Bauhaus-Universität Weimar
Fakultät Medien
Studiengang Mediensysteme

Versteckte Variablen-Modelle für spezielle Retrieval-Aufgaben

Bachelorarbeit

Christof Bräutigam
geb. am: 22.05.1979 in Rudolstadt

Matrikelnummer 40008

1. Gutachter: Prof. Dr. Benno Stein
2. Gutachter: Dr. Sven Meyer zu Eißen

Datum der Abgabe: 8. Mai 2008

Erklärung der Selbstständigkeit

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen worden ist. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Weimar, 8. Mai 2008 Christof Bräutigam

Inhaltsverzeichnis

1 Einleitung

Die Menge an Informationen, die uns durch aktuelle Technologien zur Verfügung stehen, ist immens und nimmt immer mehr zu. Der Begriff *Informationen* ist in diesem Zusammenhang jedoch missverständlich, denn im Grunde handelt es sich zum großen Teil um unstrukturierte und unkategorisierte Daten. Als Informationen im engeren Sinne bezeichnet man nur Daten, die in einem bestimmten Kontext relevant sind. Auf die Anfrage bei einer typischen Internet-Suchmaschine bekommt man beispielsweise oft sehr viele Seiten, doch welche der gelieferten Dokumente enthalten tatsächlich relevante Daten? Nur diese relevanten Daten erfüllen das Informationsbedürfnis des Anfragenden.

Die Suche nach relevanten Dokumenten in einer Dokumentkollektion *(Ranking)* und die unüberwachte Klassifikation einer Dokumentkollektion *(Clustering)* sind klassische Themen des Information Retrieval. Die für diese Aufgaben eingesetzten Systeme unterscheiden sich hinsichtlich des zugrundeliegenden Retrieval-Modells und der daraus resultierenden Anwendungsgebiete. Die vorliegende Arbeit betrachtet Modelle, die mit versteckten Variablen (engl: *hidden variables*) arbeiten und erläutert deren Funktionsweise, Vor- und Nachteile, Einsatzgebiete und Grenzen.

1.1 Gliederung

In diesem Kapitel werden Problemstellungen und die Motivation vorgestellt. Das folgende Kapitel erläutert Grundlagen, die für das Verständnis des Themas notwendig sind und zeigt einen klassischen Lösungsansatz für die genannten Probleme, der als Basis die weiterführenden Methoden Anwendung findet. Kapitel 3 beschreibt Lösungen mit versteckten Variablen und erklärt die Systeme LSI und PLSI genauer. In Kapitel 4 wird ein im Rahmen dieser Arbeit entwickeltes Softwaremodul vorgestellt, welches die im vorigen Kapitel genannten Systeme implementiert und mit dem die folgenden Experimente durchgeführt wurden. Kapitel 5 beschreibt diese Experimente zur Evaluierung der Systeme LSI und PLSI. Kapitel 6 fasst die Ergebnisse zusammen und gibt einen Ausblick auf weitere Entwicklungen auf diesem Themengebiet.

1.2 Motivation

Um in umfangreichen Dokumentsammlungen eine gezielte Suche nach Informationen zu ermöglichen, ist eine aufwendige Pflege der Daten notwendig. Klassischerweise wird zur Verwaltung ein Index oder Register mit Metainformationen angelegt, beispielsweise Karteikarten in Bibliotheken. Dieser Ansatz wurde auch auf die elektronische Datenverwaltung übertragen, birgt aber einige systemimmanente Probleme. So müssen Metadaten vor der Nutzung spezifiziert werden, sind dann oft schwer erweiterbar und ihre Verwaltung und Synchronisierung mit dem bezeichneten Datenbestand erfordert zusätzlichen Aufwand. Sie begrenzen auch die Möglichkeiten, Anfragen zu formulieren: Wenn beispielsweise das Erscheinungsjahr eines Dokuments nicht notiert ist, kann man nicht danach suchen. Ebenso ist eine in natürlicher Sprache formulierte Anfrage nur über die Vermittlung eines Menschen erfüllbar. Ein Vergleich aufgrund der tatsächlichen Daten (also des Textes der Dokumente) ist selten möglich, so kann man beispielsweise auf eine Anfrage mit einem bestimmten Text keine Arbeiten finden, die ähnliche Themen behandeln. Die Kategorisierung ist aufwendig, nicht standardisiert und basiert zumeist auf althergebrachten Ontologien, die möglicherweise nicht mehr zeitgemäß sind. Die gesamte Verwaltung muss dauerhaft von Menschen überwacht werden.

Neuere Entwicklungen versuchen den Metadaten-Ansatz mit ausdrucksstarken Auszeichnungsmechanismen und modernen Verfahren des maschinellen Lernens zu verbessern, Stichwort *Semantic Web*. Voraussetzung bleibt jedoch, dass diese Metadaten vorhanden sind bzw. neu erstellte Daten mit reichhaltigen Metainformationen versehen werden. Obwohl inzwischen sehr intelligente Ideen entwickelt wurden, dies dem Autor der Daten soweit wie möglich zu erleichtern (z.B.*tagging*) oder gar ihn selbst davon zu befreien und dies allen zu überlassen (*social tagging*), entstehen wahrscheinlich immer noch die meisten Daten ohne Metainformation.

Man kann wohl behaupten, dass die meisten Daten heute in unstrukturierter und unkategorisierter Form ohne Metainformation vorliegen. Für solche Daten werden, seit Computer in der Datenverwaltung Einsatz finden, automatische Analyseverfahren entwickelt. Aufbauend darauf lassen sich Such- und Kategorisierungsalgorithmen entwickeln, die viele der o.g. Aufgaben erledigen können. Eine typische Web-Suchmaschine kann als simples aber eingängiges Beispiel herhalten: Auf eine Anfrage, formuliert in wenigen Stichworten, erhält man viele Ergebnisdokumente. Die wenigsten dieser Ergebnisse sind von vornherein mit Stichworten versehen, sie werden aufgrund eines Vergleichs der Suchworte mit dem Inhalt aus einer großen Kollektion (dem Web) ausgewählt. Dabei speichert die Suchmaschine nicht nötigerweise alle Dokumente komplett um sie bei einer Anfrage zu durchsuchen, vielmehr wurde ein Index erstellt, der mit der Suchanfrage verglichen wird und bei einer hohen Ähnlichkeit einen Verweis auf das indexierte Dokument liefert. Dieser Index wird bei nahezu allen heutigen Suchmaschinen automatisch

aus den Inhalten der Dokumente generiert. Die Qualität der Ergebnisse zu verbessern ist ein zentrales Anliegen vieler Forschungsbemühungen in diesem Bereich.

Die Analyse natürlichsprachlicher Dokumente ist ein Forschungsgebiet, das Elemente aus Linguistik und Informatik vereint. Die entwickelten Verfahren finden Anwendung im Maschinellen Lernen und im Information Retrieval. Diese Arbeit beschäftigt sich mit Methoden zur semantischen Analyse von Dokumenten und darauf aufbauender Indexierung.

1.3 Suche in einer Dokumentkollektion

Die Suche in einer Dokumentkollektion kann als Kernproblem des Information Retrieval (IR) aufgefasst werden. Ein Nutzer mit einem Informationsbedürfnis stellt eine Anfrage an ein IR-System, daraufhin wertet dieses die Anfrage aus und liefert Ergebnisse. Sind die Ergebnisse zusätzlich hinsichtlich ihrer Relevanz zur Anfrage geordnet, bezeichnet man dies als *Ranking*. Anfrage und zu durchsuchende Dokumente müssen dafür in eine Form gebracht werden, die einen Vergleich ermöglicht. Die Einheit aus formalisierter Anfragemenge, formalisierter Dokumentmenge und Vergleichsfunktion bezeichnet man im IR als Dokumentmodell (s. Abschnitt 2.1).

1.4 Automatische Kategorisierung

Die Suche in Dokumentsammlungen ist vielleicht die anschaulichste, aber längst nicht die einzige Anwendungsmöglichkeit für IR-Systeme. Ein weiteres wichtiges Gebiet ist die automatische Kategorisierung, das *Clustering*, deren Ziel es ist, eine Dokumentkollektion in verschiedene Kategorien aufzuteilen, ohne dass jedoch ein festes Klassifikationsschema vorgegeben wäre. Ein Clusteringsystem soll die einer Kollektion innewohnende Struktur selbst erkennen und die Aufteilung vornehmen, man bezeichnet das Clustering daher auch als *unüberwacht*. Um diese Aufgabe zu erfüllen, muss ein solches System in der Lage sein, die Ähnlichkeit zwischen Dokumenten zu berechnen. Auf dieser Basis können dann Gruppen von untereinander ähnlichen Dokumenten gebildet werden.

Die Ähnlichkeit kann dabei durchaus unterschiedlich definiert sein, beispielsweise könnte man Dokumente thematisch oder nach Genre gruppieren. Das Dokumentmodell muss in diesen Fällen die nötigen Informationen bieten. Oft lassen sich gute Ergebnisse auch nur durch die Kombination verschiedener Dokumentmodelle erreichen.

2 Grundlagen

Dieses Kapitel führt wichtige Begriffe ein und stellt das Vektorraummodell als klassisches Retrievalmodell vor.

2.1 Dokumentmodell

Um Dokumentmodell und Retrievalsysteme einzuführen, muss zunächst zwischen realen Dokumenten und der Repräsentation zur maschinellen Verarbeitung unterschieden werden.

2.1.1 Begriffsdefinition

Reales Dokument Im Rahmen dieser Arbeit werden ausschließlich Textdokumente betrachtet. Ein Dokument d besteht aus einer Menge von Worten, Satzzeichen, Sonderzeichen und Leerzeichen. Der Text der vorliegenden Arbeit bildet beispielsweise ein solches reales Dokument. Eine Dokumentkollektion wird mit D bezeichnet.

Dokumentrepräsentation Zur maschinellen Verarbeitung von Dokumenten müssen diese in einer adäquaten Form repräsentiert werden, hier dargestellt durch $\mathbf{d}$. Die Dokumentrepräsentation wird auch als *formales Dokument* bezeichnet. Eine Kollektion formaler Dokumente ist entsprechend dargestellt mit $\mathbf{D}$

Anfrage Eine Anfrage q (engl: *query*) besteht analog zum realen Dokument aus Worten. Dies kann eine in Stichworten formulierte Suchanfrage oder ein ganzes Dokument sein. Die Menge aller Anfragen sei Q. Eine *formale Anfrage* $\mathbf{q}$ ist, analog zur Dokumentrepräsentation, die Repräsentation der Anfrage zur maschinellen Verarbeitung.

Das Dokumentmodell ist entscheident für die Qualität eines IR-Systems. Die formale Definition eines Dokumentmodells laut [Ste07] ist:

Definition 1 (Dokumentmodell)
Sei D eine Menge von Dokumenten und Q eine Menge von Anfragen. Ein Dokument-Modell für D,Q ist ein Tupel $\langle \mathbf{D}, \mathbf{Q}, \rho_{\mathcal{R}} \rangle$, dessen Elemente wie folgt definiert sind:

1. $\mathbf{D}$ ist die Menge der Repräsentationen der Dokumente $d \in D$. In $\mathbf{d} \in \mathbf{D}$ können Layout-, logische und semantische Sicht codiert sein.
2. $\mathbf{Q}$ ist die Menge der formalisierten Anfragen.
3. $\mathcal{R}$ ist ein Retrieval-Modell und formalisiert ein Prinzip, ein Paradigma oder eine linguistische Theorie.
   Auf der Grundlage von $\mathcal{R}$ ist die Retrieval-Funktion $\rho_{\mathcal{R}}(\mathbf{q}, \mathbf{d})$ definiert. Sie quantifiziert die Systemrelevanz zwischen einer formalisierten Anfrage $\mathbf{q} \in \mathbf{Q}$ und einer Dokumentrepräsentation $\mathbf{d} \in \mathbf{D}$:
   $\rho_{\mathcal{R}} : \mathbf{Q} \times \mathbf{D} \rightarrow \mathbf{R}$
   Die von $\rho_{\mathcal{R}}$ berechneten Werte heißen Retrieval-Werte (Retrieval Status Value, RSV).

Man versteht unter dem Begriff *Dokumentmodell* also nicht nur eine Dokumentrepräsentation, wie man dies umgangssprachlich annehmen könnte, sondern die Gesamtheit aus Dokument- und Anfragerepräsentation sowie einer Funktion, die einen quantifizierbaren Zusammenhang zwischen beiden liefert.

2.1.2 Anforderungen an ein Dokumentmodell

Formale Anforderungen: Der Inhalt der realen Dokumente d soll von den Repräsentationen $\mathbf{d}$ möglichst gut, also mit wenig Informationsverlust, abgebildet werden. Der Vergleich zwischen Dokumenten, hinsichtlich einer Ähnlichkeits- oder Diskriminierungsfunktion, muss möglich sein. Dokumentmodelle können auch geschaffen werden um ganz bestimmte Informationen zu repräsentieren, wenn dies im Retrievalsystem gefragt ist.

Praktische Anforderungen: Man kann ein Modell hinsichtlich seiner Effektivität und Effizienz betrachten. Das Modell muss so beschaffen sein, dass ein darauf basierendes Retrievalsystem seine Aufgabe effektiv durchführen kann, d.h. es muss überhaupt in der Lage sein, das gewünschte Ergebnis zu erzielen. Weiterhin sollte ein Retrievalsystem nach Möglichkeit effizient arbeiten, also die Ressourcen, meist Zeit und Speicherplatz, effizient nutzen. Je nach Einsatzzweck kann die Effizienz unterschiedlich definiert sein. Bei der Suche in einer Kollektion gilt es beispielsweise, in einer kurzen Zeit so viele relevante und so wenig irrelevante Dokumente wie möglich zu liefern. Dafür muss auch das zu Grunde liegende Dokumentmodell effizient maschinell verarbeitbar sein.

2.2 Vektorraummodell

Ein klassisches Dokumentmodell ist das Vektorraummodell [Sal75]. Die im folgenden betrachteten Modelle basieren auf Ideen, die mit dem Vektorraummodell entwickelt wurden, daher ist das Verständnis dieses Modells eine wichtige Grundlage.

