erfordert den Einsatz so genannter dokumentbasierter Algorithmen, die Schlüsselwörter auf der Grundlage eines einzelnen Dokuments extrahieren können. Es wurden mehrere grundlegend verschiedene Algorithmen implementiert. Als Referenz dient der überwachte, korpusbasierte KEA-Algorithmus. Im Vergleich dazu wird die Leistungsfähigkeit unüberwachter, dokumentbasierter Algorithmen (RSP, B&C und Cooccurrence) untersucht.

Die Extraktionsverfahren wurden in einer Java-Bibliothek implementiert. Weiterhin wurde ein Testkorpus aus wissenschaftlichen Dokumenten erstellt. Dieser Korpus stellt die Grundlage für die Evaluierung der Algorithmen dar.

Im Ergebnis der Evaluierung zeigt sich die erwartete Überlegenheit der korpusbasierten, überwachten Verfahren. Die dokumentbasierten Verfahren weichen in ihren Ergebnissen teilweise stark von einander ab. Der Cooccurrence-Algorithmus erweist sich bei allen Tests als sehr robust und liefert gute Extraktionsergebnisse. Damit ist er für den beschriebenen Einsatz gut geeignet.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Danksagung

Herrn Prof. Dr. Benno Stein danke ich für die Überlassung des Themas dieser Diplomarbeit.

Besonders möchte ich mich bei meinem Betreuer Herrn Dipl.-Inf. Sven Meyer zu Eißen bedanken für die wissenschaftliche Betreuung, die stetige Bereitschaft, meine Fragen zu beantworten und mir fachliche Hinweise zu erteilen sowie für die kritische Durchsicht bei der Abfassung der vorliegenden Schrift.

Ein großes Dankeschön geht an meine Partnerin Dana und meine Tochter Ella, die mir oft eine wohltuende und erheiternde Abwechslung waren und mich mit viel Verständnis durch die Höhen und Tiefen während der Arbeit begleitet haben.

1. Einleitung

Im August 2005 verkündete das *Yahoo! Search Weblog* 19,2 Milliarden indizierte Webdokumente[1] [Yahoo Search Blog, 2005]. Das beweist einmal mehr, dass sich das Internet, insbesondere das World Wide Web (kurz das Web), zu einem wichtigen Informationsmedium unserer Zeit entwickelt. Suchmaschinen ermöglichen das Auffinden von Dokumenten innerhalb dieser immensen und stetig wachsenden Menge von Information. Bedeutende Suchmaschinen, wie beispielsweise Google [Brin & Page, 1998], beruhen grundlegend auf dem Prinzip der Volltextindexierung sowie diversen Ranking-Verfahren. Die Suchergebnisse sind dabei sehr stark durch die vom Benutzer übergebenen Suchanfragen determiniert. Die meisten Benutzer dieser Suchmaschinen stoßen auf der Suche nach spezifischer Information jedoch schnell an ihre Grenzen. Deutlich wird dies, indem Suchanfragen immer präziser gestellt werden müssen, um eine überschaubare Menge von Ergebnissen zu erhalten. Es gilt daher umso mehr das Motto: *„Man muss nur die richtige Frage stellen."*

Die Mehrzahl der Benutzer generiert nur sehr kurze Suchanfragen, was zwangsläufig in einer Flut von Suchergebnissen mündet, die zudem häufig einen hohen Anteil irrelevanter Information enthalten. Das Filtern dieser Ergebnismenge ist für den Nutzer dann im allgemeinen nicht effektiv durchzuführen. Nach einer Studie von Spink und Jansen bestehen die meisten Anfragen aus 1-3 Wörtern und die Benutzer geben durchschnittlich 2-3 Anfrage pro Suche ein [Spink & Jansen, 2004]. Die Gründe für dieses Nutzerverhalten können sehr vielschichtig sein. Der Benutzer besitzt einerseits vielleicht nicht genügend Erfahrung in der Auswahl diskriminierender Suchbegriffe oder sein Fachwissen ist nicht tiefgründig genug. Andererseits werden aber nützliche Suchbegriffe möglicherweise auch einfach übersehen. Die Herausforderung besteht demnach in der automatisierten Unterstützung einer fokussierten Suche nach relevanter Information.

1 Die Daten, Texte, Bilder, Video- und Audiodateien, die im WWW zur Verfügung stehen, werden allgemein als Dokumente bezeichnet. Ein Dokument im Sinne dieser Arbeit ist jedoch ausschließlich ein Textdokument.

Die Wissenschaft vom Suchen nach relevanter Information in umfangreichen Dokumentenmengen bezeichnet man als Information Retrieval (IR). Modelle zur Repräsentation und Methoden zur Ähnlichkeitsbestimmung von Dokumenten sowie zur Indexierung von Dokumentenmengen oder die automatische Klassifikation sind nur einige der Schwerpunkte diese Fachgebiets der Informatik.

Konzentriert sich die Suche nach relevanter Information ausschließlich auf Textdokumente spricht man vom Text-Retrieval. Gegenstand dieser Arbeit ist ein Bereich des Text-Retrievals, der sich mit der automatischen Extraktion von Schlüsselwörtern aus Textdokumenten befasst. Ziel ist die automatische Ermittlung von Suchbegriffen zur effizienten Benutzung bestehender Suchmaschinen bei der fokussierten Suche nach relevanter Information.

Mögliche Szenarien für eine fokussierte Suche nach relevanten Dokumenten im Web sind:

- *Automatisierte Recherche.* Zu einem vorgegebenen Thema und Kontext sollen relevante Dokumente gesucht werden, z.B. Automatisierte Zusammenfassung von Umsatzvorhersagen auf der Basis von Internetdaten [Stein & Meyer zu Eissen, 2005].
- *Automatisches Aufbauen und Erweitern einer Dokumentkollektion.* Im Rahmen einer wissenschaftlichen Arbeit soll eine Kollektion aus wissenschaftlichen Veröffentlichungen zu einem bestimmten Thema automatisiert aufgebaut oder erweitert werden. Derartige Dokumentsammlungen bezeichnet man im IR auch als Korpus.
- *Web-basierte Plagiatanalyse.* Als Plagiat bezeichnet man ein durch unrechtmäßige Nachahmung oder Diebstahl geistigen Eigentums entstandenes Werk. In wissenschaftlichen Schriften bezieht sich das Plagiat dabei explizit auf die Verwendung fremder Ideen ohne einen entsprechenden Hinweis auf die zugrunde liegenden Quellen. Durch die ständig wachsende Anzahl digital verfügbarer Dokumente im Internet wird es immer einfacher, Dokumente zu einem speziellen Thema zu finden und in Teilen zu kopieren [Kleppe et al.,

2005]. Mittels einer Web-basierten Plagiatanalyse sollen identische und bearbeitete Kopien von Textabschnitten, umstrukturierte oder erweiterte Dokumente oder unautorisierte Übersetzungen erkannt werden.

Um die für die Schlüsselwortextraktion anwendbaren Techniken zu ermitteln, wird einführend ein kurzer Überblick über die Repräsentation von Textdokumenten im Computer und die Erstellung von gebräuchlichen Dokumentmodellen gegeben. Der Hauptteil befasst sich ausführlich mit der Ermittlung geeigneter Suchbegriffe aus einem Textdokument. Es wird untersucht, wie Schlüsselwörter automatisch aus einem vorhandenen Textdokument extrahiert werden können. Dazu wurden vier unterschiedliche Extraktionsverfahren implementiert und evaluiert. Die Evaluierung der Algorithmen erfolgte mittels eines im Rahmen dieser Arbeit zusammengestellten Testkorpus aus 250 wissenschaftlichen Veröffentlichungen. Mögliche Technologien zur Verbesserung der Extraktionsergebnisse runden die Betrachtungen zur automatischen Schlüsselwortextraktion ab. Abschließend wird die effiziente Generierung von Suchanfragen auf Basis der extrahierten Schlüsselwörter zur effektiven Nutzung der existierenden Web-Suchmaschinen betrachtet.

2. Repräsentation von Textdokumenten im IR

Damit ein computergestütztes IR-System ein Textdokument verarbeiten kann, muss das Dokument im Rechner gespeichert werden. Aus Effizienzgründen ist es dabei nicht notwendig, den gesamten Text in seiner natürlichen Sprache zu verwenden. Vielmehr sollte das Dokument in eine für das System verständliche Struktur überführt werden. Diese abstrakte Struktur bezeichnet man als Dokumentmodell. Ein im IR häufig verwendetes Dokumentmodell ist die Darstellung eines Dokuments *d* in Form eines Vektors aus den in *d* enthaltenen Wörtern (Terme). Dafür wird aus allen zu untersuchenden Dokumenten *D* ein *n*-dimensionaler Termvektor erzeugt, wobei *n* die Anzahl der unterschiedlichen Terme in *D* ist. Das Vokabular der zulässigen Terme sollte möglichst alle Sinn tragenden Wörter enthalten, die im Dokument bzw. der Kollektion auftreten. Diese Terme werden auch als Indexterme bezeichnet. Sie werden aus den Wörtern der Dokumente mit Hilfe verschiedener Umwandlungen ermittelt, die im Folgenden als Text-Preprocessing bezeichnet werden. Dabei kommen verschiedene Verfahren zur Anwendung:

- *Lexikalische Analyse*: Ziel ist die Überführung eines Textes in einzelne Terme (z.B. Wörter). Im einfachsten Fall geschieht dies durch die Detektion von Leerzeichen unter Beachtung einiger Sonderfälle wie Zahlen, Trennzeichen oder Interpunktionszeichen. Zur einheitlichen Darstellung der Terme wird häufig noch eine Konvertierung in Groß- oder Kleinschreibung durchgeführt.
- *Stoppwortelimination (stop word removal)*: Stoppwörter sind häufige und gleich verteilt vorkommende Terme, wie z.B. Artikel, Konjunktionen und Präpositionen. Im Allgemeinen sind etwa 20-30% der Wörter in einem Text Stoppwörter. Durch die Stoppwortelimination werden diese schlecht diskriminierenden Wörter aus Dokumenten entfernt. Somit wird der Speicheraufwand reduziert und die Verfahren beschleunigt. In der Praxis verwendet man dafür spezifische Listen, welche die Stoppwörter der jeweiligen Sprache enthalten.

- *Stammformreduktion (stemming)*: In vielen Sprachen können Wörter in verschiedenen Formen auftreten (Einzahl/Mehrzahl, Konjugation, Deklination). Die verschiedenen Formen sollten aber nicht als unterschiedliche Terme betrachtet werden, sondern in ein und denselben Term überführt werden. Ziel einer Stammformreduktion ist die Rückführung der einzelnen Wörter auf ihren Wortstamm. Dies erfolgt durch die Entfernung der flexivischen Formveränderungen wie Deklination von Substantiven und Adjektiven (Kasus, Numerus, Genus) oder Konjugation von Verben (Person, Numerus, Tempus, Modus).

  Für die meisten Sprachen ist es jedoch schwierig, die linguistisch korrekte Stammform eines Wortes ohne Wörterbuch zu finden. Für einige Sprachen gibt es aber regelbasierte Ansätze und statistische Verfahren, die ohne Wörterbuch auskommen und sehr gute Resultate liefern. Diese Reduktionsalgorithmen behandeln Fälle nicht individuell, sondern wenden implementierte Regeln auf alle potenziell vorkommenden Fälle an. Bei dieser generalisierenden Vorgehensweise wird nicht immer die linguistisch korrekte Stammform ermittelt, so dass ein gewisser Grad an fehlerhaften Analysen unvermeidlich ist. Bei der Stammformreduktion nach Porter (Porter-Stemmer) werden englische Wörter beispielsweise mit Hilfe von Präfix- und Suffixlisten automatisch in eine Stammform überführt [Porter, 1980].

Die unterschiedlichen Phasen des Text-Preprozessings sind in Abbildung 1 an einem Beispiel dargestellt.

Nach Abschluss des Text-Preprocessings wird für jedes Dokument ein Merkmalsvektor (Featurevektor) erzeugt, der für jeden Indexterm t_i des Termvektors ein entsprechendes Merkmal k_i enthält. Die Auswahl dieses Merkmals ist abhängig von der jeweiligen Datengrundlage. Die Berechnung erfolgt mit simplen statistischen Verfahren.

Das einfachste aber dennoch sehr aussagekräftige Merkmal beruht auf Luhns Forschungen zur automatischen Textanalyse. Die Erkenntnis, dass die Häufigkeit eines Wortes innerhalb eines Dokuments ein geeignetes Maß für seine Signifikanz ist [Luhn, 1958], entwickelte sich zu einer wesentlichen Grundlage des Text-Retrievals. Diese als

Termhäufigkeit $tf_{t,d}$ (term frequency) bezeichnete Größe gibt die absolute Anzahl des Auftretens eines Terms *t* im Dokument *d* an. Je häufiger ein Term in einem Dokument auftritt, desto höher ist seine Signifikanz für das Dokument.

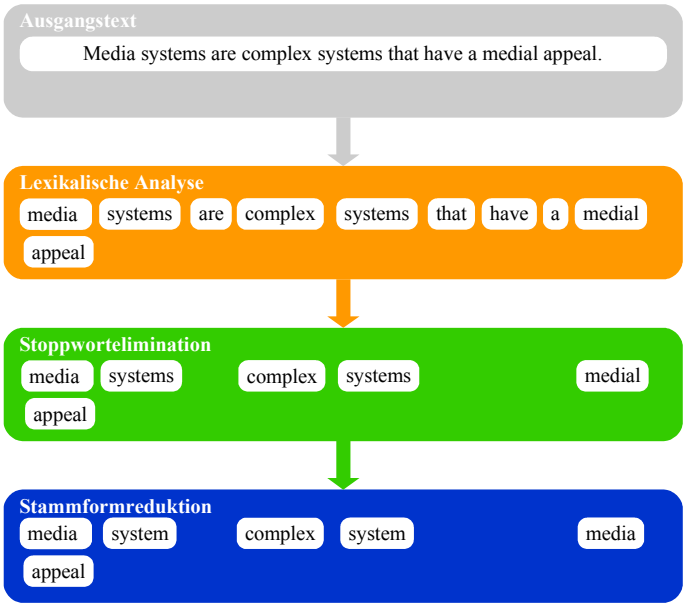

Abbildung 1: Phasen des Text-Preprocessing

Unter Anwendung der Termhäufigkeit als Merkmal *k* für das Beispieldokument aus Abbildung 1 entstehen der Termvektor **t** und der Merkmalsvektor **d** wie folgt:

$$\mathbf{t} = \begin{pmatrix} media \\ system \\ complex \\ appeal \end{pmatrix} \qquad \mathbf{d} = \begin{pmatrix} 2 \\ 2 \\ 1 \\ 1 \end{pmatrix}$$

Ein weiteres, stark diskriminierendes Merkmal auf der Grundlage einer Dokumentkollektion ist die *inverse Dokumenthäufigkeit* $idf_{t,D}$ (inverse document frequency). Sie ist ein Maß für die Verteilung eines Terms *t* innerhalb einer Dokumentkollektion *D* und damit für dessen Informationsgehalt bezüglich der Kollektion. Die Formel lautet:

$$idf_{t,D} = \log_2 \frac{size_D}{df_t},$$

wobei $size_D$ die Anzahl der Dokumente in D ist und df_t die Dokumenthäufigkeit des Terms t, also die Anzahl der Dokumente der Kollektion D, die t enthalten. Je weniger ein Term über den Dokumenten einer Kollektion verteilt ist, desto höher ist seine Diskriminationskraft für die Kollektion.

Nachdem das zu bearbeitende Dokument in eine abstrakte Form überführt wurde, können aus der Menge der Indexterme die Schlüsselwörter extrahiert werden. Es ist aber nicht immer notwendig, das Dokument in die Vektorrepräsentation zu überführen. Für einige der im folgenden beschriebenen Verfahren genügt die Überführung in eine einfache verkettete Liste aus den einzelnen Wörtern des Dokuments mittels lexikalischer Analyse.

3. Automatische Extraktion von Schlüsselwörtern

Schlüsselwörter[2] sind semantische Metadaten, d.h. sie charakterisieren ein Dokument, indem sie einen kurzen Überblick über dessen Inhalt vermitteln. Sie dienen damit der inhaltlichen Erschließung eines Dokumentes. Vom Standpunkt des Suchenden sind sie Suchbegriffe. Schlüsselwörter stammen, im Gegensatz zu Schlagwörtern, nicht aus einem kontrollierten Vokabular, sondern werden direkt dem Dokument entnommen. Turney definiert die automatische Extraktion von Schlüsselwörtern wie folgt: „*[...] the automatic selection of important, topical phrases from within the body of a document.*" [Turney, 1999] Die automatische Schlüsselwortextraktion findet beispielsweise Anwendung in der automatisierten Textzusammenfassung, der Kategorisierung von Dokumentenmengen oder der Generierung alternativer Vorschläge für Anfragen an Suchmaschinen (Query Refinement).

Das Problem der Extraktion von Schlüsselwörtern lässt sich formal wie folgt darstellen. Ein Dokument wird als Menge von Termen (mögliche Schlüsselwörter) betrachtet. Ein Term ist eine atomare Einheit eines Dokuments, wie z.B. ein Wort, eine Zahl oder eine Abkürzung. Die Aufgabe besteht in der Klassifizierung dieser Terme in Schlüsselwörter und Nicht-Schlüsselwörter.

Es sei $D=\{d_1, ..., d_n\}$, $n \geq 1$ eine Dokumentmenge. Jedes Dokument d_i ist eine Sequenz von Termen. $T_i=\{t_1, ..., t_m\}$ sei die Menge von Termen, die das Dokument d_i verwendet. Die Schlüsselwortextraktion für ein Dokument d_i bedeutet die Identifikation einer Termmenge K_i mit folgenden Eigenschaften:

- im Dokument enthalten[3]

2 In der Literatur wird neben dem Begriff Keyword Extraction häufig auch Keyphrase Extraction verwendet, da einige Extraktionsverfahren neben einzelnen Wörtern auch Wortgruppen ermitteln. Zur Vereinfachung bezieht sich der Begriff der Extraktion von Schlüsselwörtern innerhalb dieser Arbeit sowohl auf die Extraktion einzelner Wörter als auch Wortgruppen.

3 Die Teilmengenbeziehung $K_i \subseteq T_i$ gilt explizit für die Extraktion von Schlüsselwörtern. Bei der allgemeine Bestimmung von Schlüsselwörtern ist dies keine Voraussetzung, da einem Text auch Schlüsselwörter hinzugefügt werden können, die nicht im selben enthalten sind.

- eindeutig,
- ausdrucksstark,
- zusammenfassend,
- diskriminierend bezüglich anderer Dokumente,
- syntaktisch zusammenhängend

Die existierenden Verfahren können nach folgenden Kriterien klassifiziert werden:

- *Dokumentbasis*: Die Mehrzahl der Algorithmen arbeitet auf der Basis eines Korpus. Die Ergebnisse dieser Verfahren sind maßgeblich durch die spezifischen Merkmale des zu Grunde liegende Korpus bestimmt. Diese Verfahren werden im Rahmen dieser Arbeit als korpusbasiert bezeichnet. Einige Verfahren (vgl. Tabelle 1) extrahieren Schlüsselwörter aus einem einzelnen Dokument. Sie werden im Folgenden als dokumentbasiert bezeichnet. Im allgemeinen sind diese Verfahren den korpusbasierten Verfahren unterlegen, weil die aus dem Korpus resultierenden, spezifischen Merkmale, wie beispielsweise die inverse Dokumentfrequenz, zu einer starken Diskriminanzkraft der Extraktionsergebnisse führen.
- *Training*: Einige Verfahren verwenden einen überwachten Algorithmus für maschinelles Lernen. Vor dem Einsatz müssen diese Algorithmen auf einer Menge von Trainingsdaten trainiert werden. Diese Verfahren werden als überwacht bezeichnet. Algorithmen, die nicht trainiert werden müssen, heißen unüberwacht.
- *Domain-Abhängigkeit*: Extraktionsalgorithmen, die auf einem Dokumentkorpus basieren, sind im allgemeinen vom Wissensgebiet (der Domain) dieser Kollektion abhängig. Die Extraktionsergebnisse für Dokumente aus einer anderen Domain fallen daher schlechter aus, als die Ergebnisse für Dokumente aus der Domain, auf der der Algorithmus trainiert wurde. Die dem Verfahren zu Grunde liegenden Domain ist durch den Trainingskorpus determiniert und wird im Folgenden als Basisdomain bezeichnet. Dokumentbasierte Algorithmen sind im allgemeinen domainunabhängig.

- *Sprachabhängigkeit*: Die meisten Algorithmen verwenden Verfahren zur Textaufbereitung, die für jede Sprache separat implementiert werden müssen. Derartige Algorithmen werden als sprachabhängig bezeichnet.
- *Ranking der Schlüsselwortkandidaten*: Alle Extraktionsverfahren basieren hauptsächlich auf rein statistischen Ansätzen. Eine Verbesserung der Ergebnisse könnte durch Hinzunahme von externem Wissen erfolgen. In Ansätzen wird dies bereits ausgenutzt.

	Extraktionsverfahren			
	KEA	**RSP**	**Cooccurrence**	**B&C**
Dokumentbasis				
korpusbasiert	×			
dokumentbasiert		×	×	×
Training				
überwacht	×			
unüberwacht		×	×	×
Ranking				
intrinsisch	×	×	×	×
externes Wissen				
Domain-Abhängigkeit				
domainabhängig	×			
domainunabhängig		×	×	×
Sprachabhängigkeit				
sprachabhängig	×		×	×
sprachunabhängig		×		

Tabelle 1: Verfahren zur automatischen Extraktion von Schlüsselwörtern aus Text

Tabelle 1 zeigt die im Rahmen dieser Arbeit untersuchten Verfahren, die im folgenden beschrieben werden.

