Latent Semantisches Indexieren für kurze Texte

Wissenschaftliche Arbeit zur
Erlangung des Grades
Bachelor of Science
in Information Engineering
im Fachbereich
Informatik und Informationswissenschaft
der Universität Konstanz

vorgelegt von
Sebastian Rexhausen
Mainaustr. 146
78464 Konstanz

sebastian.rexhausen@web.de
Matr.Nr.: 01/475604

1. Gutachter: Prof. Dr. Harald Reiterer
2. Gutachter: Prof. Dr. Rainer Kuhlen

Konstanz, den 4. Oktober 2005

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

bash	**B**ourne **A**gain **Sh**ell
CSV	**C**omma **S**eparated **V**alues
Doc	**Doc**ument
Doc-Id	**Doc**ument **Id**entifier
ER-Diagramm	**E**ntity-**R**elationship-Diagramm
FORIS	**For**schungs**i**nformations**s**ystem Sozialwissenschaften
GESIS	**Ge**sellschaft **S**ozialwissenschaftlicher **I**nfra**s**truktureinrichtungen e.V.
GIRT	**G**erman **I**ndexing and **R**etrieval **T**estdatabase
HTML	**H**yper**t**ext **M**arkup **L**anguage
IR	**I**nformation **R**etrieval
LSI	**L**atent **S**emantic **I**ndexing
OCR	**O**ptical **C**haracter **R**ecognition
Perl	**P**ractical **E**xtraction and **R**eport **L**anguage
PHP	**P**HP: **H**ypertext **P**reprocessor
PL/pgSQL	**P**rocedural **L**anguage/**P**ost**g**res **SQL**
PR-Diagramm	**P**recision-**R**ecall-Diagramm
SGML	**S**tandard **G**eneralized **M**arkup **L**anguage
SOLIS	**So**zialwissenschaftliches **Li**teraturinformations**S**ystem
SQL	**S**tructured **Q**uery **L**anguage
SVD	**S**ingular **V**alue **D**ecomposition
TREC	**T**ext **Re**trieval **C**onference

1 Einleitung

Das Gewinnen von Informationen aus unstrukturierten Daten nimmt insbesondere mit zunehmender Ausbreitung des World Wide Webs immer mehr an Bedeutung zu. Niemals zuvor hatten so viele Menschen Zugriff auf so viele unterschiedliche Informationen (Yu et al. (2002)). Einzelne Internet-Suchmaschinen, wie zum Beispiel die der Internet-Firma Google, greifen mittlerweile auf mehr als 8 Milliarden Objekte zu (Coughran (2004)) und in Zeiten, in denen diese Unternehmen (wie z.B. Google), an der Börse notiert und sie somit gezwungen sind, ihre Marktdurchdringung immer weiter voranzutreiben, wird nicht nur die Technik, die die Informationen des WWW handhabbar machen soll, weiterentwickelt, sondern auch die mit der Zeit ansteigende Entropie auf der Festplatte des Benutzers als potentielles Einsatzgebiet von Websuchmaschinenentechnologie angesehen (Hoffmann (2004)). Ziel all dieser Suchmaschinen ist es, aus der stetig anwachsenden und immer unübersichtlicher werdenden Datenflut Wissen zu generieren und individuelle Suchbedürfnisse zu befriedigen. Die meisten der den Suchmaschinen zugrunde liegenden Information-Retrieval-Systeme vergleichen Wörter einer Suchanfrage exakt mit den Wörtern, die in den Dokumenten enthalten sind oder mit den die Dokumente beschreibenden Wörtern. Aber das Suchen nach der buchstabengenauen Übereinstimmung zwischen zwei Wörtern befriedigt in den meisten Fällen nicht das Suchbedürfnis des Nutzers - taucht ein Suchterm nicht im Dokument auf, wird dieses Dokument auch nicht in der Ergebnismenge auftauchen. Vielmehr wünscht der Nutzer darüber hinaus, dass Synonyme zu den eingegebenen Suchtermen bei der Ergebnisberechnung hinzugezogen werden, oder dass mehrdeutige Wörter in ihrer gewünschten Bedeutung mit in die Suchlogik einfließen. Die wenigsten IR-Systeme können diese Probleme der Synonymie und der Polysemie behandeln oder dieses Problem ohne aufwändige, pflegeintensive Thesauri oder Wissensdatenbanken handhaben (Yu et al. (2002)). Aktuelle Untersuchungen (Berry und Dumais (2004)) zeigen, dass der hauptsächlich auf Dimensionsreduktions-Verfahren beruhende IR-Algorithmus „Latent Semantic Indexing“ (LSI) Verbesserungen in vielen Anwendungsgebieten der IR-Systeme mit sich bringt - und auch die oben genannten Problemstellungen handhabbar macht.

LSI basiert auf der Annahme, dass zwischen Dokumenten und den darin enthaltenen Termen eine implizite übergeordnete Beziehung zueinander besteht. Der von LSI verwendete Algorithmus betrachtet die ihm zur Verfügung gestellte Dokumentenkollektion als Ganzes und berücksichtigt alle in den jeweiligen Dokumenten enthaltenen Terme sowie welche Dokumente welche Terme miteinander gemeinsam haben. Dokumente, die viele Terme miteinander teilen, sieht LSI als semantisch nah zueinander an - Dokumente, die wenig oder keine Terme miteinander gemeinsam haben, als semantisch fern. Diese einfache Regel korreliert erstaunlicherweise sehr stark mit der Vorgehensweise, wie Menschen Inhalte wahrnehmen oder wie sie z.B. eine Dokumentenkollektion klassifizieren würden (Landauer und Dumais (1997)).
Mit Hilfe der Singulärwertzerlegung (singular value decomposition—SVD) können nun Berechnungen getätigt werden, die als Grundlage für eine Anfragebearbeitung dienen. Zur Verdeutlichung der Methode von LSI sei folgendes Beispiel angeführt: Eine zu untersuchende Datenmenge enthält die beiden synonym gebrauchten Terme „Laptop“ und „Notebook“, der Nutzer gab jedoch nur „Laptop“ als Suchbegriff ein. LSI gewinnt nun aus den Resultaten der SVD die Information, dass die beiden Begriffe in den meisten Fällen im selben Kontext genutzt werden und fügt nun zur Ergebnisliste Einträge hinzu, die nur den Term „Notebook“ enthalten - LSI bringt also die latenten Beziehungen hervor.

1.1 Gegenstand der Arbeit

Motiviert durch die Arbeit des Lehrstuhls „Mensch-Computer Interaktion“ der Universität Konstanz am Projekt INVISIP[1] und die daraus entstandene Problemstellung der Volltextsuche auf Geo-Metadaten, entstand die Nachfrage nach einem geeigneten IR-System, das die verschiedenen Anforderungen des Projektes handhaben kann. Die vorhandenen Geo-Metadaten enthalten nur kurze Beschreibungen zu den einzelnen Datensätzen und der Gebrauch der Beschreibungsterme ist je nach Autor der Beschreibungen sehr inhomogen und ein geeignetes IR-System zu finden scheint diffizil. Da die Untersuchungen zu LSI sehr vielversprechende Ergebnisse in ähnlichen Bereichen lieferten, fiel die Wahl auf LSI.
Ziel dieser Arbeit ist nun einerseits ein IR-System auf Basis von LSI zu implementieren und andererseits zu testen, wie effizient und effektiv dieses LSI-System im Hinblick auf einen Dokumentenkorpus arbeitet, der vornehmlich aus kurzen Texten besteht. Um

[1] INVISIP - Informations Visualization in Site Planning. EU-Förderprojekt (IST-2000-29640). Ziel: Standortplanungsprozess durch Einsatz von Visualisierungstechniken zu vereinfachen.

eine Vergleichbarkeit mit anderen IR-Systemen zu gewährleisten, wurde für diese Arbeit allerdings nicht auf die Geo-Metadaten des INVISIP-Projekts zurückgegriffen, sondern auf den von Michael Kluck bei der GESIS[2] entwickelten GIRT4[3]-Datensatz - genauer auf die deutschsprachigen Titel einer Teilmenge der darin enthaltenen Dokumente. In einem Ausblick soll aufgezeigt werden, welche Anpassungen und Erweiterungen eine zusätzliche Steigerung der Leistungsfähigkeit des Systems nach sich ziehen könnten.

1.2 Aufbau der Arbeit

In Kapitel 2 wird zunächst auf die Geschichte von LSI eingegangen, um dann näher auf die dahinter stehende Theorie einzugehen. Nach einem Überblick über die Anwendungsgebiete wird ein State-Of-The-Art-Projekt behandelt, welches die vielen Fähigkeiten von LSI demonstriert. Zum Ende des Kapitels werden Begriffe aus der Informationswissenschaft und aus der linearen Algebra erläutert, die bei dem Verständnis der Arbeit behilflich sein sollen.
Kapitel 3 demonstriert die Funktionsweise eines IR-Systems am Beispiel LSI. Auf die einzelnen Phasen der Vorverarbeitung, Berechnung und der Anfrageverarbeitung, die jedes IR-System durchlaufen muss, wird in diesem Abschnitt näher eingegangen.
Im vierten Abschnitt der Arbeit werden die theoretischen Erkenntnisse in eine praktische Implementation eines auf LSI beruhenden IR-Systems umgesetzt. Auch hier werden die einzelnen Phasen durchlaufen und genauer auf die praktische Umsetzung eingegangen.
Nachdem in Kapitel 4 ein funktionsfähiges IR-System in die Tat umgesetzt wurde, geht es in Kapitel 5 um das Testen dieses Systems. Nach einem Einblick in vorhandene Retrieval-Tests wird mit den eigentlichen Untersuchungen der Effektivität und Effizienz des IR-Systems fortgefahren.
Schließlich soll in einem Fazit kurz ein Resümee der erarbeiteten Ergebnisse gezogen werden, um dann in einem Ausblick weitere Verbesserungsmöglichkeiten aufzuzeigen.

[2]Gesellschaft Sozialwissenschaftlicher Infrastruktureinrichtungen e.V.
[3]GIRT = German Indexing and Retrieval Testdatabase), 4. Version

2 Hintergrund

Latent Semantisches Indexieren ist ein seit 1988 existierendes automatisches algebraisches Information-Retrieval-Verfahren, das im Gegensatz zu anderen IR-Verfahren in der Lage ist, synonyme und polyseme Terme in einem Text ohne zusätzlichen Aufwand, sowohl auf Seiten der Implementation als auch der Nutzung, zu erkennen und damit das Ergebnisspektrum einer Suchanfrage zu erweitern. In diesem Kapitel wird nach einem kurzen Abriss über die Geschichte von LSI auf die Theorie, die hinter LSI steckt, eingegangen - warum funktioniert LSI, wieso liefert es bessere Ergebnisse als andere Verfahren? Daran anschließend wird ein Überblick gegeben über die Anwendungsgebiete, in denen LSI eingesetzt werden kann. Teil 2.4 wird die theoretischen Grundlagen schildern, sowohl informationswissenschaftlicher als auch mathematischer Art. Abschließend werden die bei Retrieval-Tests genutzten Testkollektionen aufgeführt, anhand derer die unterschiedlichen IR-Verfahren in ihrer Effektivität und Effizienz getestet und miteinander verglichen werden können.

2.1 Geschichte von LSI

Latent Semantic Indexing wurde Ende der 1980er Jahre bei der Firma Bellcore entwickelt. Der erste Einsatz der Technik wurde in dem Produkt „expert/expert-locator“ vollzogen und in dem Paper „An expert/expert-locating system based on automatic representation of semantic structure“ von Streeter und Lochbaum (1988) beschrieben. Erstmals unter dem Namen LSI der Öffentlichkeit vorgestellt wurde der Algorithmus auf der CHI'88 (15-19.05.1988) mit dem Vortrag: „Using Latent Semantic Analysis To Improve Access To Textual Information“ von Dumais et al. (1988). Am 13.06.1989 wurde dann das Patent „Computer information retrieval using latent semantic structure“ von der Firma Telcordia Technologies beim US-Patentamt eingereicht. Große Berühmtheit erlangte LSI dann schließlich 1990 mit der Veröffentlichung des Artikels „Indexing by Latent Semantic Analysis“ im „Journal of the American Society of Information Science“ von Deerwester et al. (1990). Dieser Artikel wird als Grundstock vieler über LSI handelnder Arbeiten ver-

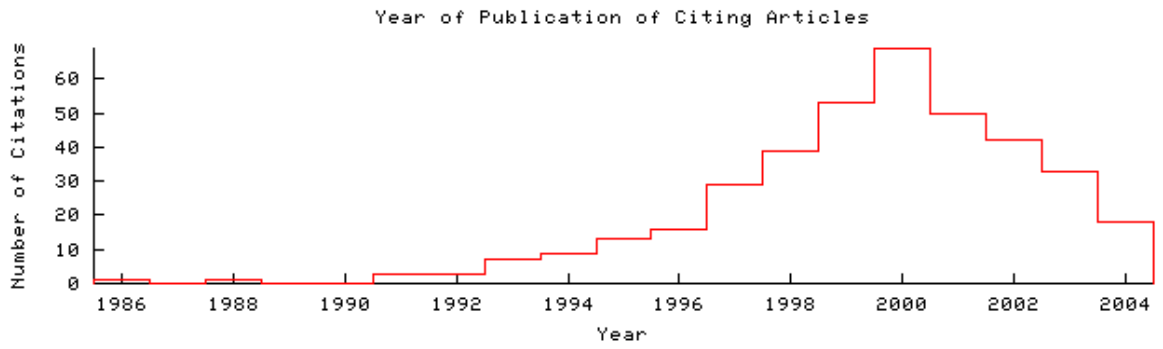

Abbildung 2.1: Verlauf der Zitierungen des Artikels Deerwester et al. (1990) pro Jahr. 502 Zitate insgesamt (Stand: 07.07.2005) - Quelle: http://citeseer.ist.psu.edu/deerwester90indexing.html

wendet. Für einen Zitierungsverlaufs-Überblick des in http://citeseer.ist.psu.edu 502 mal zitierten Papers siehe Abbildung 2.1.

2.2 Theorie hinter LSI

LSI ist ein vollautomatisches, statistisches IR-Verfahren, welches versucht - im Gegensatz zu vielen anderen IR-Verfahren - seine Urteile nicht allein auf Grund des Auftretens eines Terms (Keyword-Matching) zu ziehen, sondern die Beziehungen der Terme und den Kontexten, in denen sie auftreten (Phrasen, Absätze, Dokumente), für seine Berechnungen zu nutzen (Nakov (2001)). Um diesen Konzept- bzw. Themenraum zu generieren, benötigt es keinerlei Vorwissen oder andere Grundlagen wie z.B. Listen (Lexikon, Wörterbuch) oder Netzwerke (z.B. Thesaurus, Ontologie) und es wird außerdem die Reihenfolge bzw. Position der Terme ignoriert. Es nutzt an Stelle dessen allein die semantischen Informationen, die latent im Dokumentenkorpus vorhanden sind. Auch nachdem diese Berechnungen durchgeführt wurden, sind keinerlei manuelle Anpassungen sowohl auf Seiten des Informationssuchenden als auch auf Seiten des Informationsanbietenden mehr nötig.

LSI nutzt zunächst eine Term-Dokument-Matrize in der Terme zu Dokumenten zugeordnet werden; jede Zeile entspricht einem Term, jede Spalte einem Dokument. In den jeweiligen Zellen wird die Häufigkeit des Auftretens des Terms abgetragen und anschließend gewichtet. Der Konzeptraum ist eine niedrigdimensionale Approximation dieser Term-Dokument-Matrize. Um die Dimensionsreduktion durchzuführen, wird vorwiegend von dem SVD-Verfahren aus der linearen Algebra Gebrauch gemacht. Das Schwierige bei der Dimensionsreduktion ist die Bestimmung der optimalen Anzahl an Dimensionen

doctor	-	physician	.61
doctor	-	doctors	.79
mouse	-	mice	.79
sugar	-	sucrose	.69
salt	-	NaCl	.61
sun	-	star	.35
come	-	came	.71
go	-	went	.71
walk	-	walked	.68
walk	-	walks	.59
walk	-	walking	.79
depend	-	independent	.24
...	-	...	...
random pairs			$.02 \pm .03$

Tabelle 2.1: Ähnlichkeitswerte zwischen Termen im LSI-Konzeptraum. Quelle: Landauer (2002b)

und somit der Themen bzw. Konzepte – sie darf einerseits nicht zu klein sein, da sonst wichtige Informationen verloren gehen würden; andererseits darf sie aber auch nicht zu groß werden, da sonst irrelevante Informationen in den Daten enthalten wären.
Dieser Konzeptraum ermöglicht LSI ein Problem handzuhaben, welches die natürliche Sprache für viele IR-Systeme darstellt: Polyseme und Synonyme. Es gibt viele verschiedene Möglichkeiten eine Aussage zu formulieren – laut Furnas et al. (1984) nutzt ein Mensch nur in 10 bis 20% der Fälle die selben Wörter, um ein und dasselbe Thema zu beschreiben. LSI ist in der Lage ein Dokument als relevant zur Suchanfrage zurückzuliefern, selbst wenn das Dokument keinen einzigen Term der Suchanfrage enthält. In Tabelle 2.1 ist ein Ausschnitt zu sehen, der die Ähnlichkeitswerte zwischen Termen beruhend auf den aus dem Konzeptraum gewonnenen Ergebnissen widerspiegelt. Insgesamt kann laut Deerwester et al. (1990) durch Nutzung von LSI eine Effektivitätssteigerung von 30% gegenüber IR-Verfahren, die auf Keyword-Matching beruhen, erreicht werden.

2.3 Anwendungsgebiete

LSI kann in vielen verschiedenen Gebieten der automatischen Informationsverarbeitung eingesetzt werden. Es kann die Bedeutung eines Textes nicht direkt mit Worten beschreiben, sondern zieht all seine Urteile auf Grund der Berechnungen von semantischen Ähnlichkeiten zu anderen Objekten — und gerade dadurch eröffnet sich LSI ein sehr brei-

tes Anwendungsspektrum. Die Tragweite reicht von der normalen Suchmaschine über Bewertung von Texten (Benotungssysteme, Lernunterstützungssysteme, Plagiatfindung) bis zu sprachübergreifendem IR oder auch vielen auf Ähnlichkeiten beruhenden Verfahren wie der OCR[1], Rechtschreibkorrektur, Clustering oder auch der Modellierung des menschlichen Gehirns bzw. Gedächtnisses. Ständig aktualisierte Zusammenstellungen zu Arbeiten im Gebiet LSI finden sich auf den Webseiten von Lemaire und Dessus (2005) und Telcordia Technologies (2005); eine etwas ältere Auflistung ist auf der Webseite von Berry und Dumais (2004) zu finden. Ferner beschäftigt sich das Paper Landauer et al. (1998) mit den verschiedenen Anwendungsgebieten von LSI. Es folgt ein Beispiel, um die verschiedenen Möglichkeiten von LSI aufzuzeigen:

2.3.1 Summary Street

Summary Street bietet Schülern die Möglichkeit, ihre Zusammenfassung von Texten online einzureichen, um direkt eine automatische Bewertung darüber zu erhalten und somit die Verbesserung ihrer Texte zu ermöglichen. Die Bewertung (siehe Abbildung 2.2) kann entweder abschnittsweise oder auf den ganzen Text bezogen Rechtschreibung, Stimmigkeit, Redundanz und Grammatik bewerten oder auch Plagiate oder irrelevante Sätze (siehe Abbildung 2.3) als solche kennzeichnen. Aus dem Blick des Lehrenden können verschiedene Sichten auf die Ergebnisse der Schüler geworfen werden, so z.B. ein Überblick über mehrere Teilnehmer einer Klasse wie in Abbildung 2.4.

[1] Optical Character Recognition - Zeichenerkennung

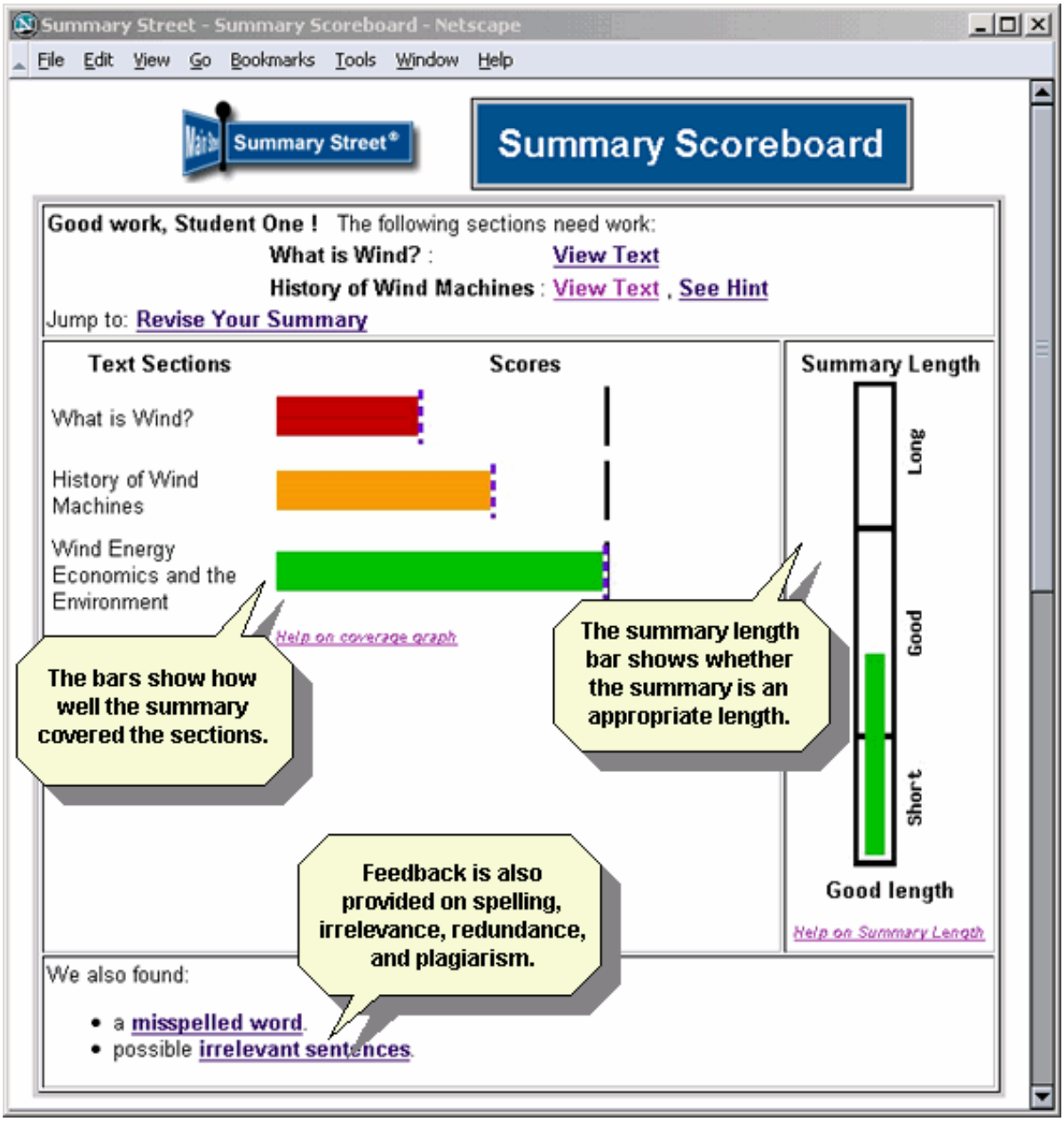

Abbildung 2.2: Screenshot: Summary Street - Schüler-Sicht: „Summary Scoreboard“ Quelle: Pearson Knowledge Technologies (2005)

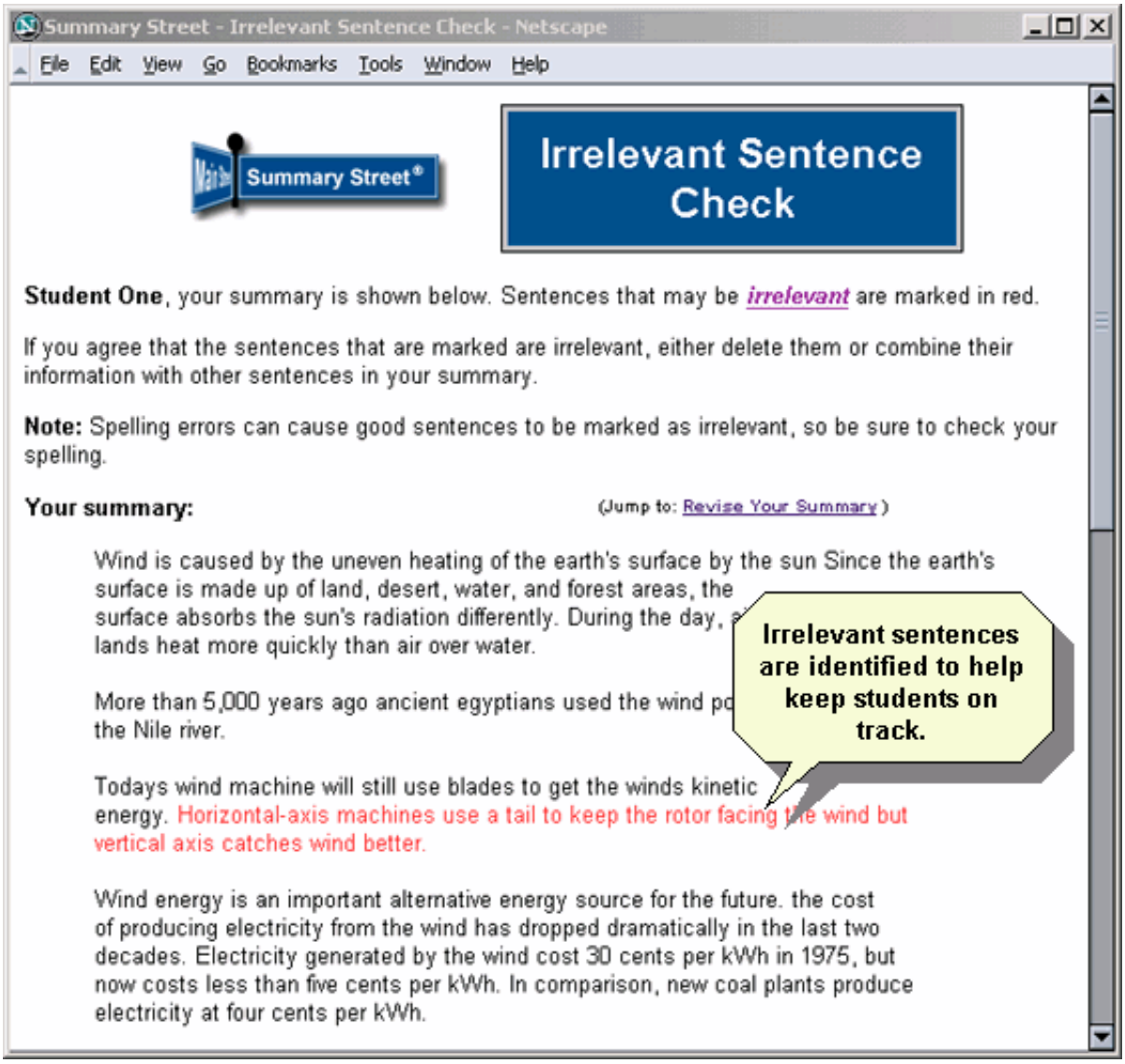

Abbildung 2.3: Screenshot: Summary Street - Schüler-Sicht: „IrrelevantSentence Check“ Quelle: Pearson Knowledge Technologies (2005)

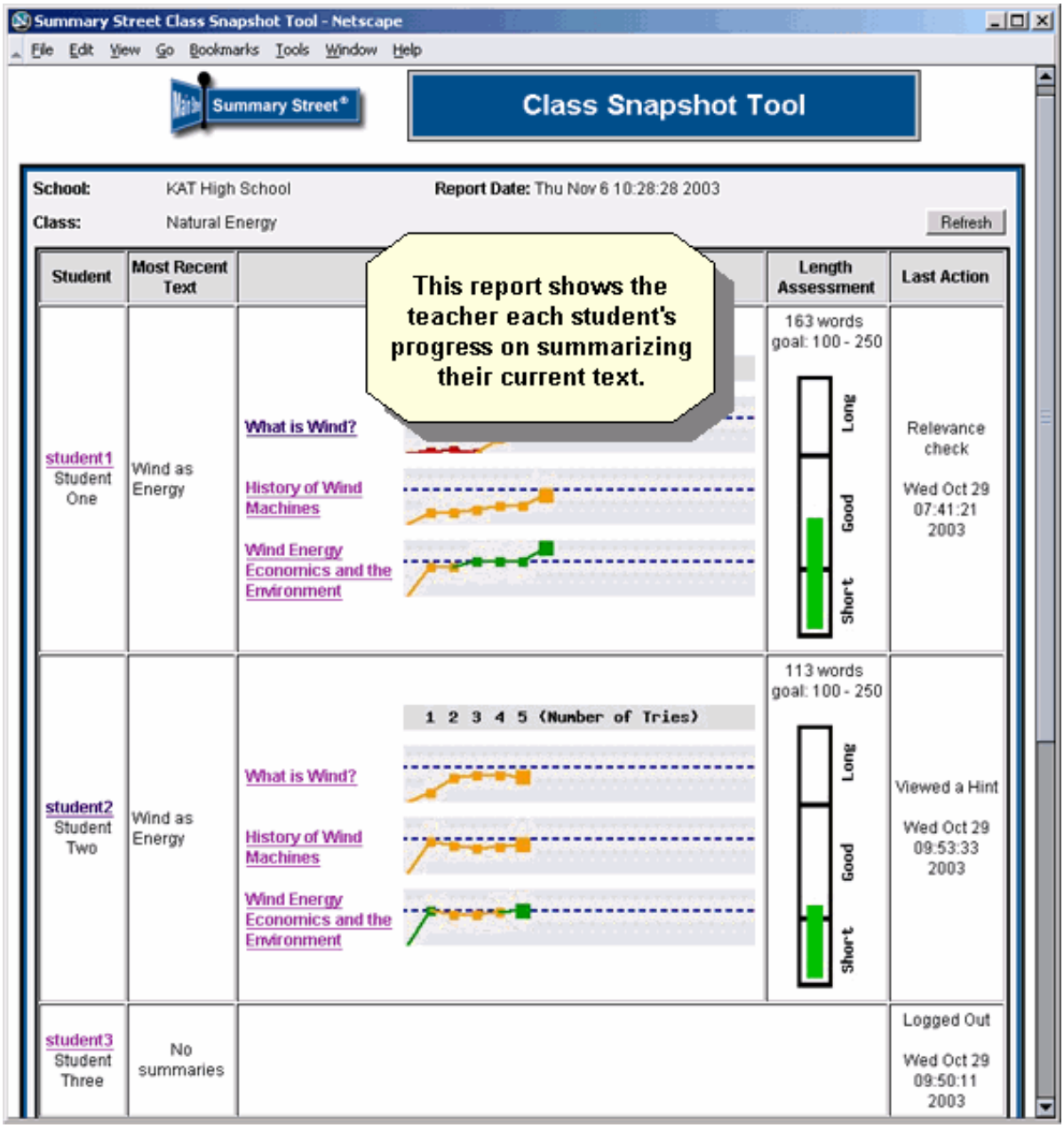

Abbildung 2.4: Screenshot: Summary Street - Lehrer-Sicht: „Class Snapshot Tool“ Quelle: Pearson Knowledge Technologies (2005)

2.4 Theoretische Grundlagen

In den beiden folgenden Unterabschnitten wird ein Abriss über informationswissenschaftliche und mathematische Grundlagen von Information-Retrieval-Systemen gegeben.