2.2.1 Dokumentrepräsentation und Retrievalfunktion

Das Vektorraummodell ist ein termbasiertes Dokumentmodell, d.h. ein Dokument $d \in D$ wird auf eine Menge von Indextermen abgebildet.

Indexterm Als Indexterm t wird hier ein Element eines formalen Dokuments bezeichnet. Indexterme werden in einem Prozess der als **Indexierung** bezeichnet wird aus den Worten der Dokumente einer Kollektion berechnet.

Die Komponenten eines Dokumentmodells laut Definition 1 sind im Vektorraummodell:

Eine formales Dokument $\mathbf{d}$ ist dargestellt als Vektor v der Dimension n, wobei jede Dimension i mit $0 \leq i < n$ einen Indexterm t_i und der Wert v_i dessen Gewicht (s. Abschnitt 2.2.2) repräsentiert. Der Vektor wird auch als *Dokumentvektor* bezeichnet.

Eine Anfrage $\mathbf{q}$ wird analog zur Dokumentrepräsentation ebenfalls als Vektor dargestellt.

Als Retrieval-Funktion $\rho_{\mathcal{R}}(\mathbf{q}, \mathbf{d})$ wird die Cosinusähnlichkeit verwendet, d.h. der Cosinus des Winkels φ zwischen zwei Dokumentvektoren. Der Cosinus ist berechenbar über die Definition des Skalarproduktes zwischen zwei Vektoren $\mathbf{v}$ und $\mathbf{w}$ mit dem eingeschlossenen Winkel φ.

$$\mathbf{v}^T\mathbf{w} = \|\mathbf{v}\| \cdot \|\mathbf{w}\| \cdot \cos(\varphi) \tag{2.1}$$

$$\cos(\varphi) = \frac{\mathbf{v}^T\mathbf{w}}{\|\mathbf{v}\| \cdot \|\mathbf{w}\|} \tag{2.2}$$

Wenn $\mathbf{v}$ und $\mathbf{w}$ normalisiert sind, gilt $\|\mathbf{v}\| = \|\mathbf{w}\| = 1$ und somit:

$$\cos(\varphi) = \mathbf{v}^T\mathbf{w} = \sum_{i=1}^{n} \mathbf{v}_i \cdot \mathbf{w}_i \tag{2.3}$$

Die Cosinusähnlichkeit bietet gegenüber anderen möglichen Ähnlichkeitsfunktionen wie beispielsweise der euklidischen Distanz eine Unabhängigkeit von der Größe des Textes (und daraus resultierenden Länge des Vektors) und hat einen definierten Wertebereich zwischen 0 (sehr unähnlich) und 1 (sehr ähnlich).

Dabei wird die Annahme zu Grunde gelegt, dass die Vektorraum-Repräsentation eines Dokuments als Sammlung von Indextermen die Information des realen Dokuments gut genug abbildet. Ein gewisser Teil der Original-Information geht natürlich verloren, beispielsweise die Reihenfolge der Worte im Text. Die Repräsentation eines Dokuments als ungeordnete Sammlung von Termen enthält implizit die vereinfachende Annahme der bedingten Unabhängigkeit (engl: *conditional independence*) zwischen den Termen.

Die Abbildung 2.1 S. 8 zeigt schematisch die Repräsentation von Dokumenten als Dokumentvektor und die Evaluation von Ähnlichkeiten im Vektorraummodell.

2.2.2 Termgewichtung

Eine sehr simple Herangehensweise zum Aufbau von Termvektoren ist es, nur einzutragen, ob ein Term in einem Dokument vorhanden ist, oder nicht. Dies wird als *Boolsches Modell* bezeichnet. Solch ein Modell hat jedoch wenig Aussagekraft. Die Idee der Termgewichtung ist, für ein Dokument wichtige Terme hervorzuheben während unwichtige Terme weniger Beachtung finden. Die Gewichtung der Indexterme ist für den Informationsgehalt des Dokumentmodells von entscheidenter Bedeutung.

Es existieren verschiedene Methoden (s. [Fer03], [BY99]), an dieser Stelle werden die beiden Verfahren vorgestellt, die für die weiteren Betrachtungen relevant sind, Termfrequenz (tf) und Termfrequenz-Inversdokumentfrequenz ($tfidf$).

Termfrequenz

Mit der Termfrequenz wird der Ansatz verfolgt, dass ein Term, der in einem Dokument häufig vorkommt, für dieses Dokument wichtiger ist als ein seltener Term. Die Termfrequenz $tf(t_i, d_j)$ gibt an, wie oft Term t_i in Dokument d_j vorkommt, ein Term bekommt also für ein bestimmtes Dokument ein stärkeres Gewicht, je häufiger er ist.

Termfrequenz-Inversdokumentfrequenz

Mit der Inversdokumentfrequenz wird der Ansatz verfolgt, dass ein Term zur Unterscheidung von Dokumenten wichtiger ist, je seltener er ist. Die Dokumentfrequenz $df(t_i)$

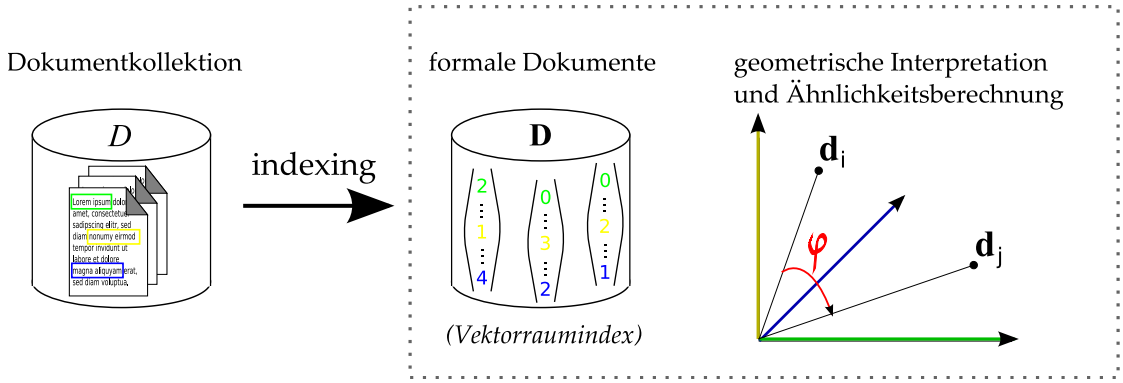

Abbildung 2.1: Schematische Darstellung der Repräsentation von Dokumenten und der Evaluierung der Dokumentähnlichkeit im Vektorraummodell.

gibt an, in wie vielen Dokumenten d Term t_j vorkommt. Dieser Wert bezieht sich also, im Gegensatz zur Termfrequenz, auf die gesamte Kollektion. Um das Diskriminierungspotenzial eines Terms herauszustellen, benutzt man die invertierte Dokumentfrequenz $idf(t_i)$, multipliziert sie zur Termfrequenz $tf(t_i)$ und erhält die Termfrequenz-Inversdokumentfrequenz $tfidf(t_i)$.

$$idf(t_i) = \ln\left(\frac{n+1}{df(t_i)+1}\right) \tag{2.4}$$

$$tfidf(t_i) = tf(t_i) \cdot idf(t_i) \tag{2.5}$$

Die Gewichtung mit $tfidf$ kombiniert die Ideen der Termfrequenz und der Inversdokumentfrequenz. Ein Term wird somit stärker gewichtet, wenn er nur in wenigen Dokumenten insgesamt vorkommt. Umgekehrt werden Terme abgewertet, die in vielen Dokumenten vorkommen. Die $tfidf$-Gewichtung ist allgemein ein stärkeres Maß als tf, da sie Wissen über die gesamte Kollektion mit einberechnet.

2.2.3 Stoppwortentfernung

Beim Indexieren einer Dokumentkollektion werden üblicherweise Worte ignoriert, die häufig oder statistisch gleich verteilt auftreten und deshalb für den Informationsgehalt eines Dokuments irrelevant sind. Die Stoppwortentfernung ist sprachspezifisch und wird meist anhand einer Stoppwortliste durchgeführt. In der deutschen Schriftsprache zählen zu den Stoppworten beispielsweise bestimmte und unbestimmte Artikel (der, die, das, ein, eine, ...).

2.2.4 Stammformreduktion

Beim Indexieren einer Dokumentkollektion kann man die Worte auf ihre Stamm- oder Grundform reduzieren (engl: *stemming*), somit werden mehrere Worte auf einen Term abgebildet. Damit lässt sich die Anzahl der generierten Indexterme reduzieren. Allerdings ist damit auch ein gewisser Informationsverlust verbunden. Je nach Retrieval-Anwendung muss man abwägen ob der Einsatz der Stammformreduktion sinnvoll ist.

2.2.5 Term-Dokument-Matrix

Die Dokumentvektoren werden zu einer Matrix der Dimension $m \times n$ (m Terme, n Dokumente) zusammengefasst. Diese Matrix wird entsprechend als Term-Dokument-Matrix (TDM) bezeichnet. Eine TDM repräsentiert also eine Dokumentkollektion. Die Darstellung als Matrix bietet die Möglichkeit, verschiedene mathematische Verfahren auf die Dokumentkollektion anzuwenden, u.a. basiert das in Abschnitt 3.2 gezeigte Modell auf einer Matrix-Faktorzerlegung. Matritzen stellen auch einen effizient implementierbaren und verarbeitbaren Datentyp für Computersysteme dar.

2.3 Evaluierung von Retrieval-Modellen

Um die Performanz eines Retrieval-Modells zu messen, gibt es verschiedene Methoden. In den Experimenten zu dieser Arbeit wird das *F-Measure* verwendet, ein Mittelwert aus den Gütemaßen *Precision* und *Recall* [Sal68]. Ein wichtiger Begriff auf den sich diese Größen beziehen, ist die *Relevanz*.

Definition 2 (Relevanz)
Die Relevanz eines Dokuments für eine Anfrage ist eine Relation $r : D \times \mathbf{Q} \rightarrow \{0,1\}$ wobei $D = \{d_1, d_2, \ldots, d_m\}$ die Menge der Dokumente und $\mathbf{Q}$ die Menge der formalisierten Anfragen bezeichnet ([Ste07]).

Relevanz ist also ein Maß für die Übereinstimmung zwischen einer formalisierten Anfrage und einem Dokument. In der dargestellten einfachen Form ist ein Dokument hinsichtlich einer Anfrage relevant (1) oder nicht (0).

2.3.1 Precision

Betrachtet wird die Menge der auf eine Anfrage gelieferten Dokumente. Sei die Teilmenge der relevanten Dokumente a und die Teilmenge der nicht relevanten Dokumente b. Die

Precision berechnet sich dann aus

$$precision = \frac{a}{a+b} \tag{2.6}$$

Precision stellt ein Maß für die Genauigkeit der Antwortmenge dar, je geringer der Anteil der irrelevanten Dokumente desto höher die Precision.

2.3.2 Recall

Betrachtet wird die Menge aller Dokumente. Sei die Teilmenge der relevanten Dokumente in der Antwortmenge a und die Teilmenge der relevanten Dokumente, die nicht in der Antwort enthalten sind c. Der Recall berechnet sich dann aus

$$recall = \frac{a}{a+c} \tag{2.7}$$

Recall stellt also ein Maß für die Vollständigkeit der Antwortmenge dar, je mehr der insgesamt relevanten Dokumente in der Antwortmenge enthalten sind, desto höher der Recall.

Der Recall ist teilweise schwierig zu bestimmen. Bei einer Suche im Web ist die Anzahl der relevanten Dokumente beispielsweise nicht bestimmbar. Bei der Gütemessung einer automatischen Kategorisierung wie in den Experimenten in Kapitel 5 ist die korrekte Kategorisierung jedoch bekannt und der Recall kann in die Berechnung aufgenommen werden.

2.3.3 F-Measure

Als F-Measure wird das gewichtete harmonische Mittel aus Precision und Recall bezeichnet:

$$F_\alpha = \frac{(1+\alpha) \cdot precision \cdot recall}{\alpha \cdot precision + recall} \tag{2.8}$$

Über den Parameter α lässt sich dabei eine Gewichtung zwischen Precision und Recall vornehmen, Werte zwischen 0 und 1 geben der Precision höheres Gewicht, Werte größer 1 bevorzugen den Recall. Beispielsweise gibt der Wert $\alpha = \frac{1}{3}$ der Precision ein dreifach höheres Gewicht gegenüber dem Recall.

Es existieren eine Reihe weitere Retrieval-Gütemaße. In den im Rahmen dieser Arbeit durchgeführten Experimenten (Kap. 5) wurde jedoch ausschließlich das F-Measure

angewendet, genauer gesagt F_1 (in den Reuters-Experimenten) und $F_{\frac{1}{3}}$ (im Spock-Experiment), daher wird auf andere Maße nicht eingegangen.

2.4 Kritik termbasierter Modelle

Ein Vorteil termbasierter Retrievalmodelle wie dem Vektorraummodell ist, dass sich die Terme einfach, direkt und automatisiert aus dem natürlichsprachlichen Text erzeugen lassen. Das Dokumentmodell liefert ausreichend Information um gute Retrievalergebnisse zu erzielen, verschiedene Termgewichtungsverfahren können die Performanz noch beträchtlich erhöhen. Die Berechnungen sind einfach und effizient implementierbar. Termbasierte Modelle finden aus diesen Gründen Verwendung in vielen Retrievalsystemen.

Termbasierte Modelle weisen jedoch auch einige Nachteile auf. Zunächst sind technische Nachteile zu nennen. Bei der Indexierung großer Dokumentkollektionen ($\geq$ 1.000 Dokumente) entstehen trotz Stopwortentfernung und Stemming sehr viele Indexterme. Deren Anzahl ist natürlich von der Beschaffenheit der Kollektion abhängig, handelt es sich etwa um Artikel eines Nachrichtendienstes oder aus wissenschaftlichen Publikationen ist das Vokabular weitaus präziser und kleiner als bei typischen Web-Dokumenten. Auch die Länge der Dokumente wirkt sich auf die Anzahl der Terme aus. Als Beispiel sei auf die Tabelle 5.2 S. 46 verwiesen.