3.1 KEA

KEA [Witten et al., 1999] wurde 1999 im Rahmen des Projekts *New Zealand Digital Library (NZDL)* entwickelt und ist derzeit eines der bekanntesten Systeme zur automatischen Extraktion von Schlüsselwörtern auf der Basis von Dokumentkollektionen. Der Algorithmus extrahiert Schlüsselwörter mit Hilfe probabilistischer Verfahren. Anhand einer Trainingsmenge von Dokumenten mit ausgewiesenen Schlüsselwörtern werden Wahrscheinlichkeiten für die Klassifikation von Termen berechnet. Dabei wird zwischen zwei Klassen unterschieden: Schlüsselwort und Nicht-Schlüsselwort. Durch ein maschinelles Lernverfahren wird ein Klassifizierer erzeugt, der für ausgewählte Terme diese Wahrscheinlichkeit berechnet.

3.1.1 Termgewichtung

KEA benutzt zwei Merkmale zur Erkennung von Termen, die sich wahrscheinlich am besten als Schlüsselwörter eignen:

- Das Termgewicht *tf.idf* [Salton & McGill, 1983] ist das Produkt aus der relativen Termhäufigkeit $wtf_{t,d}$ und inverser Dokumentfrequenz $idf_{t,d}$ und wird berechnet durch die Formel:

  $$tf.idf_{t,d,D} = wtf_{d,t} \cdot idf_{t,D}$$

  Die relative Termhäufigkeit eines Terms *t* im Dokument *d* ist der Quotient aus (absoluter) Termhäufigkeit $tf_{t,d}$ und Anzahl aller Wörter im Dokument *d* ($size_d$):

  $$wtf_{t,d} = \frac{tf_{t,d}}{size_d}$$

  Die *tf.idf*-Gewichtung ist eines der häufigsten Verfahren zur Termgewichtung. Ein Term *t* mit hohem *tf.idf*-Gewicht kommt im Dokument *d* sehr häufig vor. Die Anzahl der Dokumente in der Kollektion D, in denen t auftritt, ist aber sehr gering. Je höher das Termgewicht von *t* ist, desto wahrscheinlicher ist *t* ein signifikantes Schlüsselwort für *d*.

- Das Attribut *first occurrence* $(fo_{t,d})$ beruht auf der Annahme, dass der Grad der Signifikanz eines Terms durch die Position seines ersten Auftretens innerhalb eines Dokuments bestimmt ist. Je eher ein Term im Dokument *d* erscheint, desto

signifikanter ist er für das Dokument. Das erste Auftreten eines Terms im Dokument wird berechnet als

$$fo_{t,d} = \frac{pre_t}{size_d},$$

wobei pre_t die Anzahl der Terme in d vor dem ersten Auftreten von t ist und $size_d$ die Anzahl aller Wörter in d. Es ergibt sich ein numerischer Wert zwischen 0 und 1, der angibt an welcher relativen Position ein Term im Dokument das erste Mal erscheint. Je kleiner $fo_{t,d}$ ist, desto wahrscheinlicher ist t ein signifikantes Schlüsselwort für d.

3.1.2 Ermittlung der Schlüsselwortkandidaten

Der Algorithmus arbeitet in drei Schritten. Im ersten Schritt werden die Schlüsselwortkandidaten ermittelt. Das zu untersuchende Dokument wird durch eine lexikalische Analyse in einzelne Terme zerlegt. Die lexikalische Analyse des Dokuments unterteilt den Text durch folgende Modifikationen:

- Interpunktionszeichen, Klammern und Nummerierungen werden durch ein Termbegrenzungssymbol ersetzt.
- Apostrophe werden entfernt.
- Durch Bindestrich zusammengesetzte Wörter werden in einzelne Wörter getrennt.

Es entsteht eine Menge von logisch zusammengehörigen Einheiten, den so genannten Token. Jeder Token enthält ein oder mehrere Wörter. Danach wird jeder Token auf mögliche Schlüsselwortkandidaten untersucht. Dies erfolgt unter Anwendung der folgenden drei Regeln:

- Schlüsselwortkandidaten sind nicht länger als drei Wörter.
- Schlüsselwortkandidaten sind keine Eigennamen.
- Schlüsselwortkandidaten können nicht mit einem Stoppwort beginnen oder enden.

Aus den verbleibenden Wortsequenzen werden die häufigsten ermittelt. Sie bilden die Menge der Schlüsselwortkandidaten K', die abschließend auf ihren Wortstamm reduziert und in Kleinschreibung konvertiert werden. Dabei wird der ursprüngliche Text jedes Schlüsselwortkandidaten für den Fall einer finalen Ausgabe als Schlüsselwort gespeichert.

3.1.3 Klassifizierung

Im zweiten Schritt werden die Schlüsselwortkandidaten als Schlüsselwort ober Nicht-Schlüsselwort klassifiziert. Dazu werden für jeden Kandidaten die Merkmale *tf.idf* und *first occurrence* ermittelt. Diese dienen als Attribut für die Klassifizierung. Zur Klassifizierung wird ein *naiver Bayes Klassifizierer* verwendet.

Naiver Bayes Klassifizierer

Die folgenden Erläuterungen basieren auf Folienskripten von Benno Stein [Stein, 2005]. Das maschinelle Lernverfahren zur Vorhersage und Klassifikation von Daten beruht auf der Formel von Bayes für die bedingte Wahrscheinlichkeit.

Es seien $\langle \Omega, P(\Omega), P \rangle$ ein Wahrscheinlichkeitsraum und $A, B \in P(\Omega)$ zwei Ereignisse. Die Formel für die bedingte Wahrscheinlichkeit lautet:

$$P(B_i|A) = \frac{P(B_i) \cdot P(A|B_i)}{P(A)}$$

Die Wahrscheinlichkeit $P(B_i|A)$ wird als a-posteriori-Wahrscheinlichkeit bezeichnet. Ist $P(B_i|A_1, ..., A_p)$ die Wahrscheinlichkeit für das Eintreten von Ereignis B_i unter der Bedingung, dass die Ereignisse $A_1, ..., A_p$ gemeinsam eingetreten sind, lautet die Erweiterung auf kombinierte Ereignisse:

$$P(B_i|A_1, \dots, A_p) = \frac{P(B_i) \cdot P(A_1, \dots, A_p|B_i)}{P(A_1, \dots, A_p)}$$

Unter der Annahme (Naive-Bayes-Assumption) der stochastischen Unabhängigkeit der Ereignisse $A_1, ..., A_p$ gilt:

$$P(B_i|A_1, \dots, A_p) = \prod_{j=1}^{p} P(A_j|B_i)$$

Da die Wahrscheinlichkeit $P(B_i|A_1, ..., A_p)$ eine Konstante ist, braucht sie nicht bekannt zu sein, um das unter der Naive-Bayes-Assumption wahrscheinlichste Ereignis $B_{NB} \in \{B_1, ..., B_k\}$ zu bestimmen:

$$\underset{B_i \in \{B_1, \ldots, B_k\}}{argmax} \frac{P(Bi) \cdot P(A_1, \ldots, A_p|Bi)}{P(A_1, \ldots, A_p)} \overset{NB}{=} \underset{B_i \in \{B_1, \ldots, B_k\}}{argmax} P(B_i) \cdot \prod_{j=1}^{p} P(A_j|B_j) = B_{NB}$$

Das „Lernen“ bei diesem Klassifizierer besteht in der Ermittlung der a-priori-Wahrscheinlichkeiten $P(B_i)$ und der Wahrscheinlichkeiten für Kausalzusammenhänge $P(A_j|B_i)$ auf der Basis von Trainingsdaten. Die ermittelten Wahrscheinlichkeiten entsprechen der gelernten Hypothese, die zur Klassifikation neuer Beispiele mittels der Optimierungsformel für B_{NB} verwendet wird.

Zur vereinfachten Verwendung des Klassifizierers werden die Werte (Ereignisse) für die Attribute *tf.idf* und *first occurrence* diskretisiert. Dazu wird eine während des Trainings erstellte Diskretisierungstabelle verwendet. Anhand eines Beispiels wird das Training des Klassifizierers im Folgenden näher erläutert.

Trainingsbeispiel

Für das Training wird eine Menge von Trainingsdokumenten $L \subset D$ verwendet, die bereits vom Autor mit Schlüsselwörtern ausgezeichnet wurden. Für jedes Dokument werden die Schlüsselwortkandidaten ermittelt und die beiden Merkmale *tf.idf* und *first occurrence* berechnet. Die Schlüsselwortkandidaten werden anschließend anhand der tatsächlichen Schlüsselwörter als Schlüsselwort oder Nicht-Schlüsselwort markiert. Dieses binäre Merkmal ist das Klassifizierungsmerkmal für den maschinellen Lernalgorithmus.

Tabelle 2 zeigt beispielhaft die Ergebnisse des Trainings. Die Diskretisierung erfolgt durch ein Verfahren zur Multi-Intervall-Diskretisierung kontinuierlicher Attribute [Fayyad & Irani, 1993]. Dabei wird der Wertebereich jedes Merkmals in mehrere disjunkte Teilbereiche unterteilt. Die Werte des Attributs werden dann durch die Nummer des jeweiligen Teilbereichs ersetzt. Die Diskretisierungstabelle im Beispiel besitzt fünf Bereiche für das Attribut *tf.idf* und in vier Bereiche für das Attribut *first occurrence*. Unter Verwendung dieser Diskretisierung werden die Wahrscheinlichkeiten

für Kausalzusammenhänge ermittelt. Es entstehen je neun Wahrscheinlichkeiten für positive Klassifizierung (Schlüsselwort) und für negative Klassifizierung (Nicht-Schlüsselwort). Hat ein Schlüsselwortkandidat beispielsweise einen *tf.idf*-Wert von 0.0023, befindet er sich im *tf.idf*-Bereich 1 und ist dann mit einer Wahrscheinlichkeit von 0.2826 ein Schlüsselwort.

Zuletzt werden die a-priori-Wahrscheinlichkeiten für beide Klassen ermittelt. Damit ist das Training des Klassifizierers abgeschlossen.

Diskretisierungstabelle

Merkmal	Diskretisierungsbereiche				
	1	2	3	4	5
tf.idf	<0.0031	[0.0031,0.0045)	[0.0045,0.013)	[0.013,0.033)	≥0.033
fo	<0.0014	[0.0014,0.017)	[0.017,0.081)	≥0.081	

Wahrscheinlichkeiten für Kausalzusammenhänge $P(A_j|B_i)$

Merkmal	Wahrscheinlichkeit	Diskretisierungsbereiche				
		1	2	3	4	5
tf.idf	$P_{tf.idf}[k' \mid yes]$	0.2826	0.1002	0.2986	0.1984	0.1182
tf.idf	$P_{tf.idf}[k' \mid no]$	0.8609	0.0548	0.0667	0.0140	0.0036
fo	$P_{fo}[k' \mid yes]$	0.1952	0.3360	0.2515	0.2173	
fo	$P_{fo}[k' \mid no]$	0.0194	0.0759	0.1789	0.7333	

a-priori-Wahrscheinlichkeiten $P(B_i)$

Klasse	Trainingsinstanzen	Wahrscheinlichkeit
yes	$Y = 493$	$P_{yes} = \frac{Y}{Y+N} = 0.0044$
no	$N = 112183$	$P_{no} = \frac{N}{Y+N} = 0.9956$

Tabelle 2: exemplarisches Daten nach dem Training des Klassifizierers

Sollen nun neue Daten klassifiziert werden, wird die Klasse vorhergesagt, der sie mit der höchsten Wahrscheinlichkeit angehören.

Klassifizierung neuer Schlüsselwörter

Die Klassifizierung für jeden Schlüsselwortkandidat k' erfolgt durch die Berechnung der folgenden a-posteriori-Wahrscheinlichkeiten:

$$P_{yes}(k') = P_{yes} \cdot P_{tf.idf}[k'|yes] \cdot P_{fo}[k'|yes]$$

und

$$P_{no}(k') = P_{no} \cdot P_{tf*idf}[k'|no] \cdot P_{fo}[k'|no] .$$

Damit ist die Klassifizierung der Schlüsselwortkandidaten abgeschlossen.

3.1.4 Gewichtung und Rückgabe der Schlüsselwörter

Im dritten und letzten Schritt wird das Gewicht $p(k')$ für einen Schlüsselwortkandidaten durch folgende Formel berechnet:

$$p(k') = \frac{P_{yes}(k')}{P_{yes}(k') + P_{no}(k')} .$$

Die Schlüsselwortkandidaten werden anschließend nach dem Gewicht in absteigender Reihenfolge sortiert. Haben zwei oder mehr Kandidaten das gleiche Gewicht, so wird der tatsächliche, nicht diskretisierte *tf.idf*-Wert als Ordnungskriterium verwendet. Abschließend werden alle Terme entfernt, die Teil eines Terms mit höherer Wahrscheinlichkeit sind. Aus der verbleibenden Liste werden die ersten *r* Terme in ihrem ursprünglichen Text ausgegeben, wobei *r* die Anzahl der geforderten Schlüsselwörter ist.

KEA wurde von Ian Witten und Kollegen auf dem *Computer Science Technical Reports (CSTR)* Korpus der NZDL getestet. Dazu wurden zufällig 1300 Trainingsdokumente und 500 Testdokumente ausgewählt. Im Durchschnitt konnten zwischen 10 und 20% der vom Autor vergebenen Schlüsselwörter extrahiert werden.

3.2 RSP

Im Gegensatz zum korpusbasierten KEA-Algorithmus ist für das Verfahren der so genannten *Repeated-String-Patterns* (RSP) [Tseng, 1998] zur Extraktion von Schlüsselwörtern ein einzelnes Dokument ausreichend (dokumentbasiert). Ausgehend

von der Annahme, dass ein Dokument zu einem bestimmten Thema eine Menge von Termen in einer spezifischen Reihenfolge mehrfach beinhaltet, werden häufig auftretende Wortkombinationen als Schlüsselwortkandidaten bestimmt.

Diese Wortkombinationen kann man als n-Gramm betrachten. Ein n-Gramm ist eine geordnete Menge mit n Elementen. Der von Tseng entwickelte Algorithmus zur Detektion mehrfach vorkommender n-Gramme der Terme t eines Dokuments d besteht aus drei Schritten. Zuerst wird eine geordnete Liste $L=\{(t_i, t_{i+1}) \mid t_i, t_{i+1} \in d\}$ aus sich jeweils mit einem Term t_i überlappenden n-Grammen $G_i=(t_i, t_{i+1})$ erzeugt. Zu jedem n-Gramm G_i wird die Auftrittshäufigkeit $h_i = \#\{j \mid \exists\, G_j \in L : G_i = G_j\}$ ermittelt. Durch Hinzufügen von h_i zu jedem G_i wird die Liste L in eine geordnete Liste von Tupeln $LIST=\{(G_i, h_i) \mid G_i \in L\}$ überführt.

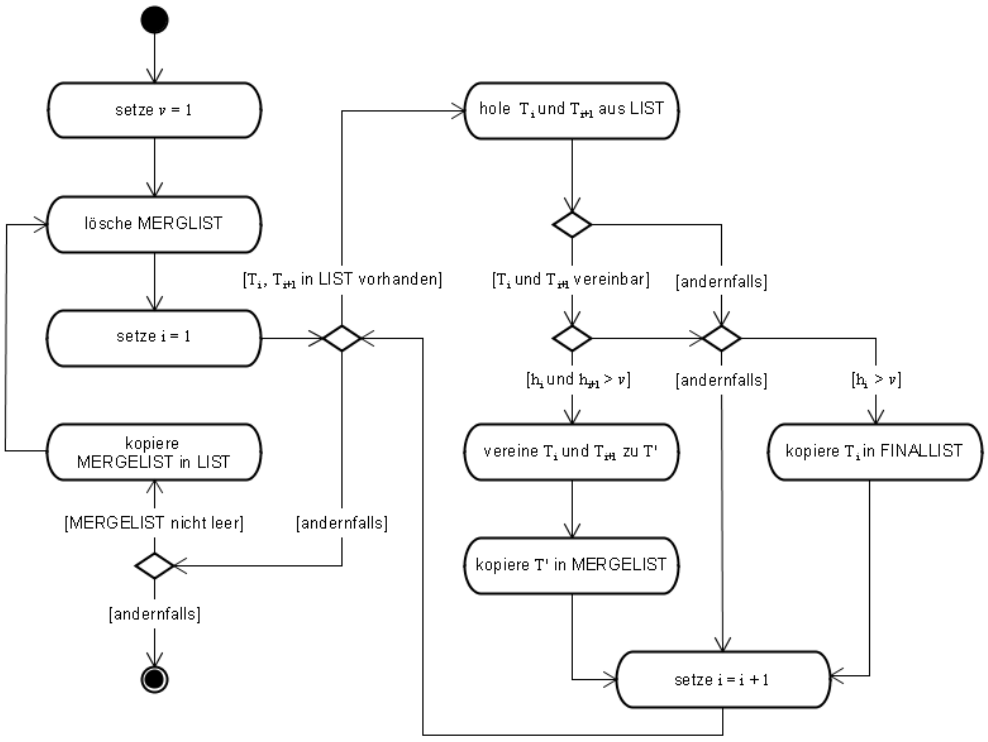

Abbildung 2: Ermittlung mehrfach vorkommender n-Gramme

Im zweiten Schritt werden mehrfach vorkommende n-Gramme ermittelt. Abbildung 3 zeigt das entsprechende UML-Aktivitätsdiagramm. Je zwei aufeinander folgende Tupel $T_i=(G_i, h_i)$ und $T_{i+1}=(G_j, h_j)$ werden miteinander verglichen. Sind die Auftrittshäufigkeiten h_i und h_j größer als eine definierte Relevanzschwelle v wird

geprüft, ob G_i und G_j zum n-Gramm G' vereinigt werden können. Zwei n-Gramme G_i und G_j können zu G' vereinigt werden, wenn gilt: $(t_{i2}, ..., t_{i\,n}) = (t_{j1}, ..., t_{i\,n-1})$. Nach erfolgreicher Vereinigung zum Tupel $T' = (G', h')$ gebildet und in eine neue Liste (*MERGELIST*) übernommen. Ist dies nicht der Fall, erfolgt eine weitere Prüfung. Wurde $h_i > v$ und G_i nicht mit dem vorangestellten n-Gramm $G_k \in T_{i-1}$ vereinigt, wird das Tupel T_i in einer weiteren Liste (*FINALLIST*) als Schlüsselwortkandidat abgelegt. Dieser Schritt wird iteriert bis keine Vereinigung der n-Gramme der in der *MERGELIST* verbleibenden Tupel mehr möglich ist.

Ein Tupel wird aufgrund seiner geringen Auftrittsfrequenz oder durch Vereinigung mit anderen Tupeln eliminiert. Abbildung 2 zeigt den Ablauf dieser Schleife und die Vereinigung an einem Beispiel. Jeder Buchstabe repräsentiert dabei einen Term.

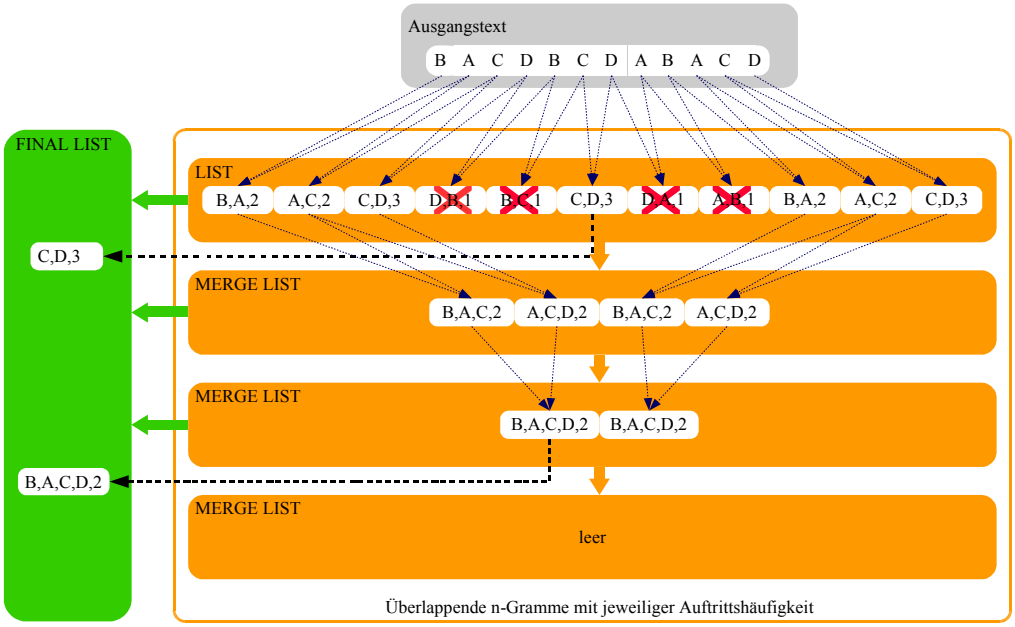

Abbildung 3: Beispiel für die Extraktion häufiger Termfolgen mit v=1.