2.4.1 Informationswissenschaftliche Grundlagen

2.4.1.1 Precision, Recall und Fall-Out

Precision und Recall sind Maße, um die Güte der Ergebnisse eines IR-Systems zu beschreiben.
Precision beschreibt die Genauigkeit des Ergebnisses eines IR-Systems, indem es das Verhältnis zwischen den relevant gefundenen Dokumenten und allen Dokumente widerspiegelt:

$$Precision = \frac{relevante\ gefundene\ Dokumente}{alle\ gefundenen\ Dokumente}$$

Recall beschreibt die Vollständigkeit des Ergebnisses indem es den Anteil zwischen den relevanten gefundenen Dokumenten und allen relevanten Dokumenten berechnet:

$$Recall = \frac{relevante\ gefundene\ Dokumente}{alle\ relevanten\ Dokumente}$$

Fall-Out ermittelt die Effektivität des Suchprozesses und beschreibt den Anteil der in den Suchergebnissen enthaltenen Menge an irrelevanten Dokumenten im Verhältnis zu allen irrelevanten Dokumenten:

$$Fall - Out = \frac{irrelevante\ gefundene\ Dokumente}{alle\ irrelevanten\ Dokumente}$$

Der Idealwert für Precision und Recall ist jeweils 1 und der für Fall-Out ist 0. Die Werte Recall und Precision haben in der Praxis eine negative Korrelation untereinander.
Weiterführende Informationen können dem Skript von Griesbaum und Bekavac (2001) oder Haenelt (1997) entnommen werden.

2.4.1.2 Index

„Indexierung ist die Abbildung relevanter Informationsinhalte eines Datensatzes (z.B. einer wissenschaftlichen Publikation, einer statistischen Zeitreihe, einer Menge betriebswirtschaftlicher Kennzahlen) auf Deskriptoren (wenn ein Thesaurus eingesetzt wird), Notationen (falls mit einem Klassifikationsmodell gearbeitet wird) oder andere Terme

mit dem Zweck, den Datensatz thematisch suchbar zu machen. Das Ergebnis einer Indexierung ist der Index.“ Quelle: (Stock, 2000, S. 81).

2.4.1.3 Boolesches Modell

Das Boolesche Modell ist ein Retrieval-Modell, dass auf Mengentheorie und Boolescher Algebra basiert. Es stellt Dokumente und Anfragen als Zusammenstellung von Indextermen dar. Durch Verwendung von Booleschen Ausdrücken - wie NOT, AND, OR - bietet es dem gewöhnlichen Benutzer eines IR-Systems ein einfach zu verstehendes Vokabular. Aus diesem Grund erfreute es sich bei frühen kommerziellen bibliographischen Informationssystemen großer Beliebtheit. Quelle: (Baeza-Yates und Ribeiro-Neto, 1999, S.20, 25).

2.4.1.4 Probabilistisches Modell

Im Gegensatz zum Booleschen Modell basiert das probabilistische Modell auf Wahrscheinlichkeitsrechnung. Es wurde erstmals von Robertson und Spärck Jones (1976) in die wissenschaftliche Diskussion eingeführt. Das Modell versucht die ideale Beantwortung einer Benutzeranfrage zunächst durch Schätzen der relevanten Dokumente zu erreichen. Anschließend wird mittels ständiger Rückfrage beim Benutzer, der auswählt, welche Dokumente relevant sind, und jeweils erneuter Suche die Ergebnisliste immer weiter verfeinert, um so der idealen Beantwortung stetig näher zu kommen. Quelle: (Baeza-Yates und Ribeiro-Neto, 1999, S.20, 30).

2.4.1.5 Vektorraummodell

Retrievalverfahren, welches Ähnlichkeiten zwischen Suchanfrage und Dokument mit Hilfe der Vektorrechnung realisiert. Dokumente, Terme und Anfragen werden als Vektoren dargestellt. Das Modell ermittelt durch ein Ähnlichkeitmaß die Ähnlichkeit zwischen Vektoren - z.B. durch das Kosinus-Maß. Je kleiner der Winkel ist (Kosinus $\rightarrow 1$), um so ähnlicher sind die beiden Vektoren zueinander.

2.4.1.6 Cross-Language Information Retrieval

Ziel eines Cross-Language-Information-Retrieval-Systems ist es, auch aus einem Datenkorpus, in dem unterschiedliche Sprachen verwendet werden, Informationen extrahieren zu können.

2.4.2 Mathematische Grundlagen

Im Folgenden werden die mathematischen Grundlagen zu den Berechnungen im Vektorraum und damit den mathematischen Voraussetzungen für LSI dargestellt.

2.4.2.1 Eigenwert, Eigenvektor

„Eigenvektoren eines linearen Operators (etwa durch eine Matrize dargestellt) sind Vektoren, auf welche die Anwendung des Operators (etwa die Multiplikation mit der Matrize) ein skalares Vielfaches ihrer selbst ergeben.
Den entsprechenden Skalar nennt man Eigenwert. (Der Nullvektor wird nicht als Eigenvektor bezeichnet, obwohl er diese Eigenschaft für jeden Skalar erfüllt.)“ Quelle: Wikipedia (2005a).

2.4.3 Testkollektionen

Zur Erreichung einer möglichst objektiven Vergleichbarkeit von IR-Systemen wurden Testkollektionen erzeugt, um einheitliche Retrieval-Güte-Tests durchführen zu können. Die Kollektionen bestehen einerseits aus einer Menge von Dokumenten und eventuellen Metadaten zu diesen Dokumenten und andererseits aus Testanfragen mit einer jeweils dazugehörigen Mengenangabe mit den zur Anfrage relevanten Dokumenten. Eine Aufstellung der Testkollektionen ist in Tabelle 2.2 zu finden.

ADI	Gebiet	Bibliothekswesen
	Inhalt	Titel, Autor, Abstract
	#Dokumente	82
	#Anfragen	35
CACM	Gebiet	Artikel aus den Communications of the ACM
	Inhalt	Titel, Autor, Quelle
	#Dokumente	3203
	#Anfragen	63
CISI	Gebiet	Bibliothekswesen, Informationswissenschaften
	Inhalt	Titel, Autor, Abstract
	#Dokumente	1460
	#Anfragen	35
CRAN	Gebiet	Aerodynamik
	Inhalt	Titel, Autor, Quelle, Abstract
	#Dokumente	1398
	#Anfragen	225
GIRT	Gebiet	Sozialwissenschaften
	Inhalt	Titel, Autor, viele weitere Metadaten
	#Dokumente	151319
	#Anfragen	25
INSPEC	Gebiet	Physik, Elektrotechnik, Elektronik, Kommunikation, Computer und Informatik, Informationstechnologie
	Inhalt	Titel, Abstracts, Klassifikation, Stichwörter
	#Dokumente	12684
	#Anfragen	77
LISA	Gebiet	Bibliothekswesen, Informationswissenschaften
	Inhalt	Titel, Abstract
	#Dokumente	6004
	#Anfragen	35
MED	Gebiet	Biomedizin
	Inhalt	Titel + Abstract in einem Feld
	#Dokumente	1033
	#Anfragen	30
TIME	Gebiet	Artikel des Time-Magazine Jahrgang 1963
	Inhalt	Volltext
	#Dokumente	425
	#Anfragen	83
UCKIS	Gebiet	Chemical Abstract Service
	Inhalt	Titel
	#Dokumente	27361
	#Anfragen	182

Tabelle 2.2: Testkollektionen

3 Funktionsweise eines IR-Systems am Beispiel LSI

Um die Funktionsweise eines IR-Systems zu verdeutlichen, werden in diesem Kapitel die einzelnen Schritte, die ein beliebiges IR-System durchlaufen muss, am Beispiel von LSI dargestellt: Vorverarbeitung, Berechnungen und Anfrageverarbeitung (siehe 3.1). Hierfür wird zuerst ein Dokumentenkorpus ausgewählt - in diesem Fall wird der Datensatz (siehe folgende Seite, Tabelle 3.1) genutzt, den Deerwester et al. in ihrem Standardwerk Deerwester et al. (1990) aufgeführt haben. Die jeweiligen Datensätze bestehen der Verdeutlichung halber nur aus den Titeln zu den jeweiligen Dokumenten. Es fällt auf, dass die ersten fünf Dokumente dem Themenbereich Mensch-Computer-Interaktion zuzurechnen sind, die letzten vier Dokumente dem Themenbereich Graphentheorie. Basierend auf diesen Datensätzen werden nun zu Beginn die einzelnen Schritte der Vorverarbeitung durchlaufen: lexikalische Analyse, Stemming, Stopwortelimination, Selektion der Indexterme und Konstruktion einer Term-Kategorisierung. Diese Schritte sind nötig, um die Daten einerseits in einer für das IR-System verarbeitbaren Form aufzubereiten und andererseits, um mit verschiedenen Verfahren Effizienz und Effektivität des Systems zu steigern. Ergebnis ist der sogenannte Index. Anschließend können darauf - je nach IR-System - unterschiedliche Berechnungen durchgeführt werden, die den Index nach bestimmten Kriterien aufbereiten, um ihn auf die Zielsetzungen eines IR-Systems hin anzupassen. Hierbei bestehen die größten Unterschiede zwischen diesen Systemen, weswegen dieser Prozess auch als Herz eines jeden Systems angesehen werden kann. Sind die Berechnungen durchgeführt, ist das System bereit für die erste Anfrage. Je nach verwendetem Konzept wird hier eine Auswahl aus den unterschiedlichen Methoden getroffen, um eine Anfrage zur größtmöglichen Zufriedenheit des Nutzers abzuarbeiten.

Doc1: Human machine interface for ABC computer applications
Doc2: A survey of user opinion of computer system response time
Doc3: The EPS user interface management system
Doc4: System and human system engineering testing of EPS
Doc5: Relation of user perceived response time to error measurement
Doc6: The generation of random, binary, ordered trees
Doc7: The intersection graph of paths in trees
Doc8: Graph minors IV: Widths of trees and well-quasi-ordering
Doc9: Graph minors: A survey

Tabelle 3.1: Dokumentensammlung

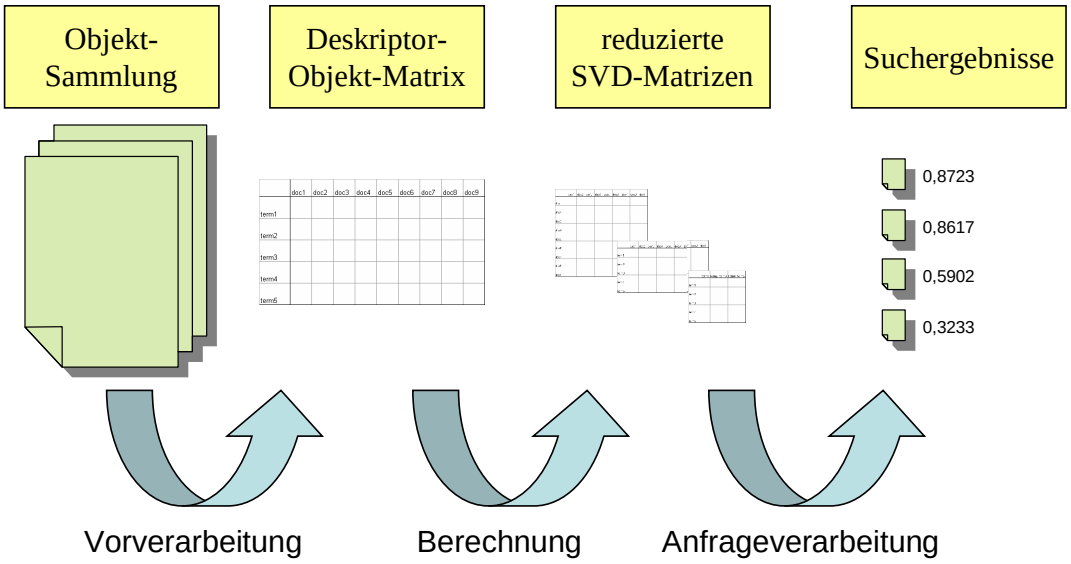

Abbildung 3.1: Verarbeitungsschritte eines IR-Systems am Beispiel LSI

3.1 Datenbeschaffung

In jedem IR-System muss dem Vorverarbeitungsprozess der Prozess der Datenbeschaffung vorgeschaltet werden. Dies wird im Allgemeinen durch Dienstprogramme (Entry-Tools, Parser für vorhandene Datensätze oder Crawler bzw. Robots) realisiert, welche die zu untersuchende Dokumentenmenge nach vorgegebenen Regeln durchforsten und speichern. Die hier verwendete Dokumentenmenge stammt aus dem Paper Deerwester et al. (1990).

3.2 Vorverarbeitung

Die Intention des Vorverarbeitungsprozesses ist es, einerseits den Dokumentenkorpus in eine für das IR-System verständliche Form aufzubereiten und andererseits Effizienz und Effektivität des IR-Systems zu steigern. Nach (Baeza-Yates und Ribeiro-Neto, 1999, S. 163-190) lässt sich der Vorverarbeitungsprozess in 5 Phasen unterteilen:

1. Lexikalische Analyse
2. Stopwortelimination
3. Stemming
4. Selektion der Indexterme
5. Konstruktion einer Term-Kategorisierung

Beim Latent Semantischen Indexieren wird auf die Konstruktion einer Term-Kategorisierung (z.B. Thesaurus, Ontologie) verzichtet, da diese über die immanente Struktur der SVD-Matrizen abgedeckt wird.
All diese Schritte haben laut (Dumais, 1990, Seite 5) jeweils nur einen geringen Einfluss (im Idealfall zwischen 1% und 5%) auf die Retrieval-Güte eines IR-Systems.

3.2.1 Lexikalische Analyse

Die lexikalische Analyse ist der Prozess des Umwandelns eines Eingabestroms von Zeichen in einen Strom von möglichen Repräsentanten für den Index eines IR-Systems und wird auch als Tokenization bezeichnet. Viele Verfahren sehen alle Zeichenfolgen, die zwischen zwei als Leerzeichen definierten Zeichen stehen, als einen Repräsentanten (Token) an. Dies reicht in den meisten Fällen jedoch nicht aus, um mit dem IR-System zufriedenstellende Ergebnisse zu erlangen. Die Frage nach der exakten Definition eines Repräsentanten lässt sich allerdings nicht so einfach beantworten:

- Wie sollen Bindestriche, Schrägstriche oder andere Sonderzeichen behandelt werden (MS-DOS, OS/2, n!)?
- Wie werden Zahlen bzw. Zählausdrücke oder gemischte Zeichenfolgen behandelt (25.3.1873, 14,3, T800-LHTEC-801)?
- Soll Groß-/Kleinschreibung beachtet werden (us $\Leftrightarrow$ US, haut $\Leftrightarrow$ Haut)?

- Wie soll mit Phrasen umgegangen werden (leg of lamb ⇒ Lammkeule)?
- Was geschieht mit im Deutschen häufig vorkommenden Komposita[1] (Donaudampfschifffahrtskapitänsmütze)?

Bei vielen Suchanfragen kann das IR-System diese Punkte ignorieren. Handelt es sich jedoch um Suchanfragen, bei denen häufig Zahlen bzw. Zählausdrücke oder gemischte Zeichenfolgen enthalten sind, z.B. Suche nach Computerteilen, die Produktbezeichnungen haben wie z.B. WD740GD, oder auch Suchanfragen mit technischen Daten mit nicht nur aus Buchstaben bestehenden Einträgen, so muss das angewandte IR-System in der Lage sein, mit einer gezielten lexikalischen Analyse auch hier relevante Daten herausfiltern zu können. Muss ein IR-System aus Texten, die einen hohen Anteil an Komposita enthalten, relevante Suchergebnisse liefern, so zieht eine Kompositazerlegung eine Senkung des Precisionwertes nach sich.
Im hier aufgeführten Beispiel von Deerwester et al. (1990) wurden nach lexikalischer Analyse Zahlen, Satz- und Sonderzeichen eliminiert, Groß-/Kleinschreibung ignoriert und Tokens als zwischen zwei Leerzeichen stehende Zeichenfolgen angesehen. Die entsprechende Dokumentenkollektion kann der Tabelle 3.2 entnommen werden.

Doc1:	human machine interface for abc computer applications
Doc2:	a survey of user opinion of computer system response time
Doc3:	the eps user interface management system
Doc4:	system and human system engineering testing of eps
Doc5:	relation of user perceived response time to error measurement
Doc6:	the generation of random binary ordered trees
Doc7:	the intersection graph of paths in trees
Doc8:	graph minors iv widths of trees and well quasi ordering
Doc9:	graph minors a survey

Tabelle 3.2: Dokumentensammlung nach lexikalischer Analyse

3.2.2 Stopwortelimination

Als Stopwort wird ein Wort bezeichnet, das für sich alleine nicht bedeutungstragend ist, häufig in einem Dokument und/oder auch häufig im ganzen Dokumentenkorpus auftreten kann. Typische Stopwörter sind Präpositionen (an, vor, durch, für, etc.), Konjunktionen (als, und, weil, falls, etc.), Pronomen (ich, er, wir, sie, etc.), Artikel (der, dem, ein, eines, etc.) und in manchen Fällen auch bestimmte geschlossene Wortklassen (Währungen,

[1] Komposita = Wort, das aus mehreren eigenständig gültigen Teilworten besteht

Monate, etc.). Eine Dokumenten-Sammlung kann nach (Baeza-Yates und Ribeiro-Neto, 1999, S. 167) durch die Elimination von Stopwörtern um teilweise mehr als 40% komprimiert werden - das Ignorieren von Stopwörtern beim Erzeugen eines Indexes kann je nach verwendetem System also enorme Speicherplatzeinsparungen und Geschwindigkeitsgewinne erwirken. Dabei geht R. Ferber in Ferber (2003) davon aus, dass Stopwörter „für die Repräsentation des Inhalts eines Dokuments nicht wichtig sind oder zumindest nicht dazu beitragen, wichtige Dokumente von unwichtigen zu unterscheiden“ und somit Stopwörter nicht zur manuellen Inhaltserschließung eines Dokuments beitragen (nach van Rijsbergen (1979) wird dies als geringe Trennschärfe bezeichnet). Somit sollte sich das Entfernen dieser Wörter nicht negativ auf die Retrievalgüte auswirken.
Welche Einflüsse die Stopwortelimination allerdings wirklich auf die Retrieval-Güte ausübt, hängt stark vom Anwendungskontext ab. Einerseits können sie eine Recall-Verschlechterung („To be or not to be“) nach sich ziehen, andererseits sind sie z.B. für eine Phrasensuche unabdingbar („State of the art“). Aus diesem Grund verzichten viele Internetsuchmaschinen beim Indizieren der Dokumentenmenge auf Stopworteliminati-on. Standardanfragen hingegen werden einer Stopwortelimination unterzogen - erst bei Phrase-Matching bzw. wenn Stopwörter extra markiert werden, ziehen sie sie zur Suchanfrage hinzu.
Sollen in einem Retrieval-System Stopwörter zur Anwendung kommen, so müssen sie in sogenannten Stopwortlisten vorgehalten werden; diese Listen enthalten in der Regel - je nach Anwendungsgebiet - laut Landauer et al. (1997) zwischen 100 und 1000 Wörter. Für beliebige Dokumentensammlungen immer die selbe Stopwortliste zu nutzen, kann und wird sich nachteilig auf die Retrieval-Güte auswirken. Da jeder Dokumentenkorpus seine eigenen Stopwörter beinhaltet - man vergleiche einen Shakespeare-Text mit einer aktuellen Internetseite -, sollte das Anpassen oder Neuerzeugen einer Stopwortliste dem Gebrauch einer vorhandenen Liste vorgezogen werden. Ein weiteres Problem tritt bei mehrsprachigen Dokumentensammlungen auf. So ist ein Wort in der einen Sprache ein Stopwort (Beispiel: deutsch - die = bestimmter Artikel ohne sinntragende Bedeutung), aber in einer anderen Sprache ein sinntragendes Wort (englisch - die = sterben). In einem mehrsprachigen Dokumentenkorpus ist also der Verzicht auf Stopwortelimination einer sprachunabhängigen Stopwortelimination vorzuziehen. Mit Hilfe von halb-/vollautomatischen Verfahren kann der Arbeitsaufwand zur Erzeugung von sowohl einsprachigen als auch mehrsprachigen Stopwortlisten enorm gesenkt werden. Folgende Verfahren zur Stopwort-Listen-Generierung stehen zur Verfügung:

- Vorhandene Listen

Es wird auf vorhandene Listen zurückgegriffen.

- Manuell
  Die Liste der Terme wird manuell auf Stopworte untersucht.

- Halbautomatisch
  Der manuelle Indexierer wird mit automatischen Verfahren unterstützt.

- Vollautomatisch
  Die Erzeugung der Stopwortliste geschieht vollautomatisch.

Das Nutzen vorhandener Stopwortlisten ist die einfachste und schnellste Methode, allerdings auch zu fehlerbehaftet. Besteht der Dokumentenkorpus z.B. aus einem Handbuch für eine SQL-Datenbank, fallen viele wichtige Terme aus dem Index (for, while, from, and). Hier sollte zumindest ein halbautomatisches Verfahren angewendet werden, in dem das Hintergrundwissen des manuellen Indexierers über den Dokumentenkorpus in die Stopwortelimination mit einfließt - automatische Verfahren haben gerade mit diesem Hintergrundwissen Probleme.
Im Falle der Beispielkollektion wird ein manuelles Verfahren verwendet, welches zu dem Ergebnis in Tabelle 3.3 führt.

Doc1:	human machin interfac ~~for~~ abc comput applic
Doc2:	~~a~~ survey ~~of~~ user opinion ~~of~~ comput syst respons tim
Doc3:	~~the~~ ep user interfac manag system
Doc4:	system ~~and~~ human system engin test ~~of~~ ep
Doc5:	relat ~~of~~ user perceiv respons time ~~to~~ error measur
Doc6:	~~the~~ generat ~~of~~ random binari order tree
Doc7:	~~the~~ intersect graph ~~of~~ path ~~in~~ tree
Doc8:	graph minor iv width ~~of~~ tree ~~and~~ well quasi order
Doc9:	graph minor ~~a~~ survey

Tabelle 3.3: Dokumentensammlung nach Stopwortelimination

3.2.3 Stemming

Unter Stemming (englisch für Stamm, Abstammung) versteht man das Verfahren, lexikalisch verwandte Wortformen auf eine grammatische Grund- (auch „weak stemming“) bzw. Stammform (auch „strong stemming“) zu reduzieren. Die Idee dahinter ist, dass die Grund- bzw. Stammform die Bedeutung des Konzeptes trägt und die verschiedenen

Wortformen dieses Konzept nur leicht modifizieren bzw. syntaktische Informationen beitragen. Ein Verlust dieser Information sollte bei einem IR-System vernachlässigbar sein und wird als Vorteil Speicherplatz- und Rechenzeiteinsparungen nach sich ziehen.
Die verschiedenen Wortformen können je nach Sprache durch syntaktische Variation in Deklination, Konjugation und Komparation entstehen. Nomina können nach Kasus (der Mann ⇔ des Mannes), Numerus (der Mann ⇔ die Männer) und Genus (Bsp. Aus dem Swahili: Wa-tu ⇔ wa-refu) dekliniert, Verben nach Person (er rennt ⇔ du rennst), Numerus (sie rennt ⇔ wir rennen), Modus (es rennt ⇔ renn!), Tempus (sie rennt ⇔ sie rannte), Genus (er trifft ⇔ er wird getroffen) konjugiert und Adjektive und Adverbien schließlich nach Steigerungsform (schnell ⇔ schneller ⇔ am schnellsten) verglichen werden. Ferner können neue Worte auch durch Derivation (greifen ⇔ un-begreif-lich) oder Komposition (Dampf, Schiff ⇔ Dampfschiff) gebildet werden. Um diese verschiedenen Wortformen auf ihre Grund- bzw. Stammform zu reduzieren, existieren verschiedene Methoden, siehe Frakes und Baeza-Yates (1992):

- Statistisch (z.B. N-Gram);
- Computerlinguistisch (z.B. Part-Of-Speech-Tagging, Successor variety stemming);
- Datenbankbasiert (z.B. Lexikon);
- Regelbasiert (z.B. Affix-Removal).

Am Weitesten verbreitet ist das Affix-Removal-Verfahren, da die Implementation sehr einfach ist, sehr viele Beispielprogramme existieren und es das schnellste der oben genannten Verfahren ist. Die zugrunde liegenden Algorithmen wurden in Kuhlen (1977) und Porter (1980) beschrieben. Sie basieren auf wenigen Verkürzungsregeln, die nacheinander auf ein Wort angewendet werden. Bei Sprachen mit größtenteils regelmäßiger Morphologie, wie dem Englischen, funktioniert dieses Verfahren relativ gut: hohe Reduzierungsrate, Anzahl der Fehler relativ gering. Sprachen hingegen wie das Deutsche bereiten mit ihrem hohen Anteil an unregelmäßiger Morphologie diesem Verfahren Probleme. Bessere Ergebnisse erzielt das Einbinden eines Lexikons - wobei dann wiederum die hohe Geschwindigkeit und die einfache Implementierung in Mitleidenschaft gezogen werden und damit ein großer Vorteil der rein regelbasierten Verfahren verloren geht.
Nähere Informationen zu dem in der Programmiersprache Snowball geschriebenen Stemmer findet sich unter http://snowball.tartarus.org/texts/introduction.html.
In unserem Beispiel wird der von Martin Porter entwickelte Snowball-Stemmer (siehe Porter (2001)) genutzt, der uns zu folgendem Ergebnis führt:

Doc1:	human machine interface for abc computer applications
Doc2:	a survey of user opinion of computer system response time
Doc3:	the eps user interface management system
Doc4:	system and human system engineering testing of eps
Doc5:	relation of user perceived response time to error measurement
Doc6:	the generation of random binari ordered trees
Doc7:	the intersection graph of paths in trees
Doc8:	graph minors iv widths of trees and well quasi ordering
Doc9:	graph minors a survey

Tabelle 3.4: Dokumentensammlung nach Stemming

3.2.4 Selektion der Indexterme

Die Selektion der Indexterme bestimmt, welche Objekte (einzelner Term oder Phrase) zu einem Index hinzugefügt werden, um das jeweilige Dokument zu repräsentieren. Es existieren drei verschiedene Verfahren:

- Volltext
  alle Terme werden zum Index hinzugefügt;
- Filterbasiert
  nur Terme, die gewissen Filterregeln entsprechen bzw. nicht entsprechen, werden zum Index hinzugezogen;
- Lexikonbasiert
  es werden z.B. nur Nomen bzw. Phrasen bestehend aus Nomen (sog. Noun Groups) zum Index hinzugefügt, die auch in einem Lexikon beinhaltet sind.

Welches dieser Verfahren bei einem IR-System zum Einsatz kommt, hängt wieder vom Anwendungskontext ab. Die meisten Internetsuchmaschinen vertrauen auf die Volltextindizierung (Yu et al. (2002)), das IR-System INQUERY (in Broglio et al. (1994) beschrieben) nutzt das lexikonbasierte Verfahren und LSI schließlich die filterbasierte Lösung. Die Filterregeln für LSI sehen wie folgt aus:

- Term kommt in mindestens einem Dokument vor.
- Term kommt nicht in allen Dokumenten vor.

Zusätzliche Regeln können z.B. wie folgt lauten:
„Term muß aus mindestens zwei Zeichen bestehen“ oder
„Term darf aus maximal 20 Zeichen bestehen“.

Dieses Beispiel verdeutlicht die zwei in der Aufzählung angegebenen Regeln: beide tragen nicht zur Unterscheidung von Dokumenten bei, sie vergrößern bei Nichtanwendung den Index und verlängern die späteren Berechnungen. Dafür wird in Kauf genommen, dass nicht mehr nach diesen Termen gesucht werden kann.
Die entsprechende Selektion der Indexterme führt im Beispiel von Deerwester et al. (1990) zu dem in Tabelle 3.5 aufgeführten Ergebnis.