Dennoch wird die Dimension des Vektorraumes schnell sehr groß, während die entstehenden TDM sehr dünn besetzt sind, d.h. zu einem großen Teil Nullwerte enthalten, weil jedes einzelne Dokument nur einen Bruchteil der insgesamt indexierten Terme enthält. Die hohe Dimension verringert die Geschwindigkeit bei der Ähnlichkeitsberechnung, weil sehr große Vektoren zu multiplizieren sind. Außerdem wird ungewolltes Rauschen in den Daten erzeugt, welches die Retrievalperformanz senkt. Die dünn besetzten Matritzen erfordern aufwendige Datenstrukturen zur Speicherung, denn auch die Nullwerte belegen Speicherplatz. Da typische TDM zum großen Teil Nullwerte enthalten, wird sehr viel Speicher verschwendet, wenn man diese Werte mit im Speicher hält.

Auch abseits der technischen Betrachtung finden sich Nachteile. Die natürliche Sprache, in der die meisten Dokumente formuliert sind, ist nicht sehr formell, dafür ist sie semantisch sehr reichhaltig. Die Bedeutung eines Textes erschliesst sich erst aus dem Zusammenwirken der lexikalischen Einheiten, ein Wort kann beispielsweise in verschiedenen Kontexten unterschiedliche Bedeutung haben. Die recht einfache Herangehensweise, Terme zu zählen, kann diese termübergreifenden Zusammenhänge nicht auflösen. Zwar haben die Terme etwas mit der Semantik (Bedeutung) des Inhalts zu tun, aber es ist nicht ausreichend, sie einzeln zu betrachten.

Bestimmte Phänomene natürlicher Sprachen, wie unterschiedliche Worte mit ähnlicher Bedeutung oder mehrdeutige Worte, beeinträchtigen die Ergebnisse der einfachen termbasierten Modelle.

Eine Herangehensweise zur Lösung der o.g. Problematik besteht darin, die Information, die im Zusammentreffen der Terme verborgen liegt, auszunutzen. Dafür wurden Modelle entwickelt, die darauf basieren, die TDM hinsichtlich der impliziten Semantik der Texte zu analysieren. Ziel ist es, die Dokumentrepräsentation vom Termraum in einen Konzeptraum zu überführen, der bei geringerer Dimension den Informationsgehalt des Termraumes behält und zusätzlich die semantische Vergleichbarkeit von Dokumenten ermöglicht.

Zwei dieser Modelle, *Latent Semantic Analysis* und *Probabilistic Latent Semantic Analysis* sind Gegenstand dieser Arbeit und werden im folgenden Kapitel genauer betrachtet.

3 Retrievalmodelle mit versteckten Variablen

Dieses Kapitel erläutert zwei Retrievalmodelle, die auf dem Vektorraummodell basieren und dabei die im Text verborgene semantische Information auszunutzen versuchen.

3.1 Idee semantischer Analyseverfahren

Termbasierte Retrievalmodelle haben systembedingte Grenzen bei der Auswertung von Suchanfragen und dem Vergleich von Dokumenten. Ein Nutzer, der eine Suchanfrage stellt, hat normalerweise eine bestimmte Vorstellung des Gesuchten, muss diese aber mit Worten formulieren. Einzelne Worte einer natürlichen Sprache geben jedoch nur unzuverlässige Hinweise auf das tatsächlich Gemeinte. Inhaltliche Mißverständnisse sind in der sprachlichen Kommunikation an der Tagesordnung. Die Semantik eines Textes ist nicht explizit aus den Worten erkennbar, sie offenbart sich dem Leser durch das kontextuelle Zusammenwirken lexikalischer Strukturen. Zur Interpretation benutzen menschliche Leser auch *common sense*. Letzteres steht einem Computer nicht zur Verfügung. Es besteht aber die Hoffnung, dass die semantische Information in den Terminterdependenzen mit geeigneten Analysemethoden verwendbar gemacht werden kann.

3.1.1 Problemstellungen bei natürlicher Sprache

Bestimmte Aspekte der natürlichen Sprache erschweren das Erkennen der Semantik. Besondere Aufmerksamkeit verdienen Synonyme und Homonyme.

Als Synonyme werden hier ganz allgemein verschiedene Begriffe mit gleicher Bedeutung bezeichnet. So sind beispielsweise die Begriffe *Bedeutung* und *Semantik* in diesem Text synonym. Synonyme senken den Recall von Termbasierten Retrievalsystemen. Wenn beispielsweise die Anfrage mit Worten formuliert wird, die in den gesuchten Dokumenten nicht vorkommen weil dort andere, bedeutungsgleiche Begriffe verwendet werden, werden diese Dokumente auch nicht gefunden.

Als Homonyme werden Worte bezeichnet, die unterschiedliche Bedeutungen haben. Bei reinem Term-Vergleich senken Homonyme die Precision eines Retrieval-Systems, da auch Dokumente geliefert werden, die zwar die Suchworte enthalten, diese aber nicht im kontextuellen Sinne des Suchenden stehen. Typische Beispiele für Homonyme sind *Jaguar*

(Bezeichnet ein Auto? Oder eine Raubkatze?) oder auch *LaTeX*, das Textsatzsystem mit dem diese Arbeit erstellt wurde.

Für beide Probleme gibt es keine zufriedenstellenden vollautomatischen Lösungen. Es existieren Ansätze, das Synonym-Problem mit Thesauri zu lösen und Homonyme mit eingeschränktem Vokabular und menschlicher Vermittlung im Indexing zu vermeiden, der Erfolg solcher Methoden ist allerdings beschränkt.

3.1.2 Lösungsansatz mit semantischer Analyse

Hier setzen die neuen Modelle an. Basierend auf der Annahme, dass die Terme eines Dokuments durchaus mit der Bedeutung des Textes in Zusammenhang stehen, wurden Analysemodelle entwickelt, die diese in den Terminterdependenzen "versteckte" Bedeutung extrahieren und verfügbar machen. Die extrahierten semantischen Entitäten werden als *Konzepte* oder *Aspekte* bezeichnet, je nach dem wissenschaftlichen Bereich aus welchem sie hervorgegangen sind. Die Bezeichnung *Konzept* wurde von Wissenschaftlern mit informationstechnischem Hintergrund geprägt [Dee90], wärend die Bezeichnung *Aspekt* aus der Linguistik stammt [Hof99].

Die Idee des *semantic indexing* ist, Dokumente hinsichtlich ihrer Zugehörigkeit zu solchen semantischen Entitäten zu indexieren. Ausgehend von der Information einer TDM, in der die Dimensionen des Suchraumes von Termen aufgespannt sind, werden semantische Analysemodelle angewendet um einen Konzeptraum aufzustellen und die Dokumentvektoren in diesen zu überführen.

Das resultierende Konzeptraummodell soll dabei bestimmten Anforderungen genügen. Es soll die Information des Termraummodells, vor allem auch in geringeren Dimensionen, möglichst vollständig erhalten und zusätzliche Information über die semantische Ähnlichkeit der Dokumente bieten.

Zu beachten ist, das die semantischen Entitäten nicht mit festen Begriffen belegt werden können, so wie ein menschlicher Leser das wahrscheinlich tun würde. Es lässt sich also nicht unbedingt eine für den Menschen sinnvolle Kategorisierung o.ä. anhand der Konzepte vornehmen, daher ist auch der Begriff "Konzept" etwas irreführend. Genau genommen sind die Konzepte vollkommen abstrakt, es lässt sich nicht einmal genau definieren, was ein Konzept eigentlich ist.

Die beiden im folgenden betrachteten Modelle LSI und PLSI unterscheiden sich hinsichtlich der Vorgehensweise, mit der semantische Informationen aus den TDM gewonnen werden und der daraus resultierenden Beschaffenheit des semantischen Raumes.

3.2 Latent Semantic Indexing (LSI)

Dieses Verfahren wurde 1990 von Deerwester et. al. vorgestellt [Dee90]. Die Entwickler adressieren die in Abschnitt 3.1 genannten Probleme und präsentieren einen algebraischen Lösungsansatz auf Basis einer Faktoranalyse der TDM. Zur Faktorisierung wird die Singulärwertzerlegung eingesetzt.

3.2.1 Singulärwertzerlegung

Die hier aufgezeigte Definintion der SVD folgt der Darstellung in [Ste07]. Ausgangspunkt sei eine $m \times n$ Matrix $\mathrm{A}_{m,n}$ mit $m > n$ vom Rang r.

Rang Der Rang einer Matrix A entspricht der maximalen Anzahl linear unabhängiger Zeilen oder Spalten. Für eine Matrix $\mathrm{A}_{m,n}$ gilt stets $rang(\mathrm{A}) \leq \min m, n$.

Die Singulärwertzerlegung von A ist definiert als

$$\mathrm{A} \stackrel{SVD}{=} \mathrm{USV}^T \tag{3.1}$$

dabei gilt

- S ist $r \times r$ diagonal, mit $r \leq min(m, n)$ und $\mathrm{S}_{0,0} \geq \mathrm{S}_{q,q} \geq \mathrm{S}_{r,r} (0 < q < r)$. Die Einträge auf der Hauptdiagonalen von S sind die Singulärwerte von A, absteigend nach Größe geordnet.
- U ist eine $m \times r$ Spalten-orthonormale Matrix.
- V^T ist eine $r \times n$ Zeilen-orthonormale Matrix.

Die Singulärwerte sind die Quadratwurzeln der Eigenwerte von AA^T bzw. A^TA. Dies lässt sich wie folgt zeigen:

Mit 3.1 (unter Beachtung von $(\mathrm{AB})^T = \mathrm{B}^T\mathrm{A}^T$) gilt

$$\mathrm{A}^T\mathrm{A} = (\mathrm{USV}^T)^T(\mathrm{USV}^T) = \mathrm{VSU}^T\mathrm{USV}^T = \mathrm{VS}^2\mathrm{V}^T \tag{3.2}$$

$$\mathrm{AA}^T = (\mathrm{USV}^T)(\mathrm{USV}^T)^T = \mathrm{USV}^T\mathrm{VSU}^T = \mathrm{US}^2\mathrm{U}^T \tag{3.3}$$

$\mathrm{A} \stackrel{SVD}{=} \mathrm{USV}^T$ lässt sich als Summe von Vektorprodukten darstellen:

$$\mathrm{A} = s_1(\mathbf{u}_1\mathbf{v}_1^T) + s_2(\mathbf{u}_2\mathbf{v}_2^T) + \ldots + s_r(\mathbf{u}_r\mathbf{v}_r^T) \tag{3.4}$$

Die Spalten von U werden als linksseitige, die Spalten von V als rechtsseitige Singulärvektoren von A bezeichnet.

Es gilt weiterhin: die Spalten von V sind Eigenvektoren von $\mathrm{A}^T\mathrm{A}$, die Spalten von U sind Eigenvektoren von $\mathrm{A}\mathrm{A}^T$.

Rangapproximation

Für die Matrix A mit dem Rang r lässt sich eine approximierte Matrix A' mit dem Rang k berechnen, indem man von den (geordneten) Singulärwerten s_n auf der Hauptdiagonalen von S nur die ersten k Werte beachtet:

$$\mathrm{A}' = \mathrm{U}_k\mathrm{S}_k\mathrm{V}_k^T \tag{3.5}$$

Dabei gilt: Für alle Matritzen $\mathrm{X}_{m,n}$ mit dem Rang höchstens k ist A' die Matrix, die A hinsichtlich der Frobenius-Norm am besten approximiert.

Die Frobenius-Norm ist definiert wie folgt:

$$\|\mathrm{A} - \mathrm{X}\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n}(\mathrm{A}_{ij} - \mathrm{X}_{ij})^2 \tag{3.6}$$

Die Rang-k-Approximation A' der Matrix A stellt also eine hinsichtlich der kleinsten Fehlerquadrate optimale Approximation dar.

3.2.2 Anwendung zur semantischen Analyse

Die in Abschnitt 3.2.1 gezeigte Matrixfaktoranalyse wird nun auf eine Term-Dokument-Matrix angewendet.

Sei $\mathrm{A}_{m,n}$ eine TDM mit m Termen und n Dokumenten. Die Matrix $\mathrm{A}^T\mathrm{A}$ (Gleichung 3.2) lässt sich interpretieren als Dokument-Dokument-Ähnlichkeitsmatrix, ein Eintrag $a_{i,j}$ repräsentiert die Ähnlichkeit von Dokument i zu Dokument j, denn $a_{i,j} = \mathbf{d}_i \cdot \mathbf{d}_j^T$ (der Eintrag $a_{i,j}$ entspricht also bei normalisierten Vektoren $\mathbf{d}$ der Kosinusähnlichkeit (Kap. 2.3)). Analog lässt sich die Matrix $\mathrm{A}\mathrm{A}^T$ (Gleichung 3.3) als Term-Term-Ähnlichkeitsmatrix interpretieren.

Die aus der Singulärwertzerlegung (s. Gleichung 3.1 S. 15) resultierenden Matritzen U mit der Dimension $m \times r$ und V^T mit der Dimension $r \times n$ werden als Term-Konzept- und

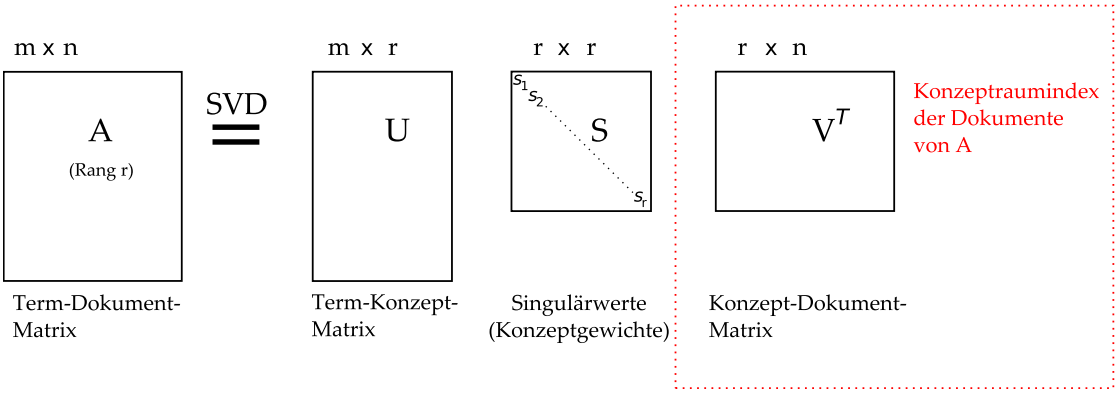

Abbildung 3.1: Singulärwertzerlegung einer Term-Dokument-Matrix mit Interpretation der resultierenden Faktoren.