Die Relevanzschwelle *v* ist der einzige Parameter des Algorithmus. Sie kann dynamisch in Abhängigkeit der Textlänge erzeugt werden.

Im nächsten Schritt werden die in der *FINALLIST* enthaltenen Tupel zur Ausgabe aufbereitet, d.h. Stoppwörter werden rekursiv vom Anfang und Ende jedes n-Gramms entfernt. Abschließend wird die Liste nach nicht genannten und daher im Rahmen dieser Arbeit noch zu bestimmenden Ranking-Kriterien sortiert und die ersten *r* Terme

ausgegeben, wobei *r die* Anzahl der geforderten Schlüsselwörter ist. Diese Ranking-Kriterien können beispielsweise die Position des ersten Auftretens oder die Auftrittshäufigkeit der Schlüsselwortkandidaten sein.

Über die Testergebnisse des Algorithmus wurde von Tseng nicht berichtet. Die Leistungsfähigkeit des Extraktionsalgorithmus ist im Folgenden noch zu ermitteln. Sie wird maßgeblich durch die richtige Auswahl der Ranking-Kriterien bestimmt.

3.3 B&C

B&C [Barker & Cornacchia, 2000] ist ein weiteres Extraktionsverfahren auf der Basis eines einzelnen Dokumentes. Der Algorithmus extrahiert ausschließlich Wortgruppen als Schlüsselwörter (Schlüsselphrasen) auf der Grundlage von *Noun Phrase Heads*. Als Noun Phrase Head bezeichnet man im Englischen das bestimmende Substantiv (*Head Noun*) einer Nominalphrase. Als Nominalphrase (*Noun Phrase*) wird hier eine Wortgruppe deklariert, die aus einem Substantiv, dem *Head Noun*, besteht und durch ein oder mehrere Substantive und/oder Adjektive näher beschrieben wird. Der Algorithmus ermittelt Nominalphrasen, die in Abhängigkeit ihrer Länge und Frequenz sowie der Frequenz des Substantivs als Schlüsselphrasen zurückgegeben werden.

Der Algorithmus besteht aus drei Abschnitten:

- Ermittlung der Nominalphrasen als Schlüsselwortkandidaten
- Bewertung bezüglich der Länge und Frequenz
- Entfernung von Rauschen und Ausgabe der Schlüsselwörter

Für die Ermittlung der Nominalphrasen wird ein Parser verwendet, der das Dokument wortweise nach Sequenzen aus Adjektiven und Substantiven durchsucht. Die gesuchten Sequenzen müssen mit einem Substantiv enden. Die Autoren benutzen dazu eine schnelle Heuristik durch einfaches Nachschlagen in Wörterbüchern (*DIPETT's dictionary* und *Collins Wordlist*). Um die Verwendung des Algorithmus unabhängig von einer Online-Verbindung zu ermöglichen und außerdem die Genauigkeit zu verbessern, wurde in der Implementierung im Rahmen dieser Arbeit diese Heuristik durch einen sehr schnellen Part-of-Speech-Tagger (PoS-Tagger) ersetzt. Dies ist ein Algorithmus, der jedem Wort in einem Satz ein so genanntes Part-of-Speech-Tag, also

eine grammatikalische Markierung, zuordnet. Um die so erhaltenen Strukturinformationen nutzen zu können, wird eine Menge von Markierern (Tags) für verschiedene Satzteile benutzt. In Tabelle 3 sind die für die Erkennung von Nominalphrasen relevanten Tags des in der Implementation verwendeten PoS-Taggers „Qtag“ gelistet.

Part-of-Speech-Tag	Beschreibung	Beispiel
NN	noun, common singular	action
NNS	noun, common plural	actions
NP	noun, proper singular	Thailand, Thatcher
NPS	noun, proper plural	Americas, Atwells
JJ	adjective, general	near
JJR	adjective, comparative	nearer
JJS	adjective, superlative	nearest

Tabelle 3: Part-of-Speech-Tags von Qtag

Bei der Bewertung der Schlüsselwortkandidaten ist zu beachten, dass längere Nominalphrasen zwar spezifisch und damit relevant für das Dokument sein können aber selten in einem Dokument wiederholt werden. So wird beispielsweise in einem Artikel über die *German Barbecue Association* diese vollständige Nominalphrase wahrscheinlich nur einmal vorkommen, jedoch die folgenden Referenzen *Barbecue Association* oder *Association* mehrmals. Ausgehend von diesen Vorbetrachtungen beschreiben Barker und Cornacchia folgenden Algorithmus für B&C:

Die absolute Häufigkeit (Frequenz) des Auftretens eines *Head Nouns h* im Dokument *d* wird als *freq*(*h*) bezeichnet. $H'=\{h'_1,...,h'_n\}$ sei die Menge der *Head Nouns* mit der höchsten Frequenz $freq(h)$. Für jedes Substantiv $h' \in H'$ werden alle Nominalphrasen $B=\{b_1,...,b_m\}$, $h' \subseteq b_i$ mit h' als *Head Noun* ermittelt. Das Bewertungskriterium für jedes $b_k \in B$ wird berechnet als:

$$score(b_k) = freq(h'_k) \cdot length(b_k),$$

mit $length(b_k)$ als Anzahl der Wörter in b_k.

Abschließend werden alle vollständig enthaltenen Nominalphrasen entfernt und die *r* Phrasen mit der höchsten Bewertung als Schlüsselwörter ausgegeben.

Der *B&C*-Algorithmus wurde von Barker und Cornacchia im Vergleich zu Peter Turney's *Extractor* [Turney, 1999] getestet. Für dreizehn Dokumente wurde jeweils eine Menge von Schlüsselwörtern mit *B&C* und eine Menge mit *Extractor* erzeugt. Zwölf Testpersonen bewerteten für jedes Dokument jeweils die beiden Schlüsselwortmengen hinsichtlich der Relevanz in ihrer Gesamtheit. In 47% der Fälle wurden die von B&C erzeugten Schlüsselwörter als relevanter bewertet, *Extractor* erzeugte zu 39% bessere Schlüsselwörter. Bei 13% wurden die von beiden Verfahren erzeugten Mengen als nicht relevant beurteilt.

3.4 Cooccurrence

Das von Matsuo und Ishizuka beschriebene Verfahren [Matsuo & Ishizuka, 2003] ist ebenfalls dokumentbasiert. Der Ansatz beruht auf der Ermittlung von Kookkurrenzen und deren Verteilungseigenschaften im Text. Als Kookkurrenz bezeichnet man das gemeinsame Auftreten zweier Terme in einem Dokument unter der Annahme, dass diese voneinander abhängig sind, wenn sie häufig gemeinsam in Dokumenten auftreten. Die Verteilungseigenschaften dieser Terme werden unter Verwendung des Chi-Quadrat-Anpassungstests untersucht.

Der Algorithmus besteht aus den sechs Schritten: Textvorverarbeitung, Auswahl der häufigsten Terme, Gruppierung ähnlicher Terme, Berechnung der erwarteten Wahrscheinlichkeit, Berechnung der χ^2-Werte und abschließende Ausgabe der Terme mit dem höchsten χ^2-Wert.

Ein Dokument besteht aus Titel, Überschriften und Sätzen. Jeder Bestandteil wird als ein Container für Terme betrachtet. Dabei werden Reihenfolge der Terme und grammatikalische Information ignoriert. T sei die Menge der unterschiedlichen Terme und T' die Menge der häufigsten Terme. Treten zwei Terme in einem Container mehrmals gemeinsam auf, wird für den weiteren Verlauf angenommen, dass sie nur einmal gemeinsam in diesem Container auftreten.

Die Vorverarbeitung des Textes besteht aus einer Stammformreduktion nach Porter, der lexikalischen Analyse zur Extraktion von Termen mit bis zu vier Wörtern und der abschließenden Stoppwortelimination. Danach werden die häufig auftretenden Terme durch Berechnung der Termfrequenz ermittelt. Tabelle 4 zeigt die häufigsten Terme am

Beispiel eines Textes von Alan Turing mit dem Titel „Computing Machinery and Intelligence“ [Turing, 1950]. Die folgenden Tabellen und Abbildungen beziehen sich auf den genannten Text.

Häufige Terme	Anzahl des Auftretens	Wahrscheinlichkeit
machine (a)	203	0,366
computer (b)	63	0,114
question (c)	44	0,079
digital (d)	44	0,079
answer (e)	39	0,070
game (f)	36	0,065
argument (g)	35	0,063
make (h)	33	0,059
state (i)	30	0,054
number (j)	28	0,050
Summe	**555**	**1,000**

Tabelle 4: Häufige Terme

Die Häufigkeit von paarweise gemeinsam auftretenden Termen (Kookkurrenzen) innerhalb eines Containers kann mittels einer symmetrischen $T \times T$-Matrix, der sog. Kookkurrenzmatrix, erfasst werden. Die Tabelle 5 zeigt in einem Ausschnitt die zehn häufigsten Terme und vier weitere Beispielterme.

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**...**	**u**	**v**	**w**	**x**
a	-	30	26	19	18	12	12	17	22	9	...	3	13	11	17
b	30	0	5	50	6	11	1	3	2	3	...	5	40	2	3
c	26	5	-	4	23	7	0	2	0	0	...	5	4	2	2
d	19	50	4	-	3	7	1	1	0	4	...	3	35	1	1
e	18	6	23	3	-	7	1	2	1	0	...	3	3	1	2
f	12	11	7	7	7	-	2	4	0	0	...	18	6	0	4
g	12	1	0	1	1	2	-	5	1	0	...	2	1	1	5
h	17	3	2	1	2	4	5	-	0	0	...	2	0	4	0
i	22	2	0	0	1	0	1	0	-	7	...	1	0	0	0
j	9	3	0	4	0	0	0	0	7	-	...	0	2	0	0
Summe	165	111	67	89	61	50	23	34	33	23	...	45	104	22	34

u: imitation, v: digital computer, w: kind, x: make

Tabelle 5: Kookkurrenzmatrix

Ausgehend von der $T \times T$-Kookkurrenzmatrix bestehen folgende Annahmen:

(1) Ist ein Term t unabhängig von einer Teilmenge $T' \subset T$, dann ist die Verteilung der Wahrscheinlichkeit des gemeinsamen Auftretens von t und T' ähnlich der Verteilung der Auftrittswahrscheinlichkeit von T' (siehe Abbildung 4).

(2) Steht ein Term t in einer semantischen Beziehung zu T', dann ist die Wahrscheinlichkeit des gemeinsamen Auftretens von t und T' höher als erwartet, d.h. die Verteilung ist abweichend (siehe Abbildung 5).

Ein Term mit den in (2) beschriebenen Abweichungen könnte eine wichtige Bedeutung für den Inhalt des Dokuments besitzen. Der Grad der Abweichung kann somit als Indikator für Schlüsselwortkandidaten genutzt werden.

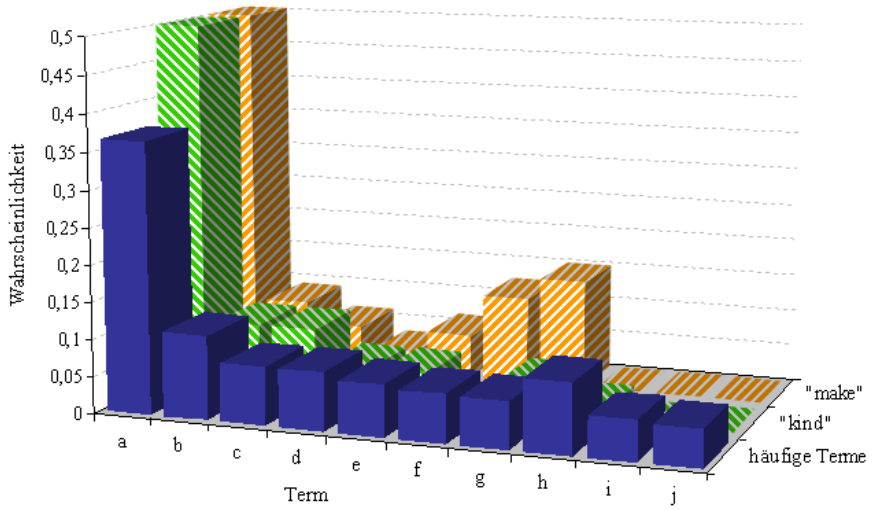

Abbildung 4: Kookkurrenzverteilung der Terme "kind" und "make"

Um die statistische Signifikanz dieser Abweichung zu ermitteln, benutzen die Autoren den Chi-Quadrat-Anpassungstest zum Testen der Verteilungseigenschaften einer statistischen Stichprobe. Der Test wird hier zur Auswertung der Abweichung zwischen erwarteten und ermittelten Häufigkeiten aus der Kookkurrenzmatrix benutzt. d.h. als Index für Abweichungen.

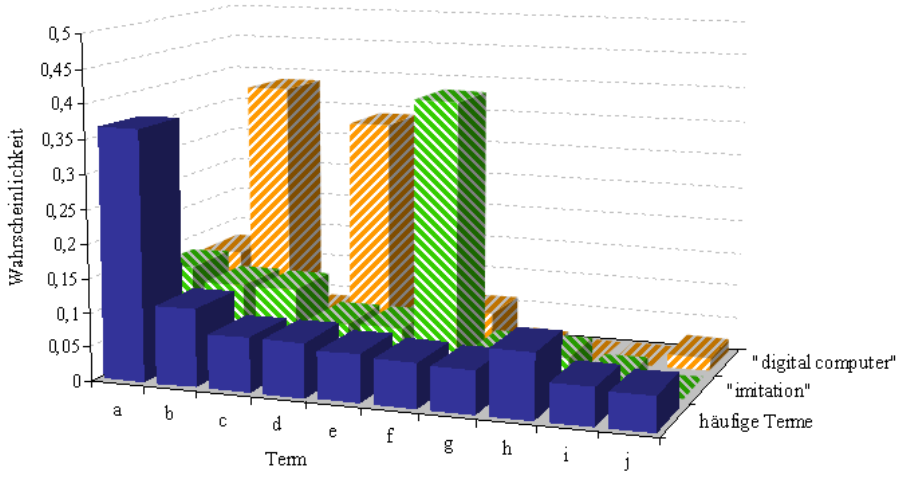

Abbildung 5: Kookkurrenzverteilung der Terme "imitation" und "digital computer"

Für jeden Term wird die Häufigkeit des gemeinsamen Auftretens mit den häufigsten Termen als Stichprobe betrachtet. Die Hypothese lautet: das gemeinsame Auftreten der häufigen Terme T' ist unabhängig vom Auftreten eines Terms t.

Die Wahrscheinlichkeit des Auftretens eines häufigen Terms $t' \in T'$ wird als erwartete Wahrscheinlichkeit $p_{t'}$ bezeichnet, n_t sei die absolute Anzahl des gemeinsamen Auftretens des Terms t mit den häufigen Termen T'. Die Häufigkeit des gemeinsamen Auftretens von t und t' wird bezeichnet als $f(t,t')$. Damit ist der statistische Wert von $\chi^2(t)$ wie folgt definiert:

$$\chi^2(t) = \sum_{t' \in T'} \frac{(f(t,t') - n_t p_{t'})^2}{n_t p_{t'}}$$

Der Term $n_t p_{t'}$ repräsentiert die erwartete Häufigkeit des gemeinsamen Auftretens, $f(t,t') - n_t p_{t'}$ stellt die Differenz zwischen gemessenen und erwarteten Häufigkeiten dar. Folglich bedeutet ein hoher Wert für $\chi^2(t)$ eine starke Abweichung.

Ausgehend von den bisherigen grundlegenden Betrachtungen wird die Berechnung von $\chi^2(t)$ durch folgende Erkenntnisse weiter modifiziert:

(1) In einem Dokument existieren Sätze mit unterschiedlicher Länge. Erscheint ein Term in einem langen Satz (Container), ist sein gemeinsames Auftreten mit anderen Termen wahrscheinlicher als in einem kurzen Satz. Unter Beachtung der Satzlänge werden folgende Modifikationen von $p_{t'}$ und n_t eingeführt:

- $p_{t'}$ ist die Summe aller Terme der Sätze in denen t' vorkommt dividiert durch die Anzahl aller Terme im Dokument.
- n_t ist die Anzahl aller Terme der Sätze in denen t vorkommt.

(2) Tritt ein Term t häufig gemeinsam mit einem bestimmten Term $t' \in T'$ auf, besitzt er einen hohen χ^2-Wert. Handelt es sich bei t um ein Attribut, so ist t nicht zwingend wichtig für den Inhalt des Textes.

Rang	χ^2	Term	Frequenz
1	593.7	Digital computer	31
2	179.3	Imitation game	16
3	163.1	future	4
4	161.3	question	44
5	152.8	internal	3
6	143.5	answer	39
7	142.8	Input signal	3
8	137.7	moment	2
9	130.7	play	8
10	123.0	output	15
...	...	...	...
558	0.6	scan	2
559	0.3	worse	2
560	0.1	eye	2

Tabelle 6: Terme mit hohem χ^2-Wert

In Tabelle 6 treten die Terme „future“ und „internal“ häufig gemeinsam mit „state“ auf („future state“ und „internal state“). Daher besitzen „future“ und „internal“ einen hohen χ^2-Wert, sind aber selbst nicht wichtig. Ein Term mit maximalem Wert wird daher von der χ^2-Berechnung subtrahiert:

$$\chi'^2(t) = \chi^2(t) - \max_{t' \in T'} \left\{ \frac{(freg(t, t') - n_t p_{t'})^2}{n_t p_{t'}} \right\}$$

Durch diese Modifikation ergibt sich ein niedriger Wert für $\chi^2(t)$, falls t explizit nur mit einem Term $t' \in T'$ gemeinsam auftritt. Tritt t aber mit mehr als einem Term $t' \in T'$ gemeinsam auf, ist $\chi^2(t)$ hoch.

Das Verfahren wird durch Clusterung von Termen weiter fortgeführt. Diese Ausführungen sind jedoch unklar formuliert und wurden in der Implementation nicht berücksichtigt.

Der Algorithmus wurde von Matsuo und Ishizuka anhand 20 technischer Dokumente getestet. Aus jedem Dokument wurden 15 Schlüsselwörter ermittelt und anschließend den Autoren zur Begutachtung vorgelegt. Ungefähr 50% der extrahierten Schlüsselwörter wurden als wichtig bewertet. Weiterhin haben die Autoren fünf unverzichtbare Schlüsselwörter im Vorfeld benannt. Im Durchschnitt wurden 60% davon gefunden. Dieses Ergebnis erscheint zunächst besser als das Ergebnis, das KEA im Test lieferte. Es ist jedoch zu beachten, dass der geringe Umfang des Testkorpus nicht unbedingt einen Schluss auf die tatsächliche Güte des Algorithmus zulässt.

4. Implementierung

Dieses Kapitel beschreibt die Implementierung einer Java-Bibliothek zur automatischen Extraktion von Schlüsselwörtern aus Text. Die Implementierung basiert auf der Grundlage des neuen *Java Development Kit (JDK) 1.5*. Im Folgenden werden die wichtigsten Details der Implementierung beschrieben.

4.1 Die Java-Bibliothek aitools.keywordextraction

Die Bibliothek ist in die in Abbildung 6 dargestellte Package-Struktur untergliedert.

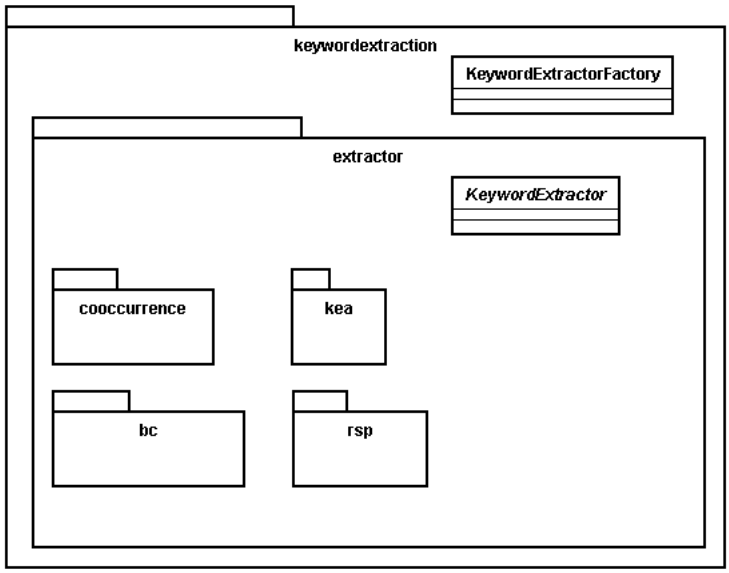

Abbildung 6: Package-Struktur der Java-Bibliothek aitools.keywordextraction

Im äußeren Package befindet sich lediglich eine Fabrikklasse. Jedes Verfahren ist in einem eigenen Package verpackt. Diese sind zum Package *extractor* zusammengefasst, in dem sich ebenfalls eine abstrakte Klasse als Basis befindet.

Die im Rahmen dieser Arbeit betrachteten Verfahren sind algorithmisch sehr unterschiedlich. Doch sie vereint eine wesentliche Gemeinsamkeit: Es wird ein Text übergeben und der Algorithmus liefert mehrere aus diesem Text stammende Wörter als Schlüsselwörter zurück. Es liegt daher nahe, eine einheitliche Schnittstelle für diese Eigenschaft zu definieren. In Java kann man dafür ein Interface oder eine abstrakte

Klasse benutzen. Die Entscheidung zugunsten der hier gewählten abstrakten Klasse beruht auf der Existenz einer Vielzahl gemeinsam genutzter Methoden, die innerhalb dieser Klasse implementiert werden können. Die Abbildung 7 zeigt die wichtigsten Klassen der Bibliothek in einem UML-Klassendiagramm.