Doc1:	human ~~machin~~ interfac ~~abc~~ comput ~~applic~~
Doc2:	survey user ~~opinion~~ comput syst respons tim
Doc3:	ep user interfac ~~manag~~ system
Doc4:	system human system ~~engin~~ ~~test~~ ep
Doc5:	~~relat~~ user ~~perceiv~~ respons time ~~error~~ ~~measur~~
Doc6:	~~generat~~ ~~random~~ ~~binari~~ ~~order~~ tree
Doc7:	~~intersect~~ graph ~~path~~ tree
Doc8:	graph minor ~~iv~~ ~~width~~ tree ~~well~~ ~~quasi~~ ~~order~~
Doc9:	graph minor survey

Tabelle 3.5: Dokumentensammlung nach Selektion der Indexterme

3.3 Berechnungen

Nach Beendigung der in Kapitel 3.2 aufgezeigten Vorverarbeitungsschritte wird mit Hilfe der gewählten Indexterme die für die SVD-Berechnungen benötigte Term-Dokument-Matrize (siehe Tabelle 3.6) erzeugt. Dumais (1990) zeigt in verschiedenen Arbeiten, dass sich die Retrieval-Güte eines IR-Systems durch die Hinzunahme eines Verfahrens, das auf die reine Auftretenshäufigkeit eines Terms angewendet wird, weiter steigern lässt: die sogenannte Termgewichtung. Hierdurch lassen sich in IR-Systemen Verbesserungen von durchschnittlich bis zu 40% Erhöhung des Precision-Wertes erreichen (Dumais, 1990, S.11). Sie haben also oft mehr Einfluss auf das Ergebnis als ein Austausch des eigentlichen Retrieval-Verfahrens. Nach der Implementierung eines Termgewichtungsalgorithmus erfolgt die Implementierung des eigentlichen Retrieval-Algorithmus, zum Beispiel des Boolschen Modells, des probabilistischen Modells oder des Vektorraummodells. In diesem Beispiel wird das auf dem Vektorraummodell basierende Verfahren LSI behandelt. Es nutzt als Dimensionsreduktionverfahren die SVD auf dessen einzelne Schritte abschließend näher eingegangen wird.

Term	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7	Doc8	Doc9
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
eps	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

Tabelle 3.6: Term-Dokument-Matrize

3.3.1 Termgewichtung

Eines der meistgenutzten Verfahren, um die Retrieval-Güte eines vektorraummodellbasierten IR-Systems weiter zu steigern, ist die Termgewichtung. Da nicht jeder Term in einer Dokumentenkollektion von gleicher Bedeutung ist, existieren verschiedene Methoden, die Terme eines Dokuments (oder später einer Anfrage) zu gewichten. Abbildung

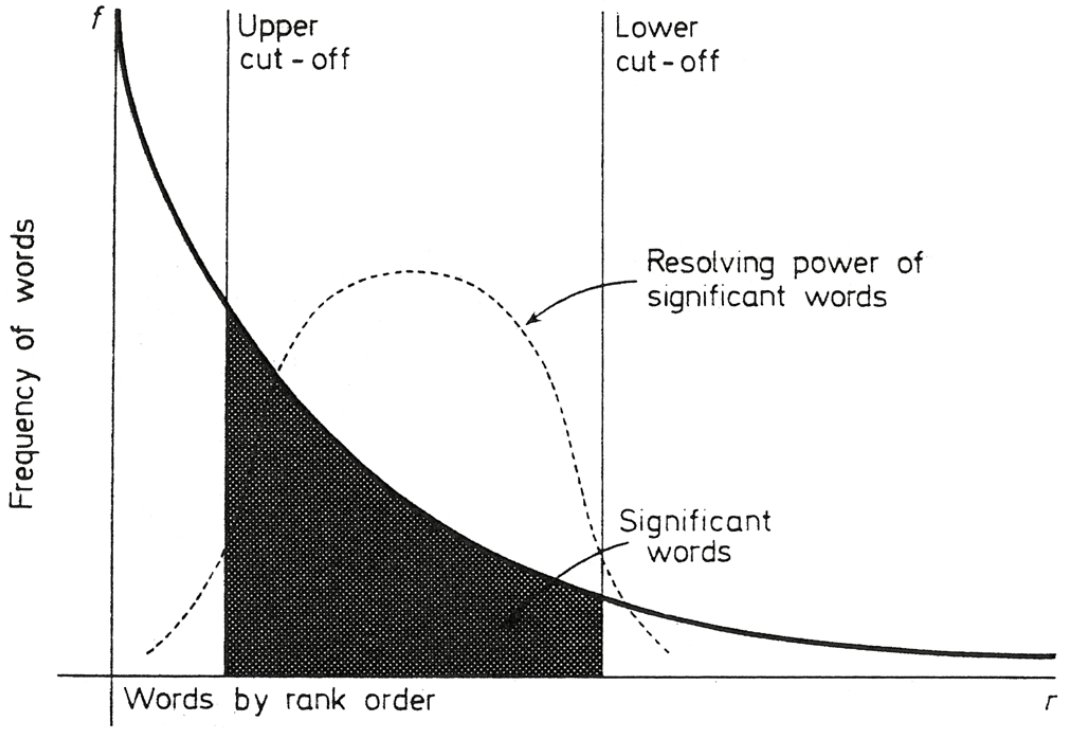

Abbildung 3.2: Term-Rang-Frequenz-Diagramm Quelle: (van Rijsbergen, 1979, S. 16)

3.2 verdeutlicht die Beziehungen zwischen Auftretenshäufigkeit eines Terms und dessen Wichtigkeit für das Dokument - die verschiedenen Gewichtungsmaße versuchen die Signifikanz, die in der Abbildung durch die gepunktete Normalverteilung dargestellt ist, nachzubilden.
Folgende Syntax wird verwendet:

- i steht für den i-ten Term in der Term-Dokument-Tabelle.
- j steht für das j-te Dokument.
- n für die Anzahl der Terme.
- m für die Anzahl der Dokumente.
- tp steht für die Position des Termes im Dokument.
- tl für die Länge eines Terms.
- tf entspricht der Termfrequenz.
- tb zeigt auf, ob ein Term in einem Dokument vorkommt.

- df beinhaltet die Anzahl der Dokumente, die den Term enthalten.
- cf ist die Corpusfrequenz, d.h. die Termhäufigkeit in der Dokumentensammlung.

Die Abschnitte 3.3.1.1 und 3.3.1.2 zeigen einige der gebräuchlichsten Verfahren, die sich in folgende Gruppen unterteilen lassen:

3.3.1.1 Lokale Gewichtungsmaße

Die lokalen Gewichtungsmaße, auch als kontextabhängige Gewichtungsmaße bzw. Within-Object-Gewichtungsmaße bezeichnet, wirken sich nur lokal auf einen einzelnen Term eines Dokuments aus - Beispiel die Term-Dokument-Matrize A (siehe 3.6): hier wirken diese Maße nur auf eine einzelne Zelle.
Folgende Syntax wird verwendet:

- Termfrequenz: tf_{ij}
  Wie oft kommt ein Term in einem Dokument vor.
- Bool: tb_{ij}
  Kommt ein Term in einem Dokument vor oder nicht.
- log(Termfrequenz + 1): $\log(tf_{ij} + 1)$
  Gleicht der einfachen Termfrequenz, jedoch werden die Effekte stark variierender Auftretenshäufigkeiten der Terme in unterschiedlichen Dokumenten durch den Logarithmus gedämpft.
- Termposition: $tp_{ij} * \omega_i$
  Gewichtung ist abhängig von der Termposition (z.B. Überschrift, Nachrichtenmeldungen).

3.3.1.2 Globale Gewichtungsmaße

Die globalen Gewichtungsmaße, auch als kontextunabhängige Gewichtungsmaße oder Between-Object-Gewichtungsmaße bezeichnet, wirken sich immer auf mehrere Einträge in der Term-Dokument-Matrize aus. Sie analysieren Beziehungen unter den Termen bzw. Dokumenten und wirken zeilen- bzw. spaltenweise. Ziel der globalen Maße ist, dass häufig vorkommende Terme weniger stark gewichtet werden und selten vorkommende stärker gewichtet werden.
Folgende Syntax wird verwendet:

- Normalisierte Worthäufigkeit: $\frac{tf_{ij}}{tfmax_{ij}}$
  Die Auftretenshäufigkeit eines Terms im Dokument wird durch die Auftretenshäufigkeit des am häufigsten vorkommenden Terms geteilt und somit werden längere Texte normalisiert.

- Dokumentenhäufigkeit: df_i
  In wie vielen Dokumenten tritt der Term auf.

- Termlänge: $tl_{ij} * \omega_{tl}$
  Die Gewichtung des Terms hängt von seiner Länge ab - je länger ein Term ist, desto stärker wird er gewichtet.

- Inverse Dokumentenhäufigkeit (IDF): $\frac{m}{df_i}$
  Terme werden invers gewichtet, indem die Anzahl aller Dokumente durch die Anzahl der Dokumente geteilt wird, die den Term enthalten. Somit wird im Dokumentenkorpus selten vorkommenden Termen ein hohes Gewicht zugeteilt und häufig vorkommenden ein niedriges. Manchmal wird auch logarithmiert, um die hohe Gewichtung sehr seltener Terme abzuschwächen: $\ln\left(\frac{\#docs}{df_i}\right) + 1$.

- Tf-Idf: $tf_{ij} * idf_i$
  Termhäufigkeit wird gewichtet, indem sie mit der IDF des Terms multipliziert wird. Terme, die selten im Dokumentenkorpus vorkommen aber häufig in einem Dokument, werden mit Tf-Idf stärker gewichtet. Sehr häufig genutztes Termgewichtungsmaß.

- Globale Häufigkeit / Dokumentenhäufigkeit: $\frac{cf_i}{df_i}$
  Verhältnis zwischen dem Vorkommen im Dokumentenkorpus und dem Vorkommen im Dokument selbst.

- 1 - Entropie: $1 - \sum_{j=1}^{m} \frac{\frac{tf_{ij}}{cf_{ij}} * \log\left(\frac{tf_{ij}}{cf_{ij}}\right)}{\log m}$
  1 - Entropie weist häufigen Termen ein niedriges Gewicht zu und seltenen Termen ein hohes.

Die Tabelle 3.7 aus (Dumais, 1990, Seite 11) listet die Auswirkungen der verschiedenen Gewichtungsmaße auf die Ergebnisse eines auf LSI beruhenden IR-Systems auf.
Weiterführende Literatur: Dumais (1990); Spärck Jones (1972); William M. Shaw et al. (1997); Spärck Jones (1997); Jung et al. (2000); Wu und Salton (1981); Salton (1971); Noreault et al. (1981).

	TF Raw	TF Normal		TF GfIdf		TF Idf		TF Entropy		Log(TF) Entropy	
ADI	.28	.30	+7%	.24	-14%	.34	+22%	.36	+30%	.36	+30%
MED	.52	.48	-6%	.55	+6%	.67	+30%	.66	+27%	.72	+39%
CISI	.11	.10	-6%	.10	-7%	.15	+36%	.16	+46%	.17	+53%
CRAN	.29	.25	-15%	.28	-5%	.40	+37%	.40	+38%	.46	+57%
TIME	.49	.32	-35%	.42	-14%	.55	+12%	.54	+10%	.59	+20%
mean			-11%		-7%		+27%		+30%		+40%

Tabelle 3.7: Auswirkungen verschiedener Termgewichtung

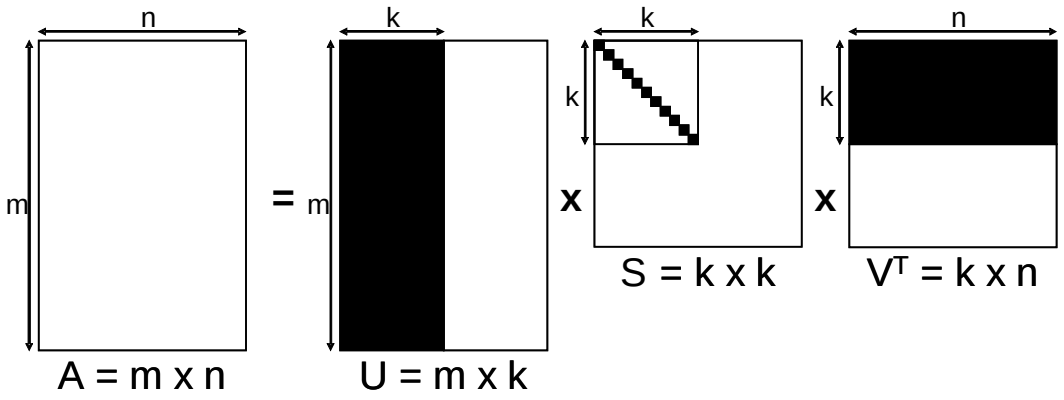

Abbildung 3.3: Aufbau und Dimension der SVD-Matrizen

3.3.2 Singulärwertzerlegung

Die Singulärwertzerlegung ist ein Verfahren aus der linearen Algebra und wird bei LSI als Dimensionsreduktionsschritt genutzt. Es kann eine beliebige $m \times n$ Matrize A in das Produkt dreier Matrizen U $(m \times m)$, S $(m \times n)$ und V $(n \times n)$ zerlegen:

$$A = U * S * V^T$$

U und V sind orthogonale Matrizen; S ist eine Diagonalmatrize. Die Diagonalelemente der Matrize S werden als „Singulärwerte“ bezeichnet.
Werden die durch die SVD-Berechnungen gewonnenen Matrizen nun auf eine vorgegebene Anzahl k reduziert ($U \rightarrow m \times k$, $S \rightarrow k \times k, V' \rightarrow k \times n$) (siehe Abbildung 3.3), so erhält man durch die Multiplikation der Matrizen eine Annäherung der ursprünglichen A.
Laut Wiemer-Hastings (1999) erzielt LSI seine hohe Retrievel-Performance hauptsächlich durch die SVD.

3.3.2.1 Berechnungen der SVD

Ausgangspunkt für die SVD-Berechnungen ist die Term-Dokument-Matrize A, wie sie in Tabelle 3.6 zu sehen ist. Die folgenden Berechnungen werden am Beispiel der Matrize A, die auch in der Arbeit Deerwester et al. (1990) genutzt wurde, durchgeführt.

$$A = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Zunächst müssen die Matrizen *tempU* und *tempV* berechnet werden. Matrize *tempU* ergibt sich aus der Multiplikation der Matrize A mit der transponierten Matrize A'. Sie wird auch als „Term-Term-Matrize“ bezeichnet, denn die resultierende Matrize *tempU* spiegelt die Beziehungen zwischen den Termen untereinander wider.

$$tempU = A * A' = \begin{pmatrix} 2 & 1 & 1 & 0 & 2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 2 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 3 & 2 & 2 & 2 & 1 & 1 & 0 & 0 & 0 \\ 2 & 1 & 1 & 2 & 6 & 1 & 1 & 3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 2 & 2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 2 & 2 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 3 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 2 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 2 & 2 \end{pmatrix}$$

Die gleiche Multiplikation wird anschließend für die Matrize *tempV* durchgeführt, jedoch in umgekehrter Reihenfolge. Somit ergeben sich auch andere Dimensionen für die resultierende Matrize *tempV* - sie spiegelt die Dokument-Dokument-Ähnlichkeit wider.

$$tempV = A' * A = \begin{pmatrix} 3 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 6 & 2 & 2 & 3 & 0 & 0 & 0 & 1 \\ 1 & 2 & 4 & 3 & 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 3 & 6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 1 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 3 & 2 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 2 & 3 \end{pmatrix}$$

Anschließend berechnet man die Singulärwerte, indem man die Eigenwerte λ_i einer[2] der beiden temporären Matrizen (um Rechenzeit zu sparen vorzugsweise von der kleineren Matrize - in unserem Beispiel also *tempV*) berechnet (siehe Gleichung 3.1), die positive Wurzel jeden Eigenwerts zieht (siehe Gleichung 3.2) und sie anschließend absteigend anordnet. Die Singulärwerte werden schließlich auf den Diagonalelementen der Matrize S (Gleichung 3.3) abgetragen - die Matrize spiegelt die Aussagekraft der Themen wider.

$$eigenwerte(tempV) = \begin{pmatrix} 0.1323 \\ 0.3138 \\ 0.7156 \\ 1.7066 \\ 2.2645 \\ 2.7045 \\ 5.5411 \\ 6.4602 \\ 11.1615 \end{pmatrix} \tag{3.1}$$

[2]Eigenwerte von $(A * B)$ = Eigenwerte von $(B * A)$

$$\Rightarrow \sqrt{(eigenwerte(tempV)} = \begin{pmatrix} 0.3637 \\ 0.5601 \\ 0.8459 \\ 1.3064 \\ 1.5048 \\ 1.6445 \\ 2.3539 \\ 2.5417 \\ 3.3409 \end{pmatrix} \tag{3.2}$$

$$\Rightarrow S = \begin{pmatrix} 3.3409 & & & & & & & & \\ & 2.5417 & & & & & & & \\ & & 2.3539 & & & & & & \\ & & & 1.6445 & & & & & \\ & & & & 1.5048 & & & & \\ & & & & & 1.3064 & & & \\ & & & & & & 0.8459 & & \\ & & & & & & & 0.5601 & \\ & & & & & & & & 0.3637 \end{pmatrix} \tag{3.3}$$

Der letzte Berechnungsschritt der SVD ist die Aufstellung der Matrizen U und V. Hierzu berechnen wir die Eigenvektoren der Matrizen *tempU* und *tempV* (Gleichung 3.4), um sie anschließend - sortiert nach den Eigenwerten (Gleichung 3.1) - spaltenweise anzuordnen. Das Rechenbeispiel wird aus Platzgründen nur für V durchgeführt und auf zwei Nachkommastellen gekürzt (die vollständigen Matrizen sind auf der beigefügten CD-ROM zu finden):

$$\Rightarrow eigenvektoren(tempV) =$$

$$\begin{pmatrix} 0.06 & 0.01 & -0.18 & 0.08 & -0.05 & -0.95 & 0.11 & -0.06 & 0.20 \\ -0.24 & -0.05 & 0.43 & 0.26 & 0.21 & -0.03 & -0.50 & 0.17 & 0.61 \\ -0.02 & -0.01 & 0.24 & -0.72 & -0.38 & 0.04 & 0.21 & -0.13 & 0.46 \\ 0.08 & 0.02 & -0.26 & 0.37 & 0.21 & 0.27 & 0.57 & -0.23 & 0.54 \\ 0.26 & 0.06 & -0.67 & -0.03 & -0.33 & 0.15 & -0.51 & 0.11 & 0.28 \\ 0.62 & -0.45 & 0.34 & 0.30 & -0.39 & 0.02 & 0.10 & 0.19 & 0.00 \\ -0.02 & 0.76 & 0.15 & 0.21 & -0.35 & 0.02 & 0.19 & 0.44 & 0.01 \\ -0.52 & -0.45 & -0.25 & 0.00 & -0.15 & 0.01 & 0.25 & 0.62 & 0.02 \\ 0.45 & 0.07 & -0.04 & -0.36 & 0.60 & -0.02 & 0.08 & 0.53 & 0.08 \end{pmatrix} \tag{3.4}$$

$$\Rightarrow V = \begin{pmatrix} 0,20 & -0,06 & 0,11 & -0,95 & -0,05 & 0,08 & -0,18 & 0,01 & 0,06 \\ 0,61 & 0,17 & -0,50 & -0,03 & 0,21 & 0,26 & 0,43 & -0,05 & -0,24 \\ 0,46 & -0,13 & 0,21 & 0,04 & -0,38 & -0,72 & 0,24 & -0,01 & -0,02 \\ 0,54 & -0,23 & 0,57 & 0,27 & 0,21 & 0,37 & -0,26 & 0,02 & 0,08 \\ 0,28 & 0,11 & -0,51 & 0,15 & -0,33 & -0,03 & -0,67 & 0,06 & 0,26 \\ 0,00 & 0,19 & 0,10 & 0,02 & -0,39 & 0,30 & 0,34 & -0,45 & 0,62 \\ 0,01 & 0,44 & 0,19 & 0,02 & -0,35 & 0,21 & 0,15 & 0,76 & -0,02 \\ 0,02 & 0,62 & 0,25 & 0,01 & -0,15 & 0,00 & -0,25 & -0,45 & -0,52 \\ 0,08 & 0,53 & 0,08 & -0,02 & 0,60 & -0,36 & -0,04 & 0,07 & 0,45 \end{pmatrix} \tag{3.5}$$

Weiterführende Literatur: Strang (2003); Golub und Loan (1989); Oksa et al. (2002)

3.3.2.2 SVD-Updating & Fold-In

Die SVD-Berechnungen brauchen bei großen Matrizen sehr viel Arbeitsspeicher und Rechenzeit. Bei sich oft ändernden Datenkollektionen ist eine komplette Neuberechnung oft nicht möglich. Es wurden zwei Verfahren entwickelt, die auf den schon berechneten Matrizen U, S und V arbeiten und ein nahezu gleiche Matrize erzeugen, wie wenn eine komplette Neuberechnung der SVD durchgeführt werden würde. Diese Verfahren ermöglichen auch das Generieren von Matrizen, deren Berechnungen vorher arbeitsspeicher- bzw. rechenzeitlimitiert waren.
Weiterführende Literatur:
Zha und Simon (1999); Götze und Rieder (1996); Manning und Schütze (1999)

3.4 Anfrageverarbeitung

Anfragen an ein IR-System werden den gleichen Vorverarbeitungsschritten unterworfen, die auch jedes Dokument durchlaufen musste, um in den Index aufgenommen zu werden. Bei Vektorraummodell-basierten Lösungen wird die Anfrage, nachdem sie die einzelnen Phasen durchlaufen hat, ebenfalls zu einem Vektor und kann genauso behandelt werden, wie alle anderen indexierten Dokumente oder Terme.

3.4.1 Ähnlichkeitsmaße

Ähnlichkeitsmaße im Vektorraummodell zeigen die Ähnlichkeit zwischen zwei Objekten auf und versuchen, diese durch einen numerischen Wert widerzuspiegeln. Es kann eine Ähnlichkeitsberechnung zwischen Dokumenten, zwischen Dokumenten und Anfragen, aber auch zwischen Termen durchgeführt werden - auf eine Term-Dokument-Matrize bezogen bedeutet dies eine spalten- (dokumentweise) bzw. zeilenweise (termweise) Berechnung. Ein perfektes, für alle Szenarien gültiges Ähnlichkeitsmaß kann aus tatsächlichen Gründen nicht existieren, weshalb eine Vielzahl von Algorithmen entwickelt wurde, die auf verschiedene Art und Weise versuchen, dem „perfekten“ Algorithmus näher zu kommen. Allein in der Arbeit von Noreault et al. (1981) wurden 67 Ähnlichkeitsmaße beschrieben, inzwischen sind weitere hinzugekommen (Spärck Jones (1997); Jung et al. (2000)). Im Folgenden werden die gebräuchlichsten Ähnlichkeitsmaße aufgezeigt (orientiert an den Arbeiten von Ferber (2003); Haenelt (2004); Anderberg (1973)).

3.4.1.1 Korrelationsmaße

Korrelationsmaße versuchen mit Hilfe verschiedener Berechnungen, die auf Korrelation (z.B. gleicher Winkel, gleiche Länge) zwischen Vektoren beruhen, Vektor-Ähnlichkeit durch einen Wert auszudrücken. Dieser Wert liegt meistens zwischen 0 und 1, kann aber auch abweichend davon sein (beim Kosinus liegen die Werte z.B. zwischen -1 und 1). Immer jedoch gilt: Je ähnlicher sich Dokumente sind, desto eher tendiert der Wert gegen 1.
Mögliche Korrelationsmaße sind:

- Matching: $match(d_j, d_k) = |d_j \cap d_k|$
  Der Matching Koeffizient ist die Anzahl der Terme, die in beiden Dokumenten enthalten sind.

- Skalarprodukt (Dot-Product): $dot(d_j, d_k) = \vec{d_j} \bullet \vec{d_k} = \sum_{i=1}^{n} w_{ij} * w_{ik}$
  Das Skalarprodukt wird durch Summe der multiplizierten Vektor-Dimensionen berechnet. Die Auftretenshäufigkeit der Terme, die in beiden Dokumenten vorkommen, wird miteinander multipliziert und die einzelnen Produkte dann aufsummiert.

- Dice-Koeffizient: $dice(d_j, d_k) = 2 * \frac{|d_j \cap d_k|}{|d_j| + |d_k|} = 2 * \frac{\sum_{i=1}^{n} w_{ij} * w_{ik}}{\sum_{i=1}^{n} w_{ij} + \sum_{i=1}^{n} w_{ik}}$
  Der Dice-Koeffizient normalisiert das Skalarprodukt, indem der Durchschnitt der beiden Termmengen gebildet wird.

- Pseudo-Kosinus-Maß: $pseudo\_cos(d_j, d_k) = \frac{|d_j \cap d_k|}{|d_j| * |d_k|} = \frac{\sum_{i=1}^{n} w_{ij} * w_{ik}}{\sum_{i=1}^{n} w_{ij} * \sum_{i=1}^{n} w_{ik}}$
  Das Pseudo-Kosinus-Maß teilt das Skalarprodukt durch das Produkt der jeweils aufsummierten Vektor-Dimensionen der beiden Dokumente.

- Kosinus-Maß: $cos(\vec{d_j}, \vec{d_k}) = \frac{|d_j \cap d_k|}{\sqrt{|d_j| \times |d_k|}} = \frac{\sum_{i=1}^{n} w_{ij} * w_{ik}}{\sqrt{\sum_{i=1}^{n} w_{ij}^2 * \sum_{i=1}^{n} w_{ik}^2}}$
  Das Kosinus-Maß spiegelt den Winkel der beiden Vektoren wider und ist Längeninvariant.

- Overlap-Maß: $overlap(d_j, d_k) = \frac{min|d_j, d_k|}{min(|d_j|, |d_k|)} = \frac{\sum_{i=1}^{n} min(w_{ij}, w_{ik})}{min\left(\sum_{i=1}^{n} w_{ij}, \sum_{i=1}^{n} w_{ik}\right)}$
  Ähnlich dem Pseudo-Kosinus-Maß, hier wird jedoch das Produkt der Summen jeweils durch das Minimum der Summanden ersetzt.

- Jaccard-Koeffizient: $jaccard(d_j, d_k) = \frac{|d_j \cap d_k|}{|d_j \cup d_k|} = \frac{\sum_{i=1}^{n} w_{ij} * w_{ik}}{\sum_{i=1}^{n} w_{ij} + \sum_{i=1}^{n} w_{ik} - \sum_{i=1}^{n} w_{ij} * w_{ik}}$
  Ähnlich dem Dice-Koeffizienten; im Unterschied zu diesem beinhaltet der Zähler die Vereinigungsmenge der Termmengen der beiden Dokumente.

In einem sehr interessanten Ansatz in der Arbeit von Jones und Furnas (1987) wurde versucht, die Korrelationsmaße geometrisch zu interpretieren und somit dem IR-Experten ein Werkzeug an die Hand zu geben, mit dem die Wahl eines passenden Ähnlichkeitsmaßes unterstützt werden kann. Der Versuch, die Evaluation eines Korrelationsmaßes zu umgehen, scheiterte - allerdings ließen sich Schlüsse auf die Eigenschaften der Maße ziehen, weshalb die Diagramme die Vorauswahl eines Korrelationsmaßes entscheidend erleichtern können.

3.4.1.2 Distanzmaße

Distanzmaße bewerten Ähnlichkeit durch den Abstand zwischen zwei Vektoren - je niedriger der Wert, um so ähnlicher sind die zwei Vektoren. Ihr Wertebereich liegt zwischen 0 und ∞ - wird oft aber zwischen 0 und 1 normalisiert. Die meisten Maße basieren auf der sog. Minkowski-Metrik $L_q(\vec{d_j}, \vec{d_k}) = \sqrt[q]{\sum_{i=1}^{n} (w_{ij} - w_{ik})^q}$, aus ihr lassen sich direkt die Manhatten-, Euklid- und die Maximumsdistanz ableiten.

- Manhattan (L1): $manhattan(\vec{d_j}, \vec{d_k}) = \sum_{i=1}^{n} (|w_{ij} - w_{ik}|)$
  Auch als City-Block bezeichnet. Hier wird die Distanz so gemessen, als ob man um einen quadratischen Häuserblock gehen müßte, um einen Punkt zu erreichen.

- Euklid (L2): $euklid(\vec{d_j}, \vec{d_k}) = \sqrt{\sum_{i=1}^{n} (w_{ij} - w_{ik})^2}$
  Die Euklidsche Distanz ist diejenige, die - wiederum auf einen Häuserblock bezogen - der Luftlinie entsprechen würde.

- Maximum (L∞): $max(\vec{d_j}, \vec{d_k}) = max\{|w_{ij} - w_{ik}|\}$
  Bei der Maximumsdistanz wird einfach die größte Ausprägung eines Vektors als Distanzmaß verwendet.

- gewichtete Euklid: $w\_euklid(\vec{d_j}, \vec{d_k}) = \sqrt{\sum_{i=1}^{n} \omega_i * (w_{ij} - w_{ik})^2}$
  Ekulidsches Distanzmaß um einen Gewichtungsfaktor erweitert - bei einem LSI-System könnte z.B. der normalisierte Singulärwert eines Themas als Gewichtungsmaß einfließen.

- Mahalanobis: $mahalanobis(\vec{d_j}, \vec{d_k}) = \sqrt{(w_{ij} - w_{ik})' * \Sigma^{-1} * (w_{ij} - w_{ik})}$
  Die Mahalanobis-Distanz unterscheidet sich von der Euklidschen durch die Gewichtung durch eine Kovarianzmatrize Σ, die die Korrelationen der Daten untereinander in die Berechnungen mit einfließen lässt. Ist die Matrize Σ eine Einheitsmatrize, so stimmt das Ergebnis mit dem der Euklidschen Distanz überein.