Konzept-Dokument-Abbildungen interpretiert. Die der Größe nach absteigend geordneten Singulärwerte s können als Gewicht der Konzepte interpretiert werden. Abbildung 3.1 S. 17 zeigt diese Anwendung schematisch.

Der Vektorraum V^T wird als *Konzeptraum* bzw. *Konzeptindex* interpretiert. Dort ist jedes Dokument d durch einen Dokumentvektor $\mathbf{d}$ der Dimension r repräsentiert, wenn r der Rang der ursprünglichen Term-Dokument-Matrix war. Mit der in 3.2.1 vorgestellten Approximation durch weglassen der $r - k$ kleinsten Singulärwerte erhält man den k-dimensionalen Vektorraum V_k^T, ebenso den Term-Konzeptraum U_k. Die Approximation wird also zur Dimensionsreduktion eingesetzt.

Abbildung 3.2 S. 18 verdeutlicht, wie die Rang-k-Approximation eingesetzt wird, um die Dimension des Konzeptraumes zu verringern.

Für den Einsatz in einem Retrievalsystem ist es essenziell, dass die Möglichkeit des Vergleichens nicht nur auf die Dokumente, die zum ursprünglichen Dokumentraum gehörten, beschränkt ist, sondern auch auf unbekannte Dokumente und Anfragen erweitert werden kann.

Ein neues Dokument oder eine Anfrage, die als Termvektor $\mathbf{q}$ der Dimension $m \times 1$ vorliegt, lässt sich über eine lineare Abbildung in die Darstellung für den Konzeptraum $\mathbf{q}'$ projizieren:

$$\mathbf{q}' = \mathbf{q}^T \mathrm{U}_k \mathrm{S}_k^{-1} \tag{3.7}$$

Abbildung 3.3 S. 18 zeigt die Projektion eines neuen Dokuments in den Konzeptraum.

Ein solches neues Dokument wird, da es in der ursprünglichen Dokumentmenge nicht enthalten ist, auch als Pseudodokument bezeichnet. Die Matrix $\mathrm{P} = \mathrm{U}_k \mathrm{S}_k^{-1}$ wird also als

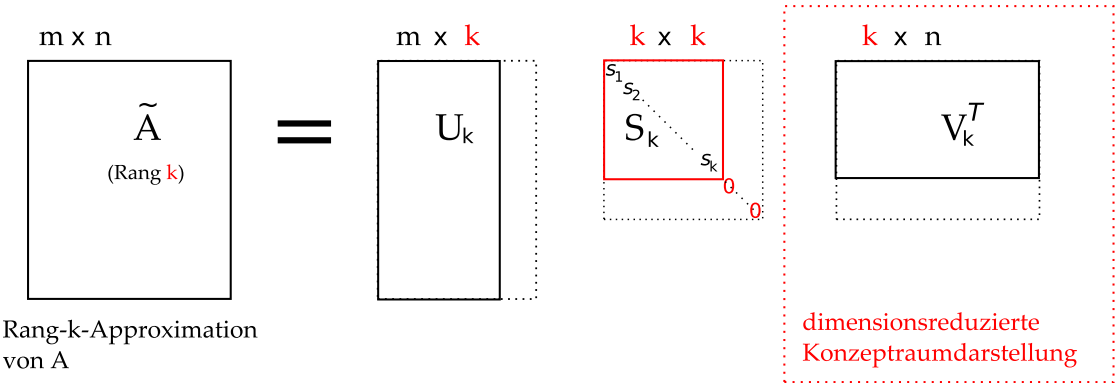

Abbildung 3.2: Anwendung der Rang-k-Approximation zur Dimensionsreduktion des Konzeptraumes.

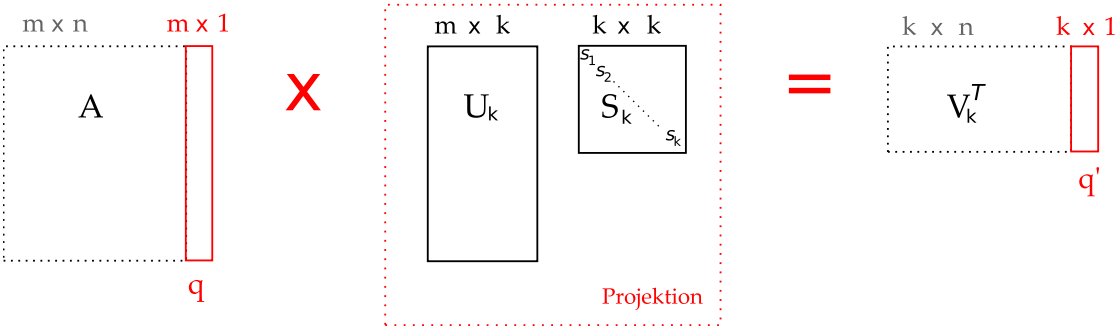

Abbildung 3.3: Projektion eines neuen Dokuments in den LSI-Konzeptraum.

Projektionsmatrix für neue Dokumente und Anfragen verwendet. Die Ergebnisse können im Konzeptraum mit allen üblichen Vektorraum-Retrievalfunktionen evaluiert werden, beispielsweise der Cosinusähnlichkeit (Kap. 2.3).

Die in Gleichung 3.7 S. 17 gezeigte Projektion wird als "einfalten" (engl: *fold-in*) von neuen Dokumenten in den Konzeptraum bezeichnet. Dabei werden laut [Dee90] die eingefalteten Dokumente im Konzeptraum beim Zentroiden ihrer enthaltenen Terme platziert.

Ebenso lässt sich ein neuer Term-Konzept-Vektor $\mathbf{t}'$ einfalten:

$$\mathbf{t}' = \mathbf{t}^T \mathrm{V}_k \mathrm{S}_k^{-1} \tag{3.8}$$

So wird es möglich, dem Konzeptraum weitere, in der ursprünglichen Kollektion nicht enthaltene oder aus Effizienzgründen zunächst nicht beachtete Terme hinzuzufügen. Ana-

log zu eingefalteten Dokumenten wird ein eingefalteter Term beim Zentroiden der enthaltenden Dokumente platziert.

Der Konzeptraum kann somit erweitert werden, was als wichtige Eigenschaft hinsichtlich des Berechnungsaufwandes der Singulärwertzerlegung angesehen wird. Hierbei ist aber zu beachten, dass sich der Konzeptraum mit jedem eingefalteten Dokument verändert und die Approximation weniger optimal wird. Ab einer bestimmten Anzahl neuer Dokumente ist es nötig, den gesamten Konzeptraum zu reindexieren. Theoretisch müsste dies sogar schon bei jedem einzelnen eingefalteten Dokument geschehen, um die Approximierung optimal zu halten, das ist jedoch in der Praxis aufgrund des hohen Aufwands nicht realisierbar. Es gibt Arbeiten, die sich mit diesem Problem beschäftigen. Beispielsweise werden in [Ber94] Verfahren vorgestellt, die den Konzeptraum neu berechnen ohne eine komplette Singulärwertzerlegung durchzuführen.

Im Rahmen dieser Arbeit spielt die Erweiterung des Konzeptraumes keine Rolle. In den Experimenten in Kap. 5 wird ein LSI-Modell trainiert und die daraus resultierende Projektionsmatrix genutzt um von der Trainingsmenge verschiedene Dokumentkollektionen in den Konzeptraum zu projizieren. Die so entstehenden Pseudodokumente werden im Konzeptraum nur evaluiert und tragen nicht zu dessen Erweiterung bei.

3.2.3 LSI-Dokumentmodell

Die Komponenten des LSI-Dokumentmodells laut Definition 1 stellen sich also wie folgt dar:

- Eine formales Dokument $\mathbf{d}$ ist dargestellt als Dokumentvektor v der Dimension k, wobei jede Dimension i mit $0 \leq i < k$ ein Konzept und der Wert v_i die Zugehörigkeit des Dokuments zu diesem Konzept bezeichnet.
- Eine Anfrage $\mathbf{q}$ wird analog zur Dokumentrepräsentation ebenfalls als Vektor im Konzeptraum dargestellt.
- Als Retrieval-Funktion $\rho_{\mathcal{R}}(\mathbf{q}, \mathbf{d})$ wird die Cosinusähnlichkeit verwendet.

3.2.4 LSI als maschinelles Lernverfahren

Die Singulärwertzerlegung einer TDM kann als Training eines LSI-Modells interpretiert werden. Dabei werden die verborgenen Konzepte gelernt und in der Projektionsmatrix gespeichert. Mit der Projektionsmatrix wird ein Konzeptraum definiert. Die Trainingsmenge sollte repräsentativ für die gesamte Kollektion sein, um eine hohe Retrievalperformanz im Konzeptraum zu erreichen.

3.2.5 Laufzeitverhalten

Der Berechnungsaufwand für LSI hängt vollständig von der SVD ab. Allgemein beträgt die Komplexität für eine Matrix mit m Zeilen und n Spalten $O(min(m^2n, mn^2))$. Es existieren jedoch verschiedene Algorithmen zur Berechnung der SVD, die für unterschiedliche Ausgangsdaten optimiert sind, beispielsweise für dünn besetzte Matritzen. Diese Algorithmen weisen ein besseres Laufzeitverhalten auf.

Die Projektion eines Term-Dokument-Vektors der Dimension m in einen Konzeptraum der Dimension k hat den Aufwand $O(m^2k)$. Die Projektion einer Kollektion mit n Dokumenten erfordert entsprechend $O(m^2kn)$. Hierbei ist zu beachten, dass die Dimension des Konzeptraumes meist sehr viel geringer ist als die Dimension des Termraumes, also $m \gg k$.

3.2.6 Kritik

Das LSI-Modell bietet einige Vorteile gegenüber dem Termbasierten Modell. Verschiedene Experimente der Entwickler (vgl. [Dee90]) und weitere Arbeiten bis heute zeigen, dass die Matrixfaktorisierung mittels Singulärwertzerlegung genutzt werden kann, um automatisiert latente semantische Informationen in einer TDM zu erkennen und zur Indexierung nutzbar zu machen.

Die Konzeptraumrepräsentation der Terme lässt sich verwenden um Synonyme zu entdecken, d.h. das Modell liefert eine implizite Verwendungsmöglichkeit als Thesaurus. Homonyme werden zwar erkannt, allerdings nicht mit zufriedenstellendem Erfolg. Die Erkennung von Homonymen ist einer der Schwachpunkte von LSI, welcher in der Weiterentwicklung zu PLSI (s. Abschnitt 3.3) besonders adressiert wird.

Dokumente und Anfragen werden durch Projektion in den Konzeptraum semantisch erweitert, d.h. die explizit formulierten Worte werden um eine implizite Bedeutung angereichert und das Retrieval im Konzeptraum kann auch hinsichtlich dieser Information erfolgen. Es gibt weiterhin Arbeiten ([Lan98]) die untersuchen, wie gut Latent Semantic Analysis in der Lage ist, menschliches Allgemeinwissen hinsichtlich des Sprachgebrauchs abzubilden.

Eine sehr interessante Anwendung des LSI-Modells wird in [Wei05] angesprochen. Dort wird gezeigt, wie ein LSI-Modell mehrsprachiges Retrieval ermöglicht: Indem verschiedene Sprachversionen eines Dokuments hinsichtlich der Analyse als ein einzelnes Dokument behandelt werden, lassen sich, mit einer Suchanfrage in einer der verwendeten Sprachen, auch Dokumente aller anderen Sprachen finden.

Beim Clustering im LSI-Konzeptraum zeigt sich, dass eine sehr geringe Dimension für die Dokumentvektoren zu sehr guten Ergebnissen führt, wobei die Information des ursprünglichen Termraumes erhalten bleibt und semantisch erweitert wird (s. Kapitel 5).

Aus algorithmischer Sicht basiert LSI mit der Singulärwertzerlegung auf einem mathematisch gut ergründeten Verfahren, welches ein vorhersagbares Ergebnis liefert. Der Berechnungsaufwand ist allerdings sehr hoch (vgl. Abschnitt 3.2.5).

Außerdem weist das Modell weitere Negativpunkte auf. Die Singulärwertzerlegung erzeugt negative Einträge in der Projektionsmatrix, was sich auf die Beschaffenheit des Konzeptraumes auswirkt und beim Einsatz der Cosinusähnlichkeit beachtet werden muss.

Die Funktionsweise von LSI ist bisher wissenschaftlich nur ansatzweise erkundet. Es gibt einige Arbeiten, die sich mit einer genaueren Analyse befassen, beispielsweise [Pap98]. Die vorliegenden Erklärungen basieren auf Theorien (vgl. [Ste07]). Es wird beispielsweise angenommen, dass sich in der Term-Dokument-Matrix durch natürlichsprachliche Phänomene wie Synonyme und wiederholte Phrasen lineare Abhängigkeiten bilden, die sich wiederum als latente semantische Konzepte interpretieren lassen. Bei der SVD werden diese linearen Abhängigkeiten aufgelöst, die berechneten Konzeptmatritzen sind Spalten- bzw. Zeilenorthonormal (also frei von Linearkombinationen). Welche Phänomene tatsächlich als semantische Konzepte interpretiert werden ist nicht bekannt.

LSI kann kein generatives Modell für den Zusammenhang von Dokumenten und Termen einer Kollektion aufweisen. Die Verwendung der Singulärwertzerlegung induziert laut [Hof99] eine Gaußsche Normalverteilung der Term-Dokument-Kookkurrenzen (d.h. gemeinsames Auftreten), die aber wissenschaftlich nicht begründbar ist.