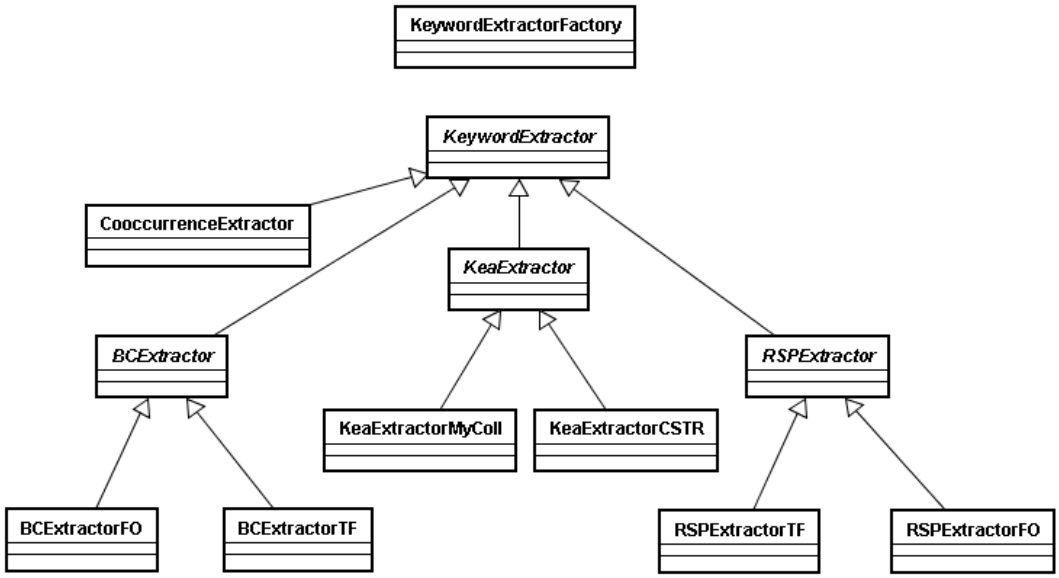

Abbildung 7: Klassendiagramm der Java-Bibliothek aitools.keywordextraction

Die abstrakte Klasse *KeywordExtractor* stellt die bereits erwähnte Schnittstelle zu anderen Anwendungen dar. Die Klasse in Abbildung 8 stellt die folgenden öffentlichen Methoden zur Verfügung:

- *getSingleWords(text : String, numberOfKeywords : int) : Vector<String>* - Gibt die extrahierten Schlüsselwörter als Vektor von einzelnen Wörtern zurück.
- *getPhrases(text : String, numberOfKeywords : int) : Vector<String>* - Extrahiert ein Algorithmus neben einzelnen Wörtern auch Wortgruppen, können die Wortgruppen mittels dieser Methode als Vektor zurückgegeben werden.
- *getName() : String* - Gibt den Name des zu Grunde liegenden Extraktions-algorithmus zurück.

- *setLanguage(language : String) : void* - Ist das Dokument in einer anderen Sprache als Englisch verfasst, muss bei sprachabhängigen Algorithmen – zur Anpassung von z.B. Stemmer oder Stoppwortlisten – die Sprache mit übergeben werden. (Funktioniert nicht bei KEA und B&C)

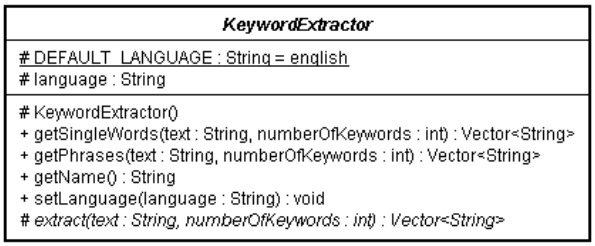

Abbildung 8: Klassendiagramm KeywordExtractor

Die abstrakte Methode *extract(text : String, numberOfKeywords : int) : Vector<String>* ist von jeder abgeleiteten Klasse zu implementieren und sollte den Algorithmus enthalten.

Weiterhin wurde eine Fabrikklasse eingeführt, welche die Erzeugung eines KeywordExtractors übernimmt. Das Klassendiagramm in Abbildung 9 zeigt die Abhängigkeiten der Fabrikklasse *KeywordExtractorFactory*.

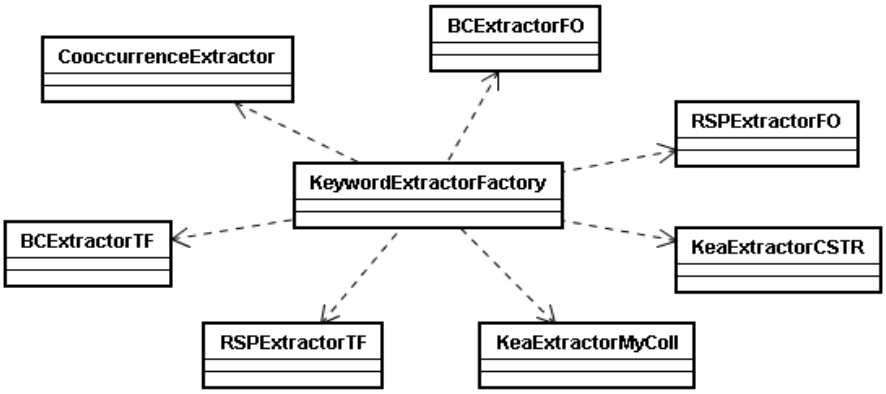

Abbildung 9: Fabrikklasse zur Erzeugung eines KeywordExtractors

Mittels dieser Klasse kann jeder implementierte KeywordExtractor instantiiert werden. Dazu verfügt die Fabrikklasse über die in Abbildung 10 dargestellten statistischen Methoden. Die Klasse selbst kann nicht instantiiert werden.

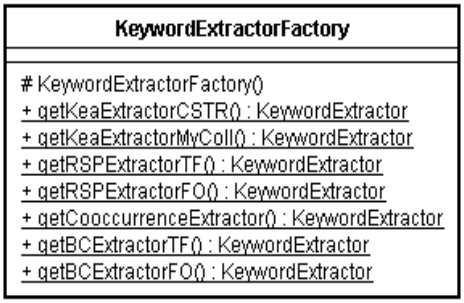

Abbildung 10: Klassendiagramm KeywordExtractorFactory

Die Algorithmen wurden im Wesentlichen nach den in Kapitel 3 erläuterten Verfahren implementiert. Einige wichtige Details zur Implementierung der Algorithmen sowie die Struktur der einzelnen Packages werden im Folgenden beschrieben.

4.2 KeaExtractor

Obwohl auch ein jar-Archiv von KEA[4] zur Verfügung gestellt wird, ist der Quellcode vollständig in die Bibliothek integriert. Dies ist in der Tatsache begründet, dass *KEA* auf der Basis von JDK 1.4 entwickelt wurde und für Enumerationen häufig der Begriff *enum* als Bezeichner verwendet wurde. Seit Java 1.5 ist *enum* jedoch ein Datentyp. Dies führt in der Folge zu einem Compilerfehler. Der Quellcode wurde daraufhin entsprechend korrigiert.

KEA wird als Konsolenanwendung ausgeliefert, die ausschließlich Textdateien als Eingabe verarbeiten kann. Die bereits beschriebene Schnittstelle sieht jedoch die Übergabe des Textes als String vor. Aus diesem Grund wurde eine Wrapper-Klasse implementiert, die den übergebenen String in eine temporäre Textdatei umwandelt und nach der Extraktion wieder löscht.

Da der Algorithmus für die Evaluation auf zwei unterschiedlichen Korpora trainiert wird, wurden zwei separate Klassen implementiert.

4 http://www.nzdl.org/Kea

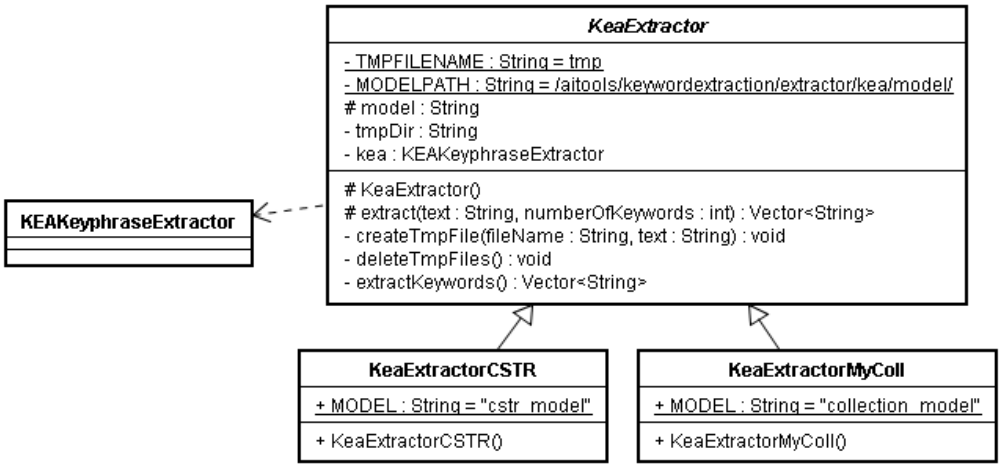

Abbildung 11: Klassendiagramm für das Package kea

Abbildung 11 zeigt die abstrakte Wrapper-Klasse *KeaExtractor* und die beiden abgeleiteten Klassen *KeaExtractorCSTR* und *KeaExtractorMyColl* für die entsprechend trainierten Modelle. Das Training des KeaExtractorMyColl basiert auf Grundlage von 80 zufällig ausgewählten Trainingsdokumenten des im Rahmen dieser Arbeit erstellten Evaluierungskorpus. Der KeaExtractorCSTR wurde auf dem in der KEA-Distribution enthaltenen Trainingskorpus trainiert. Die Evaluierung dieser auf zwei unterschiedlichen Trainingskorpora basierten Varianten soll die Abhängigkeit von den Trainingsdaten quantifizieren.

4.3 RSPExtractor

Wie bereits beim B&C-Algorithmus werden auch beim RSPExtractor zwei unterschiedliche Ranking-Kriterien evaluiert. Dabei handelt es sich wiederum um Frequenz (TF) und erstes Auftreten (FO) der Schlüsselwortkandidaten. Das Klassendiagramm in Abbildung 12 zeigt die entsprechende Implementierung des Package *rsp*.

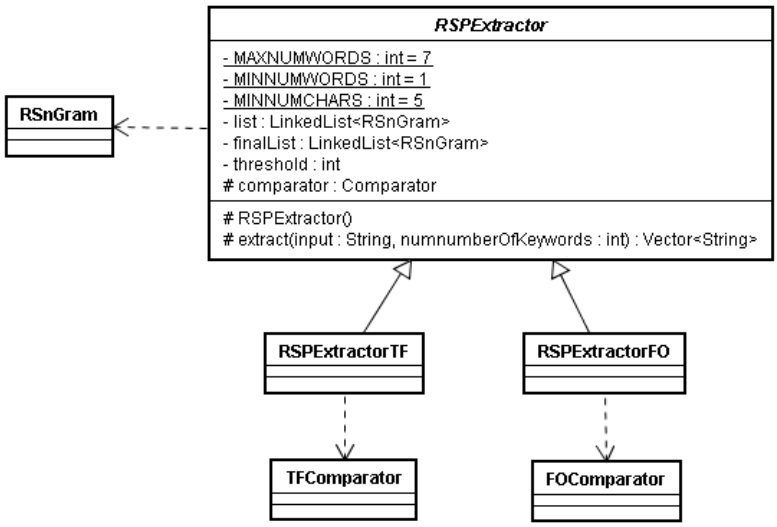

Abbildung 12: Klassendiagramm für das Package rsp

4.4 BCExtractor

Wie bereits in Kapitel 3.3 vermerkt, wurde in der hier durchgeführten Implementierung des BCExtractor zur Ermittlung der Wortarten ein Part-of-Speech Tagger (PoS-Tagger) anstelle eines Zugriffs auf die von den Autoren verwiesenen Wörterbücher benutzt. Bei dem eingesetzten freien PoS-Tagger *Qtag*[5] handelt es sich um einen probabilistischen Algorithmus, also auf der Grundlage statistischer Berechnungen. Die Nutzung dieses Taggers ist wesentlich schneller als der Zugriff auf eine Online-Ressource. Die Präzision erweist sich für die Verwendung im Extraktionsalgorithmus als ausreichend.

Das Klassendiagramm in Abbildung 13 zeigt wiederum eine abstrakte und zwei davon abgeleitete Klassen. Die Klassen *BCExtractorFO* und *BCExtractorTF* wurden erzeugt, weil zum Ranking der Schlüsselwortkandidaten zwei unterschiedliche Verfahren evaluiert werden. Der *BCExtractorTF* verwendet zur Ermittlung der Rangfolge das in der Veröffentlichung beschriebene Ranking-Verfahren (Frequenz des Head Nouns multipliziert mit der Wortlänge der längsten Nominalphrase). Im Gegensatz dazu wird beim *BCExtractorFO* als Ranking-Kriterium das erste Auftreten des Head Houns benutzt.

5 http://www.english.bham.ac.uk/staff/omason/software/qtag.html

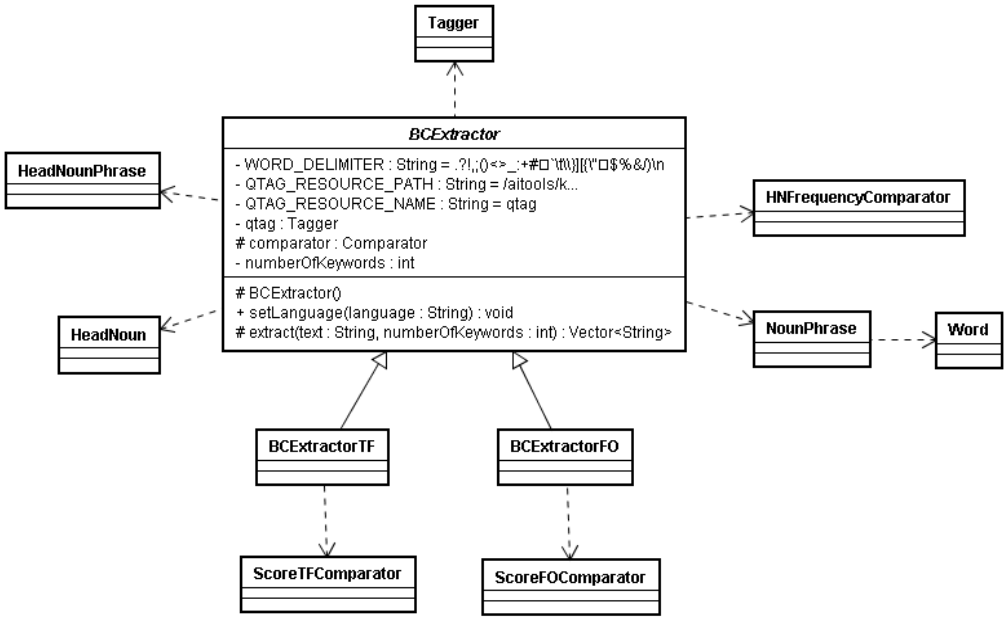

Abbildung 13: Klassendiagramm für das Package bc

4.5 CooccurrenceExtractor

Die Package-Struktur des CooccurrenceExtractors ist in Abbildung 14 zu sehen. Hier wurde auf die Implementierung der Clusterung verzichtet, da die Beschreibung des Verfahrens und damit die beschriebene Verbesserung der Extraktionsergebnisse nicht nachvollziehbar war.

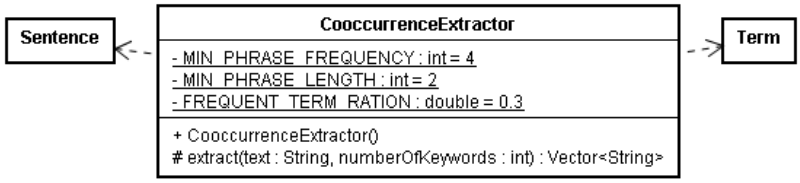

Abbildung 14: Klassendiagramm für das Package cooccurrence

Für die detaillierte Beschreibung der Implementierung der einzelnen Algorithmen sei an dieser Stelle auf die Javadoc-Dokumentation verwiesen.

5. Evaluierung

Nachdem die Implementation der Extraktionsverfahren beschrieben ist, befasst sich dieser Abschnitt mit der Evaluierung der Verfahren. In der Literatur wird allgemein über plausible Bewertungsmaßstäbe diskutiert. Dabei sind im wesentlichen zwei unterschiedliche Ansätze zu verzeichnen. Während eine Gruppe von Wissenschaftlern die nachträgliche manuelle Bewertung extrahierter Schlüsselwörter durch die Autoren favorisiert [Barker & Cornacchia, 2000] und [Matsuo & Ishizuka, 2003], wird auf der anderen Seite die Gegenüberstellung mit a-priori vom Autor vergebenen Schlüsselwörtern als Quasi-Standard betrachtet [Witten et al., 1999]. In einer Evaluierungsstudie zu KEA [Jones & Paynter, 2001] stellen Jones und Paynter die beiden Ansätze gegenüber. Zuerst ließen sie die extrahierten Terme durch eine Gruppe von Probanden manuell bewerten und sortieren. Anschließend wurden diese manuellen Bewertungen mit den vom Autor vergebenen Schlüsselwörtern verglichen. Es zeigte sich, dass diese Schlüsselwörter allgemein gute Schlüsselwörter sind. Auf Grundlage dieser Studie und nicht zuletzt, weil ein derartiger Vergleich mit bekannten Schlüsselwörtern einfacher zu realisieren ist, wird die Evaluierung der Algorithmen auf der Grundlage eines automatisch erzeugten Evaluierungskorpus mit gegebenen Schlüsselwörtern durchgeführt. Der Aufbau des Test-Frameworks sowie die automatisierte Erstellung des Evaluierungskorpus wird im Folgenden beschrieben. Anschließend werden die auf der Grundlage dieses Korpus durchgeführten Experimente diskutiert.

5.1 Der Evaluierungskorpus

Als Korpus bezeichnet man im Information Retrieval eine Sammlung von Dokumenten, die über ein bestimmtes gemeinsames Merkmal verfügen. Ein Korpus zum Evaluieren von Algorithmen zur automatischen Schlüsselwortextraktion besteht demnach idealerweise aus vollständigen Dokumenten, die bereits vom Autor mit Schlüsselwörtern ausgezeichnet wurden. Diese werden auch als tatsächliche Schlüsselwörter bezeichnet. Im Zuge der Recherche zu dieser Arbeit konnte kein für diesen Zweck geeigneter existierender Evaluierungskorpus ermittelt werden. Die zu wissenschaftlichen Zwecken frei verfügbaren Textkorpora bestehen zumeist aus einer

Vielzahl sehr kurzer Dokumente, wie Kurzbeschreibungen (abstracts) wissenschaftlicher Veröffentlichungen oder Nachrichtenmeldungen und sind häufig nicht mit Schlüsselwörtern versehen[6]. Den Anforderungen für eine hinreichende Untersuchung der Extraktionsalgorithmen werden sie somit nicht gerecht. Aus diesem Grund wurde für die Bewertung der implementierten Verfahren ein eigens für diese Studie erstellter Evaluierungskorpus verwendet.

5.1.1 Aufbau

Der Korpus besteht derzeit aus 250 wissenschaftlichen Veröffentlichungen. Die Dokumente sind frei zugänglich und wurden über den Online-Suchdienst *CiteSeer* [7] bezogen. Die Quelldokumente liegen als PDF-Datei vor. Zur automatischen Verarbeitung wurden sie in XML konvertiert. Die Schlüsselwörter wurden vom Text separiert. Des Weiteren wurden die Kurzbeschreibungen durch entsprechende XML-Tags ausgezeichnet. Abbildung 15 zeigt exemplarisch das Ergebnis dieser Umwandlung.

6 Eine der wichtigsten Quellen frei verfügbarer Testkorpora ist die TREC *Text REtrieval Conference* (http://trec.nist.gov/).