Weiterführende Literatur: Spärck Jones (1972); Manning und Schütze (1999).

3.4.2 Bestimmung der Anzahl der Themen

Die Schwierigkeit bei der Bestimmung der Anzahl der Themen bzw. Konzepte liegt darin, einerseits nicht zu viele wichtige Informationen zu verlieren, andererseits aber auch nicht zu viel Rauschen in die Relevanzberechnungen mit aufzunehmen. Verschiedene

Verfahren (siehe Dupret (2004); Oksa et al. (2002)) versuchen die Anzahl der Themen mit Heuristiken automatisch berechnen zu lassen, sind aber für die Anwendung in der Praxis noch zu unelaboriert - die meisten Umsetzungen von LSI verfahren nach der „what works best"-Methode (siehe Dumais (1990)). Die Anhaltspunkte für die Anzahl der Dimensionen schwanken je nach Autor und genutztem Dokumentenkorpus zwischen den Werten 100 (Dumais et al. (1997)), 200-346 (Dumais (1994, 1995)), 256 (Hofmann (1999)), 100 (Dumais (1990) oder zwischen 100 und 1500 (Landauer et al. (1997)).
In Abbildung 3.4 ist ein Verlauf der Singulärwerte für die Beispielkollektion abgetragen.

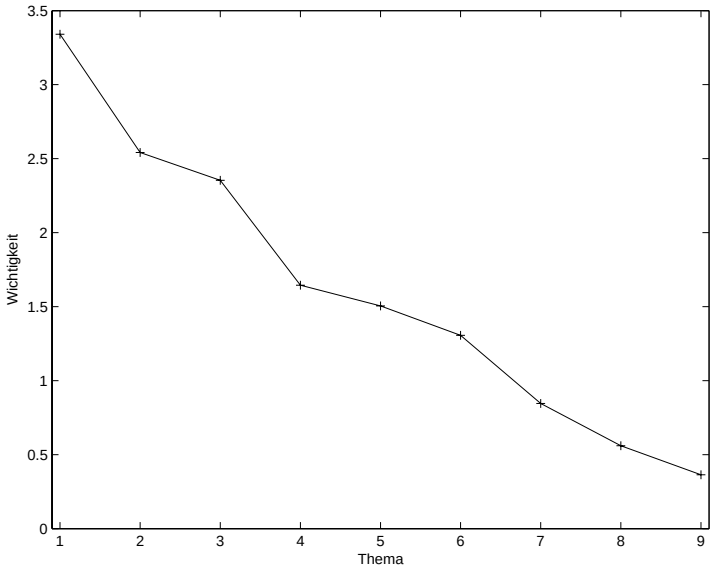

Abbildung 3.4: Verlauf der Singulärwerte für die Beispielsammlung

3.4.3 Relevance Feedback

Für viele Nutzer ist die Formulierung eines Suchbedürfnisses äußerst schwierig. Häufig liefert das IR-System dann nicht das, was der Nutzer sich erwartete. Die Methode des Relevance Feedbacks greift dieses Problem auf, indem es eine erste Suchanfrage des Nutzers bewertet und diese Bewertung in eine erneute Suchanfrage mit einbezieht, um damit den Erwartungen des Nutzers näher zu kommen. Auch hier existieren wieder manuelle und automatische Verfahren sowie Mischungen aus beiden. So besteht das manuelle Verfahren meistens darin, dass der Nutzer in der Ergebnismenge der ersten Suchanfrage die Dokumente markiert, die er als relevant zu seinem Suchbedürfnis ansieht. Diese Relevanzbewertungen fallen den Nutzern sehr viel leichter als die Formulierung der Suchan-

frage selbst (Baeza-Yates und Ribeiro-Neto (1999)). Automatische Verfahren können einerseits auf lokale Informationen (Wissen über die in der Ergebnismenge enthaltenen Dokumente und Terme) oder auf globale Informationen (Wissen über den gesamten Dokumentenkorpus und die darin enthaltenen Terme oder weiteres Hintergrundwissen wie Thesauri oder Ontologien) zurückgreifen.
Nähere Informationen zu den dahinter stehenden Algorithmen können in (Dumais, 1990, S. 11ff), Salton (1971) oder Baeza-Yates und Ribeiro-Neto (1999) nachgelesen werden.

3.5 Zwischenfazit

3.5.1 Vorteile

Zu den Vorteilen von LSI zählen: kein Erlernen von Query-Syntax nötig; Synonyme und Polyseme stellen kein Problem mehr dar; keinerlei linguistisches Vorwissen von Nöten (Thesaurus oder Wissensbasis); Terme und Dokumente werden in einem gemeinsamen Vektorraum abgebildet und werden auch gleich behandelt; Stemming und Stopwortelimination nicht unbedingt nötig; gute Ergebnisse bei verrauschten Daten; bei Cross-Language-Retrieval keine Übersetzung der Suchanfrage nötig.

3.5.2 Nachteile

Zu den Nachteilen von LSI gehören: hoher Speicherverbrauch und Rechenzeit für die Berechnung der zugrunde liegenden Matrizen; Skalierbarkeit von SVD bei großen Matrizen; schlecht für große, sich schnell ändernde Datenbestände (SVD-Updating); Interpretation der reduzierten Matrizen schwierig; optimale Anzahl an Themen nur empirisch bestimmbar; Position der Terme wird nicht beachtet; keine Phrasensuche möglich; Indexe zur Beschleunigung der Suchanfragen nur schwer erstellbar (Konzept der invertierten Listen nicht mehr anwendbar, da reduzierte Matrizen nicht mehr spärlich besetzt sind), viele Parameter beeinflussen die Retrieval-Güte.

4 Adaption und Implementation eines auf LSI beruhenden IR-Systems

In diesem Kapitel soll auf die genaue Implementation eines auf LSI beruhenden IR-Systems eingegangen werden. Das Aufgabengebiet dieses Systems sind kurze Texte, die in der Datenbank des INVISIP-Systems als Metadaten zu den jeweiligen Objekten vorgehalten werden. Um eine Vergleichbarkeit mit anderen Systemen zu gewährleisten, wird als Datensatz auf den GIRT4-Datensatz zurückgegriffen - spezieller: auf seine Überschriften. Dieser setzt sich aus 151.319 sozialwissenschaftlichen Dokumenten der Datenbanken SOLIS und FORIS zusammen - und beinhaltet zusätzlich 25 Testanfragen mit der dazugehörigen relevanten Dokumentenmenge. Bevor die Qualität eines IR-Systems getestet werden kann, muss die Funktion des Systems gewährleistet werden, was an den Beschreibungen zu den jeweiligen Abschnitten verdeutlicht werden soll. Um für die in Kapitel 5 folgenden Retrievaltests ein Vergleichsystem zu haben, werden die Implementationsschritte für zwei Verfahren durchgeführt: sowohl für das LSI-Modell als auch für das Vektorraummodell - im Folgenden soll allerdings nur auf das LSI-Verfahren eingegangen werden.

4.1 Genutzter Dokumentenkorpus

Die genutzte Testkollektion ist der GIRT4-Datensatz. Sie wurde von Michael Kluck erzeugt und beinhaltet sowohl Dokumente, Anfragen wie auch die Mengen der zu den Anfragen relevanten Dokumente. Die Dokumente werden nach Jahren und Datenbank getrennt in SGML-Dateien vorgehalten (Beispiel für ein solches Dokument siehe B) - Anfragen sehen wie folgt aus:

```
<top>
  <num> 076 </num>
```

```
  <DE-title> Politische Kultur in einer Demokratie </DE-title>
  <DE-desc> Finde Dokumente, die Formen der politischen Kultur in
            demokratischen Systemen analysieren oder beschreiben.
  </DE-desc>
  <DE-narr> Relevante Dokumente liefern Informationen, die theoretische
            Aspekte der politischen Kultur behandeln und/oder auf
            konkrete Formen innerhalb der Partei- oder Parlamentsarbeit
            oder des politischen Lebens eingehen.
  </DE-narr>
</top>
```

Die dazu relevante Dokumentenmenge befindet sich in einer extra Datei (auf der beigefügten CD-ROM im Verzeichnis „GIRT-Daten/Beispielanfragen“).
Durch Beschränkungen in der SVD-Berechnung musste der Datensatz auf 2068 Dokumente und 1449 Terme reduziert werden. Die Auswahl der Dokumente fiel dabei auf diejenigen Dokumente, die in mindestens einer Anfragemenge als relevant gekennzeichneten wurden. Die Anzahl der Terme setzt sich aus den in diesen Dokumenten vorhandenen Termen zusammen, die in mindestens einem Dokument, nicht in allen Dokumenten vorkommen und nicht als Stopwort deklariert sind (siehe Kapitel 4.4.2.3).

4.2 Genutzte Werkzeuge

Die folgende Liste zeigt die in dieser Arbeit genutzten Werkzeuge:

- PostgreSQL 8.0.0 + psql-Umgebung
  Objektrelationales Open-Source Datenbankmanagementsystem mit dem dazugehörigen interaktiven Terminal „psql“.
  http://www.postgresql.org/

- SQL
  Abfragesprache für relationale Datenbanken.

- PL/pgSQL
  Prozedurale Programmiersprache in PostgreSQL für serverseitig ausführbare Programme.
  http://www.postgresql.org/files/documentation/books/pghandbuch/html/plpgsql.html

- Snowball-Stemmer
  Ein von Martin Porter in der Programmiersprache „Snowball“ entwickelter Stammformreduzierer.
  http://snowball.tartarus.org/

- Bash
  Open-Source Unix-Kommandozeileninterpreter mit eigener Skriptsprache.
  http://www.gnu.org/software/bash/bash.html

- Perl
  Plattformunabhängige interpretierte Open-Source Programmiersprache.
  http://www.perl.org/

- Matlab
  Mathe-Software der Firma Mathworks.
  http://www.mathworks.com/

- HTML
  Auszeichnungssprache zur Beschreibung von Hypertext.
  http://www.w3.org/MarkUp/

- PHP
  Interpretierte Open-Source Skriptsprache, hauptsächlicher Nutzungsbereich ist die dynamische Webseitengenerierung.
  http://www.php.net/

- Excel
  Tabellenkalkulationsprogramm der Firma Microsoft.
  http://www.microsoft.com/germany/office/editionen/bestandteile/excel.mspx

4.3 Datenmodell

Alle Berechnungen für die späteren Anfragen und die Speicherung der dazu benötigten Daten übernimmt das Datenbankmanagementsystem PostgreSQL. Dort werden die Daten in verschiedenen Tabellen vorgehalten (siehe Abbildung 4.1), die sich wie folgt zusammensetzen:

- documents
  In dieser Tabelle wird primär eine eindeutige ID für das jeweilige Dokument gespeichert und zusätzlich mindestens ein Feld, in dem die für die Ähnlichkeitsberechnungen herangezogenen Einträge enthalten sind. In diesem Beispiel sind es die Titel der GIRT4-Datensätze.

- terms
  Hier befinden sich die Deskriptoren, die in den Dokumenten enthalten sind. Zusätzlich wird zu jedem Deskriptor das globale Gewichtungsmaß IDF gespeichert, um die späteren Berechnungen zu beschleunigen.

- term_document
  In der Term-Dokumententabelle werden die Zuordnungen von Termen zu Dokumenten zusammen mit der Auftretenshäufigkeit des Termes gespeichert. Das besondere hieran ist, dass nicht eine echte Term-Dokument-Matrize gespeichert wird, sondern nur die Koordinaten der jeweiligen Zelle. Nulleinträge werden nicht gespeichert und somit kann bei den spärlich besetzten Matrizen enorm Speicherplatz eingespart werden. Doch dieses System hat auch noch einen weiteren Vorteil: Falls ein anderes Term-Gewichtungsmaß eingeführt werden soll, so kann dies einfach durch eine weitere Spalte der Tabelle erledigt werden - und somit könnte im laufenden System das Termgewichtungsmaß geändert werden.

- query_data
  Hierin befinden sich Informationen über die einzelnen Anfragen: Start-Zeit und End-Zeit. Jede Anfrage erhält durch diese Tabelle ihre eindeutige Identifikation.

- queries
  Die Query-Tabelle beinhaltet pro Tabelleneintrag einen Suchterm der Suchanfragen der Nutzer. Es werden zusätzlich zu der eindeutigen Query_id der Anfrage (Fremdschlüssel → query_data) noch die Term_id und die Auftretenshäufigkeit des Suchterms in der Suchanfrage festgehalten. Zusätzlich könnte noch die Position des Termes gespeichert werden.

- u, s, v
  In diesen drei Matrizen befinden sich die Ergebnisse der SVD-Berechnungen. Die einzelnen Tabellen sind über Fremdschlüssel miteinander (u → s, v → s) und mit der Term- bzw. Dokument-Tabelle (u → terms, v → docs) verbunden.

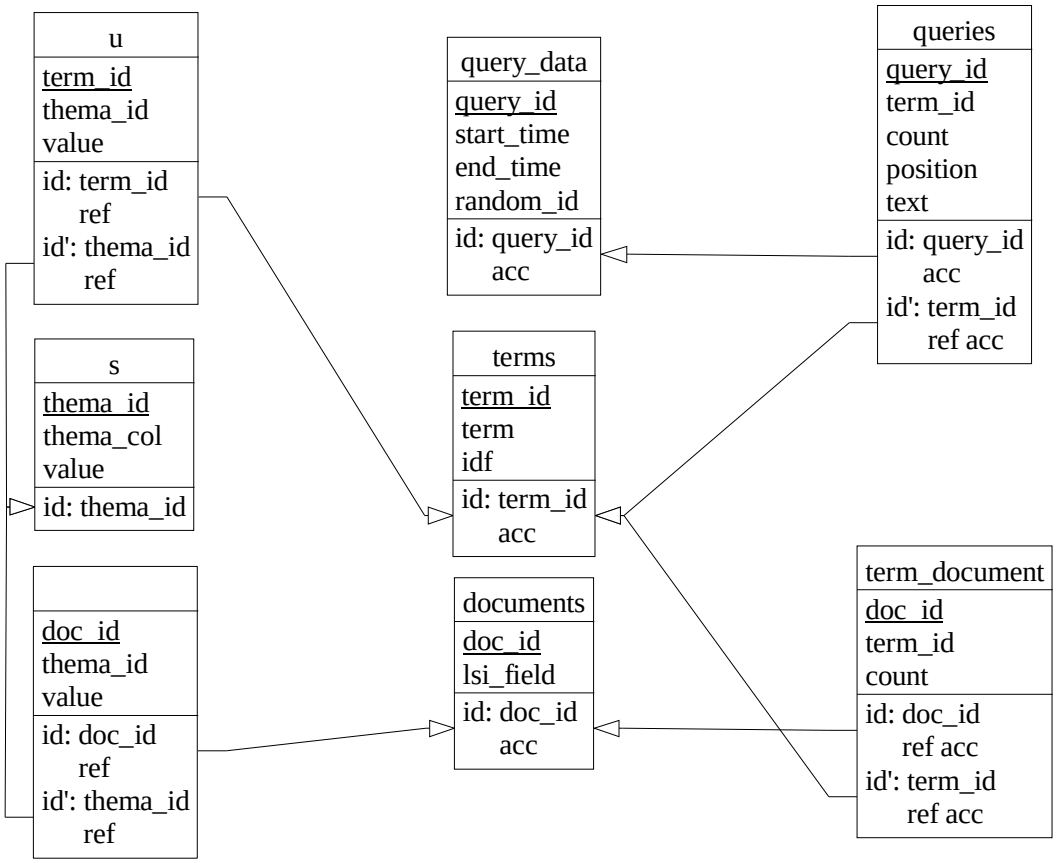

Abbildung 4.1: ER-Diagramm der für LSI-Berechnungen benötigten Tabellen

4.4 Implementation

Zur Implementation werden wie in Deerwester et al. (1990) das Termgewichtungsmaß TF-IDF, das Kosinus-Ähnlichkeitsmaß und als Dimensionsreduktionsverfahren die SVD eingesetzt.

4.4.1 Datenbeschaffung

Zur Auswahl des Dokumentenkorpusses wird auf den GIRT4-Datensatz zurückgegriffen, der pro Datenbank (SOLIS & FORIS) und Jahr jeweils eine eigene SGML-Datei bereithält. Um diese Daten aufzubereiten, wird ein Perl-Script geschrieben, welches die einzelnen Einträge in SQL-Befehle umwandelt. Diese wiederum schreiben die Daten in relationell verknüpfte Tabellen (siehe Abbildung 4.1).

4.4.2 Vorverarbeitung

Mit Hilfe eines SQL-Befehls in der psql-Umgebung werden die Dokumententitel samt den dazugehörigen Doc_Ids (mit dem Parameter `\o ausgabedatei.csv`) in eine Datei geschrieben. Anschließend wird die Datei manuell von irrelevanten Formatierungszeichen befreit und zweimal der lexikalischen Analyse übergeben - einmal um die SQL-Befehle zur Erzeugung der Term-Matrize und ein weiteres Mal um die SQL-Befehle zur Erzeugung der Term-Dokument-Matrize zu generieren. Diesen Vorgang in zwei Schritte aufzuteilen, vereinfacht den Prozess, da zur Erzeugung der Term-Dokument-Matrize die eindeutigen IDs der Terme feststehen müssen.

4.4.2.1 Lexikalische Analyse

Die lexikalische Analyse eliminiert in der hier angewandten Implementation Umlaute, Zahlen, Satz- und Sonderzeichen. Ein Perl-Script führt diese Aufgaben durch und sieht Zeichenketten, die zwischen zwei als Leerzeichen definierten Zeichen (Newline, Leerzeichen, Tabulator) stehen, als Token an. Dieser Schritt ist für beide Durchläufe gleich - sowohl für die Erzeugung der Term-Matrize als auch für die der Term-Dokument-Matrize.

4.4.2.2 Stemming

Da der Prozess der Stammformreduktion außerhalb der Datenbank durchgeführt wird, wird er in dieser Implementation vor dem Schritt der Stopwortelimination eingebunden.

Nach dem Durchlauf der lexikalischen Analyse werden für die Term-Matrize die Tokens zunächst mit dem Bash-Befehl `sort -u` alphabetisch sortiert und doppelte Elemente eliminiert, um die Dauer des Stemming-Vorgangs abzukürzen. Nach Reduktion der Terme auf ihre Stammform durchlaufen sie ein weiteres Mal den Sortierprozess `sort -u`, um evtl. mehrfach vorkommende Stammformen zu eliminieren. Für den Durchgang der Stammformreduktion der Term-Dokument-Matrize werden beide Sortierschritte ausgelassen. Das Ergebnis wird sowohl für die Phase der Term-Matrize als auch für die Term-Dokument-Matrize jeweils mit Hilfe eines Perl-Skripts in SQL-Befehle umgewandelt, die die jeweiligen Tabellen (`terms`, `A`) der Datenbank mit Inhalten füllen.

4.4.2.3 Stopwortelimininination

Nachdem die Datenbanktabellen `terms` und `A` mit den zu untersuchenden Daten versehen sind, wird die Stopwortelimination durchgeführt. Dies erfolgt bewusst nicht vor dem Einlesen der Tabelle `A`, um evtl. spätere Tests mit und ohne Stopworte durchlaufen lassen zu können.
Zunächst wird ein halbautomatisches Verfahren eingesetzt: Per SQL-Befehl werden alle Terme absteigend nach Auftretenshäufigkeit aufgelistet, um anschließend Stopworte manuell zu identifizieren und in die `stopwords`-Tabelle mit samt der `description` „manuell“ einzutragen. Nach Durchsicht von etwa 500 aus 80000 Termen und dem Identifizieren von 69 Stopworten wurde dieser Prozess auf Grund zu hohen Aufwandes abgebrochen. Entgegen der Empfehlungen aus Kapitel 3.2.2 wird als Alternative auf eine leicht angepasste vorhandene Liste zurückgegriffen; die Einträge dieser Liste werden mit der `description` „vorhandene_liste“ in die `stopwords`-Tabelle eingetragen. Um nun Stopworte aus Anfragen herausfiltern zu können, genügt das Anhängen des Befehls:

```
... AND term_id IN (SELECT term_id
                      FROM stopwords
                     WHERE description = 'vorhandene_liste')
```

Falls bei weiteren Anfragen andere Stopwortlisten bzw. andere Verfahren zur Generierung von Stopwortlisten genutzt werden, so ist dies durch das Eintragen der `term_id` und einem zu wählenden `description`-Eintrag realisierbar.

4.4.2.4 Selektion der Indexterme

Die Selektion der Indexterme gestaltet sich komplexer als der Prozess der Stopwortelimination. Eine SQL-Abfrage reicht allerdings aus, um die Filterkriterien „Term kommt

in mindestens einem Dokument“ und „Term ist in einem Dokument enthalten, das als relevant zu einem Query gilt“ wirken zu lassen (die Regel „Term kommt nicht in allen Dokumente vor“ muss nicht angewendet werden, da dies für keinen Term der Fall ist):

```
SELECT distinct(term_id)
FROM   a
WHERE  doc_id IN (
       SELECT distinct(rd.doc_id)
       FROM queries_relevant_documents rd
       WHERE rd.relevant_for_query = 1)
AND    term_id IN (
       SELECT term_id
       FROM   a
       WHERE  doc_id IN (
              SELECT distinct(rd.doc_id)
              FROM queries_relevant_documents rd
              WHERE rd.relevant_for_query = 1
              )
       GROUP BY term_id HAVING count(term_id) > 1
       )
;
```

4.4.3 Berechnungen

Die Berechnungsphase kann in zwei Teile unterteilt werden: Berechnungen auf der Datenbank, Berechnungen in Matlab. Zunächst müssen auf der Datenbank noch die globalen Termgewichtungen auf die einzelnen Zellen der Term-Dokument-Matrize angewendet werden, um sie anschließend in eine in Matlab importierbare Form aufzubereiten.

4.4.3.1 Termgewichtung

Die PL/pgSQL-Funktion `calculate_idf_values()` übernimmt die Aufgabe der Berechnung der *Idf*-Werte. Sie durchläuft die Terme der `terms`-Tabelle, die in einem Dokument enthalten sind, die auch relevant zu einer der Testanfragen sind und berechnet ihren IDF-Wert, um ihn in der `terms`-Tabelle zu speichern.
Die Berechnung des $Tf - Idf$-Maßes wird gleichzeitig mit dem Export der Term-Dokument-Tabelle vollzogen, der im nächsten Schritt genau beschrieben ist.

4.4.3.2 Singulärwertzerlegung

Um die Singulärwertzerlegung zu berechnen, wird das Programm Matlab von MathWorks genutzt - es kann aber auch jedes andere Programm genutzt werden, welches die SVD-Berechnungen durchführen kann.
Zunächst werden die Daten, die in der Datenbank in verschiedenen Tabellen vorliegen, in ein für Matlab (oder ein anderes Programm) verständliches Format umgewandelt. Hierzu wird eine PL/pgSQL-Funktion implementiert, die einerseits eine kommagetrennte Liste mit Term_ids und den Doc_ids ausgibt, und andererseits die Term-Dokument-Matrize kommagetrennt aufbereitet. Die sich daraus ergebende Matrize enthält keine Nulleinträge, womit eine Speicherplatzeinsparung von fast 50% erzielt wird. Um das Resultat der PL/pgSQL Funktion weiterverarbeiten zu können, werden einerseits die Fehlerausgaben dieses Programms - sie enthalten die Auflistung der Doc_Ids und der Term_Ids - manuell in die Dateien doc_ids.csv und term_ids.csv kopiert und andererseits die Standardausgabe in eine Datei umgeleitet. Diese Datei muss anschließend von den zusätzlichen Ausgaben (Tabellen-Überschriften, Tabellen-Unterschriften, Leerzeilen) der psql-Umgebung bereinigt werden (manueller, wenige Sekunden beanspruchender Schritt) und kann anschließend mit dem Matlab-Befehl

```
A = csvread('ausgabedatei.csv')
```

in Matlab eingelesen werden. Das Ergebnis dieses Einleseprozesses ist die Zuweisung der Term-Dokument-Matrize mit den Tf-Idf Werten zur Variablen `A`. Anschließend kann mit den SVD-Berechnungen begonnen werden.

```
[U S V] = svd(A);
```

Aus den drei Matrizen werden noch drei weitere Matrizen `s_inv` (Inverse von S), `u_sinv` (Produkt der Matrizen U und S^{-1}) und `doc_vecs` (beinhaltet die vorberechneten Dokument-Vektoren) berechnet, um die spätere Anfrageberechnung zu beschleunigen. Ein erhöhter Speicherplatzbedarf wird auf Grund des Teststatusses des Systems in Kauf genommen.

```
s_inv = inv(S);
u_sinv = U(:,1:500) * inv(S(1:500, 1:500));
doc_vecs = A' * u_sinv;
```

Anschließend werden die Variablen in die jeweiligen Dateien geschrieben.

```
csvwrite('u.csv',U(:,1:500));
csvwrite('s.csv',S(1:500,1:500));
csvwrite('v.csv',V(:,1:500));
csvwrite('s_inv.csv',s_inv(1:500,1:500));
csvwrite('u_sinv.csv',u_sinv(:,1:500));
csvwrite('doc_vecs.csv',doc_vecs(:,1:500));
```

Im nächsten Schritt werden die CSV-Dateien in SQL-Befehle konvertiert. Hierzu dient wieder ein Perl-Script, welches die verschiedenen durch Matlab berechneten Matrizen mit Hilfe der Dateien *doc_ids.csv* und *term_ids.csv* in eine für die Datenbank verständliche Syntax aufbereitet. Nach dem Einlesen der über 250MB großen SQL-Datei (in der psql-Umgebung mit dem Befehl `\i sql-input.sql`) ist die Datenbank bereit, beliebige Anfragen entgegen zu nehmen.

4.4.4 Anfrageverarbeitung

Um die Anfragen eines Benutzers entgegen nehmen zu können, wird eine Webseite erstellt. Diese enthält eine kleine Suchmaske, bei der man den gewünschten Suchtext eingeben und - wenn gewünscht - auch noch die Anzahl der Themen bestimmen kann. Anschließend kann die Suchanfrage abgeschickt werden. Die Suchmaske an sich wird in reinem HTML umgesetzt - die Bearbeitung der Suchterme und das Aufbereiten der Suchergebnisse als Webseite erledigt hingegen ein PHP-Skript. Es nimmt die Suchanfrage entgegen, führt dann neben Hilfsfunktionen die selben Schritte durch, die auch jedes Dokument beim Erzeugen des Indexes während des Vorverarbeitungsprozesses durchlaufen musste:

4.4.4.1 Lexikalische Analyse

Umlaute, Zahlen, Satz- und Sonderzeichen werden entfernt; alles, was zwischen zwei Leerzeichen steht, wird als Token angesehen. Dies wird vom selben Perl-Skript erledigt, das auch im Vorverarbeitungsprozess genutzt wurde. Die Einbindung in den PHP-Code könnte mittels des Befehls

```
passthru("echo ".$query." | perl ./tokenize.pl")
```

realisiert werden, es werden jedoch zur Vereinfachung des Aufbaus des PHP-Skripts die Schritte der lexikalischen Analyse und des Stemmings zusammen in ein gemeinsames Skript geschrieben.

4.4.4.2 Stemming

Beim Stemming der Suchanfrage wird das gleiche Programm verwendet, das auch im Vorverarbeitungsprozess für das Stemming verantwortlich war. Der nun genutzte Befehl, um die Suchanfrage in Tokens aufzuteilen und die Tokens anschließend auf ihre Stammform zu reduzieren, wird mit dem Befehl

```
passthru("echo ".$query." | perl ./tokenize.pl | ./stemwords -l german)
```

im PHP-Skript umgesetzt.
Das Ergebnis dieser beiden Schritte wird nun als Suchanfrage an die PostgreSQL-Datenbank gesendet und damit der PL/pgSQL-Funktion `lsi_get_query_results()` übergeben. Sie erwartet als Eingangsparameter den zu suchenden Text, den Schwellenwert für die als ähnlich gekennzeichneten Dokumente und die Anzahl der Themen `k`, die für die Suchanfrage herangezogen werden sollen.
Bevor der Tokenizer der Datenbank mit dem Zerlegen der Anfrage beginnt, werden durch die PL/pgSQLFunktion `lsi_create_query_data()` zunächst einige Informationen zu der aktuellen Anfrage in der Tabelle `query_data` gespeichert:

- `query_id`
  Eine eindeutige Identifikationsnummer, um die Anfrage und die darin enthaltenen Terme später eindeutig zuweisen zu können.

- `start_time`
  Hierin wird zu Beginn einer Anfrage die aktuelle Uhrzeit eingetragen, um später eine Übersicht über die Dauer der Anfragen erhalten zu können.

- `end_time`
  Wenn die Funktion `lsi_get_query_results()` abgeschlossen und somit eine Anfrage abgearbeitet ist, wird diesem Wert noch die aktuelle Uhrzeit zugewiesen.

Der Rückgabewert der Funktion `lsi_create_query_data()` ist eine über alle Anfragen eindeutige `query_id`. Anschließend kann der Tokenizer seine Arbeit aufnehmen, indem er alles, was zwischen zwei Leerzeichen steht, als Token ansieht. Die so gewonnenen Tokens enthalten jeweils den auf die Stammform reduzierten Term der ursprünglichen Suchanfrage. Zu jedem dieser Terme wird nun die `term_id` in der `terms`-Tabelle nachgeschlagen und in die Tabelle `queries` mitsamt der `query_id` eingetragen und der `count` wird auf „1“ gesetzt. Existiert schon ein Eintrag für diesen Term und diese Anfrage in

der Tabelle, so wird der `count` um „1“ inkrementiert. Für spätere Positionsbestimmungen der Terme bzw. Phrasensuche ist zusätzlich die Speicherung eines Arrays mit den Positionen des Terms möglich. In diesem Fall wird nur die Position des letzten Terms gespeichert, hierbei können aber doppelt in einer Suchanfrage vorkommende Terme nicht gesondert behandeln werden.
Nachdem die Anfragedaten in der Datenbank gespeichert wurden, fährt die PL/pgSQL-Funktion `lsi_get_query_results()` mit den Ähnlichkeitsberechnungen für sowohl das LSI-Modell als auch das Vektorraummodell fort.