Weiterhin sind die errechneten semantischen Konzepte abstrakt, lassen sich also beispielsweise nicht mit Begriffen gleichsetzen. Somit kann man für ein Dokument nur sagen, dass es zu gewissen Teilen den Konzepten 1, 2 und 3 angehört, aber nicht wofür diese stehen.

Ein großes und bisher ungelöstes Problem ist die Wahl des richtigen Parameters k (s. Abschnitt 3.2.1) für die Dimensionsreduktion. Dieser Wert muss als Modellparameter vorgegeben werden und beeinflusst den Erfolg des Retrieval entscheidend (s. Kap. 5). Der richtige Wert ist abhängig von der Zusammenstellung der Kollektion, es existieren keine dem Autor bekannten Regeln für die Wahl oder Anpassung des Wertes.

Eine interessante Arbeit, die sich mit einer Erklärung für LSI und der Wahl der Dimension beschäftigt ist [Bas05]. Die Autoren untersuchen die Verteilung der Term-Term-Verwandtschaft (s. auch Gleichung 3.3 S. 15) über das gesamte Dimensionsspektrum und zeigen, dass das Verhältnis zwischen Termen in der Form dieser Verwandtschaftskurve, bezeichnet als *curve of relatedness scores*, ersichtlich ist. Sie zeigen auch, dass es

keine optimale Dimension für alle Termpaare gibt und stellen Algorithmen vor, die statt einer fixen Dimensionswahl auf Basis der relatedness curve arbeiten.

Es ist zu erwarten, dass die Vorteile von LSI besonders dann zum Tragen kommen, wenn die Dokumentkollektion fest ist oder sich nur geringfügig ändert. Auf offenen, stark dynamischen Kollektionen sollten die Ergebnisse schwach sein. Das hängt u.a. damit zusammen, dass es auf solchen Kollektionen schwierig ist, eine repräsentative Trainingsmenge zu definieren. Außerdem erzeugen solche Kollektionen beim Indexing ein sehr umfangreiches Vokabular, das auf Grund der Komplexität des SVD-Algorithmus stark reduziert werden müsste, um das Training in annehmbarer Zeit durchführen zu können. LSI ist deshalb z.B. für Web-Suchmaschinen ungeeignet.

3.3 Probabilistic Latent Semantic Indexing (PLSI)

Dieses Verfahren, vorgestellt 1999 in [Hof99], stellt eine Weiterentwicklung von LSI dar. Ausschlaggebend für die Entwicklung waren folgende Kritikpunkte an LSI:

- LSI basiert nicht auf einem fundierten Sprachmodell
- mit der Verwendung der SVD wird eine Verteilung der Term-Dokument - Kookkurrenzen zugrundegelegt, die wissenschaftlich nicht zu rechtfertigen ist
- LSI kann zwar das Problem der Synonyme erfolgreich angehen, aber das Modell weist Probleme im Umgang mit Homonymen auf

Während bei LSI eine TDM mit Mitteln der Linearen Algebra analysiert wird, wobei offen bleibt, wie die beobachtbaren Daten eigentlich zustande kommen, ist der Ausgangspunkt der Überlegungen zu PLSI der Versuch, die Kookkurrenzen von Dokumenten und Worten in einer Dokumentkollektion mathematisch zu erschließen. Dabei wird ein statistischer Ansatz gewählt und ein Modell aufgestellt, welches diese Zusammenhänge durch bedingte Wahrscheinlichkeiten abbildet.

3.3.1 Aspektmodell

Das in [Hof99] benutzte statistische Modell zur Erschließung der Zusammenhänge von Worten und Dokumenten wird als *Aspect Model* [Hof98] bezeichnet. Das Aspect Model ist ein generatives Modell. Es erklärt das Auftreten eines Wortes in einem Dokument damit, dass das Wort von dem Dokument mit einer bestimmten Wahrscheinlichkeit generiert wird. Grundlegende Idee ist darüber hinaus die Erweiterung der beobachtbaren Kookkurrenz von Dokumenten und Worten $P(d,w)$ mit einer unbeobachteten Variable $z \in Z = \{z_1, \ldots, z_k\}$. Jede Kookkurrenz $P(d,w)$ wird mit einem z assoziiert. Die unbeobachteten Variablen z werden als latente Konzepte interpretiert.

Der Prozess, in dem Dokumente d Worte w generieren, ist im Aspektmodell wie folgt formuliert:

1. Wähle ein Dokument mit der A-priori-Wahrscheinlichkeit $P(d)$
2. Generiere ein Konzept mit der bedingten Wahrscheinlichkeit $P(z|d)$
3. Generiere ein Wort mit der bedingten Wahrscheinlichkeit $P(w|z)$

Dokumente generieren also Konzepte und diese wiederum generieren die Worte. Bildlich dargestellt ist dieser Prozess in Abbildung 3.4 S. 24.

Das Zusammentreffen von Dokumenten und Worten ist explizit beobachtbar und wird beispielsweise bei der Indexierung einer Dokumentkollektion quantifiziert (vgl. Abschnitt 2.2.1). Diese Kookkurrenz ausgedrückt als Wahrscheinlichkeit $P(d,w)$ ist darstellbar als

$$P(d,w) = P(d)P(w|d), P(w|d) = \sum_{z \in Z} P(w|z)P(z|d) \tag{3.9}$$

Durch eine Umformung unter Anwendung des Bayestheorems

$$P(z|d) = \frac{P(z|d)P(z)}{P(d)}$$
$$P(d,w) = P(d) \sum_{z \in Z} P(w|z)P(z|d)P(z)\frac{1}{P(d)}$$

erhält man die zu 3.9 äquivalente Darstellung

$$P(d,w) = \sum_{z \in Z} P(z)P(w|z)P(d|z) \tag{3.10}$$

Die Darstellung 3.10 ist im Hinblick auf Worte w und Dokumente d symmetrisch, dieser Umstand ist für die Anwendung vorteilhaft.

Die A-priori-Wahrscheinlichkeit $P(z)$ sowie die bedingten Wahrscheinlichkeiten $P(w|z)$ und $P(d|z)$ stellen die Parameter des PLSI-Modells dar. Ziel einer Berechnung ist es nun, für diese Parameter Werte zu finden, so dass die bekannten Kookkurrenzen $P(d,w)$ möglichst gut approximiert werden.

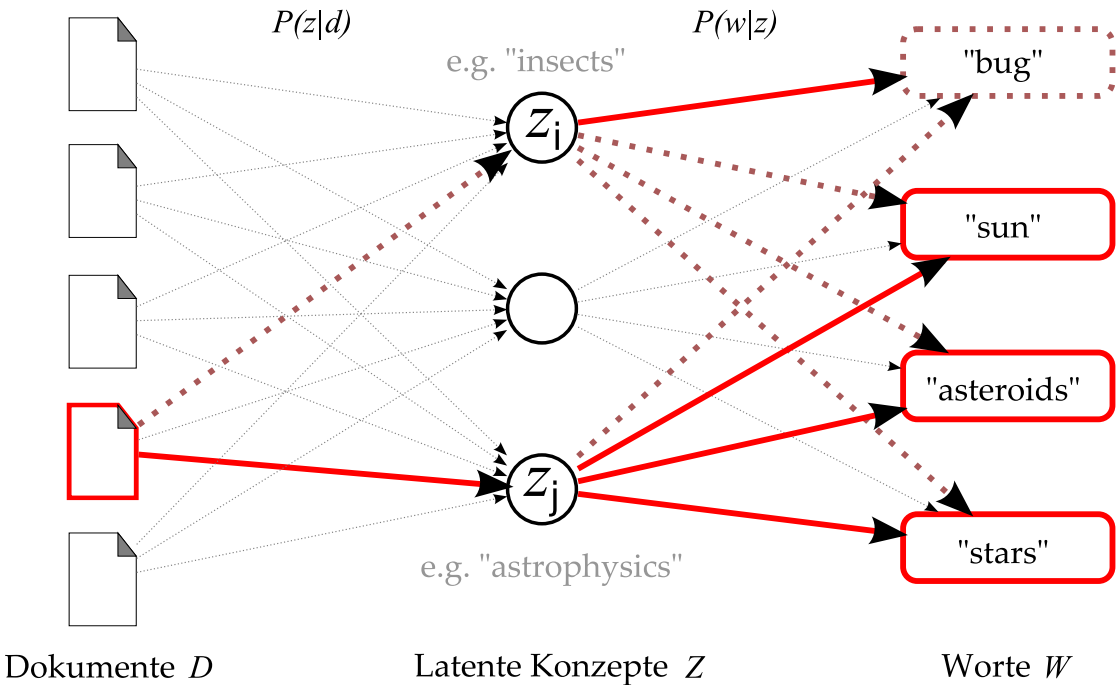

Abbildung 3.4: Prinzip der Generierung von Dokument-Wort-Kookkurrenzen im Aspektmodell (Quelle: [Wei05]).

Voraussetzung für die Berechnung ist die Annahme, dass in Abhängigkeit von der latenten Klasse z die Worte w unabhängig von den Dokumenten d generiert werden (Annahme der bedingten Unabhängigkeit).

3.3.2 Maximum Likelihood Schätzung

Das Teilgebiet der Statistik, das sich mit dem Problem des Schätzens von unbekannten Parametern einer Grundgesamtheit auf Basis einer Stichprobe beschäftigt, nennt man *inferentielle Statistik*. Mit dem Aspektmodell wird die Kookkurrenz von Worten und Dokumenten auf ein solches Problem abgebildet. Eine Schätzung der Modellparameter, die das Modell dem tatsächlichen System optimal anpasst, wird als Maximum Likelihood Schätzung (engl: *Maximum Likelihood Estimation, MLE*) bezeichnet.

Der englische Begriff *Likelihood* wird oft als *Wahrscheinlichkeit* übersetzt, diese Übersetzung ist aber nicht korrekt, da die Wahrscheinlichkeit anders definiert ist.

Likelihood

Der Begriff *Likelihood* bezeichnet den Grad der Anpassung der Parameter eines auf Basis beobachteter Daten (z.B. Experimentergebnisse, Stichproben) erstellten Modells an die tatsächlichen Parameter eines Systems, welches diese Daten erzeugt hat. Als *Maximum Likelihood* werden die Modellparameter bezeichnet, die die beobachteten Daten am wahrscheinlichsten erzeugen. Die Likelihood-Funktion L beschreibt die Likelihood-Verteilung im Parameterraum. Das Maximum Likelihood kann auf analytischem Wege berechnet werden, wenn es möglich ist, das Extremum der Likelihood-Funktion zu bestimmen. Dafür müssen aber die Modellparameter bekannt sein.

Zu beachten ist dabei, dass man aus dem Maximum Likelihood nicht auf die realen Parameter schliessen kann, diese bleiben weiterhin unbekannt. Einerseits kann man nur die Ergebnisse einer begrenzten Anzahl von Experimenten untersuchen, andererseits können die Parameter des Modells nur einen Teil der Komplexität des tatsächlichen Systems abbilden. Durch Verwendung bestimmter statistischer Methoden können die Modellparameter beliebig genau den beobachteten Daten angepasst werden, was allerdings auch einen beliebig hohen Aufwand erfordert.

Im Falle des Aspektmodells sind zudem unbeobachtete Parameter, entstanden durch die Einführung der latenten Klassen z, einzuberechnen. Das Maximum Likelihood kann für solche komplexe Problemstellungen nicht auf analytischem Wege berechnet werden. Die Standardmethode zur Berechnung des Maximum Likelihood unter Beachtung unbeobachteter Parameter ist ein iteratives Optimerungsverfahren, das als *Erwartungsmaximierung* (engl: *Expectation Maximization, EM*) bezeichnet wird.

3.3.3 Expectation Maximization

Der Expectation Maximization-Algorithmus wird wie folgt beschrieben [Bor04]: Eine EM-Iteration besteht aus zwei Schritten, *Expectation* (E) und *Maximization* (M). Im E-Schritt werden die unbeobachteten Parameter auf Grundlage der beobachteten Daten und der aktuellen Schätzung der Modellparameter geschätzt. Im speziellen Fall der ersten Iteration wird für die Modellparameter oft eine zufällige Belegung gewählt. Im M-Schritt wird die Likelihood-Funktion maximiert, unter der Annahme dass die unbekannten Parameter bekannt sind. An Stelle der unbekannten Parameter wird hier die Schätzung aus dem E-Schritt verwendet. Es lässt sich beweisen dass dieses Verfahren konvergiert, weil in jeder Iteration das Likelihood erhöht wird.

Angewendet auf das Problem der Wort-Dokument-Kookkurrenz (s. auch 3.10) werden in [Hof99] die folgenden Gleichungen entwickelt:

E-Schritt:

$$P(z|d,w) = \frac{P(z)P(w|z)P(d|z)}{\sum_{z' \in Z} P(z')P(w|z')P(d|z')} \tag{3.11}$$

M-Schritt:

$$P(w|z) \propto \sum_{d \in D} n(d,w)P(z|d,w) \tag{3.12}$$

$$P(d|z) \propto \sum_{w \in W} n(d,w)P(z|d,w) \tag{3.13}$$

$$P(z) \propto \sum_{d \in D} \sum_{w \in W} n(d,w)P(z|d,w) \tag{3.14}$$

Dabei bezeichnet $n(d,w)$ die Anzahl des Auftretens von Term w in Dokument d, also die Termfrequenz tf.

3.3.4 PLSI als maschinelles Lernverfahren

Die Optimierung der Parameter $P(z)$, $P(d|z)$ und $P(w|z)$ durch den EM-Algorithmus auf Grundlage der beobachteten Daten $P(d,w)$ wird als Training eines PLSI-Modells interpretiert. Dabei wird die Charakteristik der Dokumentkollektion gelernt, die zum Training eingesetzt wird. Ein in diesem Zusammenhang auftretendes Problem ist, dass die Parameter zu sehr auf die Trainingsmenge angepasst werden. Diesen Effekt bezeichnet man als *Überanpassung* (engl: *Overfitting*). Überangepasste Parameter sind zu sehr auf die Trainingsmenge, die nur eine Auswahl der Grundgesamtheit darstellt, spezialisiert und liefern schlechte Ergebnisse für Dokumente, die nicht in der Trainingsmenge enthalten sind.