7 http://citeseer.ist.psu.edu

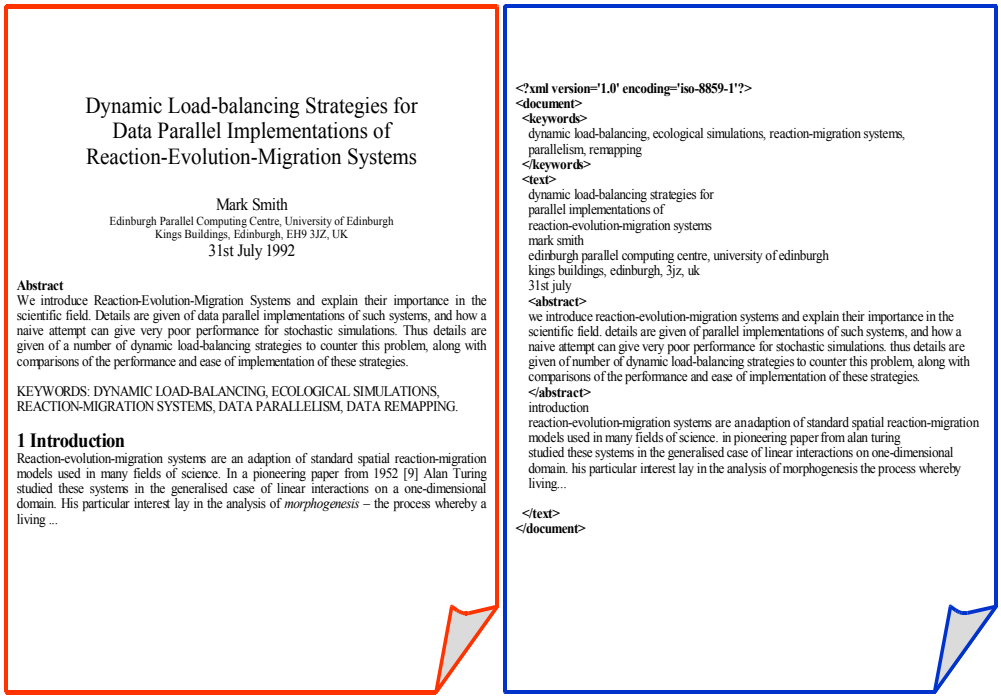

Abbildung 15: Original (PDF) und automatisch erzeugtes XML-Dokument

Da KEA im Rahmen dieser Arbeit auf diesem Korpus neu trainiert werden sollte, wurde der Korpus zunächst in ein Drittel Trainingsdokumente für KEA (Trainingskorpus) und zwei Drittel Testdokumente (Testkorpus) zufällig unterteilt. Es stehen 80 Trainings- und 170 Testdokumente für die Evaluierung zur Verfügung. Folgende statistische Daten wurden für den Testkorpus ermittelt:

Durchschnittliche Schlüsselwortanzahl:	10,5
Maximale Schlüsselwortanzahl:	29
Minimale Schlüsselwortanzahl:	2
Durchschnittlicher Anteil der Schlüsselwörter im gesamten Text:	97%
Durchschnittlicher Anteil der Schlüsselwörter in der Kurzbeschreibung:	65%
Durchschnittliche Wortzahl im gesamten Text:	6150
Durchschnittliche Wortzahl in der Kurzbeschreibung:	120

Tabelle 7: Aufbau des Testkorpus

Die Verteilung der Anzahl der Schlüsselwörter pro Dokument im Testkorpus ist in Abbildung 16 dargestellt.

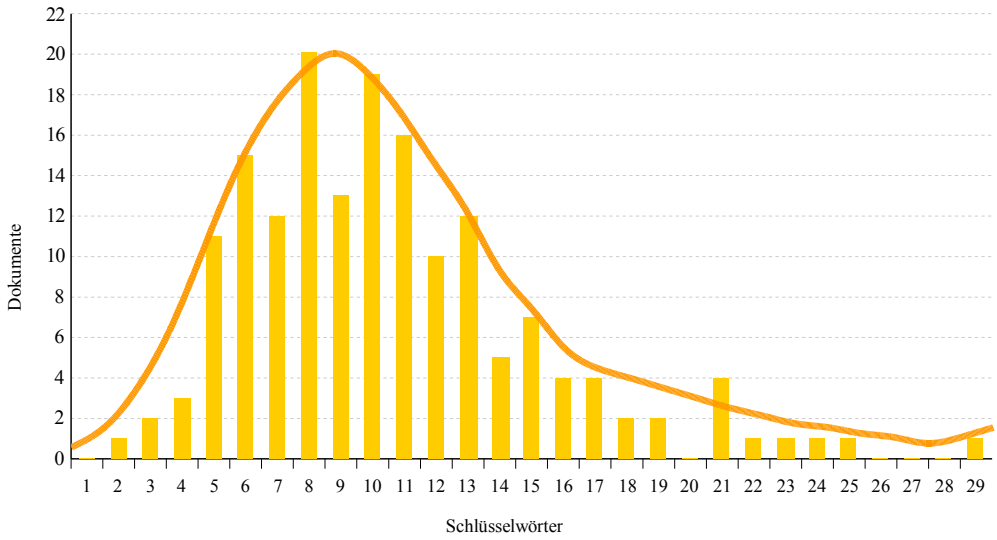

Abbildung 16: annähernde Normalverteilung der Schlüsselwörter pro Dokument

Forschungen [Turney, 1997] haben gezeigt, dass bei der Vergabe von Schlüsselwörtern durch die Autoren durchschnittlich 80% dieser Schlüsselwörter tatsächlich im Dokument enthalten sind. Der im Rahmen dieser Arbeit erzeugte Evaluierungskorpus weist jedoch einen durchschnittlichen Anteil von 97% der im Text enthaltenen Schlüsselwörter auf. Der Korpus eignet sich damit außerordentlich gut für die gewählte Evaluierungsmethode, nämlich den Vergleich der extrahierten Wörter mit den vom Autor vergebenen, da die zu betrachtenden Algorithmen nur im Text enthaltene Wörter ermitteln können. Bei der Bewertung der Evaluierungsergebnisse ist zu beachten, dass der durchschnittliche Recall das Maximum von 1 nicht erreichen kann.

5.1.2 Erstellung

Da eine manuelle Erstellung eines umfangreichen Korpus extrem zeitaufwändig ist, entstand die Java-Anwendung „*CopusBuilder*“, mit deren Hilfe der Evaluierungskorpus automatisiert erstellt wurde und erweitert werden kann.

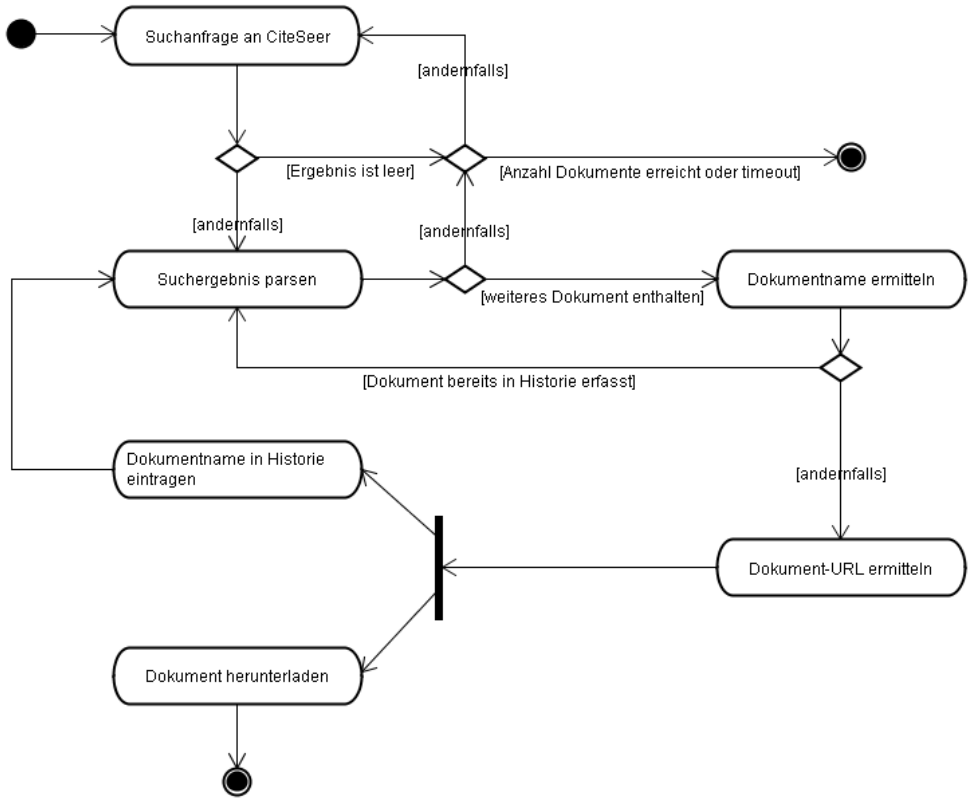

Abbildung 17: Aktivitätsdiagramm für die Dokumentbeschaffung

Die Anwendung besitzt einen zweistufigen Aufbau. Das Aktivitätsdiagramm in Abbildung 17 zeigt den Ablauf der ersten Stufe. Hier werden die Quelldokumente beschafft. Dazu wird eine Anfrage an den Online-Suchdienst *CiteSeer* beispielsweise mit dem Suchbegriff „keywords“ gesendet. Als Antwort wird eine mehrseitige Ergebnisliste zurückgegeben, die anschließend nach enthaltenen Dokument-URLs durchsucht wird. Gefundene URLs werden einer Warteschlange (queue) hinzugefügt. Ein Download-Manager arbeitet die Warteschlange ab und speichert die heruntergeladenen PDF-Dokumente in einem Verzeichnis. Weiterhin wird eine Historie in Form einer Liste erzeugt, um mehrfache Downloads zu vermeiden. Somit kann das Programm zu einem späteren Zeitpunkt wieder gestartet werden.

In der zweiten Stufe werden die PDF-Dokumente in XML-Dokumente konvertiert. Dazu werden die PDF-Dateien zunächst als Text eingelesen. Ist dies nicht möglich, werden sie gelöscht. Ein *XMLStreamWriter* erzeugt aus jedem erfolgreich konvertierten PDF-Dokument eine XML-Datei. Können die bereits vergebenen Schlüsselwörter nicht korrekt aus dem Text ermittelt werden oder ergeben sich anderweitige Probleme

während der Konvertierung, wird ein Hinweis in einer Logdatei erzeugt. In Abbildung 18 ist das entsprechende Aktivitätsdiagramm dargestellt. In diesem Fall müssen die in der Logdatei gelisteten Dokumente manuell korrigiert werden.

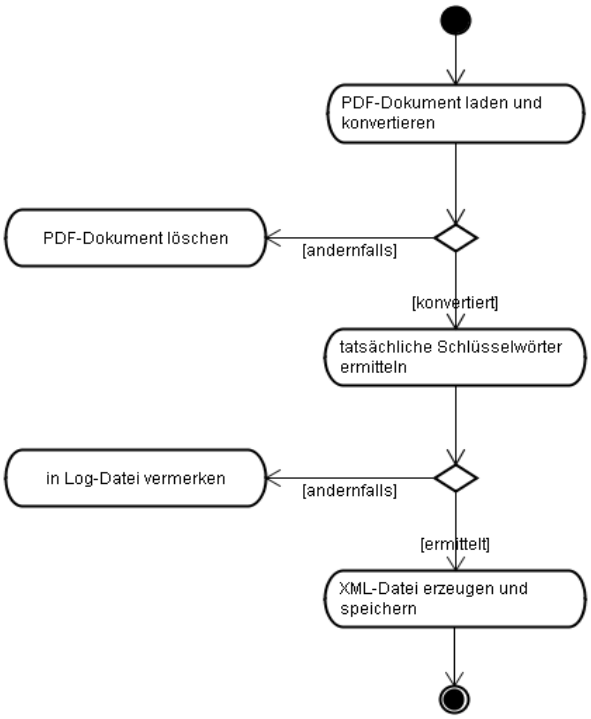

Abbildung 18: Aktivitätsdiagramm für die Konvertierung in XML

Während in der erste Stufe des CorpusBuilders die Ermittlung der Dokument-URLs und der Download der PDF-Dokumente durch Threads parallel verlaufen, arbeitet die zweite Stufe die Dokumente sequentiell ab. Die Anwendung wird mit folgenden Parametern gestartet:

```
java
aitools.keywordextraction.evaluation.collection.CorpusBuilder.java
-d absPathToPDF         -x absPathToXML         -n numberOfXmlDocuments
-s "queryString"
```

Nach der Erzeugung des Evaluierungskorpus mittels des CorpusBuilders kann dieser zur Evaluierung der Extraktionsalgorithmen eingesetzt werden.

5.2 Aufbau des Test-Frameworks

Zur automatisierten Evaluierung der implementierten Bibliothek wurde neben dem Evaluierungskorpus ein Test-Framework entwickelt. Folgende Anforderungen wurden dabei berücksichtigt:

(1) Alle Dokumente in einem Verzeichnis sollen seriell verarbeiten werden.

(2) Der Verzeichnispfad ist als Parameter zu übergeben.

(3) Die zu testenden Extraktionsalgorithmen sind ebenfalls als Parameter zu übergeben.

(4) Es sollen beliebig viele Extraktionsalgorithmen in einem Testdurchlauf berücksichtigt werden.

(5) Die Anzahl der zu ermittelnden Schlüsselwörter soll als Parameter zu übergeben sein.

(6) Die Anzahl der zu extrahierenden Schlüsselwörter soll in einem Intervall mit flexiblem Schrittmaß möglich sein.

(7) Es soll zwischen Extraktion aus dem gesamten Text (langer Text) und Extraktion aus der Kurzbeschreibung (kurzer Text) ausgewählt werden können.

(8) Für jedes Dokument soll eine detaillierte Zusammenfassung der Ergebnisse erstellt werden können (ausführlicher Report).

(9) Es gibt eine kurze Zusammenfassung über alle Dokumente im Verzeichnis (kurzer Report).

(10) Es kann eine Statistik über alle Dokumente eines Verzeichnisses erzeugt werden.

(11) Precision, Recall und F-Measure können sowohl für jedes Dokument als auch für das gesamte Verzeichnis ermittelt werden.

(12) Alle Reports werden als Textdatei von Typ *.csv erzeugt. Dies ermöglicht den einfachen Import in ein Auswertungsprogramm (z.B. OpenOffice Calc).

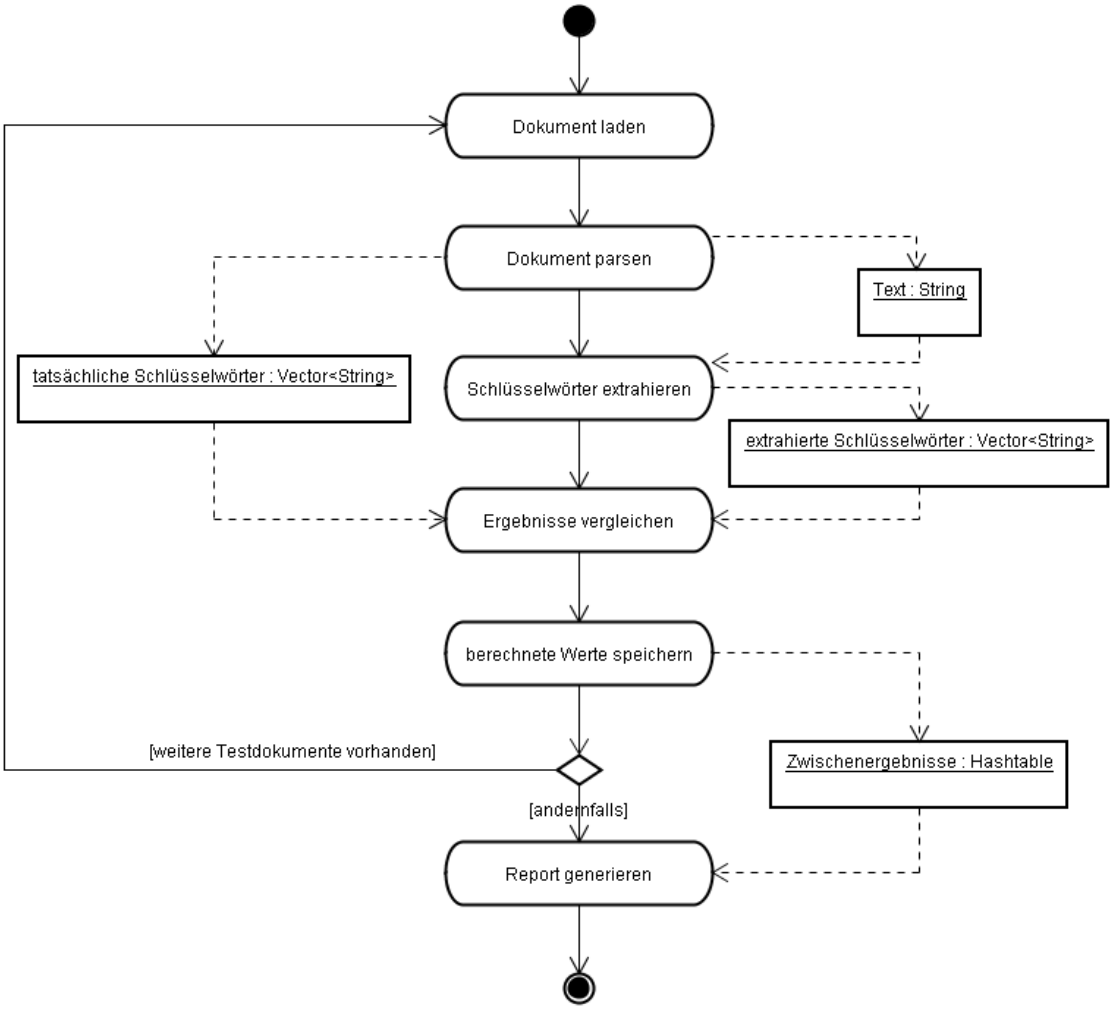

Abbildung 19: Aktivitätsdiagramm der Evaluation

Die Realisierung dieser Anforderungen wurde als Konsolenanwendung im Package aitools.keywordextraction.evaluation implementiert. Das Aktivitätsdiagramm in Abbildung 19 stellt den Ablauf der Evaluation dar. Die Anwendung *KeywordEvaluation* wird wie folgt aufgerufen:

```
java aitools.keywordextraction.evaluation.KeywordEvaluation.java
-p absPathToCollection -t task -nmax maxNumberOfKeywords
-nmin minNumberOfKeywords -step stepInIntervall [-context] [-rsptf]
[-rspfo] [-keacstr] [-keamy] [-cooc] [-bctf] [-bcfo] -tags tags
[-longRep]
```

Die Parameter bedeuten im Einzelnen:

- `-p`: absoluter Pfad zum zu testenden Verzeichnis
- `-t`: „cs“ zur Erzeugung einer Statistik, „test“ zum Testen der Algorithmen
- `-nmax`: maximale Anzahl der zu extrahierenden Schlüsselwörter
- `-nmin`: minimale Anzahl der zu extrahierenden Schlüsselwörter
- `-step`: Schrittweite im Intervall [nmin, nmax]
- `-context`: ContextExtractor verwenden
- `-rsptf`: RSPExtractorTF verwenden
- `-rspfo`: RSPExtractorFO verwenden
- `-keacstr`: KeaExtractorCSTR verwenden
- `-keamy`: KeaExtractorMyColl verwenden
- `-cooc`: CooccurrenceExtractor verwenden
- `-bctf`: BCExtractorTF verwenden
- `-bcfo`: BCExtractorFO verwenden
- `-tags`: Text, der für Extraktion verwendet werden soll (a = abstract, t = text)
- `-longRep`: ausführlichen Report erzeugen

Für eine detailliertere Beschreibung der Implementierung des Frameworks sei an dieser Stelle ebenfalls auf die Javadoc-Dokumentation verwiesen.

5.3 Bewertung von Retrieval-Ergebnissen

Die Beurteilung der Extraktionsergebnisse beruht auf der Tatsache, dass die „richtigen“ Schlüsselwörter bekannt sein müssen, um die Ergebnisse des Systems zu bewerten. Dazu werden die vom Autor vergebenen Schlüsselwörter mit den extrahierten Wörtern verglichen. Dafür verwendet man im Information Retrieval die Größen *Precision* und *Recall* und das daraus resultierende *F-Measure*. Precision beantwortet die Frage: „Wie genau ist die Antwort?”, Recall: „Wie vollständig ist die Antwort?“.

Gegeben sei ein Dokument D, die Termmenge $K=\{k_1, ..., k_n\}$ der extrahierten Schlüsselwörter sowie die Termmenge der relevanten (vom Autor vergebenen) Schlüsselwörter $K^*=\{k^*_1, ..., k^*_m\}$.

Für jeden extrahierten Term k_i eines Dokuments wird die vorhergesagte Klasse ($k_i \in K^*$) mit der tatsächlichen Klasse ($k_i \in K^*$ oder $k_i \notin K^*$) verglichen. Die Menge der korrekt klassifizierten Terme $K^+=\{k^+_1, ..., k^+_m\}$ sei $K^+=K \cap K^*$. Abbildung 20 zeigt schematisch das Szenario dieser Klassifikation.

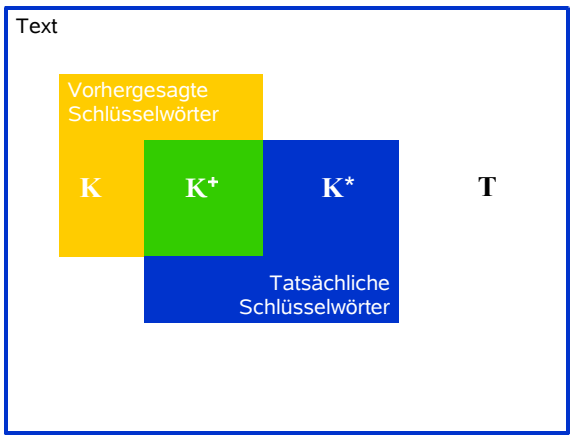

Abbildung 20: Klassifikationsszenario

Die aus diesen Mengen resultierenden Klassifikationsmöglichkeiten sind in Tabelle 8 dargestellt. Die Anzahl a der Elemente einer Menge A sei $a=|A|$.