4.4.4.3 Stopwortelimination

Die Stopwortelimination ist ein dem Verfahren immanenter Prozess, denn Terme, die nicht im Index des IR-Systems zu finden sind, werden auch nicht zur Suchanfrage hinzugezogen. Dies trifft in diesem IR-System auch auf die Stopworte zu.

4.4.4.4 Selektion der Queryterme

Wie auch schon bei der Stopwortelimination werden nur Terme genutzt, die auch im Index des IR-Systems zu finden sind. Dadurch fallen einige Terme, die in den Beispielsuchanfragen für die GIRT-4-Daten enthalten sind, durch diesen Indexprozess, da sie der Filterregel „Term kommt in mindestens einem Dokument vor“ nicht gerecht werden - so bei der Anfrage mit der Nummer 90 „Zeitungslesen“ oder mit dem Term „Homosexualität“ aus der Anfrage 94. Beide Terme finden sich nur in der Überschrift eines Dokumentes im Index. Somit wird z.B. zu der Anfrage 90, die nur aus diesem einen Term besteht, kein einziges relevantes Dokument geliefert.

4.4.4.5 Ähnlichkeitsmaß

Als Ähnlichkeitsmaß wird wie in vielen LSI-Implementationen das Kosinus-Maß gewählt, da es ein relativ gutes Verhältnis von Implementationsaufwand und Retrievalgüte besitzt. Die Umsetzung der Kosinus-Berechnung wird in mehrere Teilschritte zerlegt; sie ist im Prinzip für LSI und VSM gleich. Im Folgenden wird nur auf die LSI-Implementation eingegangen. Die PL/pgSQL-Funktion

```
lsi_get_query_results()
```

iteriert über alle Dokumente und ruft zunächst die Funktion

```
lsi_calc_cosine_similarity_between_the_query_and_a_doc_for_k()
```

auf. Sie erwartet als Eingangsparameter die `query_id`, die `doc_id` des zu vergleichenden Dokuments und schließlich die Anzahl der Themen, ausgedrückt durch die Dimension der Vektoren, für die die Ähnlichkeitsberechnungen durchgeführt werden sollen.
Zur Veranschaulichung der folgenden Erklärung noch einmal das Kosinus-Maß:

$$cos(\vec{d_j}, \vec{d_k}) = \frac{\sum_{i=1}^{n} w_{ij} * w_{ik}}{\sqrt{\sum_{i=1}^{n} w_{ij}^2 * \sum_{i=1}^{n} w_{ik}^2}}$$

Die Funktion `lsi_calc_cosine_similarity_between_the_query_and_a_doc_for_k()` berechnet zunächst den Zähler der Kosinus-Formel: die Summe der multiplizierten Termgewichte für die übereinstimmenden Dimensionen der beiden Vektoren. Anschließend wird der Nenner berechnet, indem die Wurzel aus dem Produkt der Summe der Quadrate der beiden Vektoren gezogen wird.

4.4.4.6 Bestimmung der Anzahl der Themen

Eine valide Bestimmung der Anzahl der Themen kann erst nach der Durchführung der Retrieval-Tests erfolgen - erste Beispiel-Anfragen lassen allerdings einen optimalen Wert zwischen 50 und 75 Themen erwarten.

4.4.4.7 Relevance Feedback

Ein vom IR-System gestütztes Relevance Feedback wird nicht durchgeführt. Relevance Feedback in diesem System ist möglich, muss aber manuell durchgeführt werden, indem die Titel, die als relevant identifiziert werden, manuell zur Suchanfrage hinzugezogen werden. Das Ergebnis entspricht dem durch ein von einem IR-System unterstützten Relevance Feedback.

4.5 Zwischenergebnis

Nach den ersten Suchergebnissen zeigten sich beim VSM einige auffällig schlechte Ergebnisse: Die Anfragen 81, 92, 95, 97, 100 liefern sehr wenige Ergebnisse, allerdings sind diese häufig Volltreffer.
Anfrage 90 verursacht bei beiden Systemen einen Totalausfall - bei beiden Verfahren wird kein Dokument als relevant zur Suchanfrage ausgegeben. Nach kurzen Nachforschungen

stellte sich heraus, dass der Term „Zeitunglesen“ nicht in den Titeln der gewählten Dokumente zu finden ist, weshalb er auch nicht im Index des Systems gespeichert ist. Bei einer manuellen Umformulierung der Suchanfrage zu „Zeitung lesen“ erschienen die erhofften Ergebnisse.
Die LSI-Ergebnisse für die geringe Anzahl an Testanfragen mit nur wenigen unterschiedlichen Dimensionen erfüllen nicht die in LSI gesteckten Erwartungen - ein genaues Urteil kann zu diesem Zeitpunkt aber noch nicht gefällt werden, da die einzelnen Parameter zu große Auswirkungen auf die Retrieval-Güte haben.

5 Evaluation

In diesem Kapitel soll genauer auf Testverfahren und deren Ergebnisse eingegangen werden, die für die Implementation des in Kapitel 4 beschriebenen LSI-Systems durchgeführt werden. Zunächst soll allerdings eine Analyse vorhandener Retrieval-Tests durchgeführt werden, um im Voraus evtl. Schlüsse auf die Ergebnisse des gewählten Dokumentenkorpusses ziehen zu können. Danach wird ein kurzer Einblick in die Effizienz des Systems gegeben, indem Speicherverbrauch, Rechenzeit zur Erzeugung des Indexes und die Dauer einer Anfrage näher analysiert werden. Nachdem diese Schritte durchgeführt wurden, finden die eigentlichen Retrieval-Tests statt. Jede der 25 Anfragen wird insgesamt 100 Mal an die Datenbank gestellt - für verschiedene Anzahl an Konzeptdimensionen; beginnend bei 5 und dann in 5er-Schritten bis 500. Dabei werden jedes Mal auch die Ähnlichkeitsberechnungen für das Vektorraummodell durchgeführt, um eine direkte Vergleichbarkeit mit einem IR-System zu ermöglichen - so können Schwankungen, die durch minimal verschiedene Implementationen entstehen, ausgeschlossen werden. Für eine Auswahl an Konzeptdimensionen (meistens 25, 50, 75, 100, 125 - bei Auffälligkeiten auch mehr) werden dann mit Hilfe von interpolierten Precision-Recall-Diagrammen und den durchschnittlichen Precision- und Recall-Werten die Ergebnisse der jeweiligen Anfrage analysiert. Zusätzlich werden die Ergebnisauflistungen der Anfragen genauer betrachtet, um evtl. Rückschlüsse auf das hinter LSI stehende Modell ziehen zu können. In einem Fazit sollen die Ergebnisse und somit die LSI-Implementierung bewertet werden.

5.1 Analyse vorhandener Retrieval-Tests

Vor Beginn der eigentlichen Retrieval-Tests werden zunächst vorhandene Retrieval-Test des LSI-Verfahrens analysiert und überprüft, ob sie sich evtl. auf das durch INVISP gestellte Szenario übertragen lassen. Die Aufstellung der verschiedenen Testkollektionen sind aus Dumais (1990) und Ferber (2003) entnommen und in der um die GIRT4-Testdatenbank erweiterten Tabelle 5.1 eingetragen. Bei der Testkollektion CISI schnei-

[1] Die Relevanzurteile in UCKIS sind teilweise unvollständig.

Name	#Docs	#Terms /Doc	Anf	#Terms /Anf	RelDocs /Anf	(RD/A) /#Docs	LSI AVG Precision
ADI	82	16	35	5	5	6.10%	.29
CISI	1460	45	35	8	50	3.42%	.11
Cranfield	1400	28.7	225	8	7.2	0.51%	.39
LISA	6004	39.7	35	16.5	10.8	0.18%	.20
MED	1033	50	30	10	23	2.23%	.52
TIME	425	190	83	8	4	0.94%	.40
GIRT (Titel)	151319	9.1	25	2.88	91.36	0.0006%	
GIRT (LSI-Test)	2068	6.0	25	2.88	91.36	4.42%	

Tabelle 5.1: Zusammensetzung Test-Kollektionen Quelle: Ferber (2003)

det LSI sehr schlecht ab. Die genauen Gründe für dieses Ergebnis lassen sich aus Tabelle 5.1 nicht entnehmen – die quantitativen Daten entsprechen ungefähr denen von MED, bei dem LSI ein sehr gutes Ergebnis erzielt. Auch ein genaueres Betrachten der Precision-Recall-Diagramme (siehe Abbildung 5.1) führen zu keinen möglichen Folgerungen für ähnliche Dokumentenkorpusse. Es müsste also auf die genaue Zusammensetzung der Testkollektion eingegangen werden, um Rückschlüsse für das schlechte Abschneiden ziehen zu können - also auch nicht von dem guten Ergebnis von MED. Insgesamt läßt sich in den quantitativen Daten also keine Korrelation zur Retrieval-Güte erkennen. Ein genaueres Studieren der Dokumentenkollektionen würde allerdings sehr viel Zeit in Anspruch nehmen und die Verlässlichkeit der daraus gewonnenen Erkenntnisse müsste in Frage gestellt werden. Ein Test mit dem eigenen Dokumentenkorpus lässt sich also nicht umgehen.

5.2 Analyse der LSI-Ergebnisse des GIRT-4-Datensatzes

Zunächst wird im Unterkapitel 5.2.1 auf die Effizienz der LSI-Implementierung eingegangen - Fragen nach dem Umgang mit Ressourcen wie Rechenzeit und Speicherplatz werden geklärt.

5.2.1 Effizienz

Effizienz bei IR-Systemen bedeutet, wie sparsam das Verfahren mit Ressourcen eines Rechnersystems umgeht. In diesem Unterkapitel sollen der Speicherverbrauch und die

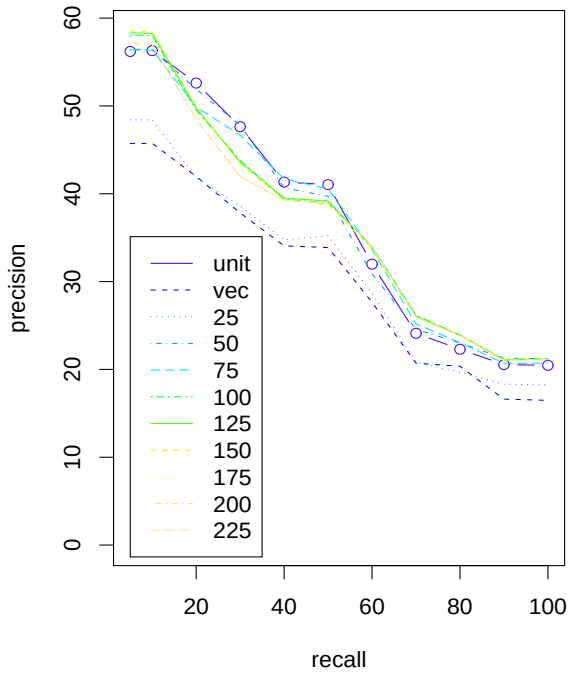

Abbildung 5.1: Precision-Recall-Diagramm für Testkollektion ADI. Quelle: Dupret (2004)

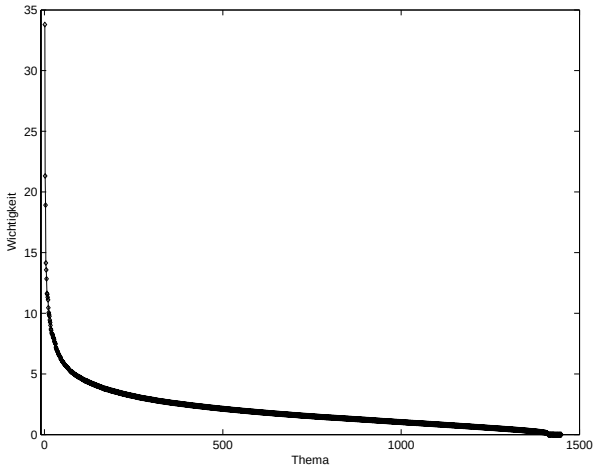

Abbildung 5.2: Verlauf der Singulärwerte für den GIRT4-Datensatz

verschiedenen Berechnungszeiten genauer betrachtet werden.

5.2.1.1 Speicherverbrauch

Der Speicherverbrauch der für die Ähnlichkeitsberechnung benötigten Datenbanktabellen läßt sich aus folgender Formel berechnen (Tabellen mit Metadaten zu Termen und Dokumenten werden nicht behandelt, da diese in jedem System gespeichert werden müssen):

$$(\#Terme * k + k + \#Docs * k) * Datentyp\ in\ Byte \tag{5.1}$$

Der Verbrauch eines normalen Vektorraummodells wird durch folgende Formel berechnet:

$$\#Terme * \#Docs * Datentyp\ in\ Byte \tag{5.2}$$

Für einen Dokumenten-Korpus mit 1000 Termen und 2000 Dokumenten und einem 4 Byte großen Datentyp ergäbe dies für LSI bei 100 Themen einen Verbrauch von

$$(1000 * 100 + 100 + 2000 * 100) * 4\ Byte = 500100\ Byte \approx 0,5\ MB \tag{5.3}$$

Für das Vektorraummodell würde dies folgende Rechnung bedeuten:

$$1000 * 2000 * 4\ Byte = 8000000\ Byte \approx 8\ MB \tag{5.4}$$

Gerade bei großen Term-Dokument-Matrizen würde die Nutzung eines auf LSI beruhenden IR-Systems enorme Speicherplatzeinsparungen nach sich ziehen.

5.2.1.2 Berechnung des Indexes

Hier liegt ein großes Manko von LSI - das genutzte SVD-Verfahren hat eine Laufzeit von $\sigma\left(n^{2k^3}\right)$. Aber neben der Laufzeit existiert in der Praxis ein noch viel größeres Problem: der Arbeitsspeicherverbrauch. Bei der Berechnung der SVD-Matrizen des gekürzten GIRT4-Korpusses (2068 Dokumente, 1449 Terme) benötigte Matlab über 800MB Arbeitsspeicher - bei nur geringfügig größeren Matrizen war es nicht mehr im Stande, die SVD-Berechnung durchzuführen.

5.2.1.3 Anfragedauer

Wie in Abbildung 5.3 zu sehen, ist der Verlauf der durchschnittlichen Anfragedauer für verschiedene Anzahl an Themen nahezu linear. Die Berechnungszeit läßt sich durch die

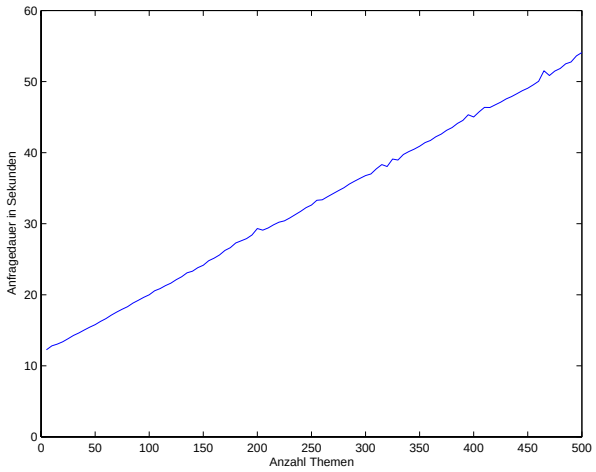

Abbildung 5.3: Verlauf der Anfragedauer für verschiedene Anzahl Themen

Funktion

$$f(x) \approx 0,084 * x + 11,705 \tag{5.5}$$

approximieren. Da sich jede Anfrage aus mehreren Funktionen zusammensetzt (siehe Kapitel 4.4), lässt sich die Konstante der Funktion weiter aufteilen. Die Berechnung der Ergebnisse des Vektorraummodells benötigt durchschnittlich etwas unter einer Sekunde, die nun noch übrigen Hilfsfunktionen verbrauchen den Rest, also ungefähr 11 Sekunden. Welche dieser Hilfs-Funktionen den größten Anteil dieser Rechenzeit verbraucht, müsste durch weitere Tests genauer untersucht werden.
Wie ausschlaggebend allerdings die Anzahl der Themen bzw. der Terme ist, kann nur durch weitere Tests mit verschiedenen Größen genauer spezifiziert werden. Höchstwahrscheinlich werden sie die Steigung der Gleichung 5.5 jeweils linear bzw. – wenn sowohl Termanzahl als auch Dokumentenanzahl erhöht werden – exponentiell erhöhen bzw. verringern. Die Arbeit von Letsche (1996) geht genauer auf diese Problemstellung ein.
Die Anfragedauer bei 50 Themen von durchschnittlich 16,831 Sekunden scheint je nach Anwendungsgebiet auszureichen - im Alltagseinsatz eines Suchsystems wie INVISIP oder MedioVis – bei denen die Anfragedauer im Millisekundenbereich liegt – sollten allerdings weitere Optimierungen durchgeführt werden. Dies kann durch Deaktivieren von verschiedenen Hilfsfunktionen, aber vor allem durch Externalisieren der Matrizen-Berechnungen, realisiert werden.

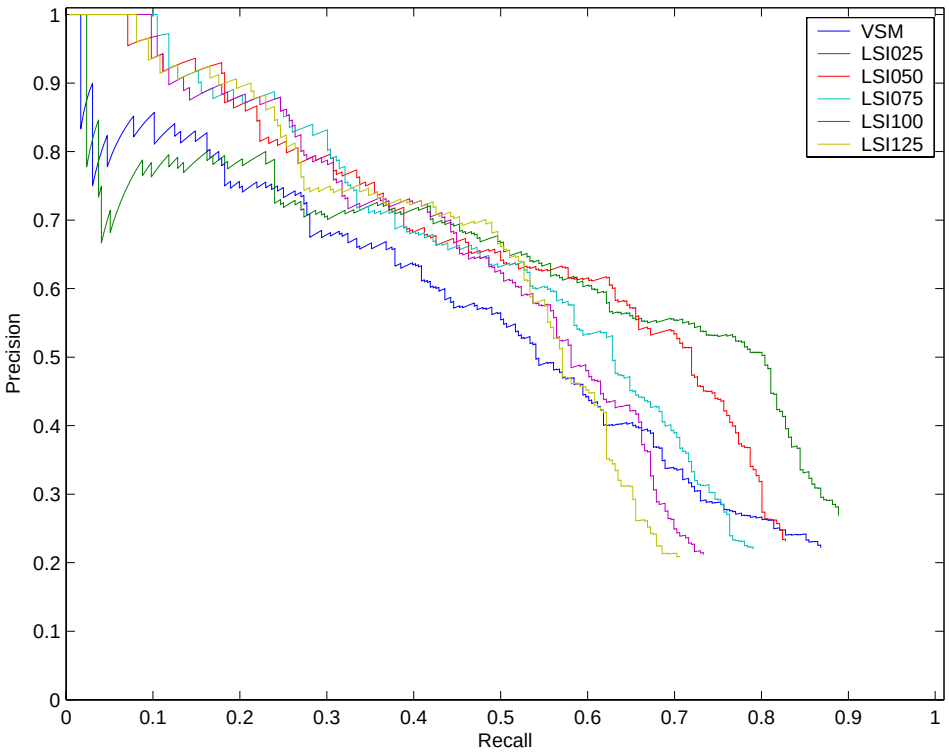

Abbildung 5.4: Precision-Recall-Diagramm für Anfrage 76 ohne Interpolation

5.2.2 Retrieval-Ergebnisse

Die Retrieval-Tests wurden mit den 25 Beispielanfragen (siehe Anhang C) des GIRT4-Datensatzes durchgeführt. Die Nummerierung der Anfragen beginnt bei „Query 76“ und endet bei „Query 100“. Für jede dieser Anfragen wurden die Ähnlichkeitswerte für das Vektorraummodell und LSI (in 5er-Schritten von 5 bis 500 Themen) ermittelt. Die Ergebnisse dieser Anfragen befinden sich als Excel-Tabellen, nach Anfragen getrennt, auf der beigefügten CD-ROM im Verzeichnis „Query-Tests“. Anschließend wurden die Precision-, Recall- und Fall-Out-Werte (siehe Kapitel 2.4.1.1) berechnet, um daraus Precision-Recall-Diagramme – wie in Abbildung 5.4 – zu generieren. Eine übersichtlichere Version (das sogenannte 11-Punkt-Verfahren) interpoliert die Ergebnisse bei den Werten von 0 bis 1 in den Intervallen 0,1. Um die Güte eines IR-Systems darzustellen wird normalerweise ein interpoliertes Precision-Recall-Diagramm über den Durchschnitt aller Anfragen erstellt. Da die GIRT-Daten allerdings nur 25 Anfragen bereithalten kann auf eine genauer auf die Ergebnisse der einzelnen Anfragen eingegangen werden:
(Der VSM-Graph ist immer blau, die Farbe und Anzahl der Themen bei LSI variieren von Test zu Test.)

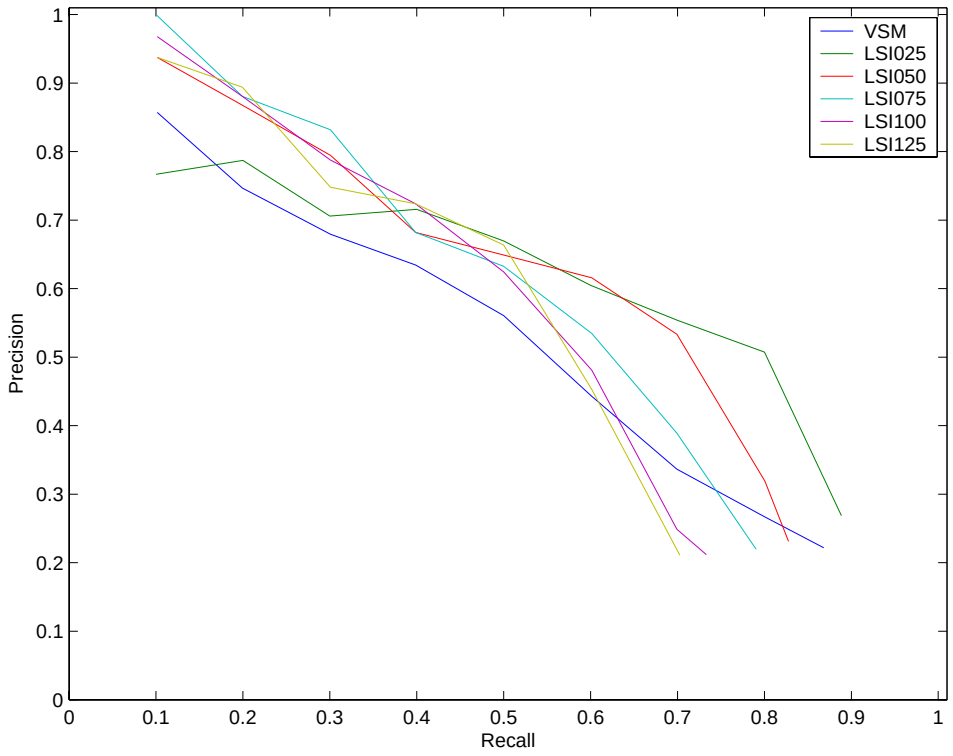

Abbildung 5.5: Precision-Recall-Diagramm für Anfrage 76 mit Interpolation

Query76

„Politische Kultur in einer Demokratie“
Anzahl relevante Dokumente: 296

- VSM
  Gutes Ergebnis.
  Durchschnittliche Precision: 0,62
  Durchschnittlicher Recall: 0,42

- LSI
  Gute Ergebnis für k $\approx$ 100.
  Durchschnittliche Precision für k = 100: 0,58
  Durchschnittlicher Recall: 0,45
  Andere Wert für k liefern einen besseren Durchschnitt (z.B. für k = 5: Precision 0,70; Recall 0,49), obwohl für den Nutzer die Ergebnisse schlechter sind. Für k = 100 sind die ersten 28 Ergebnisse Volltreffer, für k = 5 nur die ersten 8. Aus diesem Grund wird nicht weiter auf die Maße der durchschnittlichen Precision und des

durchschnittlichen Recalls eingegangen, obwohl in vielen Arbeiten über LSI (z.B. Deerwester et al. (1990); Dumais et al. (1997)) dieses Maß als Vergleichskriterium zu anderen Verfahren genutzt wurde. Anstelle dessen werden die Qualitätsaussagen anhand der Precision-Recall-Diagramme getroffen.

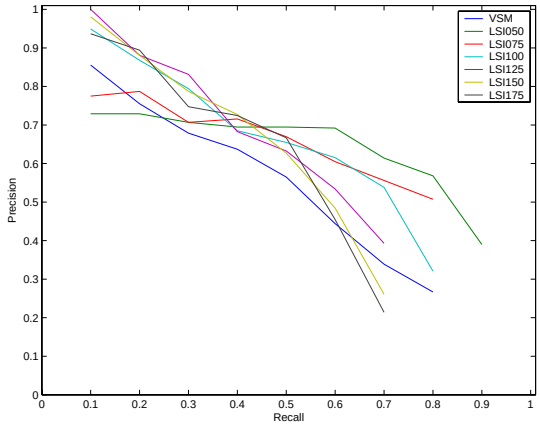

Abbildung 5.6: Anfrage 76

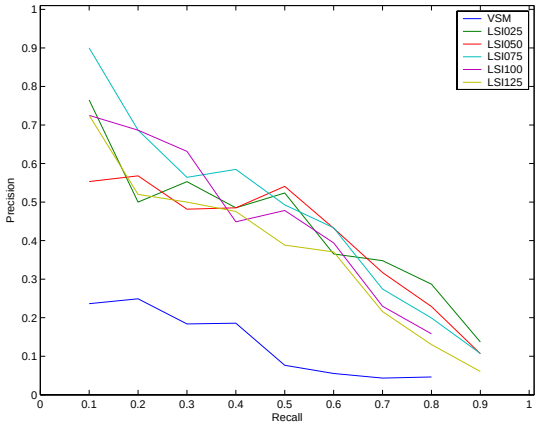

Abbildung 5.7: Anfrage 77

Query77

„Politische Partizipation von DDR-Frauen“
Anzahl relevante Dokumente: 33

- VSM
  Schlechtes Ergebnis.
  Durchschnittliche Precision: 0,51
  Durchschnittlicher Recall: 0,08

- LSI
  Gutes Ergebnis für k $\approx$ 75.
  Durchschnittliche Precision für k = 75: 0,86
  Durchschnittlicher Recall: 0,12
  LSI arbeitet durchweg 30% besser als VSM.

Query78

„Bildung in der Türkei“
Anzahl relevante Dokumente: 24

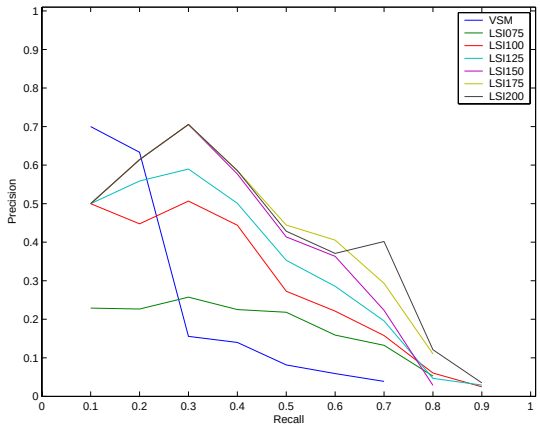

Abbildung 5.8: Anfrage 78

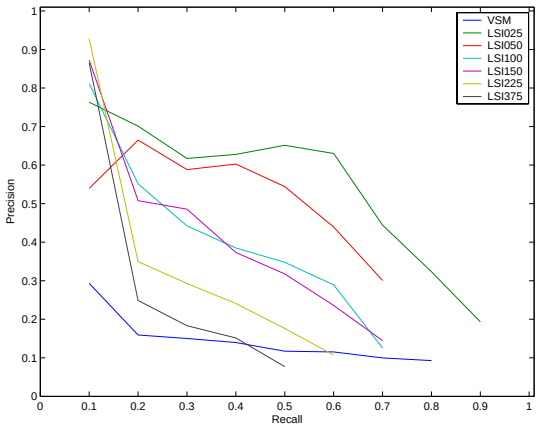

Abbildung 5.9: Anfrage 79

- VSM
  Zu Beginn bessere Ergebnisse als die getesteten Konzeptdimensionen LSIs.
  Durchschnittliche Precision: 0,68
  Durchschnittlicher Recall: 0,05

- LSI
  Relativ schlechte Ergebnis für kleine k (für k = 25 ist unter den ersten 20 Dokumenten kein einziges Relevantes zu finden), beste Ergebnisse für k ≈ 175, höhere Anzahl an Konzeptdimensionen muss getestet werden.
  Durchschnittliche Precision für k = 175: 0,82
  Durchschnittlicher Recall: 0,08
  Die Anfänge der Ergebnislisten, die durch LSI berechnet wurden, sind relativ schlecht, nehmen mit steigendem k aber stetig zu. Weitere Tests mit höheren k sollten durchgeführt werden, um eine mögliche Steigerung der Retrieval-Güte zu überprüfen.

Query79

„Volksentscheide in der Demokratie“
Anzahl relevante Dokumente: 129

- VSM
  Sehr schlechtes Ergebnis im Vergleich zu LSI, Precision kann nur zu Beginn über 20% gehalten werden.
  Durchschnittliche Precision: 0,52
  Durchschnittlicher Recall: 0,13

- LSI
  Zu Beginn relativ gute Ergebnis für hohe k, über den kompletten Verlauf des Diagramms ist allerdings k ≈ 25 vorzuziehen.
  Durchschnittliche Precision für k = 25: 0,82
  Durchschnittlicher Recall: 0,27
  Je nach Zielsetzung kann hier zwischen hohem k (≈ 225; dadurch guten Ergebnissen zu Beginn der Ergebnisliste) oder niedrigem k (≈ 25; mehr Treffer insgesamt) gewählt werden.