Bezogen auf die Likelihood-Funktion L hat man mit überangepassten Parametern ein lokales Maximum gefunden. Allgemein sollen die Parameter aber so gewählt sein, dass das Modell die Grundgesamtheit gut abbildet, man bezeichnet das Modell dann als gut *generalisierend*. Man möchte optimalerweise das globale Maximum von L finden.

Um Überanpassung zu vermeiden, wird üblicherweise eine zweite Kollektion aus der Grundgesamtheit erstellt und die Parameter während des Trainings auf dieser Validationsmenge überprüft. Wenn sich die Anpassung auf der Validationsmenge von einer

Iteration zur folgenden verschlechtert, hat man eine Überanpassung auf die Trainingsmenge erkannt und kann das Training abbrechen.

Benuzt man die in 3.3.3 gezeigten Formeln zum Training, werden die Parameter von Iteration zu Iteration stark verändert und Überanpassung kann auftreten. In [Hof99] wird daher auch ein modifiziertes Verfahren vorgestellt, das Überanpassung erkennt und in diesem Falle die Anpassungsgeschwindigkeit der Parameter absenkt, um zu vermeiden dass sich der Algorithmus in lokalen Maxima von L "verfängt". Dieses Verfahren wird als *Tempered Expectation Maximization* bezeichnet.

3.3.5 Tempered Expectation Maximization

In [Nea98] wird gezeigt, dass die Ergebnisse des EM-Algorithmus analog durch Minimierung einer Funktion, bekannt als *Helmholtz free energy* erreicht werden können. Diese Funktion basiert auf Beobachtungen physikalischer Systeme, denen zufolge mittels kontrollierter, langsamer Erhitzung oder Abkühlung von Werkstoffen gleichmäßigere, glatte Oberflächenstrukturen entstehen. In [Hof99] wird die Helmholtz free energy für das Aspektmodell definiert:

$$\begin{aligned} \mathrm{F}_\beta = &- \beta \sum_{d,w} n(d|w) \sum_z \tilde{P}(z;d,w) \log P(d,w|z)P(z) \\ &+ \sum_{d,w} n(d|w) \sum_z \tilde{P}(z;d,w) \log \tilde{P}(z;d,w) \end{aligned} \tag{3.15}$$

$$\tilde{P}(z;d,w) = \frac{[P(z)P(d|z)P(w|z)]^\beta}{\sum_{z'} [P(z')P(d|z')P(w|z')]^\beta} \tag{3.16}$$

Der Parameter β mit Werten < 1 dämpft den Einfluss der A-posteriori-Wahrscheinlichkeiten bzw. verringert die Anpassungsgeschwindigkeit. Mit $\beta = 1$ unterscheidet sich dieses Verfahren nicht von der Vorgehensweise in Abschnitt 3.3.3. Der Algorithmus wird wie folgt angewendet:

1. Setze $\beta = 1$ und führe EM durch bis Overfitting auftritt.
2. Verringere $\beta = \eta\beta$ mit $\eta < 1$ und führe eine TEM-Iteration aus.
3. Solange sich die Performanz auch auf den Validationsdaten verbessert, fahre fort mit TEM-Iterationen, ansonsten fahre fort mit Schritt 2.
4. Stoppe wenn β einen bestimmten Wert unterschreitet und weitere Iterationen die Performanz nur noch unwesentlich verbessern.

3.3.6 PLSI-Dokumentmodell

Die Parameter $P(z)$, $P(w|z)$ und $P(d|z)$ lassen sich in Matrixschreibweise formulieren und den Matritzen des LSI-Modells (s. Abschnitt 3.2.1) entsprechend interpretieren:

$$P(z) = diag(P(z_k))_{k,k} = \mathrm{S}_k \tag{3.17}$$
$$P(w|z) = (P(w_j|z_k))_{j,k} = \mathrm{U}_k \tag{3.18}$$
$$P(d|z) = (P(d_i|z_k))_{i,k} = \mathrm{V}_k \tag{3.19}$$

3.18 ist interpretierbar als Term-Konzept-Mapping und 3.19 als Dokument-Konzept-Mapping. Die Diagonalmatrix 3.17 enthält die A-priori-Wahrscheinlichkeiten für die latente Klassen z. Die transponierte Matrix 3.19 ist als Konzeptraum interpretierbar, in dem die Dokumente als Dokumentvektoren der Dimension k dargestellt sind.

Um Anfragen q in den Konzeptraum zu überführen, werden die Parameter $P(w|z)$ eines trainierten Modells verwendet und per Expectation Maximization die bedingten Wahrscheinlichkeiten $P(z|q)$, also die Konzepte, welche die Anfrage am wahrscheinlichsten generieren würden, berechnet (vgl. [Wei05]).

$$P(z|q,w) = \frac{P(z|q)P(w|z)}{\sum_{z'} P(z'|q)P(w|z')} \tag{3.20}$$
$$P(z|q) = \frac{\sum_{w} n(q,w)P(z|q,w)}{\sum_{w,z'} n(q,w)P(z'|q,w)} \tag{3.21}$$

Die Komponenten des PLSI-Dokumentmodells laut Definition 1 stellen sich also wie folgt dar:

- Eine formales Dokument $\mathbf{d}$ ist dargestellt als Dokumentvektor v der Dimension k, wobei jede Dimension i mit $0 \leq i < k$ ein Konzept und der Wert v_i die bedingte Wahrscheinlichkeit $P(z_i|d)$ bezeichnet.
- Eine Anfrage $\mathbf{q}$ wird analog zur Dokumentrepräsentation ebenfalls als Vektor im Konzeptraum dargestellt.
- Als Retrieval-Funktion $\rho_{\mathcal{R}}(\mathbf{q}, \mathbf{d})$ wird die Cosinusähnlichkeit verwendet.

3.3.7 Aufwand

Jeder einzelne Schritt des EM-Algorithmus ist abhängig von der Anzahl der Dokumente d, Worte w und latenten Konzepte z (vgl. Abschnitt 3.3.3). Damit ergibt sich ein Aufwand von $O(dwz)$. Allgemein lässt sich die Laufzeit mit $O(N^3)$ abschätzen. Da es sich um einen iterativen Algorithmus handelt, kann auch die Anzahl der Iterationen mit einbezogen werden, diese ist hinsichtlich des Algorithmus aber als zuvor festgelegte Konstante anzusehen, die, ebenso wie die vier Schritte für eine Iteration, in der Abschätzung üblicherweise nicht explizit aufgeführt ist.

3.3.8 Kritik

Mit dem Aspektmodell ist eine solide mathemathische Grundlage für die Dokument-Wort-Kookkurrenzen definiert. Experimente auf verschiedenen Kollektionen zeigen, dass das Modell in der Lage ist, auch in Fällen in denen LSI versagt, gute Ergebnisse zu bringen. So wird beispielsweise in [Hof99] die Fähigkeit zur Erkennung und korrekten Auflösung von Homonymen demonstriert. Die wesentlichen Designziele sind somit erfüllt.

Negative Kritikpunkte und Ansätze für weitere Verbesserungen werden in [Ble03] angeführt. PLSI definiert das statistische Modell nur für die Dokumentkollektion, mit der das Modell trainiert wird. Damit ist nicht geklärt, wie Dokumente außerhalb der Trainingskollektion gewichtet werden sollen. Die Anzahl der Parameter ist abhängig von der Größe der Trainingskollektion, was wiederum zu Problemen mit Überanpassung führt, die bei PLSI nicht komplett beseitigt werden können.

3.4 Gegenüberstellung LSI - PLSI

Beide Verfahren basieren auf dem Vektorraummodell, zeigen Möglichkeiten zur semantischen Analyse und können zur automatischen Indexierung von Dokumenten in einem Konzeptraum eingesetzt werden. Die Herangehensweise ist jedoch sehr unterschiedlich. Die Unterschiede sollen hier noch einmal dargestellt werden.

LSI basiert mit der Singulärwertzerlegung auf einem Matrixfaktorisierungsverfahren und definiert kein Modell mit dem Dokumente oder Worte generiert werden. Im Gegensatz dazu basiert PLSI von vornherein auf einem solchen generativen Modell und macht sich statistische Verfahren zu nutze. Die Singulärwertzerlegung ist ein deterministischer Algorithmus, während der EM-Algorithmus nichtdeterministisch ist und bei wiederholten Experimenten nur tendenziell die gleichen Ergebnisse liefert. Die optimale Approximierung ist bei LSI über die Frobenius-Norm definiert während PLSI das Maximum Likelihood heranzieht.

3.5 Allgemeine Kritik

Beide vorgestellten Retrievalmodelle mit versteckten Variablen zeigen gegenüber den Termbasierten Modellen signifikante Verbesserungen in der Retrievalperformanz. Sie sind jedoch deutlich komplexer in der Anwendung und das Training ist sehr aufwendig. Daraus ergibt sich für beide Verfahren die gleiche Einschränkung hinsichtlich des Einsatzgebietes. Auf geschlossenen, statischen Dokumentkollektionen sind die Resultate sehr gut, weil die Modelle mit repräsentativen Teilmengen trainiert werden können. Auf dynamischen und stark heterogenen Kollektionen wie etwa dem Web ist dies nicht möglich, hier sind die Methoden praktisch unbrauchbar.

4 Softwaremodul für aitools

Im Rahmen dieser Arbeit wurde ein Framework entwickelt, das die vorgestellten Modelle implementiert und in den Experimenten (s. Kapitel 5) Anwendung fand. Dieses Kapitel behandelt den Aufbau und die Benutzung des Frameworks.

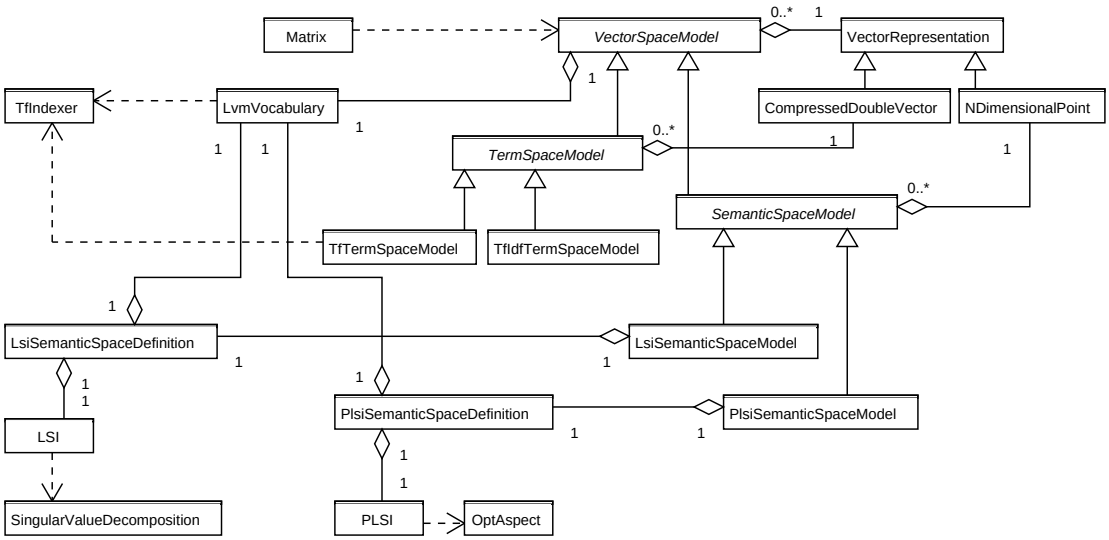

Abbildung 4.1: Schematischer Aufbau des LVM-Frameworks

4.1 Aufbau

Das Framework ist in Java implementiert. Kernkomponenten sind Klassen, die verschiedene Vektorraummodelle implementieren und Methoden darauf zur Verfügung stellen. Die vorgestellten Klassen befinden sich, solange nicht anders angegeben, im Package *de.aitools.latentvariablemodels.lvmcomponents*. Das Framework ist als Komponente für das *aitools* IR-Framework der Professur Web Technology and Information Systems an der Bauhaus-Universität Weimar konzipiert und nutzt einige der darin vorhandenen Funktionalitäten. Das System wurde vor allem entwickelt um Experimente schnell und einfach durchführen zu können.

Die Abbildung 4.1 S. 31 zeigt den schematischen Aufbau des Frameworks. Die wichtigen Klassen werden in Abschnitt 4.1.2 erläutert.

4.1.1 Referenzierte Bibliotheken

Auf eine Implementation der grundlegenden Algorithmen, SVD für LSI (s. Abschnitt 3.2.1) und TEM für PLSI (s. Abschnitt 3.3.5), wurde im Rahmen dieser Arbeit verzichtet, stattdessen wurden auf externe Bibliotheken zurückgegriffen.

Singulärwertzerlegung

Für die Singulärwertzerlegung existieren eine Reihe verschiedener Implementationen. In diesem Falle wird die Implementation aus dem Java-Matrix Package *Jama*, verfügbar unter `http://math.nist.gov/javanumerics/jama/Jama-1.0.2.jar` (letzter Zugriff: 3.5.2008), eingesetzt. Dokumentation, Quellcode und weitere Informationen zu diesem Paket sind ebenfalls unter der angegebenen URL verfügbar.

Tempered Expectation Maximization

Für den TEM-Algorithmus gibt es nur sehr wenige Implementationen. Verwendet wurde eine Referenzimplementation aus dem Paket *PennAspect*, verfügbar unter `http://www.cis.upenn.edu/datamining/software_dist/PennAspect/pa1-01.jar` (letzter Zugriff: 3.5.2008) und dokumentiert in [Sch03].

aitools

Verschiedene Hilfsklassen und Algorithmen aus dem *aitools*-Framework wurden genutzt. Dazu gehören die abstrakte Klasse *VectorRepresentation* und deren Implementationen *CompressedDoubleVector* und *NDimensionalPoint*, die die Cosinusähnlichkeit implementieren, die Graphstruktur *UndirectedGraph* und eine Implementation des Clusteringalgorithmus *MajorClust*.