	Schlüsselwörter	keine Schlüsselwörter
Extrahierte Terme	$\|K^+\|=\boldsymbol{a}$	$\|K \setminus K^*\|=\boldsymbol{b}$
Nicht extrahierte Terme	$\|K^* \setminus K\|=\boldsymbol{c}$	$\|T \setminus (K \cup K^*)\|=\boldsymbol{d}$

Tabelle 8: Möglichkeiten der Klassifikation

Die Precision berechnet sich dann durch:

$$precision=\frac{a}{a+b} \quad \in[0,1]$$

Sie spiegelt das Verhältnis der erhaltenen relevanten Schlüsselwörter im Bezug zur Gesamtanzahl der extrahierten Schlüsselwörter wieder. Ein hoher Wert bedeutet, die meisten erhaltenen Schlüsselwörter sind relevant.

Recall ist definiert als

$$recall = \frac{a}{a + c} \quad \in [0,1] .$$

und bezeichnet das Verhältnis zwischen der Anzahl der erhaltenen relevanten Schlüsselwörter und der Gesamtanzahl der relevanten Schlüsselwörter. Hoher Recall bedeutet, von allen relevanten Schlüsselwörtern wurden die meisten gefunden. Der so genannte Precision-Recall-Graph zeigt den Zusammenhang zwischen Precision und Recall. In diesem Graph wird auf der y-Achse die Precision und auf der x-Achse der Recall aufgetragen. So wird versucht, ein Bewertungsmaß zu schaffen, dass beide Größen mit einbezieht. Abbildung 21 zeigt den typischen Verlauf solch eines Graphen.

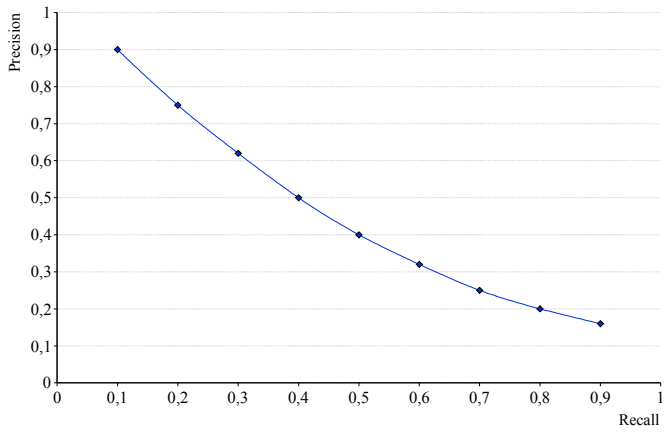

Abbildung 21: Precision-Recall-Graph

Ein Vergleichswert, der einfacher zu handhaben ist als separate Recall- und Precision-Werte, ist das F-Measure [van Rijsbergen, 1979]. Es handelt sich hierbei um das harmonische Mittel aus Precision und Recall:

$$F\text{-}Measure = \frac{2 \cdot precision \cdot recall}{precision + recall} \quad \in [0,1]$$

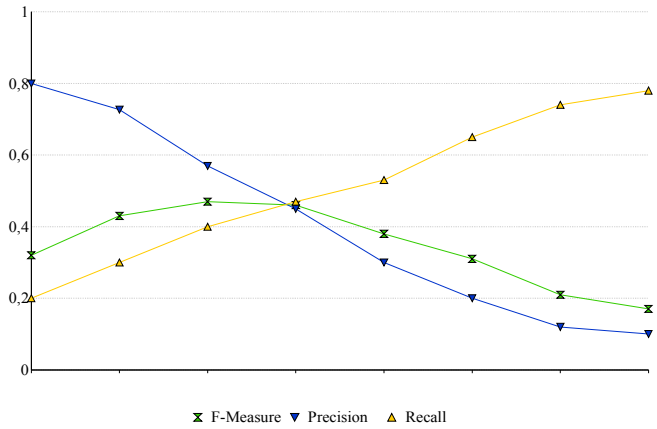

Abbildung 22: Beispiel für Precision, Recall und F-Measure

Abbildung 22 zeigt exemplarisch den Zusammenhang zwischen Precision, Recall und F-Measure. Je größer das F-Measure ist, desto besser sind die Retrieval-Ergebnisse des Systems. Die Optimierung von Algorithmen im Information Retrieval hat das Ziel, das F-Measure zu maximieren und damit sowohl Precision als auch Recall zu optimieren.

5.4 Experimente und Ergebnisse

Die implementierten Algorithmen wurden auf der Grundlage des erstellten Evaluierungskorpus mittels des entwickelten Test-Frameworks evaluiert. Das Training des KEA-Algorithmus basiert sowohl auf Grundlage der zufällig ausgewählten Trainingsdokumente des Evaluierungskorpus (KeaExtractorMyColl), als auch auf dem in der KEA-Distribution enthaltenen Trainingskorpus (KeaExtractorCSTR).

Die Experimente wurden jeweils für alle n Dokumente des Testkorpus durchgeführt. Dabei wurde zuerst der gesamte Text jedes Dokuments als Grundlage der Extraktion verwendet. Anschließend wurden die Tests mit der Kurzbeschreibung des Dokuments wiederholt, um den Einfluss der Textlänge und die daraus resultierenden Einsatzmöglichkeiten zu ermitteln. Für jeden Algorithmus wurden Precision P_i und Recall R_i pro Dokument d_i separat ermittelt. Die durchschnittlichen Werte für Precision P und Recall R pro Algorithmus berechnen sich dann nach den Formeln:

$$P = \frac{1}{n}\sum_{i=1}^{n} P_i \qquad\qquad R = \frac{1}{n}\sum_{i=1}^{n} R_i$$

Das F-Measure als harmonisches Mittel wird berechnet als

$$F = \frac{2 \cdot P \cdot R}{P + R}$$

Die im Folgenden beschriebenen Experimente werden in zwei Typen unterteilt:

- *variable Extraktion:* Extraktion einer festgelegten Anzahl *r* von Schlüsselwörtern auf einem definierten Intervall,
- *statische Extraktion*: Extraktion einer in Abhängigkeit vom zu Grunde liegenden Dokument gewählten Anzahl *r* von Schlüsselwörtern, wobei *r* der Anzahl *a* der jeweils vom Autor vergebenen Schlüsselwörter entspricht.

In beiden Experimenten wird die Qualität der Extraktionsergebnisse der verschiedenen Algorithmen untereinander verglichen. Weiterhin wird der Einfluss der Dokumentlänge auf die Extraktionsergebnisse untersucht.

Bei der Interpretation der Ergebnisse ist weiterhin zu beachten, dass der Anteil der Schlüsselwörter durchschnittlich 0,9% aller im Text enthaltenen Wörtern beträgt. Das bedeutet, dass die Wahrscheinlichkeit ein Schlüsselwort durch zufälliges Ziehen zu erhalten bei 0,001 liegt. Ein Verfahren, das Schlüsselwörter mit einer Wahrscheinlichkeit von 0,3 korrekt klassifiziert, verbessert das Ergebnis um das 300-fache.

5.4.1 Variable Extraktion

Unter Berücksichtigung der durchschnittlichen Anzahl der vom Autor vergebenen Schlüsselwörter pro Dokument und ihrer Verteilung auf dem Testkorpus wurde für die Anzahl *r* der pro Dokument zu ermittelnden Schlüsselwörter das Intervall $[4,18] := \{k \in \mathbb{N} \mid 4 \leq 2k \leq 18\}$ gewählt. Die folgenden Diagramme (Abbildungen 23, 24 und 25) stellen Precision, Recall und F-Measure der variablen Extraktion auf der Basis des gesamten Dokuments über dem Intervall *r* dar.

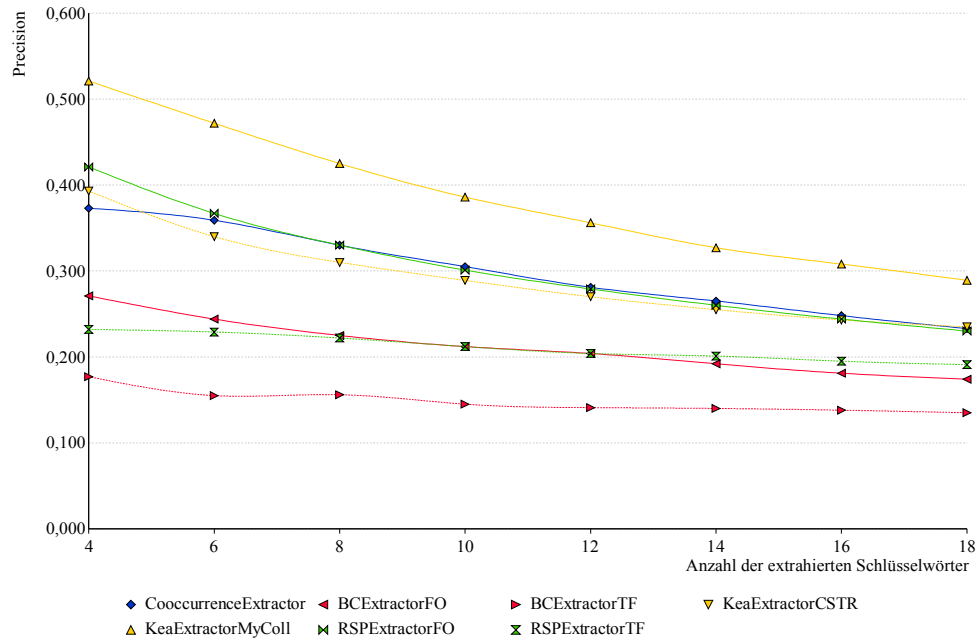

Abbildung 23: Precision für die variable Extraktion aus langem Text

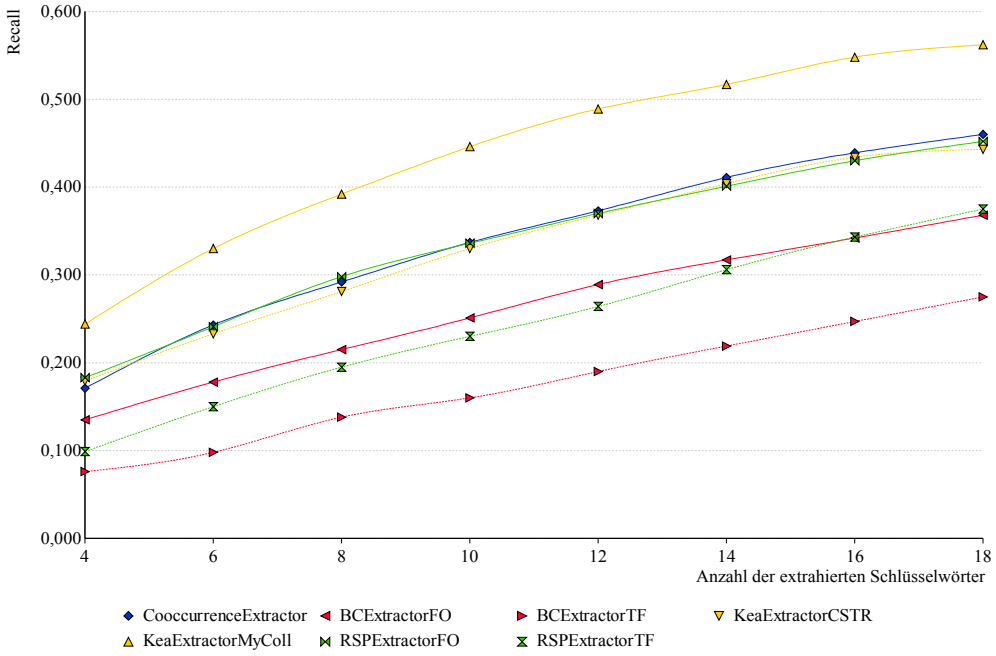

Abbildung 24: Recall für die variable Extraktion aus langem Text

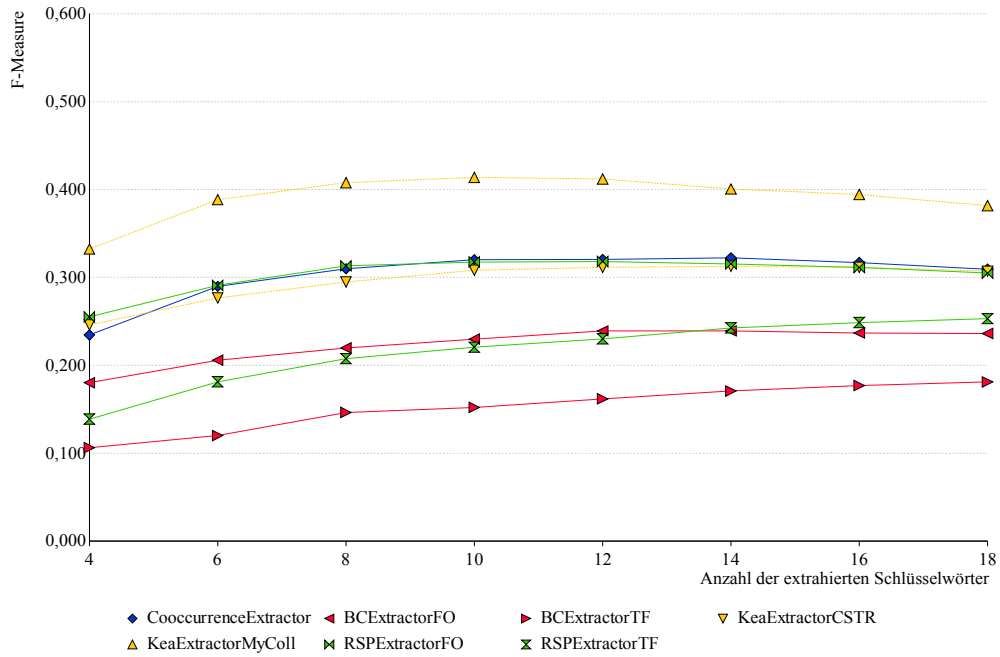

Abbildung 25: F-Measure für die variable Extraktion aus langem Text

Die Kurven zeigen eine ähnliche Performance von CoocurrenceExtractor, dem auf dem CSTR-Korpus trainierten KEA und dem Repeated-String-Pattern-Extractor mit *first occurrence* als Ranking-Kriterium. Das um ca. ein Drittel bessere Ergebnis für den auf dem Trainingskorpus trainierten KeaExtractorMyColl ist auf die Tatsache zurückzuführen, dass die Menge der Testdokumente und die Menge der Trainingsdokumente dem gleichen Wissensgebiet (Domain) angehören, der wissenschaftlichen Veröffentlichungen im Bereich der Informatik. Da der CSTR-Korpus auch in dieses Gebiet fällt, sind die Ergebnisse des KeaExtractorCSTR ebenfalls plausibel.

Der Unterschied resultiert aus der verschiedenen durchschnittlichen Dokumentlänge der beiden Trainingskorpora. Der CSTR-Korpus besteht aus 80 Kurzbeschreibungen technischer Berichte mit im Durchschnitt 120 Wörtern. Im Gegensatz dazu beinhaltet der neue Trainingskorpus 80 vollständige wissenschaftliche Veröffentlichungen. Die durchschnittliche Wortanzahl beträgt 6150 Wörter pro Dokument. Dies entspricht ebenfalls der Struktur des Testkorpus.

Weiterhin ist zu erkennen, dass die Version des RSP-Algorithmus mit *first occurrence* als Ranking-Merkmal wesentlich bessere Ergebnisse erzeugt, als durch die Verwendung der Termfrequenz realisiert werden kann. Gleiches gilt für die Modifikation des B&C-Algorithmus. Hier erreicht das alleinige Ranking mit *first occurrence* ebenfalls bessere Ergebnisse als das von Barker und Cornaccia beschriebene Verfahren. Dies lässt die Schlussfolgerung zu, dass das erste Auftreten eines Terms im Dokument ein starkes Kriterium für die Klassifizierung als Schlüsselwort darstellt.

Die Precision-Recall-Kurve in Abbildung 26 zeigt die jeweils beste Version der vier untersuchten Extraktionsalgorithmen für die Extraktion aus langen Texten.

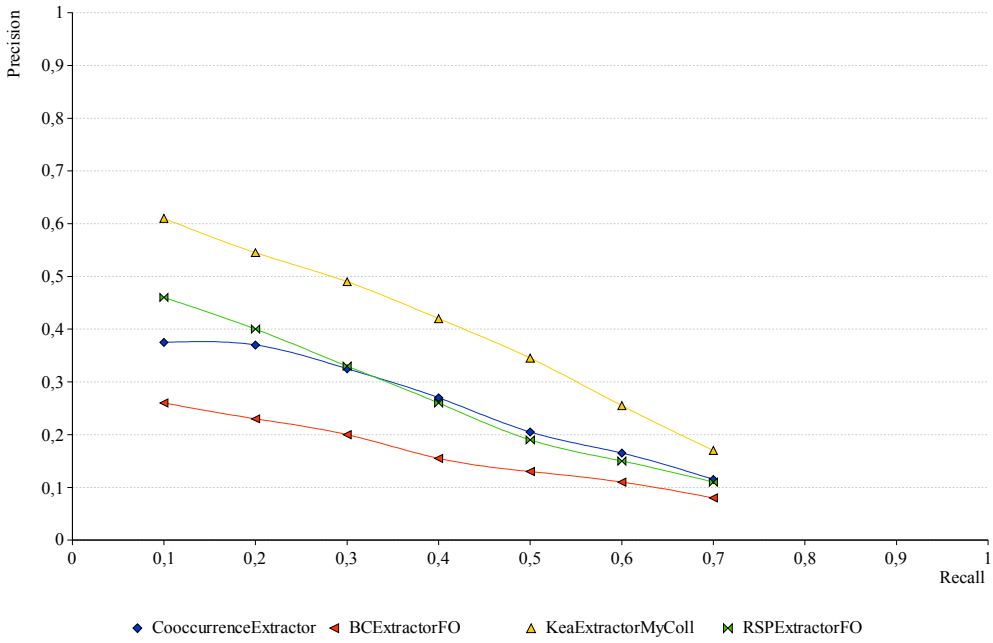

Abbildung 26: Precision-Recall-Kurve

Witten und Kollegen stellten bereits den Einfluss der Dokumentlänge auf die Extraktionsergebnisse von KEA fest. Die Ergebnisse für die Extraktion aus einem kompletten Dokument verbesserten sich gegenüber der Extraktion aus der entsprechenden Kurzbeschreibung um 80% [Witten et al., 1999]. Die Wiederholung der variablen Extraktion unter lediglicher Verwendung der Kurzbeschreibungen der

Dokumente des Testkorpus soll die Abhängigkeit aller implementierten Algorithmen von der Textlänge ermitteln. Die Resultate für kurze Texte sind in Abbildung 27 illustriert.

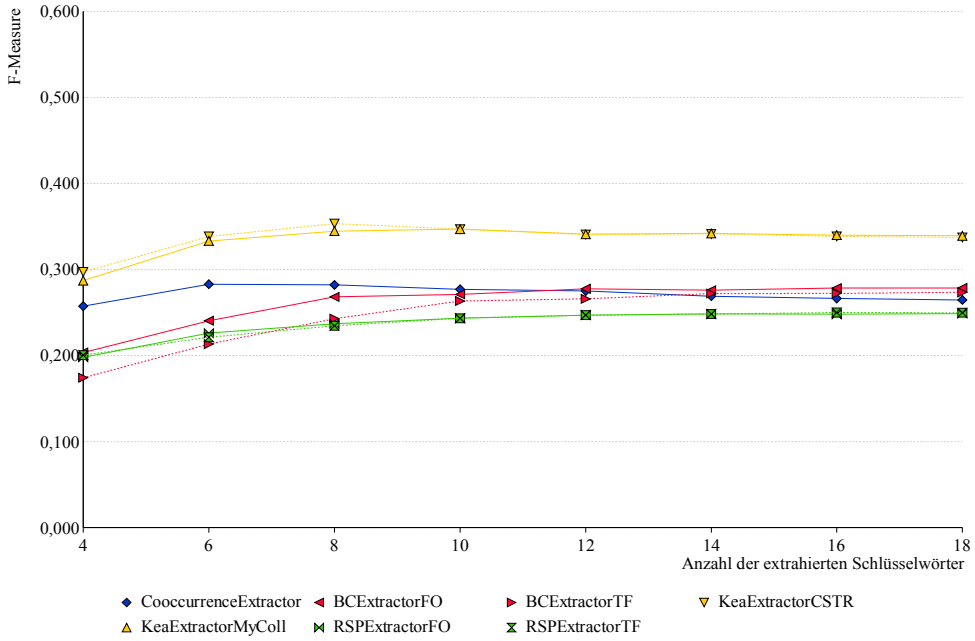

Abbildung 27: F-Measure für die variable Extraktion aus kurzem Text

Obwohl zu erwarten wäre, dass die Extraktionsergebnisse aufgrund der kürzeren Textlänge der Kurzbeschreibungen insgesamt schlechter ausfallen, trifft diese Annahme jedoch nicht für alle Algorithmen zu. Die Ergebnisse für kurzen Text weichen unterschiedlich stark von den Extraktionsergebnissen für langen Text ab und weisen teilweise sogar eine deutliche Verbesserung auf. Es ist daher eine differenzierte Betrachtung der einzelnen Algorithmen notwendig, die im Folgenden jeweils anhand der F-Measure-Diagramme durchgeführt wird.

KEA

Die Veränderung der Ergebnisse zeigt hier deutlich die allgemeine Abhängigkeit überwachter Lernverfahren von den jeweiligen Trainingsdaten. Während sich der auf kurzen Texten trainierte KeaExtractorCSTR durchschnittlich um 20 % verbesserte, ist eine etwa gleich große Verschlechterung der Ergebnisse des auf langen Texten trainierten KeaExtractorMyColl festzustellen.