Query80

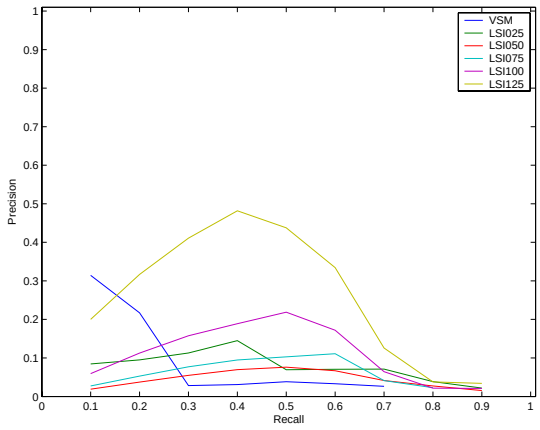

Abbildung 5.10: Anfrage 80

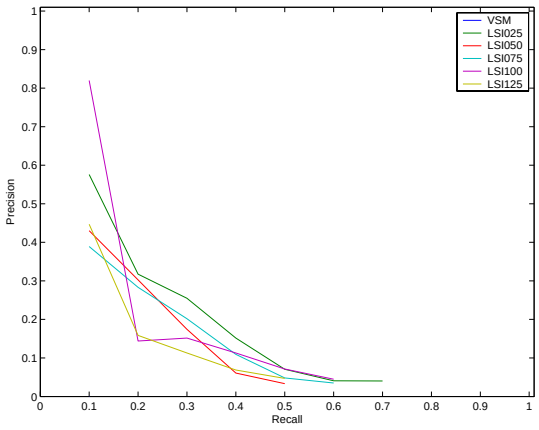

Abbildung 5.11: Anfrage 81

„Industrielle und ökonomische Entwicklung Indiens“
Anzahl relevante Dokumente: 14

- VSM
  Zu Beginn besseres Ergebnis als LSI; ab einem Recallwert von 30% fast keine Treffer mehr.
  Durchschnittliche Precision: 0,62
  Durchschnittlicher Recall: 0,02

- LSI
  Sehr schlechtes Ergebnis für kleine k, interessanter „Buckel“ für k = 125 → für größere k weitere Tests durchführen. Mangels Tests vorerst k ≈ 125.
  Durchschnittliche Precision für k = 125: 0,82
  Durchschnittlicher Recall: 0,05

Unbedingt weitere Tests durchführen. Anscheinend kann bei Anfragen, die nur eine kleine Menge an relevanten Dokumenten enthalten, mit großen k eine enorme Steigerung der Retrieval-Güte erreicht werden. Somit könnte bei Dokumentenkollektionen, in denen nur wenige Dokumente als relevant zu einer Suchanfrage gelten, ein hoher k-Wert in Betracht gezogen werden, um diese Anfragen befriedigen zu können. Dabei könnten die schlechteren Ergebnisse bei Suchanfragen mit vielen relevanten Dokumenten in Kauf genommen werden.

Query81

„Ausbildungsabbruch“
Anzahl relevante Dokumente: 53

- VSM
  Laut Precision-Recall-Diagramm keine Ergebnisse für VSM, allerdings werden 5 Dokumente gefunden und diese sind allesamt Volltreffer. Jedoch fallen die Dokumente durch die Interpolation der Ergebnisse aus den Daten.
  Durchschnittliche Precision: 0,06
  Durchschnittlicher Recall: 1,00

- LSI
  Zu Beginn gutes Ergebnis für k $\approx$ 100 - sonst eher schlechtes Ergebnis.
  Durchschnittliche Precision für k = 100: 0,47
  Durchschnittlicher Recall: 0,09
  Das schlechte Ergebnis für VSM und LSI liegt wahrscheinlich daran begründet, dass der Term „Ausbildungsabbruch“ selten in der Dokumentenmenge zu finden ist (insgesamt 5 mal). Eine Kompositazerlegung würde bei einer solchen Anfrage für eine enorme Steigerung des Recalls sorgen.

Query82

„Berufliche Bildung von Immigranten“
Anzahl relevante Dokumente: 42

- VSM
  Schlechtes Ergebnis, VSM besser als alle getesteten LSI-Konzeptdimensionen.
  Durchschnittliche Precision: 043,
  Durchschnittlicher Recall: 0,11

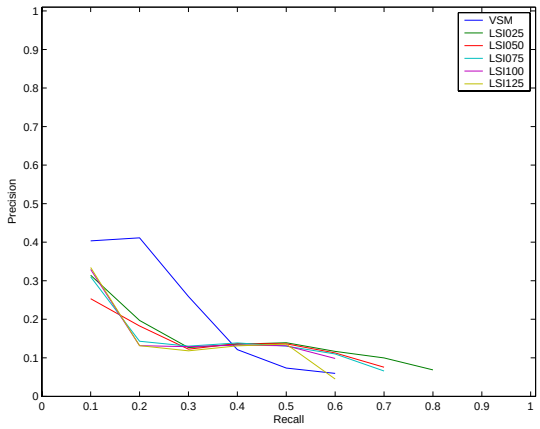

Abbildung 5.12: Anfrage 82

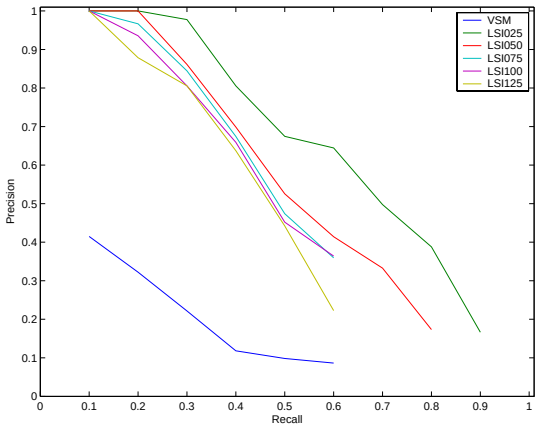

Abbildung 5.13: Anfrage 83

- LSI
  Sehr schlechtes Ergebnis für k bis 125, höhere Anzahl an Dimensionen muss getestet werden.
  Durchschnittliche Precision für k = 25: 0,64
  Durchschnittlicher Recall: 0,09

Query83

„Medien und Krieg“
Anzahl relevante Dokumente: 145

- VSM
  Weit abgeschlagen im Vergleich zu LSI.
  Durchschnittliche Precision: 0,42
  Durchschnittlicher Recall: 0,16

- LSI
  Hervorragende Ergebnisse für LSI, k ≈ 25.
  Durchschnittliche Precision für k = 25: 0,76
  Durchschnittlicher Recall: 0,36
  Anscheinend werden die beiden Themen „Medien“ und „Krieg“ in den ersten 25 Dimensionen des Konzeptraumes sehr gut behandelt. Höhere Dimensionen verschlechtern das Ergebnis nach und nach. Evtl. für noch kleinere k testen.

Query84

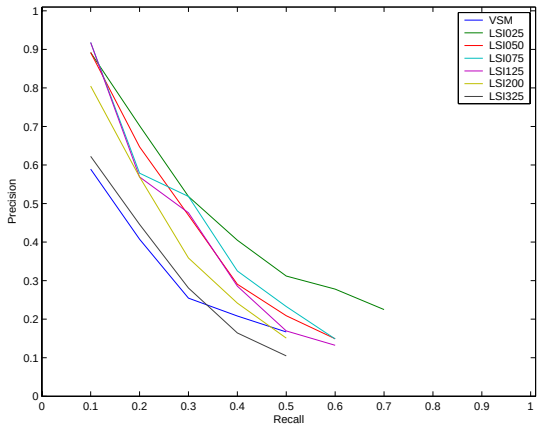

Abbildung 5.14: Anfrage 84

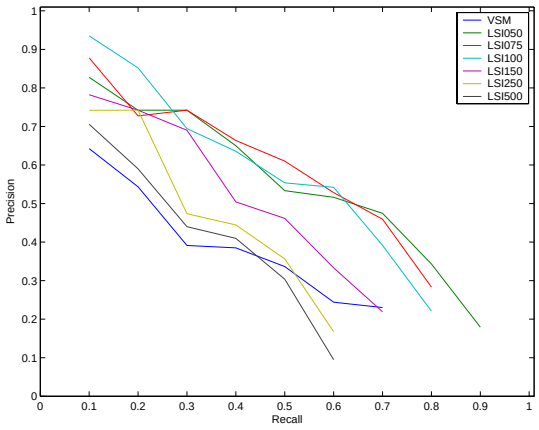

Abbildung 5.15: Anfrage 85

„Neue Medien im Unterricht“
Anzahl relevante Dokumente: 165

- VSM
  Durchschnittliches Ergebnis.
  Durchschnittliche Precision: 0,35
  Durchschnittlicher Recall: 0,28

- LSI
  Gutes Ergebnis für k $\approx$ 25.
  Durchschnittliche Precision für k = 25: 0,60
  Durchschnittlicher Recall: 0,29

Query85

„Europäische Bildungssysteme im Vergleich“
Anzahl relevante Dokumente: 144

- VSM
  Durchschnittliches Ergebnis.
  Durchschnittliche Precision: 0,47
  Durchschnittlicher Recall: 0,36

- LSI
  Gutes Ergebnis für k ≈ 100.
  Durchschnittliche Precision für k = 100: 0,70
  Durchschnittlicher Recall: 0,32
  Optimaler k-Wert wahrscheinlich zwischen 75 und 100.

Query86

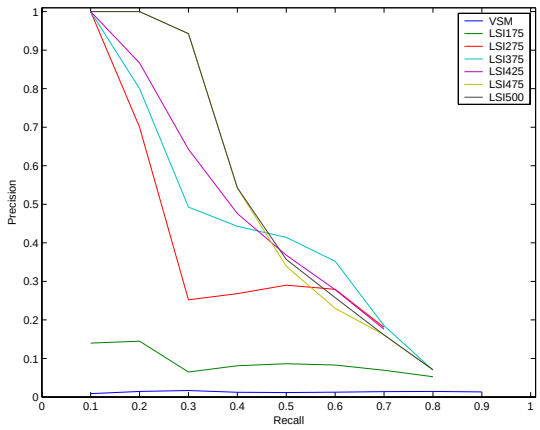

Abbildung 5.16: Anfrage 86

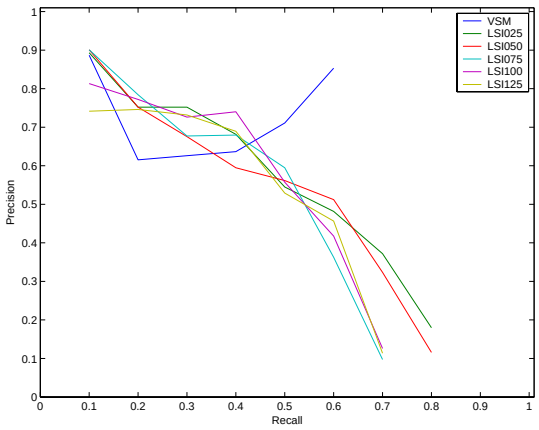

Abbildung 5.17: Anfrage 87

„Politische Symbole in Osteuropa“
Anzahl relevante Dokumente: 7

- VSM
  Sehr schlechtes Ergebnis für VSM, erster Treffer nach 80 Dokumenten.
  Durchschnittliche Precision: 0,64
  Durchschnittlicher Recall: 0,01

- LSI
  Hervorragende Ergebnisse für hohe k, k ≈ 500; evtl. mit noch höheren k testen.
  Durchschnittliche Precision für k = 500: 0,81
  Durchschnittlicher Recall: 0,03
  Bei k zwischen 25 und 125 erzielt LSI ähnlich schlechte Ergebnisse wie VSM - erst ab k = 175 verbessern sich die Ergebnisse langsam. Ergebnis bestätigt die Vermutung bei kleiner Anzahl an relevanten Dokumenten eine hohe Anzahl an Konzeptdimensionen zu wählen.

Query87

„Integration Behinderter in Schulen“
Anzahl relevante Dokumente: 91

- VSM
  Sehr gutes Ergebnis für VSM, mit einem steilen Anstieg ab 40% Recall (hier werden fast nur noch relevante Dokumente gefunden).
  Durchschnittliche Precision: 0,23
  Durchschnittlicher Recall: 0,66

- LSI
  Gutes Ergebnis für LSI, k ≈ 100; aber besser noch mit weiteren Anzahlen an k testen.
  Durchschnittliche Precision für k = 100: 0,65
  Durchschnittlicher Recall: 0,22
  Um sicher zu gehen, sollten noch weitere Tests mit höheren k durchgeführt werden. Wie sich aber aus dem Diagramm ergibt, fallen die Ergebnisse ab k = 125 wieder ab.

Query88

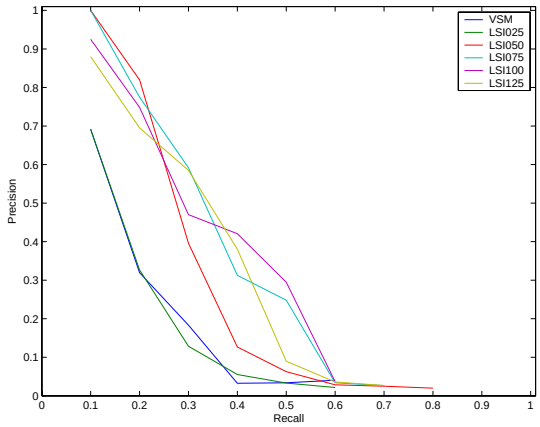

Abbildung 5.18: Anfrage 88

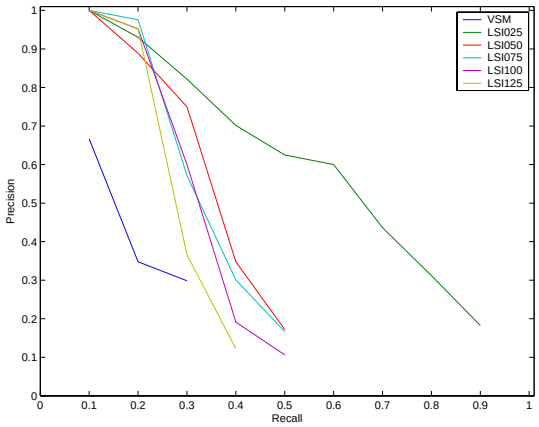

Abbildung 5.19: Anfrage 89

„Sport im Nationalsozialismus“
Anzahl relevante Dokumente: 23

- VSM
  Durchschnittliches Ergebnis, ab 40% Recall fast keine Treffer mehr.
  Durchschnittliche Precision: 0,32
  Durchschnittlicher Recall: 0,07

- LSI
  Sehr gutes Ergebnis für LSI, k $\approx$ 75.
  Durchschnittliche Precision für k = 75: 0,63
  Durchschnittlicher Recall: 0,06
  Evtl. noch weitere Tests mit höheren k durchführen, um Ergebnis zu validieren.

Query89

„Berufsethik im Journalismus“
Anzahl relevante Dokumente: 200

- VSM
  Durchschnittliches Ergebnis, relativ „kleine“ Ergebnismenge (382 Dokumente).
  Durchschnittliche Precision: 0,25
  Durchschnittlicher Recall: 0,35

- LSI
  Sehr gutes Ergebnis für k $\approx$ 25.
  Durchschnittliche Precision für k = 25: 0,71
  Durchschnittlicher Recall: 0,39
  Auch dieses Ergebnis bestätigt die Vermutung: Hohe Anzahl an relevanten Dokumente $\rightarrow$ niedrige Anzahl an Konzeptdimensionen.

Query90

„Zeitunglesen“
Anzahl relevante Dokumente: 125
Für beide Modelle keine Ergebnisse, was darin begründet liegt, dass der Term „Zeitunglesen“ in keinem der Titel vorkommt. Wird die Suchanfrage in „Zeitung lesen“ umgewandelt, so erscheinen die erwarteten Ergebnisse. Eine mögliche Lösung wäre einerseits eine Kompositazerlegung (VSM würde enorm davon profitieren) oder andererseits das

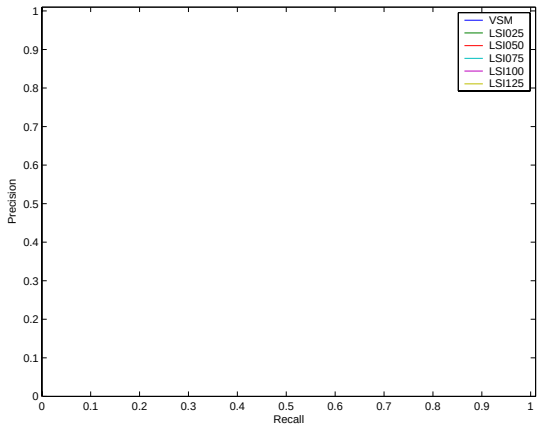

Abbildung 5.20: Anfrage 90

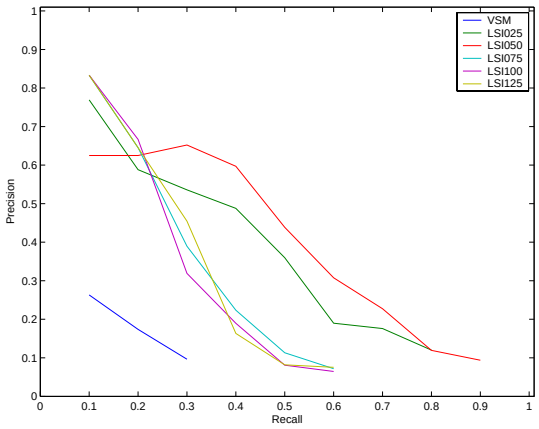

Abbildung 5.21: Anfrage 91

Trainieren mit einer größeren Dokumentenmenge (LSI könnte wahrscheinlich den Kontext von „Zeitunglesen“ aus der höheren Anzahl an Dokumenten herausfinden und somit bessere Ergebnisse liefern).
Diese Anfrage wird aus den Durchschnittsberechnungen herausgenommen.

Query91

„Qualität von Bildungssystemen“
Anzahl relevante Dokumente: 100

- VSM
  Schlechtes Ergebnis.
  Durchschnittliche Precision: 0,21
  Durchschnittlicher Recall: 0,16

- LSI
  Durchschnittliches bis gutes Ergebnis für LSI, k $\approx$ 25, wahrscheinlich aber zwischen 25 und 50.
  Durchschnittliche Precision für k = 25: 0,68
  Durchschnittlicher Recall: 0,20

Query92

„Parteimitglieder“
Anzahl relevante Dokumente: 103

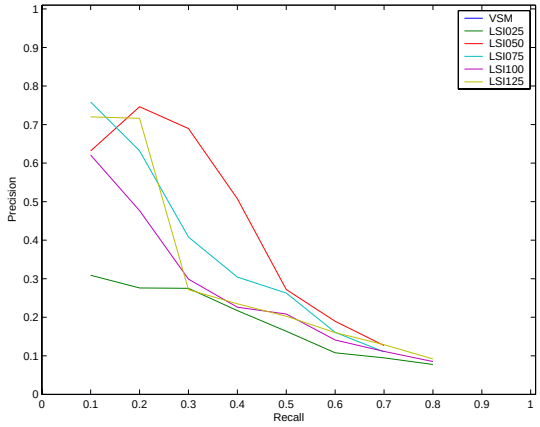

Abbildung 5.22: Anfrage 92

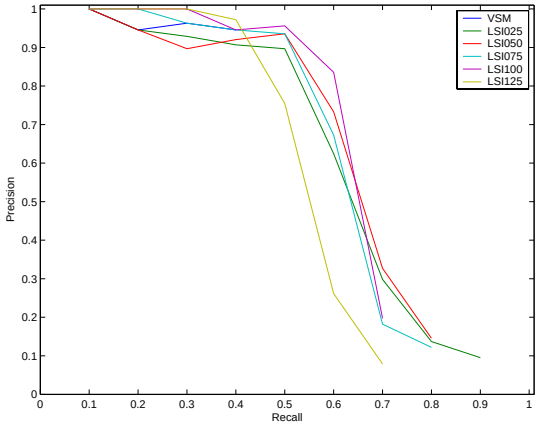

Abbildung 5.23: Anfrage 93

- VSM
  5 Volltreffer, danach keine weiteren Ergebnisse.
  Durchschnittliche Precision: 0,03
  Durchschnittlicher Recall: 1,00

- LSI
  Durchschnittliche Ergebnis für k $\approx$ 125.
  Durchschnittliche Precision für k = 125: 0,61
  Durchschnittlicher Recall: 0,19
  Tests für größere k müssen durchgeführt werden - optimale Anzahl an Konzepten wahrscheinlich höher als 125.

Query93

„Burnout-Syndrom“
Anzahl relevante Dokumente: 87

- VSM
  Sehr gutes Ergebnis für VSM, sehr hohe durchschnittliche Recall-Werte.
  Durchschnittliche Precision: 0,22
  Durchschnittlicher Recall: 0,96

- LSI
  Nahezu ideales Ergebnis für LSI, k $\approx$ 100.
  Durchschnittliche Precision für k = 100: 0,70
  Durchschnittlicher Recall: 0,25

Die ersten 30 Ergebnisse Volltreffer. Optimaler Wert für k liegt wahrscheinlich zwischen 100 und 125 - weitere Tests könnten dies validieren.

Query94

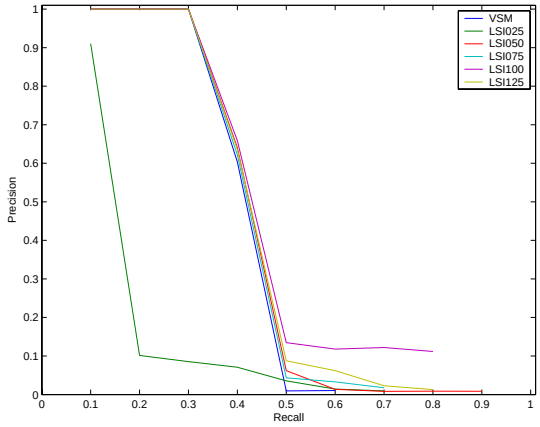

Abbildung 5.24: Anfrage 94

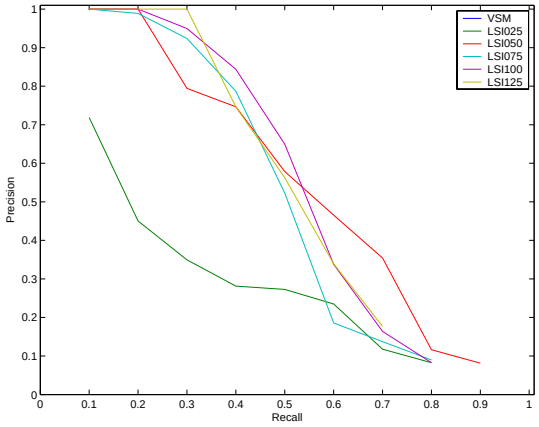

Abbildung 5.25: Anfrage 95

„Homosexualität und Coming-out“
Anzahl relevante Dokumente: 11

- VSM
  Schlechtes Ergebnis, 4 Volltreffer, danach erst nach über 500 Dokumenten wieder ein Treffer.
  Durchschnittliche Precision: 0,48
  Durchschnittlicher Recall: 0,03

- LSI
  Sehr gutes Ergebnis für LSI, k $\approx$ 100.
  Durchschnittliche Precision für k = 100: 0,79
  Durchschnittlicher Recall: 0,05
  Weitere Tests für größere k sollten durchgeführt werden - optimaler Wert evtl. höher als 100.

Query95

„Sonntagsarbeit“
Anzahl relevante Dokumente: 81

- VSM
  Zunächst 12 Volltreffer, danach keine weiteren Ergebnisse.
  Durchschnittliche Precision: 0,08
  Durchschnittlicher Recall: 1,00

- LSI
  Hervorragende Ergebnisse für LSI, k ≈ 125.
  Durchschnittliche Precision für k = 125: 0,69
  Durchschnittlicher Recall: 0,20
  Aber weitere Tests nötig, da k = 100 in Teilbereichen besser ist als k = 125 - entweder Wert zwischen 100 und 125 oder größer als 125.

Query96

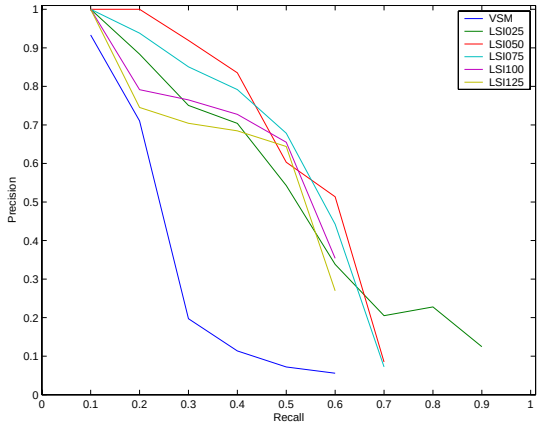

Abbildung 5.26: Anfrage 96

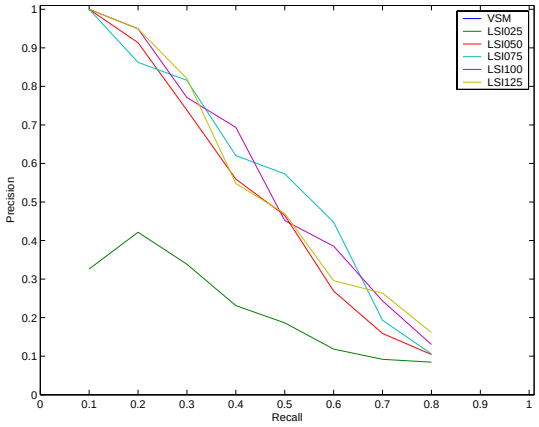

Abbildung 5.27: Anfrage 97

„Kosten der Berufsausbildung“
Anzahl relevante Dokumente: 76

- VSM
  Zu Beginn recht gut, doch ab 20% stark abfallend.
  Durchschnittliche Precision: 0,45
  Durchschnittlicher Recall: 0,13

- LSI
  Sehr gutes Ergebnis für k ≈ 50.
  Durchschnittliche Precision für k = 50: 0,64

Durchschnittlicher Recall: 0,22
Optimaler Wert wahrscheinlich zwischen 50 und 75.

Query97

„Sekten“
Anzahl relevante Dokumente: 94

- VSM
  17 Volltreffer, danach keine weiteren Ergebnisse.
  Durchschnittliche Precision: 0,07
  Durchschnittlicher Recall: 1,00

- LSI
  Sehr gutes Ergebnis für k ≈ 100.
  Durchschnittliche Precision für k = 100: 0,72
  Durchschnittlicher Recall: 0,22
  Optimaler Wert wahrscheinlich zwischen 100 und 125 - weitere Tests nötig.

Query98

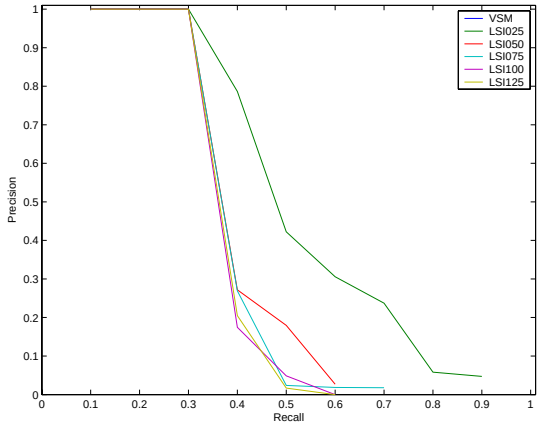

Abbildung 5.28: Anfrage 98

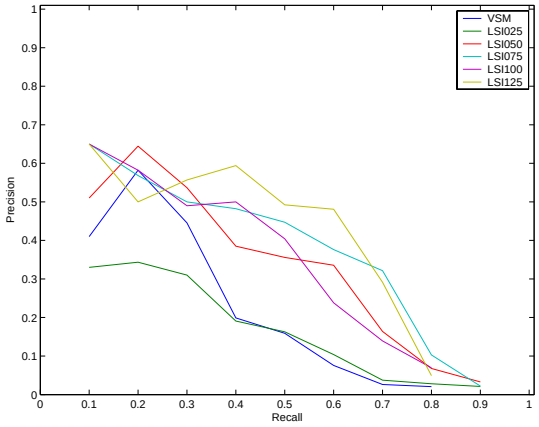

Abbildung 5.29: Anfrage 99

„Außenpolitik Kanadas“
Anzahl relevante Dokumente: 19

- VSM
  7 Volltreffer, VSM liefert gute Ergebnisse.

Durchschnittliche Precision: 0,23
Durchschnittlicher Recall: 0,98

- LSI
  Sehr gutes Ergebnis für k ≈ 25.
  Durchschnittliche Precision für k = 25: 0,82
  Durchschnittlicher Recall: 0,08

Query99

„Sport und Jugendarbeit“
Anzahl relevante Dokumente: 21

- VSM
  VSM zu Beginn recht schlecht, kurzes Hoch bei 20% aber dann wieder starker Abfall.
  Durchschnittliche Precision: 0,67
  Durchschnittlicher Recall: 0,05

- LSI
  Durchwachsenes Ergebnis für LSI, k ≈ 125.
  Durchschnittliche Precision für k = 125: 0,
  Durchschnittlicher Recall: 0,
  Weitere Tests nötig, optimaler k-Wert wahrscheinlich höher als 125.