4.1.2 Klassenhierarchie der Vektorraummodelle

VectorSpaceModel

Die abstrakte Basisklasse *VectorSpaceModel* stellt einen allgemeinen Vektorraum dar und bildet somit den kleinsten gemeinsamen Nenner aller beschriebenen Modelle. Die Elemente dieses Raumes sind formale Dokumente. Jeder Vektorraum besitzt zusätzlich ein eigenes Vokabular. Der Zugriff auf die Dokumente des Vektorraumes kann einzeln, als Collection oder als Matrixdarstellung erfolgen. Weiterhin lässt sich die Anzahl der

Dokumente, die Dimension des Vektorraums und das Vokabular erfragen. Von *VectorSpaceModel* sind zwei weitere abstrakte Klassen abgeleitet, *TermSpaceModel* und *ConceptSpaceModel*.

TermSpaceModel

Die Klasse *TermSpaceModel* implementiert einen Vektorraum, dessen Dokumente durch Termvektoren repräsentiert werden. Die Termvektoren werden durch die *aitools*-Klasse *CompressedDoubleVector* implementiert, da diese Klasse eine effizientere Speicherung dünn besetzter Vektoren erlaubt. Die erweiterte Funktionalität umfasst den Zugriff auf die Dokumentfrequenz der Terme und eine auf der Dokumentfrequenz basierende Dimensionsreduktionsmethode. Abgeleitet von *TermSpaceModel* sind konkrete Klassen, deren Termvektoren mit Termfrequenz (*TfTermSpaceModel*) oder Termfrequenz-Inversdokumentfrequenz (*TfIdfTermSpaceModel*) gewichtet sind.

SemanticSpaceModel

Die Klasse *SemanticSpaceModel* implementiert einen Vektorraum, dessen Dokumente durch Konzeptvektoren repräsentiert werden. Die Konzeptvektoren sind durch die *aitools*-Klasse *NDimensionalPoint* implementiert. Eine konkrete Konzeptraumimplementierung stellt die abgeleitete Klasse *LsiConceptSpaceModel* dar.

TfTermSpaceModel

Die Klasse *TfTermSpaceModel* implementiert einen Vektorraum, dessen Dokumente durch tf-gewichtete Termvektoren repräsentiert werden. Ein Objekt dieser Klasse stellt den Ausgangspunkt für alle Experimente dar, die direkte Erstellung eines *TfTermSpaceModel* aus einer realen Dokumentkollektion wird vom *TfIndexer* geleistet. Aus einem *TfTermSpaceModel*-Objekt lässt sich ein *TfIdfTermSpaceModel*-Objekt erzeugen und mit einem der beiden Termraummodelle können die Konzeptraummodelle trainiert werden.

TfIdfTermSpaceModel

Die Klasse *TfIdfTermSpaceModel* implementiert einen Vektorraum, dessen Dokumente durch $tfidf$-gewichtete Termvektoren repräsentiert werden. Ein *TfIdfTermSpaceModel*-Objekt lässt sich aus einem *TfTermSpaceModel*-Objekt erzeugen. Die Klasse bietet kaum erweiterte Funktionalität, sie dient vor allem der Übersicht bei Experimenten.

LsiSemanticSpaceModel

Die Klasse *LsiSemanticSpaceModel* implementiert einen Vektorraum, dessen Dokumente durch LSI-Konzeptvektoren repräsentiert sind. Zusätzlich enthält ein *LsiSemanticSpaceModel* ein Objekt der Klasse *LsiSemanticSpaceDefinition*, welches den Konzeptraum definiert. Ein *LsiSemanticSpaceModel* wird mit einem *TermSpaceModel* trainiert, dabei wird

das *TermSpaceModel* als Term-Dokument-Matrix interpretiert, mittels Singulärwertzerlegung analysiert, die resultierenden Matritzen gespeichert und der Konzeptraumindex erstellt. Einem existierenden *LsiSemanticSpaceModel*-Objekt können *TermSpaceModel*-Objekte übergeben werden, die dann automatisch in den Konzeptraum projiziert werden. Als zusätzliche Funktionalität lässt sich die Dimension des Konzeptraumes festlegen.

PlsiSemanticSpaceModel

Die Klasse *PlsiSemanticSpaceModel* implementiert einen Vektorraum, dessen Dokumente durch PLSI-Konzeptvektoren repräsentiert werden. Es verhält sich ansonsten analog zum LSI-Konzeptmodell. Die Definition des Konzeptraumes wird in einem *PlsiSemanticSpaceDefinition*-Objekt gehalten. Beim Training müssen mehrere Parameter übergeben werden (s. Abschnitt 4.2.2 S. 35), u.a. muss die Dimension des Konzeptraumes bereits beim Training festliegen.

4.1.3 Konzeptraumdefinitionsklassen

Die Klassen *LsiSemanticSpaceDefinition* und *PlsiSemanticSpaceDefinition* werden genutzt, um trainierte Modelle zu serialisieren. Sie enthalten je ein Objekt der Klassen *LSI* bzw. *PLSI* und ein *LvmVocabulary*-Objekt. *LSI* und *PLSI* kapseln die tatsächlichen Trainingsfunktionen und die Trainingsergebnisse, d.h. die Matritzen $\mathrm{U}, \mathrm{S}, \mathrm{V}$ im Falle von LSI und die trainierten Parameter $P(z), P(z|w), P(z|d)$ (ebenfalls in Matrixdarstellung) im Falle von PLSI. Die Definitionsklassen werden weiterhin eingesetzt um Objekte der Konzeptraummodelle damit zu initialisieren. Die so erstellten Konzeptraum-Objekte sind leer bis die Dokumente eines Termraum-Objekts hinzugefügt werden, welche dabei automatisch in den Konzeptraum projiziert werden.

4.1.4 Vokabular

Die Klasse *LvmVocabulary* speichert das Vokabular, welches beim Indexieren eines Termraummodells erstellt wird. Beim Training eines Konzeptraummodells wird das Vokabular in die Konzeptraumdefinition übernommen. Dies ist wichtig, um die Konsistenz von serialisierten Konzeptmodellen und neu indexierten Testkollektionen zu gewährleisten. Die Testkollektionen, die in den Konzeptraum projiziert werden sollen müssen mit dem gleichen Vokabular indexiert werden wie die Trainingskollektion des Konzeptraummodells, damit die Termvektoren zur Projektionsmatrix kompatibel sind.

4.1.5 Indexer

Die Klasse *TfIndexer* ist für die Erstellung eines *TfTermSpaceModel* aus einer Kollektion realer Dokumente zuständig. Dabei wird das Vokabular für das Vektorraummodell mit erzeugt. Als weitere wichtige Funktion leistet der *TfIndexer* im Zusammenspiel mit einem übergebenen Vokabular die konsistente Indexierung von neuen Dokumenten, die einem bestehenden Konzeptraummodell hinzugefügt werden sollen.

4.2 Anwendungsbeispiele

Dieser Abschnitt soll einige konkrete Beispiele zeigen, wie das LVM-Framework zu benutzen ist.

4.2.1 Indexierung einer Dokumentkollektion

Listing 4.1: Indexierung einer Dokumentkollektion

```
1 TfIndexer indexer =
2     new TfIndexer(Stemmer.createStemmer(Locale.ENGLISH),
3                   Locale.ENGLISH, false);
4 for (String s : collection) indexer.addDocument(s);
5 TfTermSpaceModel tf = indexer.indexTf();
```

Die Plaintextdokumente der Kollektion *collection* werden dem Indexer übergeben, die Methode *indexTf()* liefert ein *TfTermSpaceModel*-Objekt. Soll ein bestehendes Vokabular beim Indexing verwendet werden, wird die Methode *indexTf(LvmVocabulary voc)* aufgerufen.

4.2.2 Training der Konzeptraummodelle

LSI

Listing 4.2: Training eines LSI-Modells

```
1 LsiSemanticSpaceModel lsi =
2     LsiSemanticSpaceModel.analyze(TermSpaceModel tm);
3
4 LsiSemanticSpaceModel lsi =
5     LsiSemanticSpaceModel.train(TermSpaceModel tm);
6
7 LsiSemanticSpaceDefinition def = lsi.getDefinition();
```

Das Beispiel zeigt zwei Möglichkeiten, ein LSI-Modell zu trainieren. In beiden Fällen wird eine TDM des übergebenen Termraummodells erstellt und darauf eine Singulärwertzerlegung durchgeführt. Die Methoden unterscheiden sich im Resultat, der Aufruf von *analyze()* liefert ein Konzeptraummodell, in dem die Dokumente des Termraummodells verfügbar sind, der Aufruf von *train()* liefert einen leeren Konzeptraum, dem Dokumente hinzugefügt werden können. In beiden Fällen kann mit *getDefinition()* auf die Konzeptraumdefinition zugegriffen werden.

PLSI

Listing 4.3: Training eines PLSI-Modells

```
1 PlsiSemanticSpaceModel lsi =
2     PlsiSemanticSpaceModel.train(
3         TfTermSpaceModel train,
4         TfTermSpaceModel validate,
5         int restarts,
6         int latentClasses,
7         int maxIterations
8     );
```

Analog zum LSI-Training gibt es die Methoden *analyze()* (mit allen Parametern die auch für *train()* nötig sind und gleichen Resultaten wie bei LSI erklärt) und *getDefinition()*, die hier nicht noch einmal gesondert aufgeführt werden. Das Training eines PLSI-Modells ist deutlich aufwendiger. Der TEM-Algorithmus benötigt zunächst außer der Trainingsmenge auch eine Validationsmenge, beide müssen die selbe Dimension haben (d.h. gleiche Anzahl Dokumente und Terme). Die Validationsmenge ist nötig, um Überanpassung zu erkennen (s. Abschnitt 3.3.5 S. 27). Zum Training muss hier ein tf-gewichtetes Termmodell verwendet werden, die $tfidf$-Gewichte führen bei der Multiplikation mit mehreren Wahrscheinlichkeiten zu einem Unterlauf des Zahlentyps. Dieses Problem könnte mit einer passenden Skalierung umgangen werden. Der Algorithmus ist weiterhin darauf ausgelegt, mehrmals mit zufälligen Werten gestartet zu werden, um vielleicht ein besseres lokales Maximum der Likelihood-Funktion zu finden (s. Abschnitt 3.3.4 S. 3.3.4). Die Anzahl der Restarts wird beim Training angegeben, empfohlen ist ein Wert von mindestens 3. Höhere Werte können zu einem besseren Ergebnis führen, sie verlängern aber die Laufzeit des Trainings erheblich.

Auch die Anzahl der latenten Klassen bzw. Dimensionen des Konzeptraumes muss schon beim Training angegeben werden. Dies unterscheidet das PLSI-Training vom LSI-Training, dort wird ein Modell mit dem vollen Rang der Trainings-Matrix erstellt und die gewünschte Dimension kann beim trainierten Modell gewählt werden. Dieser Aspekt erschwert das Experimentieren mit PLSI, denn um das Verhalten in verschiedenen Dimensionen zu evaluieren muss für jede Dimension einzeln ein Modell trainiert werden.

Die gewünschte Anzahl latenter Klassen beeinflusst zudem die Laufzeit des Trainings, zwar weniger stark als die Anzahl der Restarts, aber dennoch merklich.

Zuletzt muss noch ein Wert für die maximale Anzahl der Iterationen angegeben werden. Prinzipiell stoppt der TEM-Algorithmus abhängig von dem β-Wert (s. Abschnitt 3.3.5), die maximalen Iterationen beeinflussen den Algorithmus dann, wenn der Wert zu gering gewählt ist und der voreingestellte Abbruchswert für β nicht erreicht wird.

Der β-Wert wird mit 1.0 gestartet und beim Auftreten von Überanpassung mit dem Multiplikator 0.92 verringert bis zu einem Minimum von 0.6. Diese Werte sind im PennAspect-Paket vorgegeben und wurden weder modifiziert noch über die Schnittstelle zum Training eines PLSI-Modell exportiert.

Zeitaufwand für das Modelltraining

An dieser Stelle soll eine praktische Betrachtung des Zeitaufwandes beim Training stattfinden. Obwohl beide Algorithmen eine ähnliche Komplexitätsklasse haben (s. Abschnitte 3.2.5 und 3.3.7) zeigen sich starke Unterschiede in der Trainingszeit.

Konzeptmodell	Dimension der Trainingsmatrix	Trainingsdauer
LSI	1000×1000	1m35s
LSI	9998×2129	5h23m36s
PLSI (30)	1000×1000	28m05s
PLSI (40)	1000×1000	36m32s
PLSI (50)	5185×1750	17h31m25s

Tabelle 4.1: Trainingszeit für ausgewählte LSI- und PLSI-Konzeptmodelle. Der Wert in Klammern bei PLSI zeigt die Anzahl der latenten Klassen für die das Modell trainiert wurde. Die Zeiten wurden auf einem aktuellen Rechner gemessen (PIV, 3GHz, 2GB RAM).

Tabelle 4.1 zeigt, dass das LSI-Training für kleine Trainingsmengen recht schnell geht. Viele Experimente wurden mit Modellen durchgeführt die etwa die Ausmaße des ersten Beispiels, 1000×1000 haben. Dies entspricht einer Reuters-Kollektion mit 1000 Dokumenten und einer Dokumentfrequenzreduktion von 20 (s. Tabelle 5.2 S. 5.2). Das zweite LSI-Beispiel zeigt die Trainingszeit für die umfangreichste Spock-Instanz (s. Beschreibung der Experimente in Abschnitt 5.2.4), dieses Training dauert erheblich länger. Obwohl bei den Reuters-Experimenten größere Kollektionen möglich gewesen wären, wurde darauf weitgehend verzichtet um die Ergebnisse mit PLSI vergleichen zu können. Bei PLSI ist schon das Training auf kleinen Trainingsmengen sehr zeitaufwendig. Die Werte für PLSI sind zudem nur für einen Start des Algorithmus gemessen, bei mehreren Restarts muss der entsprechende Faktor hinzugerechnet werden.