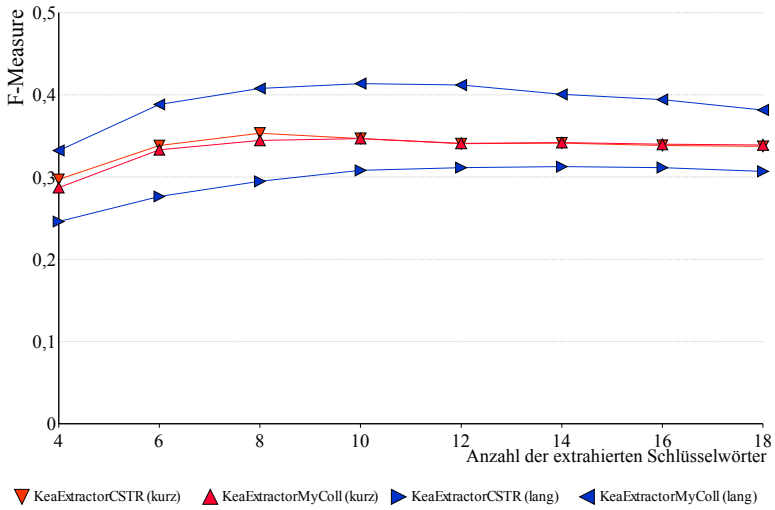

Abbildung 28: F-Measure KeaExtractor

Um eine eventuelle Abhängigkeit dieser Ergebnisse von der unterschiedlichen Trainingsgrundlage[8] auszuschließen, wurde der Algorithmus nochmals auf der Grundlage der Kurzbeschreibungen des Trainingskorpus trainiert (KeaExtractorShort).

	Anzahl extrahierter Schlüsselwörter							
	4	**6**	**8**	**10**	**12**	**14**	**16**	**18**
Precision								
KeaExtractorCSTR	0,467	0,423	0,386	0,351	0,329	0,320	0,312	0,308
KeaExtractorShort	0,472	0,425	0,387	0,352	0,331	0,322	0,314	0,310
Recall								
KeaExtractorCSTR	0,218	0,282	0,326	0,343	0,353	0,366	0,370	0,373
KeaExtractorShort	0,221	0,285	0,328	0,346	0,357	0,371	0,375	0,379
F-Measure								
KeaExtractorCSTR	0,297	0,338	0,353	0,347	0,341	0,341	0,339	0,337
KeaExtractorShort	0,301	0,341	0,355	0,349	0,344	0,345	0,342	0,341

Tabelle 9: Ergebnisse für Extraktion aus kurzem Text

8 Obwohl die CSTR-Abstracts ebenfalls Dokumente aus dem Bereich der Informatik sind, könnte sich durch unterschiedliche Wahl der Schlüsselwörter im Vergleich zum Trainingskorpus eine negative Beeinflussung der Extraktionsergebnisse einstellen.

Die Ergebnisse für den Vergleich zwischen KeaExtractorCSTR und KeaExtractorShort sind in der Tabelle 9 dargestellt. Die Werte weisen eine vernachlässigbare Abweichung der Extraktionsergebnisse auf. Eine eventuelle Beeinflussung durch die unterschiedlichen Trainingsdaten ist somit nicht gegeben.

RSPExtractor

Die Extraktion der Schlüsselwörter beruht beim RSPExtractor auf der Ermittlung häufig vorkommender n-Gramme. Die Wahrscheinlichkeit für das mehrfache Auftreten bestimmter Wortkombinationen ist für kurze Texte wesentlich kleiner als für lange Texte. Diese Tatsache wird durch die Extraktionsergebnisse in Abbildung 29 bestätigt.

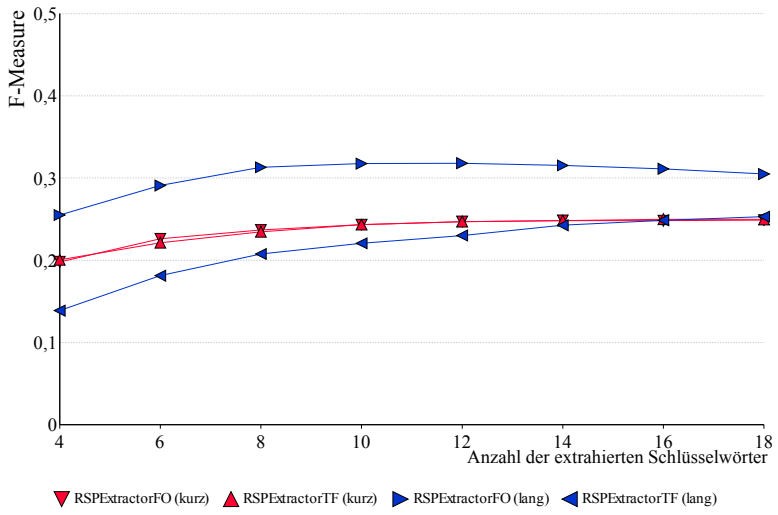

Abbildung 29: F-Measure RSPExtractor

BCExtractor

Die F-Measure-Kurve des BCExtraktors in Abbildung 30 zeigt eine deutliche Verbesserung der Extraktionsergebnisse für kurze Texte. Der linguistische Ansatz dieses Verfahrens zeigt hier eine hohe Wirkungskraft. Dies ist ein Indiz für den hohen Informationsgehalt der zu Grunde liegenden Kurzbeschreibungen.

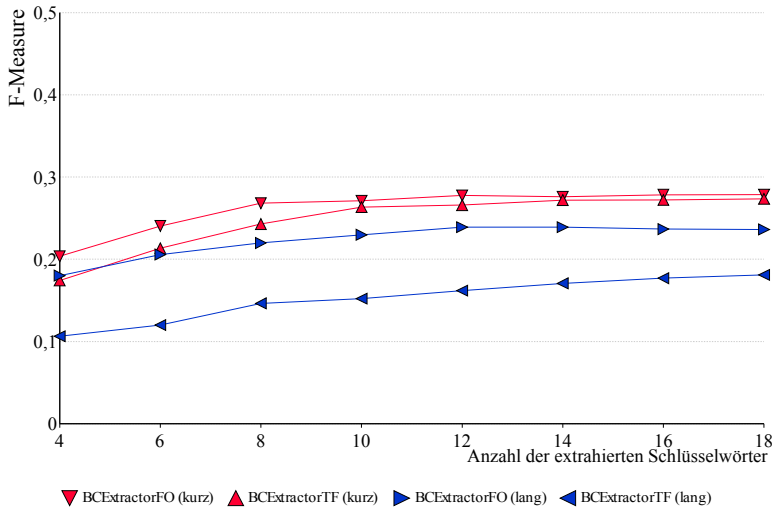

Abbildung 30: F-Measure BCExtractor

CooccurrenceExtractor

Die Qualität der Extraktionsergebnisse des CooccurrenceExtraktors ist ebenfalls von der Textlänge abhängig. Die Abbildung 28 zeigt, dass diese Abhängigkeit jedoch geringer als die der bisher betrachteten Verfahren ist.

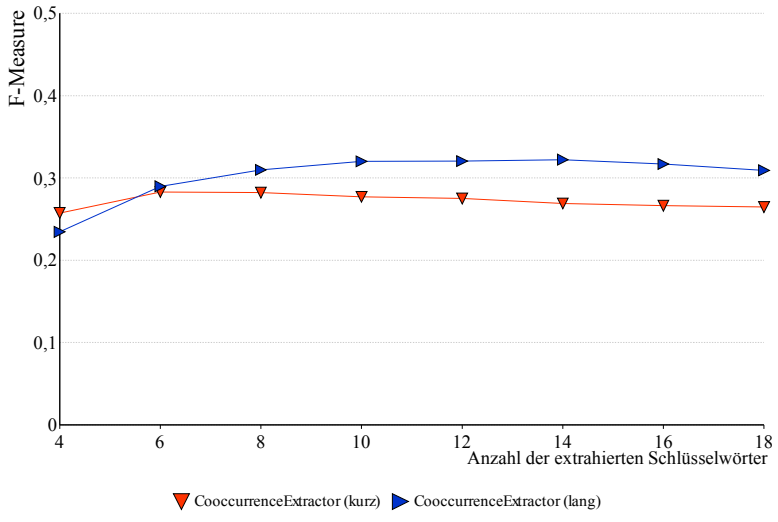

Abbildung 31: F-Measure CooccurrenceExtractor

5.4.2 Statische Extraktion

Durch die statische Extraktion werden für jedes Dokument genau soviel Schlüsselwörter ermittelt wie vom Autor vergeben wurden. Es gilt: $\forall d \in D$: $r_d = |K_d^*| = |K_d|$. Die Werte für Precision und Recall sind in diesem Fall identisch ($a+b=a+c$). Abbildung 32 zeigt das Ergebnis der Extraktion aus langen Texten und Abbildung 33 aus kurzen Texten.

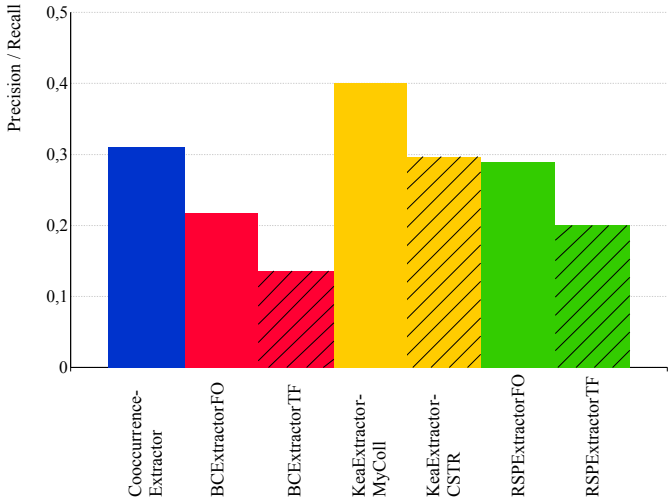

Abbildung 32: Precision / Recall für langen Text

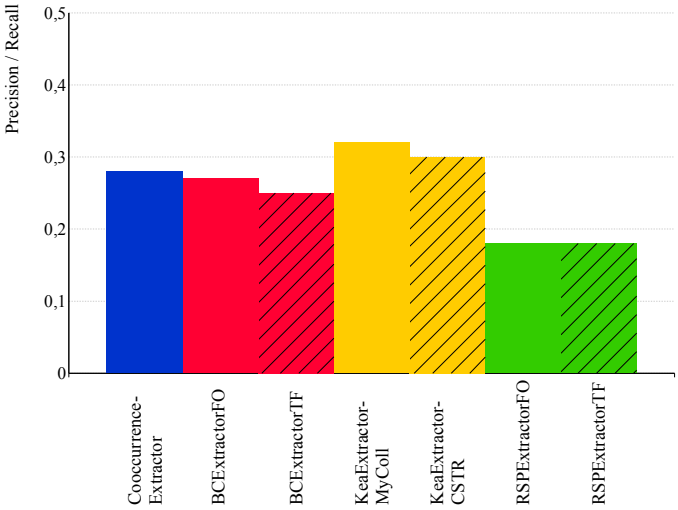

Abbildung 33: Precision / Recall für kurzen Text

Die Ergebnisse der statischen Extraktion können als durchschnittliche Klassifikationsrate der jeweiligen Algorithmen interpretiert werden, die für den Evaluationskorpus in Tabelle 10 aufgelistet sind.

Algorithmus	**Klassifikationsrate in %**	
	langer Text	kurzer Text
CooccurrenceExtractor	31,0	28,2
BCExtractorFO	21,2	26,5
BCExtractorTF	14,0	24,9
KeaExtractorMyColl	40,3	31,8
KeaExtractorCSTR	29,5	33,1
RSPExtractorFO	28,8	17,8
RSPExtractorTF	19,6	17,7

Tabelle 10: Klassifikationsrate der Algorithmen

6. Verbesserung der Extraktionserbnisse

Betrachtet man die Extraktionsergebnisse für eine größere Anzahl von Schlüsselwörtern kann man feststellen, dass der Recall innerhalb von 60 extrahierten Schlüsselwörtern signifikant zunimmt. Das Diagramm in Abbildung 34 zeigt die Recall-Kurven für die jeweils besten Varianten der evaluierten Algorithmen. Der maximale Recall liegt für 60 extrahierte Schlüsselwörter bei allen Verfahren um den Wert 0,7.

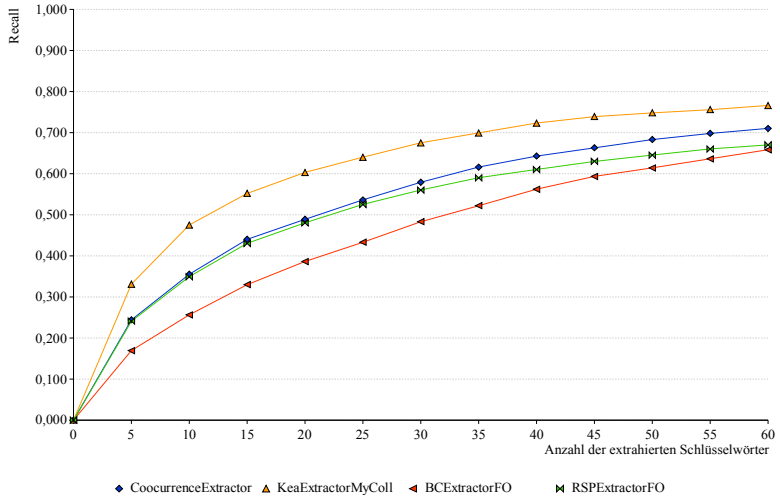

Abbildung 34: aktuelle Recall-Kurve der evaluierten Algorithmen

Hier ergibt sich ein möglicher Ansatz zur Verbesserung der Ergebnisse. Ziel ist es, das Ranking der extrahierten Schlüsselwörter so zu verändern, das der maximal erreichbare Recall bereits innerhalb der ersten 10 Schlüsselwörter bestmöglich angenähert wird. Der Anstieg der Recall-Kurve soll in diesem Bereich maximiert werden (siehe Abbildung 35). Es stellt sich die Frage, mit welchen Mitteln dieses Ziel zu realisieren ist.

Eine viel versprechende Methode besteht in der Generierung und Erschließung von externem Wissen, d.h. zusätzlicher Information zur Entscheidung. Diese Information kann in verschiedener Form vorliegen. Klassische Quellen sind beispielsweise Thesauri, Wörterbücher oder Klassifikationsschemata (Ontologien). Diese Quellen sind jedoch für jeden Einsatzzweck angepasst.

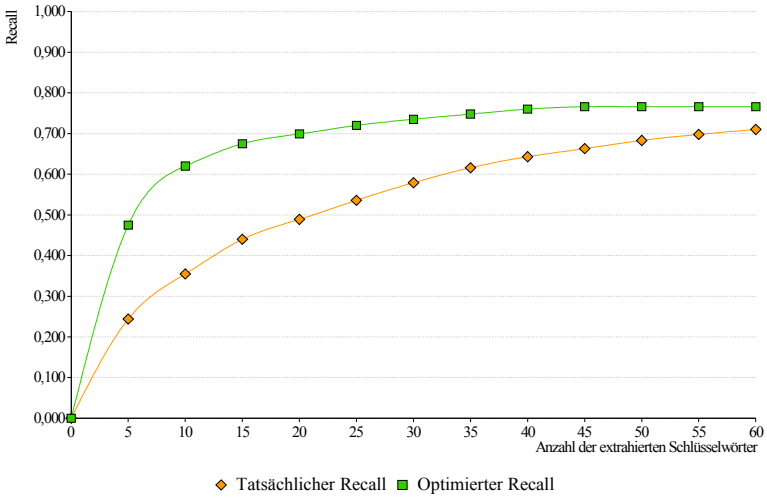

Abbildung 35: mögliche Recall-Optimierung

Das Internet stellt dagegen eine universelle Wissensquelle von ständig wachsendem Ausmaß dar. Zur Generation und Erschließung von externem Wissen aus dem Internet können Methoden des Data-Minings angewendet werden[9]. Im Folgenden werden zwei Verfahren zur Verbesserung der Extraktionsergebnisse auf der Basis von Data-Mining im Internet (Web-Mining) erläutert.

6.1 PMI-IR

Gegenstand der folgenden Betrachtung ist die Verbesserung der semantischen Relation der extrahierten Terme. Schlüsselwörter aus einer Domäne sind untereinander stark assoziiert, d.h. sie treten häufig gemeinsam innerhalb von Dokumenten der Domäne auf. Die bisher betrachteten Verfahren ermitteln Schlüsselwörter ohne explizite Beachtung des semantischen Zusammenhangs auf statistischer Basis. Diese Einschränkung kann dazu führen, dass zwar die Mehrheit der extrahierten Schlüsselwörter das Dokument gut beschreiben, andere scheinen jedoch „Außenseiter“ ohne eine klare semantische Beziehung zur Mehrheit der Schlüsselwörter und damit zum Dokument zu sein. Das als PMI-IR (*Pointwise Mutual Information* und *Information Retrieval)* bezeichnete

9 Als Data-Mining bezeichnet man das systematische Entdecken und Extrahieren unbekannter Informationen aus großen Datenmengen.

Verfahren von Turney führt zu einer Verbesserung hinsichtlich der Kohärenz der extrahierten Schlüsselwörter [Turney, 2002]. Das Verfahren benutzt die Transinformation (*mutual information)* zur Messung der statistischen Assoziationen zwischen zwei Termen.

Ausgangspunkt ist die Annahme, dass Terme, die häufig gemeinsam auftreten, in einer semantischen Beziehung zueinander stehen. Der Algorithmus umfasst Anfragen an eine Internet-Suchmaschine und die Anwendung statistischer Verfahren auf das Resultat dieser Anfragen. Die Benutzung einer Internet-Suchmaschine ermöglicht es dabei, sehr effektiv auf einen Korpus von über 100 Mrd. Wörtern zuzugreifen.

Von einer Menge $K'=\{k'_1, ..., k'_m\}$ der Schlüsselwortkandidaten wird eine Teilmenge $K''\subset K'$ mit $K''=\{k''_1, ..., k''_n\}$, $n \ll m$ der Terme mit der größten Häufigkeit gebildet. Es wird angenommen, dass ein Term g, der in semantischer Relation zu einem oder mehreren Termen $k''\in K''$ steht, ein kohärenteres, d.h. signifikantes Schlüsselwort ist. Für jeden Term k' wird ein Kohärenzwert errechnet, basierend auf der statistischen Assoziation mit jedem Term k'', $k' \neq k''$. Das Kohärenzmerkmal besteht aus $2n$ Bestandteilen:

$$\text{score\_AND}_i(k'_j)=\frac{hits(k'_j \text{ AND } k''_i)}{hits(k'_j)}$$

und

$$\text{score\_NEAR}_i(k'_j)=\frac{hits(k'_j \text{ NEAR } k''_i)}{hits(k'_j)},$$

wobei $hits(query)$ die Anzahl der Treffer für eine Suchanfrage *query* an eine Internet-Suchmaschine ist. Die Suchanfrage wird durch die Operatoren AND bzw. NEAR gebildet. Die AND-Anfrage liefert als Ergebnis die Anzahl aller Dokumente, in denen k' und k" innerhalb des Dokuments vorkommen. Das Ergebnis der NEAR-Anfrage besteht in der Anzahl der Dokumente, in denen k' und k" im Abstand von maximal zehn Wörtern enthalten ist.

Turney verwendet die AltaVista-Suchmaschine für die Suchanfragen zur Bestimmung der Merkmale. Durch den Einsatz von PMI-IR konnten die Ergebnisse von KEA um ca. 25% verbessert werden [Turney, 2003].

Seit der Übernahme von AltaVista durch Yahoo! 2003 wird die von Turney verwendete Anfragesyntax von AltaVista nicht mehr unterstützt, sodass der PMI-IR Algorithmus in der beschriebenen Form aktuell nicht angewendet werden kann. Sollte eine adäquate Suchanfrage in Zukunft durch eine andere Suchmaschine bereitgestellt werden, kann PMI-IR als Modifikation wieder eingesetzt werden.

6.2 Kontextinformation

Dieses Verfahren zur Optimierung der Extraktionsergebnisse beruht auf der Ausnutzung von Kontextinformationen, die im Zuge einer Suche mit einer Internetsuchmaschine gewonnen werden. Es beruht auf der Idee von Benno Stein und Sven Meyer zu Eißen zur Ausnutzung von externem Wissen.