Query100

„Kneipenkultur“
Anzahl relevante Dokumente: 10

- VSM
  VSM und LSI beide zu Beginn recht gut, um dann bei 30% einzuknicken auf nahezu 0-Niveau.
  Durchschnittliche Precision: 0,15
  Durchschnittlicher Recall: 1,00

- LSI
  Exakt gleichverteilte, aber gute Ergebnisse für LSI, k ≈ 100.

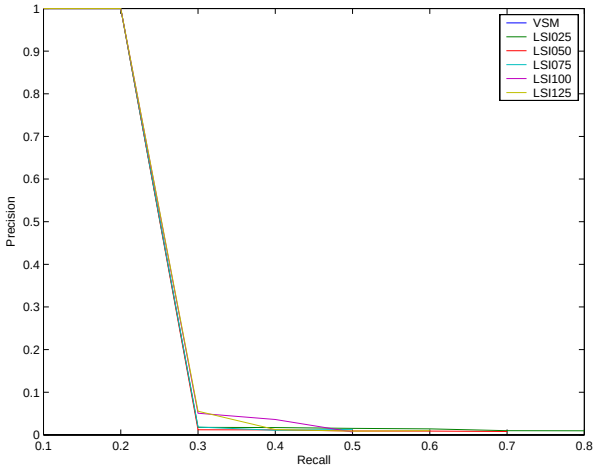

Abbildung 5.30: Anfrage 100

Durchschnittliche Precision für k = 100: 0,40
Durchschnittlicher Recall: 0,02
Tests mit größeren k sind nötig - vielleicht ähnlich überraschendes Ergebnis wie bei Anfrage 86.

5.3 Zwischenergebnisse

Die Analyse des in Kapitel 4 implementierten IR-Systems gestaltete sich viel komplexer als zunächst angenommen. Insgesamt wurden über 2500 Anfragen an das System gestellt um die jeweiligen Aspekte des Systems und deren Qualität zu überprüfen. Doch ein großes Manko von LSI konnte auch ohne Anfragen an das System lokalisiert werden. Die Berechnung der SVD-Matrizen. Der erzeugte Index hat zwar einen hochoptimierten Speicherplatzverbrauch, allerdings muss dies durch die enorm arbeitsspeicherlimitierten Berechnungen erkauft werden. Nicht ganz so schlimm sieht der Ressourcenverbrauch aus, wenn man andere Programme zur SVD-Berechnung nutzt als Matlab: In der Arbeit von Tang et al. (2004) wurde versucht LSI auf einer Teilmenge der TREC-Datensammlung (insgesamt 2GB groß) zu implementieren - allerdings mußte die Dokumentenkollektion auf 15% ihrer eigentlichen Ausmaße geschrumpft werden um die SVD-Berechnungen zu ermöglichen - und der Arbeitsspeicherverbrauch lag trotzdem bei 1,7GB.
Nachdem die Hürde genommen wurde konnten erste Tests über die Geschwindigkeit des Systems durchgeführt werden. Diese Betrug zu Beginn dieser Untersuchungen 40min pro Anfrage - Versuche die Anfragedauer auf ein erträgliches Minimum zu schrumpfen scheiterten kläglich. Doch dann stellte sich heraus, dass der Arbeitsspeicher des Rechners,

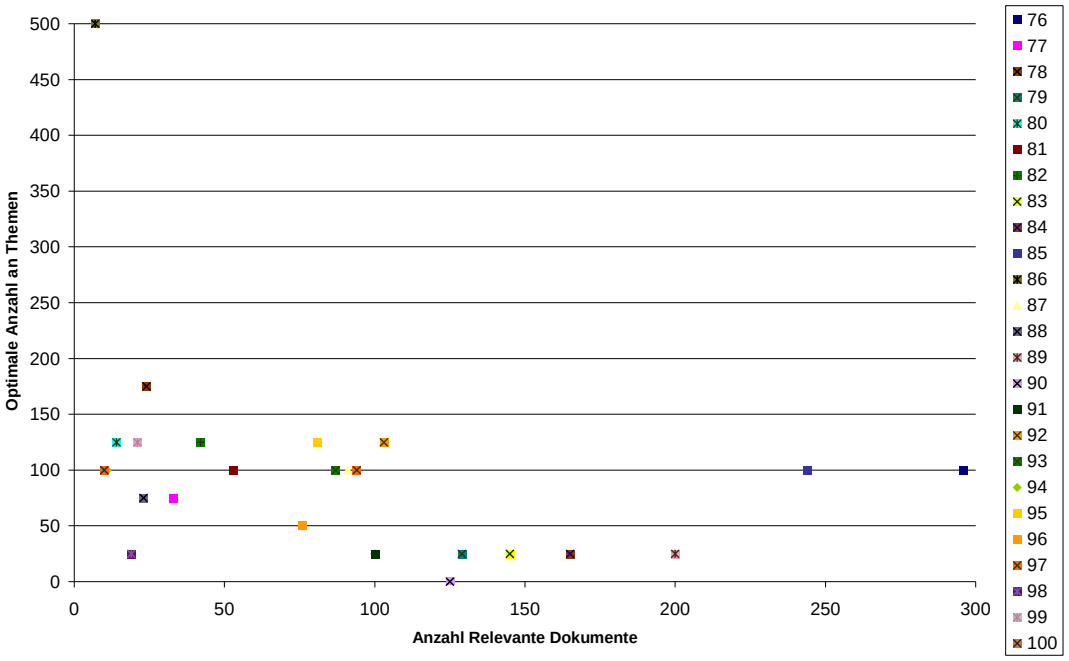

Abbildung 5.31: Verhältnis Anzahl relevante Dokumente zu Themen

auf dem das Datenbankmanagementsystem lieg, ständig an seine Grenzen geführt wurde. Nachdem dieses Problem behoben wurden dauerten die Anfragen nur noch zwischen 12 und 60 Sekunden - je nach gewählter Anzahl an Konzeptdimensionen. Für ein im Tageseinsatz einsatzfähiges System noch viel zu langsam. Die Anfragedauer sollte durch Optimierungen in den Millisekundenbereich gedrückt werden.
Der Speicherverbrauch der SVD ist zu vernachlässigen, jedes Vektorraummodell benötigt ein vielfaches eines LSI-Systems.
Kommen wir nun zur Analyse der Anfrageergebnissen: Die Tests müßten um endgültige Aussagen zu treffen für die noch fehlende Anzahl an Dimensionen durchgeführt werden - gerade die Anfragen 80, 81, 82, 92, 95 und 99 lassen optimale k-Werte jenseits der 125 vermuten. LSI liefert über die ganzen Anfragen gesehen gute bis sehr gute Ergebnisse. Das Vektorraummodell erreicht im Durchschnitt $\tilde{3}0\%$ schlechtere Precision-Werte als LSI. Es kann LSI also eine Gute Retrieval-Güte bescheinigt werden.
Zum Schluß der Retrieval-Tests sei noch auf die Abbildung 5.31 hingewiesen - sie zeigt die Verteilung der Einzelnen Anfragen im Verhältnis von Anzahl relevanter Dokumente zur optimalen Anzahl an Themen.

6 Zusammenfassungen & Ausblick

Das Versprechen, ausnahmslos bessere Ergebnisse zu liefern als das Vektorraummodell, konnte LSI nur in einem Fall nicht ganz erfüllen (siehe Anfrage 82 auf Seite 62). Den hohen Erwartungen konnte LSI also auch dann gerecht werden, wenn der genutzte Dokumentenkorpus - wie der angepasste GIRT4-Datensatz - vornehmlich aus kurzen Texten besteht und die Anfragen aus durchschnittlich 2,5 Termen bestehen. Die Vorteile gegenüber dem Vektorraummodell stechen teilweise sogar noch deutlicher hervor, als dies bei größeren Dokumentenkorpussen der Fall wäre (siehe Anfragen 77, 83, 86 und 91). Eine Pauschalisierung, wie diese Ergebnisse erreicht werden können, lässt sich aber nicht vornehmen. Ähnlich dem, wie Nakov (2000a) es so schön formuliert hat („The usage of LSI is a kind of art and is dependent on a variety of parameters that must be tuned very carefully in order to achieve better results“), ist die Anwendung von LSI nicht mit dem Kochen nach Rezept zu vergleichen - vielmehr muss mal hier mal da ein bisschen Pfeffer (Termgewichtung), eine Prise Salz (Ähnlichkeitsmaß), verschiedene Kräuter (Nutzen von Stopwortelimination) oder etwas Zucker hinzugegeben (Anzahl Konzepte) werden, um den Geschmack den eigenen Wünschen anzupassen. Genau wie im richtigen Leben kann man zwar abschätzen, ob der Geschmack einer Mahlzeit jemandem mundet. Definitive Aussagen können aber erst getroffen werden, wenn derjenige das Essen probiert hat. So auch bei einem auf LSI beruhenden IR-System: ohne die genauen Zutaten und ihre Dosierung zu kennen, kann man die Retrieval-Güte nicht vorhersagen. Erst nach ausgiebigem Testen des Systems mit verschieden strukturierten Anfragen können Schlüsse auf die Retrievalgüte gezogen werden.
Eine Regel fürs LSI-Kochrezept - für die Menge an „Zucker“, die hinzugegeben werden soll - konnte allerdings für die GIRT4-Daten gefunden werden: Wenn sich zu einer Anfrage eine vermutlich hohe Anzahl an relevanten Dokumenten im Korpus befindet, so sollte eine niedrige Anzahl an Dimensionen gewählt werden - für den GIRT4-Korpus ist der Anhaltspunkt 25 Dimensionen. Wenn die zu vermutende relevante Ergebnismenge allerdings sehr wenig Dokumente enthält, so muss eine sehr hohe Anzahl an Konzept-Dimensionen gewählt werden (beim GIRT4-Datensatz ≈ 500). Bei Ergebnismengen zwischen diesen

beiden Extremen stellten sich Werte um 125 Konzeptdimensionen als wirksam heraus. Ob sich diese Regel auch auf andere Dokumentensammlungen übertragen lässt, muss durch genauere Test validiert werden.
Vielleicht kann man noch einen Schritt weiter gehen und behaupten, dass die Anzahl der Konzepte auf die Verteilung der Terme im Dokumentenkorpus zurückgeführt und mit einer Formel berechnet werden kann. Nehmen wir wieder die auf Seite 25 genutzte Abbildung (6.1) als Demonstrationsobjekt:

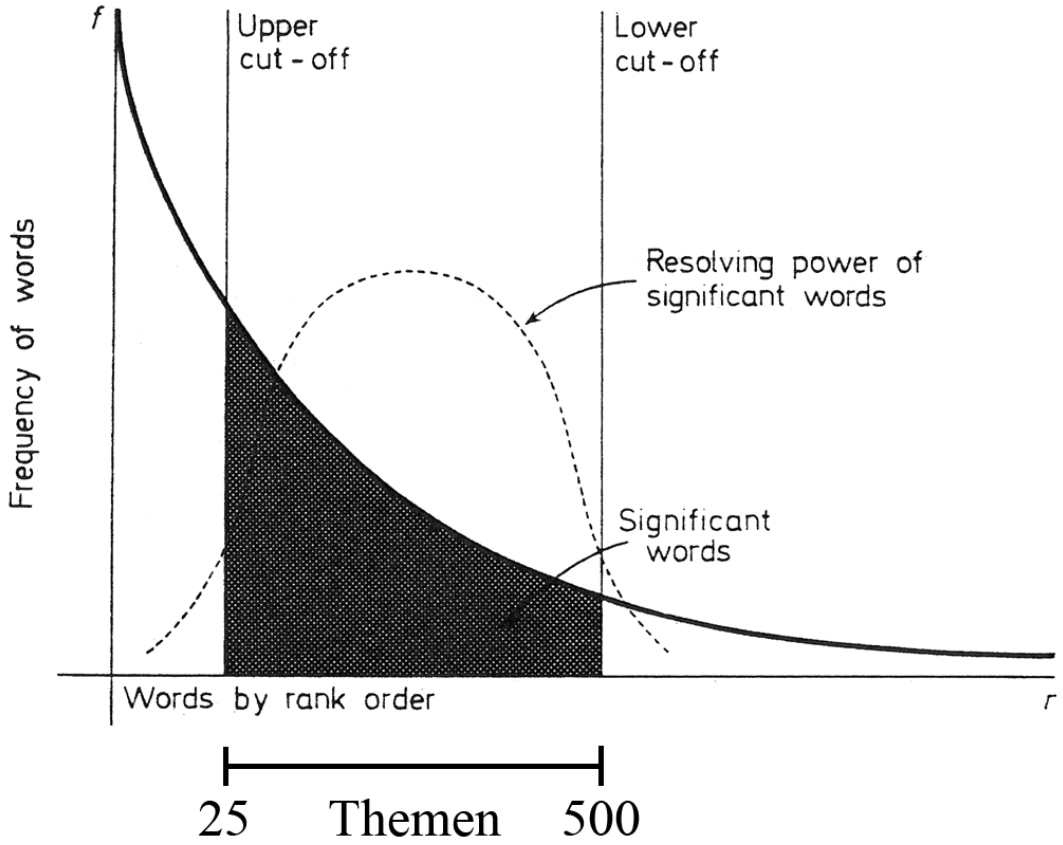

Abbildung 6.1: Term-Rang-Frequenz-Diagramm erweitert um Bestimmung der Anzahl der Konzepte

Je seltener ein Term in der Dokumentenkollektion auftritt, um so höher ist sein Rang. Nehmen wir an, ein Term kommt sehr selten in der Dokumentenkollektion vor (z.B. „ökonomische“ in diesem Korpus nur 4 Mal), so spricht dies dafür, dass er bei den Berechnungen der Konzeptdimensionen nur sehr wenig Einfluss auf die vorderen Dimensionen hat - er wird wahrscheinlich erst in einer der höheren Dimensionen stärkeren Einfluß nehmen können ($k \rightarrow 500$). Würde man nun, bevor die eigentliche Suchanfrage an die SVD-Matrizen geschickt wird, eine Auftretensanalyse der in der Suchanfrage

enthaltenen Terme vornehmen, so könnte man anhand des am seltensten auftretenden Term der Suchanfrage die Anzahl der Konzeptdimensionen festmachen (evtl. müsste ein Gewichtungsverfahren auf die Auftretenshäufigkeit angewendet werden und nicht allein die Auftretenshäufigkeit als Maß für diese Bestimmung). Bei in den GIRT4-Titeln selten auftretenden Termen konnte die These nachvollzogen werden - bei anderen Korpussen oder größeren Dokumentkollektionen müsste das Verfahren näher evaluiert werden.
Falls die These validiert werden kann, so müsste die Festlegung für eine Anzahl an Themen nicht nur ein einziges Mal für den kompletten Korpus vollzogen werden, sondern für jede Anfrage neu, da die Unterschiede zwischen den Dokumenten einer Dokumentenkollektion sich nicht über eine starre Anzahl an Konzeptdimensionen behandeln lassen können.

6.1 Optimierungsmöglichkeiten

Das größte Problem von LSI stellt die für die Dimensionreduktion genutzte SVD dar: Bei sich schnell ändernden Dokumentensammlungen und bei großen Term-Dokumenten-Matrizen lässt sich die Standard-SVD nicht nutzen. In diesem Bereich wird allerdings intensiv geforscht, wie die Arbeiten Braun et al. (2002); Oksa et al. (2002) belegen. Alternativen zur SVD sind rar gesät und wenig elaboriert.
Je nach Anwendungsgebiet könnte auch eine Kompositazerlegung sinnvoll sein. Anfrage 90 (siehe Seite 67) ist ein Paradebeispiel für eine Steigerung der Retrieval-Güte mit Hilf der Wortzerlegung in Komposita. Wie sich dies allerdings auf die Retrieval-Performance der anderen Anfragen auswirkt, kann wiederum nur durch Tests evaluiert werden.

6.1.1 Boolean LSI

Das Nutzen von Boolean-Operatoren wäre in einigen Anfragen (z.B. „Industrielle und ökonomische Entwicklung Indiens“) der Testkollektion hilfreich gewesen. Die Studien Nakov (2000b); Jeliazkov und Nakov (2002); Baek et al. (2000) gehen auf die Erweiterung LSIs mit Boolean-Operatoren ein und zeigen in vielen Anwendungsgebieten Vorteile gegenüber dem Standardverfahren.

6.1.2 Probabilistic LSI

In der Arbeit von Hofmann (1999) wird als Dimensionsreduktion nicht die SVD genutzt sondern auf Bayes beruhende Wahrscheinlichkeitsberechnungen. Wie in Abbildung 6.2

zu sehen sind die Ergebnisse sehr vielversprechend - eine genauere Betrachtung gerade im Hinblick auf die Berechnungs-Probleme der SVD erscheint sehr sinnvoll.

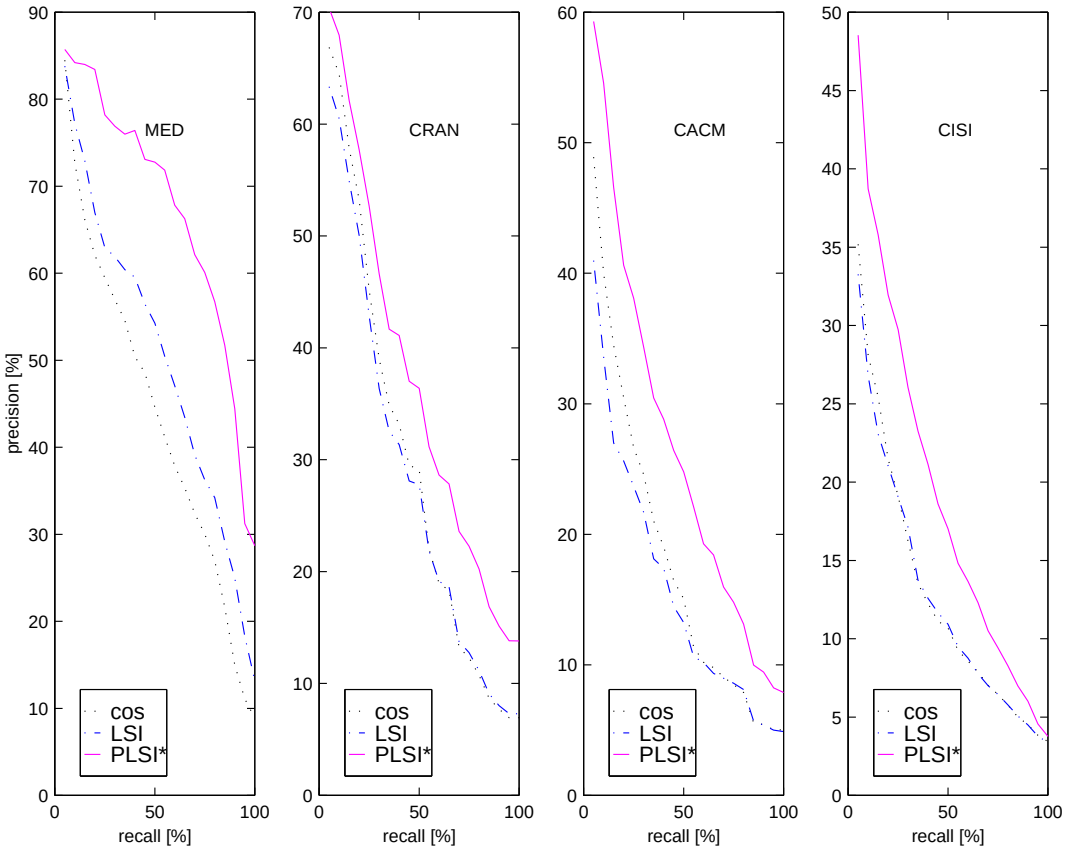

Abbildung 6.2: Retrievalergebnisse für Probabilistic LSI (Y-Achsenbelegung verzerrt)

6.1.3 Sparsification

Die Arbeiten Kontostathis et al. (2004); Gao und Zhang (2003); Kontostathis et al. (2005) beschäftigen sich mit der „Sparsification“ der durch die SVD entstandenen Matrizen. Aus den Matrizen U und V werden Einträge entfernt, die ein niedriges Gewichtungsmaß haben. Mit diesem Verfahren können 70 - 90% der Einträge der Matrizen entfernt werden (siehe Abbildung 6.3), was zur Folge hätte, dass die Matrizen und die Anfragebearbeitung weniger Ressourcen benötigen. In den meisten Fällen wird dies erreicht ohne einen Nachteil bei der Güte der Ergebnisse zu erleiden - in manchen Fällen geht sogar eine Verbesserung der Retrieval-Ergebnisse einher, da unwichtige Einträge das Ergebnis nicht mehr „verrauschen“ können. Beim Entfernen zu vieler Daten kann die Performance allerdings auch sinken (siehe Abbildung 6.4).

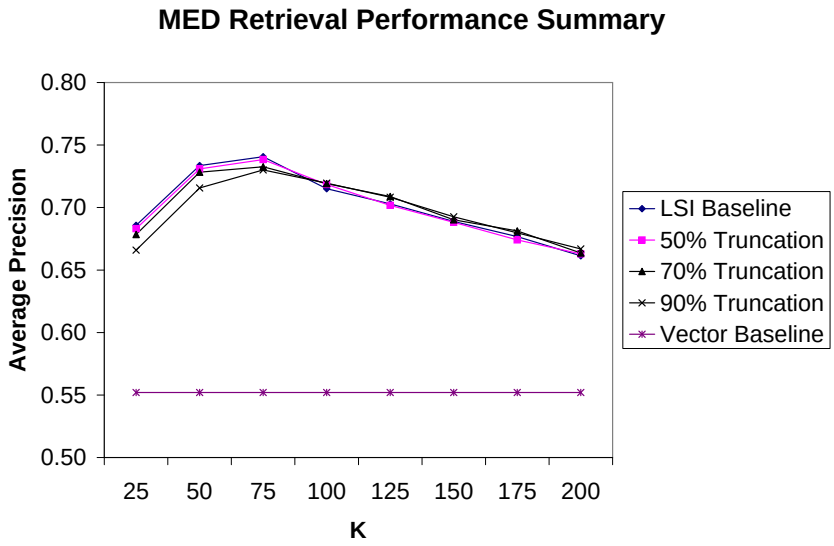

Abbildung 6.3: Sparsification Performance bei der MED-Kollektion Quelle: Kontostathis et al. (2005)

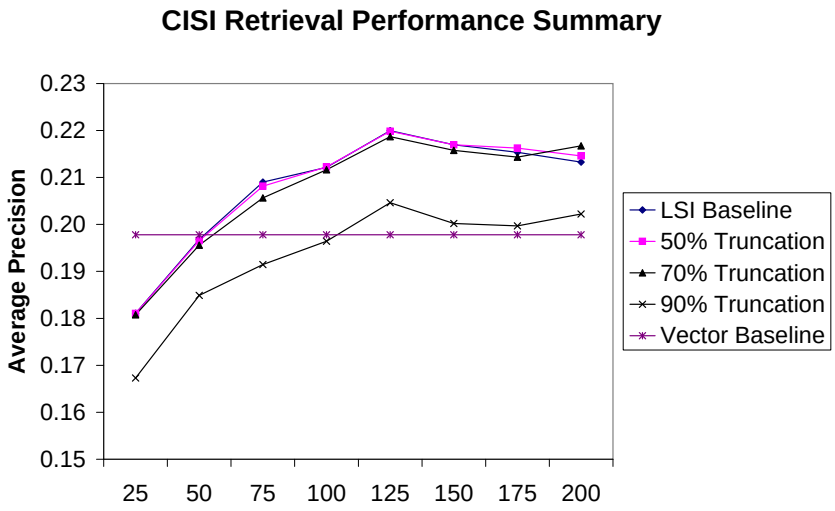

Abbildung 6.4: Sparsification Performance bei der CISI-Kollektion Quelle: Kontostathis et al. (2005)

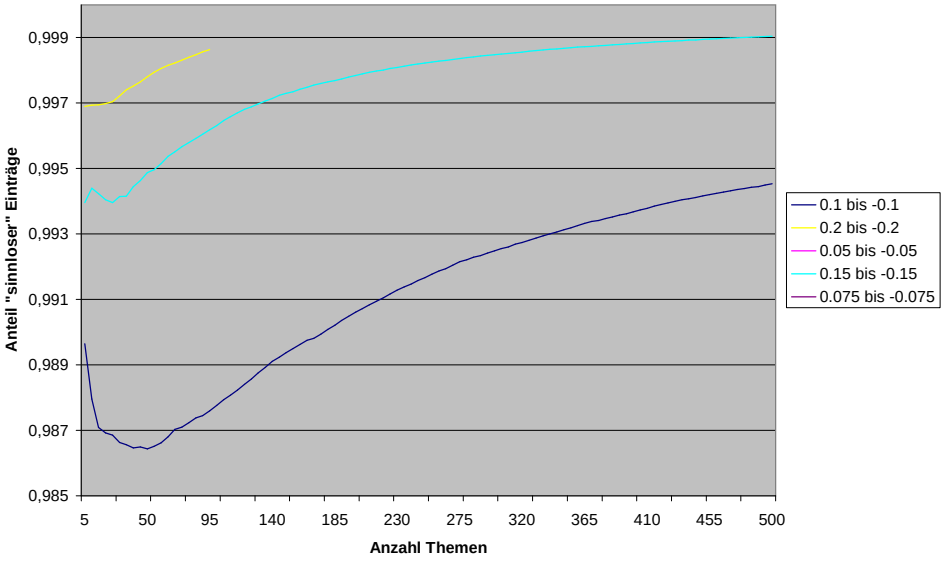

Abbildung 6.5: Verteilung der Zelleneinträge mit „sinnlosen“ Werten

Inspiriert durch die Arbeiten über Sparsification der SVD-Matrizen, wurden für den GIRT4-Dokumentenkorpus kurze Tests auf Sparsification durchgeführt:
Anzahl der Einträge der Matrize U (bei 500 Themen):
724.500
Anzahl der Einträge der Matrize U (bei 500 Themen) die zwischen 0.01 und -0.01 liegen:
302.179
→ 41,7% der Werte liegen zwischen 0.01 und -0.01.
Diese Berechnungen wurden für verschiedene Intervalle durchgeführt und in einem Diagramm (Abbildung 6.5) abgetragen. Das Intervall zwischen 0,1 und -0,1 liefert zu Beginn einen steilen Abfall, um dann seine negative Steigung langsam in eine positive umzuwandeln, die nach kurzer Zeit aber immer mehr abnimmt. Kleinere Intervalle wie 0,05 bis -0,05 lieferten flachere Kurven. Meine Hypothese ist nun, dass man mit Hilfe dieser Graphen die optimale Anzahl an Themen für einen Korpus finden kann. Man berechnet zunächst für viele Intervalle den jeweiligen Graphen. Dann versucht man die Funktionen der jeweiligen Graphen zu berechnen und sucht anschließend den Graphen, der in irgendeinem Abschnitt die größte Steigung besitzt. Berechnet man nun für diesen Graphen die Nullstelle der 1. Ableitung, so erhält man den Wert bei dem seine Steigung 0 beträgt: Der Wert für die optimale Anzahl an Themen ist gefunden.
Auch hier müssen weitere Tests erst beweisen, ob die These gehalten oder verworfen werden kann.

6.1.4 Erweiterung um Phrasen-Suche

Inspiriert durch die verschiedenen Suchanfragen des GIRT4-Datensatzes („Politische Kultur“, „ökonomische Entwicklung“) und auch den eigenen Vorlieben und Erfahrungen im Bereich der Phrasensuche wurden nähere Überlegungen über die Einbindung einer Phrasensuche in LSI getätigt. Die untersuchten Methoden (Salton (1965); Fagan (1987); Sheridan und Smeaton (1992)) schienen mir unflexibel und würden die Möglichkeiten von LSI zu sehr beschneiden. Im Folgenden soll näher auf das Konzept eines auf LSI beruhenden „Weak-Phrase-Matching“ oder „Synonym-Phrase-Matching“-Verfahrens eingegangen werden: Jede Suchanfrage wird in einzelne Tokens zerlegt und zu jedem Token eines zuvor als Phrase gekennzeichneten Abschnitts wird eine Term-Ähnlichkeitsanfrage an den Vektorraum des LSI-Modells gestellt. Überschreitet ein Term eine gewisse Mindestähnlichkeit, so wird er zusammen mit dem Ursprungsterm zu einer Menge hinzugefügt. Dieses Verfahren wird für jeden Term der Phrase durchgeführt. Nachdem nun die Termmengen generiert wurden, kann mit der eigentlichen Phrasen suche begonnen werden. Grundlage hierfür ist das Speichern der Position eines jeden Terms - in der in Kapitel 4 beschriebenen Implementation sind die ersten Vorarbeiten für eine effektive Positionsspeicherung vollzogen. Nun wird in der Menge des ersten Terms nach direkten Nachfolgern gesucht, die in der Menge des zweiten Terms enthalten sind. Nachdem dies bis zur letzten Termmenge durchgeführt wurde, kann theoretisch schon eine Ergebnismenge ausgegeben werden - aber wahrscheinlich müsste zunächst noch ein Gewichtungsmaß auf die Termmengen angewendet werden. Der große Vorteil dieser Phrasensuche wäre, dass sie auch Phrasen findet, die nicht exakt mit der eingegebenen Phrase übereinstimmen - dies hört sich auf den ersten Eindruck eher widersinnig an. Doch gegeben sei folgendes Beispiel: Der Nutzer tippt als Phrase „Laptop Computer“ ein - bei einer normalen Phrasensuche würden nur Dokumente gefunden, die den Term Laptop enthalten. Bei der hier dargestellten Phrasensuche würden auch Dokumente gefunden, die nur die Phrase „Notebook Computer“ enthalten. Wenn eine normale Phrasensuche durchgeführt werden soll, dann kann das Gewichtungsmaß besonders kritisch eingestellt werden, wenn eine etwas freizügigere Phrasensuche vollzogen werden soll, so kann das Maß nach und nach gelockert werden. Aber wie bei allen anderen Hypothesen gilt auch hier, dass erst ein lauffähiges System die Qualität und Nützlichkeit eines solchen Systems validieren kann.

Literaturverzeichnis

[Adamson und Bush 1973]
ADAMSON, G. W. ; BUSH, J. A.: A method for the automatic classification of chemical structures. In: *Information Storage and Retrieval* 9 (1973), S. 561–568

[Anderberg 1973]
ANDERBERG, Michael R.: *Cluster Analysis for Applications - Probability and Mathematical Statistics*. New York, NY, USA : Academic Press, 1973. – 359 S. – ISBN 0–12–057650–3

[Baek et al. 2000]
BAEK, Dae-Ho ; LIM, HeuiSeok ; RIM, Hae-Chang: Latent semantic indexing model for Boolean query formulation (poster session). In: *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA : ACM Press, 2000. – ISBN 1–58113–226–3, S. 310–312

[Baeza-Yates und Ribeiro-Neto 1999]
BAEZA-YATES, Ricardo A. ; RIBEIRO-NETO, Berthier A.: *Modern Information Retrieval*. ACM Press / Addison-Wesley http://citeseer.ist.psu.edu/baeza-yates99modern.html. – ISBN 0–201–39829–X

[Bartell et al. 1992]
BARTELL, Brian T. ; COTTRELL, Garrison W. ; BELEW, Richard K.: Latent Semantic Indexing is an Optimal Special Case of Multidimensional Scaling. In: *Research and Development in Information Retrieval*, 161-167

[Berry und Dumais 2004]
BERRY, Michael W. ; DUMAIS, Susan: *LSI-Related Publications (Articles, Reports, and Books)*. http://www.cs.utk.edu/~lsi/papers/index.html. Version: may 2004

[Braun et al. 2002]
Braun, Tracy D. ; Ulrey, Renard ; Maciejewski, Anthony A. ; Siegel, Howard J.: Parallel approaches for singular value decomposition as applied to robotic manipulator Jacobians. In: *Int. J. Parallel Program.* 30 (2002), Nr. 1, S. 1–35. http://dx.doi.org/http://dx.doi.org/10.1023/A:1013270420397. – DOI http://dx.doi.org/10.1023/A:1013270420397. – ISSN 0885–7458

[Broglio et al. 1994]
Broglio, John ; Callan, James P. ; Croft, W. B. ; Nachbar, Daniel W.: Document Retrieval and Routing Using the INQUERY System. In: *Overview of the Thrid Text REtrieval Conference (TREC-3)*, 29-38

[Chung und O'Neil 1997]
Chung, G. ; O'Neil, G.: *Methodological Approaches to online scoring of essays.* citeseer.ist.psu.edu/chung97methodological.html. Version: 1997

[Coughran 2004]
Coughran, Bill: *Google's index nearly doubles.* http://googleblog.blogspot.com/2004/11/googles-index-nearly-doubles.html. Version: nov 2004

[Deerwester et al. 1990]
Deerwester, Scott C. ; Dumais, Susan T. ; Landauer, Thomas K. ; Furnas, George W. ; Harshman, Richard A.: Indexing by Latent Semantic Analysis. In: *Journal of the American Society of Information Science* 41 (1990), Nr. 6, 391-407. http://citeseer.ist.psu.edu/deerwester90indexing.html

[Dennis 2003]
Dennis, Simon: *Extracting Lexical Semantics from Text.* http://lsa.colorado.edu/~simon/LexicalSemantics/. Version: 2003

[Dennis et al. 2003]
Dennis, Simon ; Landauer, Tom ; Kintsch, Walter ; Quesada, Jose: Latent Semantic Analysis: Theory, Use and Applications. In: Alterman, Richard (Hrsg.) ; Kirsh, David (Hrsg.): *Proceedings of the 25th Annual Meeting of the Cognitive Science Society, 2003.* Cognitive Science Society

[Dumais 1990]
Dumais, Susan T.: *Enhancing Performance in Latent Semantic Indexing Retrieval.* citeseer.ist.psu.edu/dumais92enhancing.html. Version: 1990

[Dumais 1994]
DUMAIS, Susan T.: *Latent Semantic Indexing (LSI) and TREC-2.* citeseer.ist.psu.edu/19248.html. Version: 1994

[Dumais 1995]
DUMAIS, Susan T.: *Latent Semantic Indexing (LSI): TREC-3 Report.* citeseer.ist.psu.edu/431.html. Version: 1995

[Dumais et al. 1988]
DUMAIS, Susan T. ; FURNAS, George W. ; LANDAUER, Thomas K. ; DEERWESTER, Scott C. ; HARSHMAN, Richard: Using latent semantic analysis to improve access to textual information. In: *CHI '88: Proceedings of the SIGCHI conference on Human factors in computing systems.* ACM Press. – ISBN 0–201–14237–6, 281–285

[Dumais et al. 1997]
DUMAIS, Susan T. ; LANDAUER, Thomas K. ; LITTMAN, Michael: *Automatic cross-linguistic information retrieval using Latent Semantic Indexing.* citeseer.ist.psu.edu/dumais97automatic.html. Version: jul 1997

[Dupret 2004]
DUPRET, Georges: Latent Semantic Indexing with a Variable Number of Orthogonal Factors. In: *RIAO 2004 - Session 13: Improving Retrieval*, 2004

[Fagan 1987]
FAGAN, J.: Automatic phrase indexing for document retrieval. In: *SIGIR '87: Proceedings of the 10th annual international ACM SIGIR conference on Research and development in information retrieval.* New York, NY, USA : ACM Press, 1987. – ISBN 0–89791–232–2, S. 91–101

[Ferber 2003]
FERBER, Reginald: *Information Retrieval - Suchmodelle und Data-Mining-Verfahren für Textsammlungen und das Web.* dpunkt-verlag http://information-retrieval.de/irb/irb.html. – ISBN 3898642135

[Frakes und Baeza-Yates 1992]
FRAKES, William B. ; BAEZA-YATES, Ricardo: *Information retrieval: data structures and algorithms.* Upper Saddle River, NJ, USA : Prentice-Hall, Inc., 1992. – ISBN 0–13–463837–9

[Furnas et al. 1984]
FURNAS, G W. ; LANDAUER, T K. ; GOMEZ, L M. ; DUMAIS, S T.: Statistical semantics: analysis of the potential performance of keyword information systems. (1984), S. 187–242. ISBN 0–89391–146–1

[Gao und Zhang 2003]
GAO, Jing ; ZHANG, Jun: Sparsification strategies in latent semantic indexing. In: BERRY, M. W. (Hrsg.) ; POTTENGER, W. M. (Hrsg.): *Text Mining Workshop*. San Francisco, CA, may 2003, S. 93–103

[Gebhardt 1981]
GEBHARDT, Friedrich: *Dokumentationssysteme*. Berlin - Heidelberg - New York : Springer-Verlag, 1981. – 331 S. – ISBN 3–540–10744–4

[Golub und Loan 1989]
GOLUB, Gene H. ; LOAN, Charles F. V.: *Matrix Computations*. second. Baltimore, Maryland, USA : Johns Hopkins University Press, 1989

[Griesbaum und Bekavac 2001]
GRIESBAUM, Joachim ; BEKAVAC, Bernard: *Bewertung und Evaluierung, Retrievaltests*. http://www.inf-wiss.uni-konstanz.de/CURR/winter0102/IR/v11_bewertung.pdf. Version: dec 2001

[Götze und Rieder 1996]
GÖTZE, Jürgen ; RIEDER, Peter: SVD-Updating Using Orthonormal Micro-Rotation. In: *Journal of VLSI Signal Processing (Special Issue on Array Optimization and Adaptive Tracking Algorithms)* 14 (1996), S. 7–18

[Haenelt 1997]
HAENELT, Karin: *Evaluierung von Information Retrieval Systemen*. http://kontext.fraunhofer.de/haenelt/kurs/folien/IR-Evaluierung01.pdf. Version: oct 1997

[Haenelt 2004]
HAENELT, Karin: *Ähnlichkeitsmaße für Vektoren*. Kursfolien - Hauptseminar Computerlinguistik. http://kontext.fraunhofer.de/haenelt/kurs/InfoRet/termine_folien.html. Version: nov 2004

[Harman 1986]
HARMAN, Donna W.: An experimental study of factors important in document ranking. In: *SIGIR '86: Proceedings of the 9th annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA : ACM Press, sep 1986. – ISBN 0–89791–187–3, S. 186–193

[Hoffmann 2004]
HOFFMANN, Horst-Joachim: *Google nimmt Bestände von Universitätsbibliotheken in Index auf*. http://www.heise.de/newsticker/meldung/54218. Version: dec 2004

[Hofmann 1999]
HOFMANN, Thomas: Probabilistic Latent Semantic Indexing. In: *Proceedings of the 22nd Annual ACM Conference on Research and Development in Information Retrieval*. Berkeley, California, August 1999, 50-57

[Hull 1996]
HULL, David A.: Stemming Algorithms: A Case Study for Detailed Evaluation. In: *Journal of the American Society of Information Science* 47 (1996), Nr. 1, 70-84. citeseer.ist.psu.edu/hull96stemming.html

[Jeliazkov und Nakov 2002]
JELIAZKOV, Jivko S. ; NAKOV, Preslav I.: Extended Boolean Operations in Latent Semantic Indexing Search. In: *Spring Conference of Bulgarian Mathematicians Union*. Bulgaria, Apr 2002

[Jones und Furnas 1987]
JONES, William P. ; FURNAS, George W.: Pictures of relevance: a geometric analysis of similarity measures. In: *Journal of the American Society for Information Science* 38 (1987), Nr. 6, S. 420–442. http://dx.doi.org/10.1002/(SICI)1097-4571(198711)38:6<420::AID-ASI3>3.0.CO;2-S. – DOI 10.1002/(SICI)1097–4571(198711)38:6<420::AID–ASI3>3.0.CO;2–S. – ISSN 0002–8231

[Jung et al. 2000]
JUNG, Yunjae ; PARK, Haesun ; DU, Ding zhu: An Effective Term-Weighting Scheme for Information Retrieval / Department of Computer Science and Engineering - University of Minnesota. Version: feb 2000. citeseer.ist.psu.edu/446757.html. 4-192 EECS Building, 200 Union Street SE, Minneapolis, MN 55455-0159 USA, feb 2000 (00-008). – Forschungsbericht. – Elektronische Ressource

[Kaufman und Rousseeuw 1990]
KAUFMAN, Leonard ; ROUSSEEUW, Peter J.: *Finding Groups in Data - An Introduction to Cluster Analysis*. New York, NY, USA : Wiley Series in Probability and Mathematical Statistics, 1990. – 342 S. – ISBN 0–471–87876–6

[Kontostathis et al. 2004]
KONTOSTATHIS, April ; POTTENGER, William M. ; DAVISON, Brian D.: *Assessing the Impact of Sparsification on LSI Performance*. citeseer.ist.psu.edu/article/kontostathis04assessing.html. Version: 2004

[Kontostathis et al. 2005]
KONTOSTATHIS, April ; POTTENGER, William M. ; DAVISON, Brian D.: *Identification of Critical Values in Latent Semantic Indexing*. citeseer.ist.psu.edu/article/kontostathis05identification.html. Version: 2005

[Kraaij und Pohlmann 1996]
KRAAIJ, Wessel ; POHLMANN, Renée: Viewing Stemming as Recall Enhancement. In: *Proc. of SIGIR '96*, 40–48

[Kramer 2005]
KRAMER, André: Such, Programm! - Elf kostenlose Desktop-Tools schnüffeln um die Wette. In: *c'T - Magazin für Computertechnik* (2005), jun, Nr. 13, S. 170–177

[Kuhlen 1977]
KUHLEN, Rainer: *Deutsche Gesellschaft für Dokumentation: DGD-Schriftenreihe*. Bd. 5: *Experimentelle Morphologie in der Informationswissenschaft*. 1. München : Verlag Dokumentation, 1977. – ISBN 3–7940–3624–7

[Landauer et al. 1997]
LANDAUER, T. ; LAHAM, D. ; REHDER, B. ; SCHREINER, M.: *How well can passage meaning be derived without using word order*. citeseer.ist.psu.edu/landauer97how.html. Version: 1997

[Landauer 2002a]
LANDAUER, Thomas K.: *Applications of Latent Semantic Analysis*. citeseer.ist.psu.edu/547704.html. Version: aug 2002

[Landauer 2002b]
LANDAUER, Thomas K.: *On the computational basis of learning and cognition: Ar-*

guments from LSA. http://lsa.colorado.edu/papers/Ross-final-submit.pdf. Version: 2002

[Landauer und Dumais 1997]
LANDAUER, Thomas K. ; DUMAIS, Susan T.: Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction and Representation of Knowledge. In: *Psychological Review* 104 (1997), Nr. 2, S. 211–240

[Landauer et al. 1998]
LANDAUER, Thomas K. ; FOLTZ, Peter W. ; LAHAM, Darrell: Introduction to Latent Semantic Analysis. In: *Discourse Processes* 25 (1998), S. 259–284

[Lawrence et al. 1999]
LAWRENCE, Steve ; GILES, C. L. ; BOLLACKER, Kurt D.: Autonomous citation matching. In: *AGENTS '99: Proceedings of the third annual conference on Autonomous Agents*. New York, NY, USA : ACM Press, 1999. – ISBN 1–58113–066–X, S. 392–393

[Lemaire und Dessus 2005]
LEMAIRE, Benoît ; DESSUS, Philippe: *Readings in Latent Semantic Analysis for Cognitive Science and Education*. http://www-leibniz.imag.fr/~blemaire/lsa.html. Version: 2005

[Letsche 1996]
LETSCHE, T.: *Toward large-scale information retrieval using latent semantic indexing*, Diplomarbeit, aug 1996. citeseer.ist.psu.edu/letsche96toward.html. – Elektronische Ressource

[Lo et al. 2005]
LO, Rachel Tsz-Wai ; HE, Ben ; OUNIS, Iadh: Automatically Building a Stopword List for an Information Retrieval System. In: *Proceedings of the 5th Dutch-Belgian Information Retrieval Workshop (DIR'05)*. De Uithof, Utrecht University, Utrecht, the Netherlands, 2005

[Manning und Schütze 1999]
MANNING, Christopher D. ; SCHÜTZE, Hinrich: *Foundations of Statistical Natural Language Processing*. 1. Cambridge, Massachusetts, USA : MIT Press, 1999. – 620 S. – ISBN 0–262–13360–1

[Nakov 2000a]
NAKOV, Preslav: Getting Better Results with Latent Semantic Indexing. In: *Proceedings of the Students Presentations at ESSLLI-2000*. Birmingham, UK, August 2000, 155-156

[Nakov 2000b]
NAKOV, Preslav: Web Personalization Using Extended Boolean Operations with Latent Semantic Indexing. In: *Artificial Intelligence: Methodology, Systems, Applications*, 189-198

[Nakov 2001]
NAKOV, Preslav: Latent Semantic Analysis for German Literature Investigation. In: *Proceedings of the International Conference, 7th Fuzzy Days on Computational Intelligence, Theory and Applications*. London, UK : Springer-Verlag, 2001. – ISBN 3–540–42732–5, S. 834–841

[Netousek 1997]
NETOUSEK, Thomas: *Verbesserung des Retrievalergebnisses durch Aufbau unscharfer Thesauri und Wortstammbildungen*. Wien : Dissertation Wirtschaftsuniversität Wien, 1997

[Noreault et al. 1981]
NOREAULT, Terry ; MCGILL, Michael ; KOLL, Matthew B.: A performance evaluation of similarity measures, document term weighting schemes and representations in a Boolean environment. In: *SIGIR '80: Proceedings of the 3rd annual ACM conference on Research and development in information retrieval*. Kent, UK : Butterworth & Co., 1981. – ISBN 0–408–10775–8, S. 57–76

[Oksa et al. 2002]
OKSA, Gabriel ; BECKA, Martin ; VAJTERSIC, Marian: Parallel SVD computation in updating problems of latent semantic indexing. In: *ALGORITMY 2002 - 16th Conference on Scientific Computing*. Bratislava : Technical University Press, sep 2002. – ISBN 80–227–1750–9, S. 113–120

[Park und Ewerbring 1991]
PARK, Haesun ; EWERBRING, L. M.: An algorithm for the generalized singular value decomposition on massively parallel computers. In: *ICS '91: Proceedings of the 5th*

international conference on Supercomputing. New York, NY, USA : ACM Press, 1991. – ISBN 0–89791–434–1, S. 136–145

[Pearson Knowledge Technologies 2005]
PEARSON KNOWLEDGE TECHNOLOGIES: *Summary Street - Sample Screens*. http://www.k-a-t.com/cu_sample.shtml. Version: 2005

[Porter 1980]
PORTER, Martin F.: An algorithm for suffix stripping. In: *Program* Bd. 14, 1980, S. 130–137

[Porter 2001]
PORTER, Martin F.: *Snowball-Stemmer*. http://snowball.tartarus.org/. Version: oct 2001

[van Rijsbergen 1979]
RIJSBERGEN, C. J.: *Information retrieval*. 2. Butterworths http://www.dcs.glasgow.ac.uk/Keith/Preface.html. – ISBN 0–408–70929–4

[Robertson und Spärck Jones 1976]
ROBERTSON, S. E. ; SPÄRCK JONES, Karen: Relevance weighting of search terms. In: *Journal of the American Society for Information Science* 27 (1976), Nr. 10, S. 129–146

[Salton 1965]
SALTON, Gerard: Automatic Phrase Matching. In: *Internatinal Conference on Computational Linguistics*, 1965

[Salton 1971]
SALTON, Gerard: *The SMART Retrieval System - Experiments in Automatic Document Processing*. Englewood Cliffs, NJ : Prentice Hall, 1971. – 556 S

[Sheridan und Smeaton 1992]
SHERIDAN, Paraic ; SMEATON, Alan F.: The application of morpho-syntactic language processing to effective phrase matching. In: *Inf. Process. Manage.* 28 (1992), Nr. 3, S. 349–369. http://dx.doi.org/http://dx.doi.org/10.1016/0306-4573(92)90080-J. – DOI http://dx.doi.org/10.1016/0306–4573(92)90080–J. – ISSN 0306–4573

[Spärck Jones 1972]
SPÄRCK JONES, Karen: A Statistical Interpretation of Term Specificity and its Application in Retrieval. In: *Journal of Documentation* 28 (1972), S. 11–21

[Spärck Jones 1997]
SPÄRCK JONES, Karen: Search term relevance weighting given little relevance information. (1997), S. 329–338. ISBN 1–55860–454–5

[Stock 2000]
STOCK, Wolfgang G.: *Informationswirtschaft: Management externen Wissens*. München - Wien : Oldenbourg Wissenschaftsverlag GmbH, 2000. – ISBN 3–486–24897–9

[Strang 2003]
STRANG, Gilbert: *Lineare Algebra*. Berlin : Springer, 2003. – 656 S. – ISBN 3–540–43949–8

[Streeter und Lochbaum 1988]
STREETER, L. A. ; LOCHBAUM, K. E.: An expert/expert-locating system based on automatic representationof semantic structure. In: *Artificial Intelligence for Applications, 1988., Proceedings of the Fourth Conference on AI Applications*, 1988. – ISBN 0–8186–0837–4, S. 345–350

[Tang et al. 2004]
TANG, C. ; DWARKADAS, S. ; XU, Z.: *On scaling latent semantic indexing for large peer-to-peer systems*. citeseer.ist.psu.edu/tang04scaling.html. Version: 2004

[Telcordia Technologies 2005]
TELCORDIA TECHNOLOGIES: *References to Papers on LSI*. http://lsi.research.telcordia.com/lsi/LSIpapers.html. Version: 2005

[Wiemer-Hastings 1999]
WIEMER-HASTINGS, Peter: How Latent is Latent Semantic Analysis? In: *Proceedings of the Sixteenth International Joint Congress on Artificial Intelligence*. San Francisco : Morgan Kaufmann, 1999, S. 932–937

[Wikipedia 2005a]
WIKIPEDIA: *Eigenwertproblem*. http://de.wikipedia.org/wiki/Eigenwertproblem. Version: jul 2005

[Wikipedia 2005b]
WIKIPEDIA: *Normierter Raum*. http://de.wikipedia.org/wiki/Normierter_Raum. Version: jun 2005

[William M. Shaw et al. 1997]
WILLIAM M. SHAW, Jr. ; BURGIN, Robert ; HOWELL, Patrick: Performance standards and evaluations in IR test collections: vector-space and other retrieval models. In: *Inf. Process. Manage.* 33 (1997), Nr. 1, S. 15–36. http://dx.doi.org/10.1016/S0306-4573(96)00044-1. – DOI 10.1016/S0306–4573(96)00044–1. – ISSN 0306–4573

[Wu und Salton 1981]
WU, Harry ; SALTON, Gerard: A comparison of search term weighting: term relevance vs. inverse document frequency. In: *SIGIR '81: Proceedings of the 4th annual international ACM SIGIR conference on Information storage and retrieval.* New York, NY, USA : ACM Press, 1981. – ISBN 0–89791–052–4, S. 30–39

[Xu und Croft 1998]
XU, Jinxi ; CROFT, W. B.: Corpus-Based Stemming using Cooccurrence of Word Variants. In: *ACM Transactions on Information Systems* 16 (1998), Nr. 1, 61–81. citeseer.ist.psu.edu/article/xu98corpusbased.html

[Xu und Croft 2000]
XU, Jinxi ; CROFT, W. B.: Improving the effectiveness of information retrieval with local context analysis. In: *ACM Transactions on Information Systems* 18 (2000), Nr. 1, 79–112. citeseer.ist.psu.edu/xu00improving.html

[Yu et al. 2002]
YU, Clara ; CUADRADO, John ; CEGLOWSKI, Maciej ; PAYNE, J. S.: *Patterns in Unstructured Data - Discovery, Aggregation, and Visualization.* http://javelina.cet.middlebury.edu/lsa/out/lsa_intro.htm. Version: 2002

[Zha und Simon 1999]
ZHA, Hongyuan ; SIMON, Horst D.: On Updating Problems in Latent Semantic Indexing. In: *SIAM J. Sci. Comput.* 21 (1999), Nr. 2, S. 782–791. http://dx.doi.org/http://dx.doi.org/10.1137/S1064827597329266. – DOI http://dx.doi.org/10.1137/S1064827597329266. – ISSN 1064–8275

A Stopwortlisten

Manuell erstellte Liste (gestemmt)

all, als, am, an, auch, auf, aus, bei, beim, bis, das, dem, den, der, des, die, durch, ein, er, erst, es, fur, geg, gibt, hab, hat, ich, ihr, im, in, ins, ist, kann, man, mehr, mit, nach, neu, noch, nur, oder, ohn, sein, seit, sich, sie, sind, so, statt, uber, um, und, uns, unt, vom, von, vor, war, was, wenn, wer, werd, wie, wird, zu, zum, zur, zwisch

Vorhande Liste (gestemmt) - manuell angepaßt

aber, all, allein, als, also, and, anstatt, auch, auf, aus, auss, ausserhalb, behalt, bei, beid, beim, beinah, bevor, bin, bis, bist, bitt, dah, damit, danach, dank, dann, darub, daruberhinaus, darum, das, dass, dein, dem, den, denn, der, des, deshalb, die, dies, dir, doch, dort, durch, durft, eig, eigent, ein, einfach, einig, entgeg, entwed, erschein, etwas, fast, fertig, fort, fur, ganz, geg, gegenub, gehalt, geht, gemacht, gemass, genau, genug, gerad, getan, getrennt, gewes, gibt, grundlich, gut, hab, habt, hallo, hast, hat, hatt, haufig, hier, hint, ich, ihm, ihn, ihr, imm, ind, inn, innerhalb, irgendwelch, irgendwo, iss, ist, jed, jedoch, jemal, jemand, jen, jetzt, jung, kann, kaum, kein, kommt, konn, konnt, konntet, mach, macht, mal, man, mehr, mein, meist, mich, mir, mit, muss, musst, nach, nachd, nebenan, nein, nen, net, nicht, niemand, nirgendwo, nix, noch, nun, nur, oben, obwohl, oder, oft, ohn, pro, quot, roll, sagt, schein, schon, sehr, sei, seid, seien, sein, seit, selb, sich, sie, sind, sogar, solch, soll, sollt, sond, statt, stet, tatsach, tief, tun, tut, uber, und, uns, unt, unterhalb, usw, viel, vielleicht, von, vor, vorbei, vorh, wahrend, wann, war, wart, was, wed, weg, weil, weit, welch, wem, wen, wenig, wenn, wer, werd, werdet, wess, wie, wied, will, wir, wird, wirklich, wirst, wohin, wohl, wurd, ziemlich, zum, zur, zusamm, zwar, zwisch

B GIRT4-Beispiel-Dokument

<DOC>
<DOCNO>GIRT-DE20008977</DOCNO>
<DOCID>GIRT-DE20008977</DOCID>
<TITLE-DE>Ganzheitliche Humanpsychologie und ganzheitliche Human-
medizin oder: die Psyche als Heilkraft</TITLE-DE>
<AUTHOR>Schleifer, Horst</AUTHOR>
<PUBLICATION-YEAR>2000</PUBLICATION-YEAR>
<LANGUAGE-CODE>DE</LANGUAGE-CODE>
<CONTROLLED-TERM-DE>Psyche</CONTROLLED-TERM-DE>
<CONTROLLED-TERM-DE>Heilung</CONTROLLED-TERM-DE>
<CONTROLLED-TERM-DE>Schulmedizin</CONTROLLED-TERM-DE>
<CONTROLLED-TERM-DE>kognitive Faktoren</CONTROLLED-TERM-DE>
<CONTROLLED-TERM-DE>Alternative</CONTROLLED-TERM-DE>
<CONTROLLED-TERM-DE>psychosomatische Faktoren</CONTROLLED-TERM-DE>
<CONTROLLED-TERM-DE>Mensch</CONTROLLED-TERM-DE>
<CONTROLLED-TERM-DE>psychosoziale Faktoren</CONTROLLED-TERM-DE>
<CONTROLLED-TERM-DE>Therapie</CONTROLLED-TERM-DE>
<CONTROLLED-TERM-DE>Krankheit</CONTROLLED-TERM-DE>
<CONTROLLED-TERM-DE>Psychotherapie</CONTROLLED-TERM-DE>
<CONTROLLED-TERM-DE>Denken</CONTROLLED-TERM-DE>
<CONTROLLED-TERM-DE>ganzheitlicher Ansatz</CONTROLLED-TERM-DE>
<METHOD-TERM-DE>empirisch</METHOD-TERM-DE>
<METHOD-TERM-DE>Theoriebildung</METHOD-TERM-DE>
<METHOD-TERM-DE>Grundlagenforschung</METHOD-TERM-DE>
<METHOD-TERM-DE>Befragung</METHOD-TERM-DE>
<CLASSIFICATION-TEXT-DE>Sozialmedizin, Medizin</CLASSIFICATION-TEXT-DE>
<CLASSIFICATION-TEXT-DE>Medizinsoziologie</CLASSIFICATION-TEXT-DE>
<ABSTRACT-DE>Ebenso wie in der akademischen Psychologie das

```
'Geistig-seelische' aus dem Forschungsgegenstand weitgehend
ausgeblendet bleibt (sic!), so führt auch in der natur-
wissenschaftlich begründeten sog. Schulmedizin die Einbeziehung
erfahrungswissenschaftlicher Erkenntnisse, auch sog. Alternativer
Heilverfahren bis in die Gegenwart ein Schattendasein. Mit der
Forderung, den ganzen Menschen in den Blickpunkt der Heilkunst zu
rücken, wird der Grundgedanke des Projekts strukturiert: Nach einer
Charakterisierung der jeweiligen Vorgehensweisen bei der Therapie
erkrankter Menschen - auf der Basis unterschiedlicher Heilmethoden
in der psychologischen bzw. medizinischen Therapie - sollen die
'essentials' des Heilungsprozesses herausgearbeitet werden. Dies
geschieht primär durch die Bilanzierung einschlägiger Untersuchungs-
ergebnisse aus Psychotherapie und Humanmedizin, deren Aktualisierung
über selbst geführte Interviews mit renommierten Vertretern dieser
Heilverfahren 'gesichert' werden soll. Letztlich geht es um die
Vermittlung des Grundkonzeptes systemtheoretischen Denkens, d.h. um
den Versuch, somatische, kognitive und psychosoziale Prozesse auch
für die psychologische und medizinische Therapie unter besonderer
Berücksichtigung der 'Psyche als Heilkraft' herauszuarbeiten.
</ABSTRACT-DE>
</DOC>
```

Die kompletten GIRT4-Daten befinden sich im Verzeichnis „GIRT-SGML-Dateien“ auf der beigefügten CD-ROM.

C GIRT-Retrieval-Test: Anfragen

Query76: Politische Kultur in einer Demokratie
Query77: Politische Partizipation von DDR-Frauen
Query78: Bildung in der Türkei
Query79: Volksentscheide in der Demokratie
Query80: Industrielle und ökonomische Entwicklung Indiens
Query81: Ausbildungsabbruch
Query82: Berufliche Bildung von Immigranten
Query83: Medien und Krieg
Query84: Neue Medien im Unterricht
Query85: Europäische Bildungssysteme im Vergleich
Query86: Politische Symbole in Osteuropa
Query87: Integration Behinderter in Schulen
Query88: Sport im Nationalsozialismus
Query89: Berufsethik im Journalismus
Query90: Zeitunglesen
Query91: Qualität von Bildungssystemen
Query92: Parteimitglieder
Query93: Burnout-Syndrom
Query94: Homosexualität und Coming-out
Query95: Sonntagsarbeit
Query96: Kosten der Berufsausbildung
Query97: Sekten
Query98: Außenpolitik Kanadas
Query99: Sport und Jugendarbeit
Query100: Kneipenkultur

Weitere Details zu den Anfragen sind in der Datei „clef-2003-girt_de.txt“ auf der beigefügten CD-ROM zu finden.