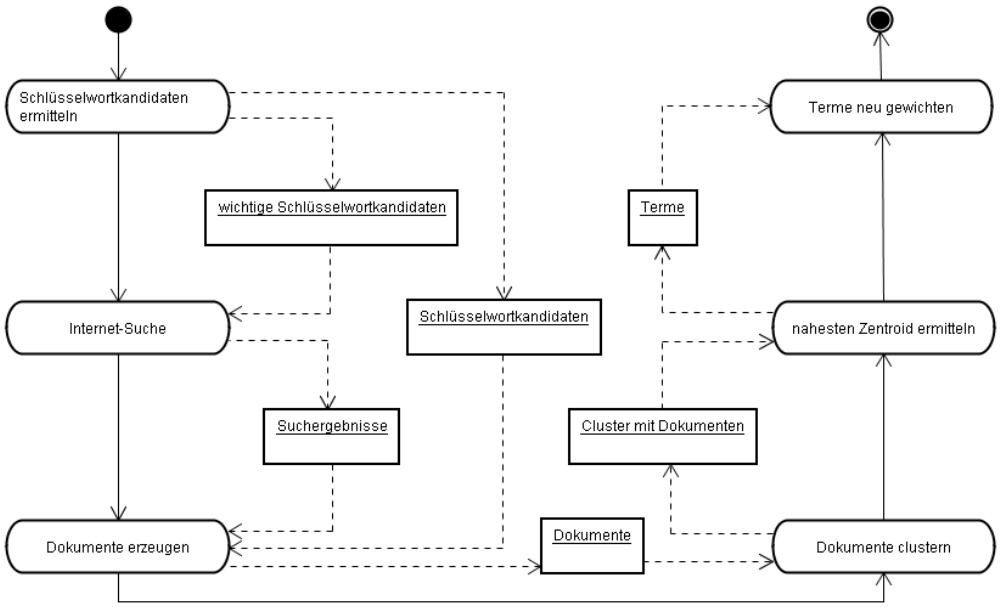

Abbildung 36: Aktivitätsdiagramm für die Extraktion mit Kontextinformation

Mittels eines initialen Extraktionsverfahrens (Bootstrap-Algorithmus) wird die Menge $K' = \{k'_1, ..., k'_m\}$ der Schlüsselwortkandidaten erzeugt. Die Terme der Teilmenge $K'' \subset K'$ mit $K'' = \{k''_1, ..., k''_n\}$, $n \ll m$ mit dem größten Gewicht werden anschließend als Anfrage an eine Suchmaschine übergeben. Die Kontextinformation kann dann durch die Auswertung der Ergebnisse der Suchanfrage ermittelt werden. Dazu werden die in der Liste der Suchergebnis enthaltenen Textschnipsel (snippets) durch eine Clusteranalyse

in k Cluster unterteilt. Die Zentroiden $Z=\{z_1, ..., z_k\}$ der Cluster werden berechnet und der Zentroid $z^* \in Z$, in dessen Cluster sich die Repräsentation des Ausgangsdokumentes befindet (closest centroid), wird extrahiert. Die Aufgabe besteht nun in der Ermittlung der Terme des Zentroiden z^* mit der höchsten Diskriminanzkraft gegenüber allen anderen Clusterzentroiden z_i. Abbildung 36 stellt ein mögliches System schematisch dar.

Die Evaluation der Extraktionsverfahren hat gezeigt, dass die Precision mit Zunahme der Anzahl der extrahierten Schlüsselwörter abnimmt. Daraus abgeleitet sollten die höchstgewichteten Terme K'' des Bootstrap-Algorithmus zur Suchanfrage verwendet und gleichzeitig zur Menge der finalen Schlüsselwörter K hinzugefügt werden.

Für die Ermittlung der Diskriminanzkraft der verbleibenden Terme $K^*=K'/K''$ sind verschiedene statistische Ansätze zu untersuchen. Ein im Rahmen dieser Arbeit getestetes Verfahren ist die Auswertung der Entropie der Terme über alle Zentroiden Z. Für jeden Term k^* kann die Entropie $E(k^*)$ berechnet werden als (vgl. [Lochbaum & Streeter, 1989]):

$$E(k^*)=1+\frac{1}{\log_2|Z|}\sum_{z\in Z} P(z,k^*)\cdot\log_2 P(z,k^*) \text{ mit } P(z,k^*)=\frac{tf(z,k^*)}{\sum_{z\in Z} tf(z_i,k^*)}.$$

In einem Experiment wurden durch den ContextExtractor 60 Schlüsselwortkandidaten ermittelt und anschließend die vier höchst gewichteten Terme als Suchanfrage übergeben (n=4, m=60). Die als Resultat erhaltenen 100 Textschnipsel wurden durch ein *k-means* Clusterverfahren in $5 \leq k \leq 15$ Cluster unterteilt. Obwohl dieses Vorgehen plausibel erscheint, konnte das Ergebnis ad hoc nicht verbessert werden. An dieser Stelle sind weitere Forschungen über die Ursachen nötig, die jedoch über den Rahmen dieser Arbeit hinausgehen.

7. Anwendung: Fokussierte Suche im Internet

In den vorhergehenden Kapiteln wurden verschiedene Verfahren zur automatischen Extraktion von Schlüsselwörtern aus Textdokumenten untersucht. Dieses Kapitel beschreibt, wie die ermittelten Schlüsselwörter für die eingangs skizzierten Szenarien der fokussierten Suche eingesetzt werden können. Die extrahierten Schlüsselwörter sollen nun als Suchbegriffe für die Eingabe in Internet-Suchmaschinen dienen. Dazu müssen Suchanfragen automatisch generiert werden.

Der einfachste Fall besteht in der Konkatenation aller ermittelten Suchbegriffe zu einer einzigen Suchanfrage. Im ungünstigsten Fall liefert die Suchmaschine dann jedoch kein Dokument oder falls vorhanden das Ausgangsdokument zurück. Daher müssen Methoden zur Erzeugung mehrerer partieller (multipler) Suchanfragen aus der Menge der Schlüsselwörter entwickelt werden. Ziel ist die Vergrößerung des Ergebnisraums bei gleichzeitiger Optimierung der Retrieval-Ergebnisse [Belkin et al., 1993]. Die im Folgenden vorgestellten Algorithmen zur Generierung multipler Suchanfragen stellen eine mögliche Lösung der Aufgabe dar.

7.1 Automatische Generierung multipler Suchanfragen

Aus einer gegebenen Menge von gewichteten Schlüsselwörtern K wird die Teilmenge der Schlüsselwörter mit dem höchsten Gewicht als Menge von Suchbegriffen $S \supseteq K$, $S = \{s_1, s_2, ..., s_n\}$ gebildet und als *„lange Suchanfrage“* an eine oder mehrere Suchmaschinen übergeben. Die Qualität der Suchergebnisse kann verbessert werden indem aus der Menge S mehrere sich überlappende Teilanfragen (multiple queries) formuliert werden. Die jeweiligen Ergebnisse werden nach Abschluss der Suche zu einer Liste zusammengefasst und in einer Rangfolge ausgegeben. Forschungen [Saracevic & Kantor, 1988] haben gezeigt, dass die Wahrscheinlichkeit der Relevanz eines Dokumentes relativ zu seiner Auftrittshäufigkeit in den Suchergebnissen der einzelnen Teilanfragen ist.

Die Generierung multipler Suchanfragen bedeutet die sinnvolle Ermittlung mehrerer sich überschneidender Teilmengen S_j aus der Menge der Suchbegriffe S, wobei gilt:

$$\bigcup_{j=1}^{p} S_j = S$$

Shapiro und Taska [Shapiro & Taska, 2003] beschreiben hierfür zwei unterschiedliche Algorithmen.

7.1.1 OEQ – Open End Query

Der OEQ-Algorithmus erzeugt eine feste Anzahl von Teilanfragen aus einer variablen Anzahl von nach einer Rangfolge sortierten Suchbegriffen. Die Teilanfragen werden ausgehend vom Suchbegriff s_i mit dem höchsten Gewicht rekursiv durch Hinzufügen des jeweils nächsten Suchbegriffs s_{i+1} gebildet. Unterschreitet die Anzahl der Suchergebnisse für eine derart erstellte Suchanfrage eine definierte Relevanzschwelle g, so wird der zuletzt hinzugefügte Term vor Hinzufügen des nächsten Terms aus der Suchanfrage entfernt. Tabelle 11 zeigt die Ermittlung der ersten Teilanfrage an einem Beispiel.

#	Schritt	Suchbegriffe								Anzahl der
		s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	Suchergebnisse
1	Anfrage mit	s_1								631
2	Anfrage mit	s_1	s_2							276
3	Anfrage mit	s_1	s_2	s_3						112
4	Anfrage mit	s_1	s_2	s_3	s_4					67
5	Anfrage mit	s_1	s_2	s_3	s_4	s_5				0
6	**Anfrage mit**	**s_1**	**s_2**	**s_3**	**s_4**		**s_6**			**37**
7	Anfrage mit	s_1	s_2	s_3	s_4		s_6	s_7		25
8	Anfrage mit	s_1	s_2	s_3	s_4		s_6		s_8	3

Tabelle 11: Erstellung der Teilanfrage S_1 mit g = 30

Die ermittelte Teilanfrage 1 lautet $S_1 = \{s_1, s_2, s_3, s_4, s_6\}$. Dieser Vorgang wird n mal wiederholt. Für acht Suchbegriffe entstehen somit acht Teilanfragen. Die Relevanzschwelle g wurde von Shapiro und Taska experimentell ermittelt.

7.1.2 CEQ – Close End Query

Im Gegensatz zu *OEQ* erzeugt der CEQ-Algorithmus eine variable Anzahl an Teilanfragen aus einer festen Anzahl von Suchbegriffen. Die Anzahl der möglichen Kombinationen einer Suchanfrage aus *k* Termen aus *S* kann berechnet werden durch:

$$C(n,k)=\frac{n!}{(n-k)!*k!}$$

Gesucht sind nun alle Kombinationen von mindestens *m* Termen aus *S,* die wie folgt ermittelt werden können:

$$\sum_{i=m}^{n} C(n,i)$$

Tabelle 12 zeigt die Ermittlung der Teilanfragen mit einer Mindestlänge der Teilanfragen von $m=6$ aus acht vorgegebenen Suchbegriffen. Die Mindestlänge *m* wurde hier ebenfalls experimentell ermittelt.

#	Suchbegriffe								Anzahl der Teilanfragen	
	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8		
6	s_1	s_2	s_3	s_4	s_5	s_6			C(8,6) =	28
7	s_1	s_2	s_3	s_4	s_5	s_6	s_7		C(8,7) =	8
8	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	C(8,8) =	1
									Summe der Teilanfragen	37

Tabelle 12: Ermittlung der Teilanfragen für m = 6 und n = 8

Nach Shapiro und Taska ist der CEQ-Algorithmus am besten geeignet für $7 \leq n \leq 9$ Schlüsselwörter.

Nach der Generierung der Teilanfragen werden diese an eine oder mehrere Suchmaschinen übermittelt und die jeweiligen Suchergebnisse empfangen. Aus einer Anzahl von *p* Teilanfragen entsteht eine Menge von von *p* Ergebnislisten. Diese müssen anschließend zu einer Ergebnisliste zusammengeführt werden. Die Aufbereitung dieser Listen ist Gegenstand des nächsten Abschnitts.

7.2 Vereinigung multipler Suchergebnisse

Nach Absenden von p Teilanfragen $(S_1, S_2, ..., S_p)$ werden p Suchergebnislisten $(R_1, R_2, ..., R_p)$ zurückgegeben. Jede Liste R_j besteht aus einer Menge verschiedener Dokument-URLs $R_j=\{r_{j1}, r_{j2}, ..., r_{jkj}\}$, die in einer bestimmten Rangfolge geordnet sind. Diese entsteht nach Kriterien der jeweiligen Suchmaschine. Es gilt nun alle erhaltenen Listen R_j zu vereinen und eine Rangfolge der Dokumente nach ihrer Relevanz zu erstellen.

Die von Shapiro und Taska [Shapiro & Taska, 2003] beschriebenen Vereinigungsalgorithmen werden im Folgenden erläutert.

7.2.1 MCQ – Merge Close Query

Dies ist die einfachste Methode, mehrere Listen miteinander zu vereinen. Die maximale Anzahl der Dokumente einer Ergebnisliste R_j sei wiederum m. Aus allen Ergebnislisten werden die ersten m Dokument-URLs extrahiert und einer gemeinsamen Liste F angefügt. Diese Liste enthält zunächst auch alle Duplikate einer URL, die in allen Ergebnislisten vorkommen. Die URLs werden in alphabetischer Reihenfolge sortiert. Anschließend werden alle mehrfach vorkommenden URLs bis auf eine Instanz entfernt. Das Gewicht einer URL entspricht der Anzahl ihres Auftretens in F. Schlussendlich werden die verbleibenden URLs wiederum beginnend mit dem größten Gewicht ausgegeben. Haben zwei URLs das gleiche Gewicht, entscheidet das Gewicht der entsprechenden Teilanfrage $W(S_j)$ über die Reihenfolge.

7.2.2 *MOQ* – Merge Open Query

Der MOQ-Algorithmus benutzt zur Gewichtung der Suchergebnisse das Gewicht der jeweiligen Teilanfrage sowie den Rang eines Dokuments in der zugehörigen Ergebnisliste.

Die maximal betrachtete Länge (Anzahl der Dokumente) einer Ergebnis-Liste R_i sei m und $|R_j|$ die tatsächliche Länge von R_j. Dann ist m_j die Anzahl der Dokumente von R_j mit $m_j=min(m,|R_j|)$. Das Gewicht einer Dokument-URL wird bestimmt als:

$$W(r_{jk})=W(S_j)\cdot(m_j-k+1)\ .$$

$W(S_j)$ ist das Gewicht der zugehörigen Teilanfrage und wird wie folgt berechnet:

$$W(S_j) = \sum_{k=1}^{|S_j|} (W(s_{jk}))/|S_j| \ ,$$

wobei $|S_i|$ die Anzahl der Suchbegriffe in S_i und $W(s_{ik})$ das Gewicht des Suchbegriffs s_{ik} ist.

Nach Berechnung der Gewichte der Dokument-URLs aller Ergebnislisten werden alle mehrfach enthaltenen URLs entfernt. Dabei wird jeweils das Gewicht kumuliert. Nach Entfernung aller Duplikate werden die verbleibenden URLs beginnend mit dem größten Gewicht ausgegeben.

Auf die Implementierung und Evaluierung der fokussierten Suche wurde im Rahmen dieser Arbeit verzichtet.

8. Zusammenfassung und Ausblick

Im Rahmen dieser Arbeit entstand eine Java-Bibliothek, die unterschiedliche Ansätze zur automatischen Extraktion von Schlüsselwörtern aus Textdokumenten vereint. Weiterhin wurde ein Framework zur Evaluation der Algorithmen sowie ein umfangreicher, automatisiert erweiterbarer Evaluierungskorpus aus wissenschaftlichen Dokumenten erzeugt. Die Umwandlung der Quelldokumente ins XML-Format gewährleistet dabei einen einfachen Zugriff auf die Daten.

Die Evaluierung der implementierten Verfahren zeigt den Vorteil der auf einem Korpus basierenden Algorithmen gegenüber den dokumentbasierten Algorithmen. Am Beispiel der unterschiedlichen Trainingskorpora des KEA-Algorithmus wird jedoch auch die Abhängigkeit der korpusbasierten Verfahren von der Trainingsdomain deutlich. Für eine allgemeine Anwendung zur fokussierten Suche nach relevanter Information im Internet ist jedoch ein domainunabhängiges Extraktionsverfahren notwendig.

Die Extraktionsergebnisse der dokumentbasierten, domainunabhängigen Algorithmen sind sehr unterschiedlich. Generell zeigt sich, dass den statistischen Merkmalen Auftrittshäufigkeit eines Terms im Text (Termfrequenz) und Position des ersten Auftretens im Text (*first occurrence*) ein wesentliches Gewicht bei der Klassifizierung als Schlüsselwort zukommt. Dies ist besonders deutlich an den stark abweichenden Ergebnissen der unterschiedlichen Implementationen des RSPExtractors zu erkennen.

Der linguistische Ansatz der Extraktion von Nominalphrasen des BCExtractors führte in Abhängigkeit der Textlänge zu sehr unterschiedlichen Ergebnissen. Den linguistischen Verfahren sollte jedoch bei der Verbesserungen der Extraktionsverfahren in Zukunft eine größere Aufmerksamkeit zuteil werden.

Die für den korpusbasierten KEA-Algorithmus festgestellte Abhängigkeit der Extraktionsergebnisse von der Textlänge [Witten et al., 1999] konnte für die dokumentbasierten Verfahren nicht pauschal nachgewiesen werden.

Das von Matsuo und Ishizuka entwickelte Verfahren der Kookkurrenz-Ermittlung in Verbindung mit dem Chi-Quadrat-Anpassungstest stellte sich sowohl für kurze als auch für lange Texte als sehr robust dar. Für die aufgezeigten Methoden zur Recall-Optimierung durch externes Wissen ist dieses dokumentbasierte Verfahren am besten geeignet.

Literaturverzeichnis

[Barker & Cornacchia, 2000] Ken Barker, Nadia Cornacchia: "Using Head Noun Phrases to Extract Document Keyphrases" in Proceedings of the 13th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence, S. 40-52, 2000

[Belkin et al., 1993] Nicholas J. Belkin, C. Cool, W. Bruce Croft, James P. Callan: "The effect multiple query representations on information retrieval system performance" in Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval, S. 339 - 346, 1993

[Brin & Page, 1998] S. Brin and L. Page: "The anatomy of a large-scale hypertextual Web search engine" in Computer Networks and ISDN Systems,Vol. 30, S. 107 - 117, 1998

[Fayyad & Irani, 1993] Usama M. Fayyad, Keki B. Irani: "Multi-interval discretization of continous-valued attributes for classification learning" in Proceedings of the 13th IJCAI, S. 1022-1027, 1993

[Jones & Paynter, 2001] Steve Jones, Gordon W. Paynter: "Human evaluation of Kea, an automatic keyphrasing system" in Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries, S. 148 - 156, 2001

[Kleppe et al., 2005] Alexander Kleppe, Dennis Braunsdorf, Christoph Loessnitz, Sven Meyer zu Eissen: "On Web-based Plagiarism Analysis" in Proceedings of the Second International Workshop on Text-Based Information Retrieval, Universitaet Koblenz-Landau, 2005

[Lochbaum & Streeter, 1989] K. E. Lochbaum, L. A. Streeter: "Combining and comparing the effectiveness oflatent semantic indexing and the ordinary vector space model for informationretrieval" in Information Processing and Management, 25(6), S. 665 - 676, 1998

[Luhn, 1958] H. P. Luhn: "The automatic creation of literature abstracts" in IBM Journal of Research and Development, 2, S. 159 - 165, 1958

[Matsuo & Ishizuka, 2003] Y. Matsuo, M. Ishizuka: "Keyword Extraction from a Single Document using Word Co-occurrence Statistical Information" in International Journal on Artificial Intelligence Tools, Vol.13, No.1, S. 157-169, 2003

[Porter, 1980] M. F. Porter: "An algorithm for suffix stripping" in Program, 14(3), S. 130 - 137, 1980

[Salton & McGill, 1983] G. Salton, M. J. McGill, Introduction to modern information retrieval, McGraw-Hill, Inc. New York, NY, USA, 1983

[Saracevic & Kantor, 1988] T. Saracevic, P. Kantor: "A Study of Information Seeking and Retrieving. Part III. Searchers, Searches and Overlap." in JASIS, 39(3), S. 197 - 216, 1988

[Shapiro & Taska, 2003] Jacob Shapiro, Isak Taska: "Constructing Web Search Queries from the User's Information Need Expressed in Natural Language" in Proceedings of the 2003 ACM symposium on Applied computing, S. 1157 - 1162, 2003

[Spink & Jansen, 2004] Amanda Spink, Bernard J. Jansen: "A study of Web search trends", http://www.webology.ir/2004/v1n2/a4.html, 2004

[Stein & Meyer zu Eissen, 2005] Benno Stein, Sven Meyer zu Eissen: "Technologies for Market Analysis from Internet Data" in Proceedings of the WWW/Internet 2005 Conference, Lisbon, Portugal, 2005

[Stein, 2005] Benno Stein: "Unit. Bayes-Klassifikation", http://www.uni-weimar.de/medien/webis/teaching/lecturenotes/webtec-advanced/part-machine-learning/unit-bayesian-learning.ps.pdf, Stand: 28. Apr 2006

[Tseng, 1998] Yuen-Hsien Tseng: "Multilingual Keyword Extraction for Term Suggestion" in Proceedings of the 21st annual international ACM SIGIR conference on Research and Development in Information Retrieval, Melbourne, Australia, S. 377-378, 1998

[Turing, 1950] Alan Mathison Turing: "Computing machinery and intelligence" in VOL. LIX. No.236, S. 433, 1950

[Turney, 1997] Peter D. Turney: "Extraction of Keyphrases from Text: Evaluation of Four Algorithms" in National Research Council, Institutefor Information Technology, Technical Report ERB-1051, 1997

[Turney, 1999] Peter D. Turney: "Learning to Extract Keyphrases from Text" in National Research Council, Institutefor Information Technology, Technical Report ERB-1057, 1999

[Turney, 2002] Peter D. Turney: "Mining the Web for Lexical knowledge to Improve Keyphrase Extraction: Learning from Labeled and and Unlabeled Data", http://iit-iti.nrc-cnrc.gc.ca/iit-publications-iti/docs/NRC-44947.pdf, Stand: 28. Oktober 2004, 17:14 Uhr

[Turney, 2003] Peter D. Turney: "Coherent Keyphrase Extraction via Web Mining" in Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03), S. 434-439, 2003

[van Rijsbergen, 1979] C. J. van Rijsbergen, Information Retrieval, Butterworths, London, 1979

[Witten et al., 1999] Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, Craig G. Nevill-Manning: "KEA: Practical Automatic Keyphrase Extraction" in Proceedings of the fourth ACM Conference on Digital Libraries, Berkeley, California, United States, S. 254-255, 1999

[Yahoo Search Blog, 2005] : "Our Blog is Growing Up - And So Has Our Index", http://www.ysearchblog.com/archives/2005_08.html, Stand: 08. August 2005, 16:15 Uhr