4.2.3 Konzeptindexerstellung

Listing 4.4: Konzeptindexerstellung

```
1 LsiSemanticSpaceModel lsi =
2     LsiSemanticSpaceModel.createFromDefinition(
3                                 LsiSemanticSpaceDefinition def);
4 lsi.addTermVectors(tsm);
```

Das Beispiel zeigt, wie ein LSI-Modell aus einer Konzeptraumdefinition erstellt wird. Dieses Modell enthält noch keine Dokumente. Mit der Methode *addTermVectors(TermSpaceModel tsm)* werden die Termvektoren des spezifizierten Termraummodells in den Konzeptraum projiziert. Eine Methode gleichen Namens wird auch vom *PlsiSemanticSpaceModel* angeboten.

4.2.4 Clustering und F-Measure-Berechnung

Die Evaluierung im Konzeptraum geschieht über die Hilfsklasse *ClusterTools*, welche die Methode *getFmeasure(VectorSpaceModel vsm)* anbietet. Diese Methode clustert alle Dokumente im übergebenen Modell, vergleicht das Ergebnis mit einem zuvor per *ClusterTools.setReferenceClustering(Clustering c)* festgelegten Referenzclustering, berechnet das F-Measure und liefert dieses zurück.

5 Experimente

Mit den folgenden Experimenten soll die Leistungsfähigkeit der in Kapitel 3 eingeführten semantischen Indexierungsmethoden als Basis für ein Clustering im Vergleich mit Standardverfahren, d.h. tf- und $tfidf$-gewichtetes Termraummodell, untersucht werden.

5.1 Fragen

Neben der hauptsächlichen Frage, wie performant ein Konzeptmodell im Vergleich mit den Termraumindexierungsverfahren ist, sollen einige weitere Aspekte untersucht werden:

Bei welcher Dimension liefert ein Modell mit latent semantischen Variablen das beste Ergebnis? Die Frage nach der richtigen Dimension und ihrer möglichen Abschätzung im voraus gehört zu den schwierigsten Aspekten bei den latent semantischen Methoden und kann auch hier nicht beantwortet werden. Die optimale Dimension lässt sich nur durch Experimente herausfinden.

Wie gut wird die Information des Termraummodells beim Training eines Konzeptraummodells übernommen und wie wirken sich verschiedene Termgewichtungsmethoden aus? Die Aussagekraft des Termraummodells, mit dem ein Konzeptraummodell trainiert wird, soll in diesem erhalten bleiben. Es ist daher zu vermuten, dass ein stärkeres Termraummodell beim Training auch zu einer besseren Performanz des Konzeptraummodells führt. Es bietet sich aber auch die Möglichkeit, beim Training und der zu projizierenden Testkollektion verschiedene Termgewichtungsverfahren einzusetzen. Lässt sich mit einer solchen Kombination ein besseres Ergebnis erzielen?

Weiterhin soll betrachtet werden, wie sich die Methoden auf Kollektionen mit unterschiedlicher Charakteristik verhalten, beispielsweise auf Kollektionen die sehr ähnliche oder sehr verschiedene Themen abdecken.

5.2 Experimentbeschreibung

Dieser Abschnitt soll dazu dienen, die Experimente nachvollziebar zu beschreiben.

5.2.1 Clustering

Clustering bezeichnet die unüberwachte automatische Kategorisierung einer Menge von Dokumenten. Dies wird als eine spezielle Retrieval-Anwendung angesehen.

Die Evaluation der Methoden beschränken sich im Rahmen dieser Arbeit auf Clusteringexperimente. Obwohl weitere Versuche, besonders hinsichtlich der Fähigkeit zur semantischen Anfrageerweiterung wünschenswert wären, wurde darauf aus zwei Gründen verzichtet. Zum einen wurden solche Experimente schon in den einführenden Arbeiten ([Dee90], [Hof99]) und seitdem in vielen weiteren Arbeiten durchgeführt, die Ergebnisse sind gut dokumentiert. Zum anderen sind die Experimente wegen der Laufzeit der zugrunde liegenden Algorithmen (Singulärwertzerlegung und Expectation Maximization, s. auch Abschnitt 3.2.5 und 3.3.7) sehr zeitaufwendig. Die Anwendung der Konzeptraumindexierung als Grundlage für das Clustering von Dokumentkollektionen wurde hingegen bisher selten untersucht.

5.2.2 Betrachtung hinsichtlich der praktischen Anwendbarkeit

Im Falle von LSI dienen trainierte Modelle als Projektion, der implizit berechnete Konzeptindex der Trainingskollektion wird ignoriert. Über diese Projektion wird für die Testkollektionen ein Konzeptindex erstellt und geclustert. Diese Vorgehensweise wurde gewählt, da sie als einzige praktisch anwendbar erscheint. Als Anwendungsbeispiel sei an dieser Stelle die Dokumentsammlung eines Unternehmens genannt. Es ist viel zu aufwendig, die gesamte Sammlung zu analysieren, ein Modell muss mit einem repräsentativen Auszug trainierbar sein und damit soll ein Konzeptindex für die weiteren Dokumente erstellt werden können. Im Falle von PLSI wurde analog vorgegangen. Die Methode, einen PLSI-Konzeptindex für neue Dokumente zu berechnen (s. Gleichung 3.21 S. 28), wurde jedoch aufgrund des zu hohen Aufwands für den EM-Algorithmus modifiziert, Konzeptgewichte werden hier mittels der im Modelltraining gelernten Wahrscheinlichkeiten direkt berechnet (s. Abschnitt 5.3.3).

5.2.3 Testkollektionen

Die Experimente wurden mit zwei verschiedene Dokumentkollektionen durchgeführt, zum einen dem Reuters-Korpus, zum anderen dem Spock-Trainingskorpus. Die gewähl-

ten Kollektionen adressieren verschiedene problematische Fälle, wie im folgenden erläutert wird.

Reuters

Auf dem Reuters-Korpus wurde ein Großteil der Experimente im Rahmen dieser Arbeit durchgeführt. Es handelt sich dabei um eine kategorisierte Sammlung von Nachrichtenartikeln, die von der Nachrichtenagentur Reuters speziell für die Forschungsarbeit bereitgestellt und häufig in Experimenten referenziert wird. Der Reuters-Korpus wird als Beispiel für eine abgeschlossene Kollektion angesehen. Die Dokumente sind von professionellen Journalisten erstellt, das verwendete Vokabular und die Länge der Artikel entsprechen einem gewissen Standard. Die thematische Einordnung des Inhalts ist festgelegt und soll als Kategorisierungsgrundlage dienen.

Mit dem Reuters-Korpus werden im Rahmen dieser Arbeit vier verschiedene Charakteristika simuliert, die Dokumentkollektionen aufweisen können:

1. Kollektionen mit ähnlicher Anzahl Dokumente in den Themengebieten und verschiedenen Themen. Ein Beispiel wäre eine Sammlung von wissenschaftlichen Dokumenten der Gebiete Informatik, Biologie und Geschichte, wobei für jedes Thema 1000 Dokumente vorliegen.
2. Kollektionen mit unterschiedlicher Anzahl Dokumente in jedem Themengebiet und verschiedenen Themengebieten. Ein Beispiel wären die Sammlung aus Punkt 1 mit 500, 1000 und 1500 Dokumenten in den einzelnen Fachgebieten.
3. Kollektionen mit ähnlicher Dokumentanzahl in den Kategorien und sehr ähnlichen Themen. Ein Beispiel wäre eine Sammlung von Dokumenten über Algorithmentheorie, Maschinelles Lernen, Information Retrieval und Programmiersprachen mit jeweils 1000 Dokumenten.
4. Kollektionen mit unterschiedlicher Dokumentanzahl in den Kategorien und ähnlichen Themen. Ein Beispiel wäre die Kollektion aus Punkt 3 wenn 200, 600, 1200 und 2000 Dokumente in den einzelnen Kategorien vorliegen.

Es ist zu erwarten, dass LSI und PLSI auf diesen Kollektionen eine bessere Performanz gegenüber dem Vektorraummodell mit tf- und $tfidf$-Gewichten liefern (s. Abschnitt 3.5).

Um vergleichbare und allgemeingültige Experimentergebnisse zu erhalten, wurden verschiedene Maßnahmen getroffen.

Für jede der vier o.g. Kollektionscharakteristika wurden aus dem Reuters-Korpus 10 Kategorien gewählt. Die Zusammensetzung der einzelnen Experimente ist in Tabelle 5.1

S. 42 dargestellt. Die Experimente werden der Einfachheit halber mit den Kürzeln exp0, exp1, exp2 und exp3 bezeichnet.

- exp0: verschiedene Kategorien, gleiche Clustergröße
- exp1: ähnliche Kategorien, gleiche Clustergröße
- exp2: verschiedene Kategorien, ungleiche Clustergröße
- exp3: ähnliche Kategorien, ungleiche Clustergröße

Die Experimente exp0 und exp2 enthalten nicht 10 vollkommen verschiedene Kategorien, da der zur Verfügung stehende Teil des Reuters-Korpus nur 4 Hauptkategorien (CCAT, ECAT, GCAT, MCAT) enthält, die wiederum in mehrere Unterkategorien aufgeteilt sind. Die Kollektionen wurden daher mit einer Mischung aus allen Hauptkategorien zusammengestellt. Exp1 und exp3 sind nicht nur aus Unterkategorien einer Hauptkategorie aufgebaut, weil keine Hauptkategorie 10 Unterkategorien mit ausreichend vielen Dokumenten aufweist. Die gewünschten Charakteristika werden dennoch ausreichend gut abgebildet.

Für exp0 und exp1 wurden 100 Dokumente aus jeder Kategorie gezogen, bei exp2 und exp3 lag die Anzahl der Dokumente zwischen 20 und 230. Die genaue Zusammensetzung findet sich im Anhang in Tabelle B.1 S. 83. Insgesamt enthält jede Kollektion 1000 Dokumente.

Experiment	Kategorien
exp0	C11, C21, C33, E11, E71, G15, GSCI, GSPO, M12, M14
exp1	E11, E12, E211, E311, E511, E71, M11, M12, M131, M14
exp2	C11, C21, C33, E11, E71, G15, GSCI, GSPO, M12, M14
exp3	E11, E12, E211, E311, E511, E71, M11, M12, M131, M14

Tabelle 5.1: Kategorieauswahl für die verschiedenen Kollektionscharakteristiken auf dem Reuters-Korpus

Für jedes Experiment (exp0 – exp3) wurden über 20 Testkollektionen (bezeichnet als coll0 – coll20) mit den in Tabelle 5.1 gezeigten Kategorieverteilungen zufällig aus dem Reuters-Korpus zusammengestellt. Das bei den Experimentergebnissen angegebene F-Measure wurde als Durchschnittswert auf je 20 Testkollektionen ermittelt.

Die Experimentkollektionen festzulegen, statt bei jedem Experimentdurchlauf neue Kollektionen zufällig mit der gewünschten Verteilung zusammenzustellen, erwies sich im Laufe der Experimente in mehrerer Hinsicht als sinnvoll. Erstens dauert die Erstellung einer Kollektion eine gewisse Zeit. Zweitens ist es ausreichend ein Referenzergebnis einmal auszurechnen. Drittens erlaubt es einen fairen Vergleich zwischen verschiedenen Experimenten.

Spock Trainingskorpus

Eine weitere Dokumentkollektionscharakteristik wird mit dem Trainingskorpus der *Spock Challenge* betrachtet. Diese Challenge, initiiert im Frühjahr 2007 von den Betreibern der Personensuchmaschine spock.com (*http://www.spock.com*), hatte zur Aufgabe, Webseiten, die Personennamen enthalten, tatsächlichen Personen zuzuordnen. Dieses Problem ist interessant und schwierig, da ein Name kein eindeutiger Hinweis auf eine Person ist. Zur Unterscheidung mehrerer Personen gleichen Namens müssen Dokumentmodelle entwickelt werden, die zusätzliche Information aus den Dokumenten für einen Vergleich verfügbar machen. Die *Spock Challenge* konnte ein Forschungsteam der Bauhaus-Universität Weimar für sich entscheiden.

Der Trainingskorpus umfasst 25.000 HTML-Dokumente und wurde mit der korrekten Kategorisierungsinformation geliefert. Er wird im Rahmen der folgenden Experimente verwendet, da er einen kategorisierten Auszug des Web darstellt (die Bezeichnung "repräsentativer Querschnitt" ist im Kontext des WWW eher unangebracht). Der Fokus liegt darauf, die Dokumente Personen zuzuordnen und aufgrund der Dokumentinformationen verschiedene Personen mit gleichem Namen korrekt auseinanderzuhalten (*person resolution problem*), eine vollkommen andere Aufgabe also, als die thematische Kategorisierung bei den Reuters-Experimenten.

Auf dieser Kollektion ist ein schlechtes Resultat der Konzeptraummodelle zu erwarten, da, wie schon zitiert (s. Abschnitt 3.5), die Anwendung auf solchen heterogenen Kollektionen sehr schwierig ist.

5.2.4 Experimentablauf

Reuters-Experimente

Alle Experimente folgen einem festen Ablauf. Es wird vorausgesetzt, dass für die Testkollektionen eine korrekte Kategorisierung, also ein Referenzclustering, bekannt ist, welches zum Vergleich mit dem berechneten Clustering herangezogen wird.

1. Die Dokumente einer Testkollektion werden indexiert und zu einer TDM zusammengefasst.
2. Für die Dokumente wird ein Konzeptraumindex erstellt.
3. Auf Basis der Ähnlichkeitsinformation im Konzeptraum wird ein Ähnlichkeitsgraph erstellt. Als Maß dient die Cosinusähnlichkeit (s. Abschnitt 2.3).
4. Der Ähnlichkeitsgraph wird mit MajorClust geclustert.

5. Das Resultat wird mit dem Referenzclustering verglichen und auf dieser Basis das F-Measure berechnet.

Die F-Measure-Berechnung erfolgt mit F_1 (s. Abschnitt 2.3.3), Precision und Recall gehen zu gleichen Teilen in das Ergebnis ein.

Spock-Experiment

Die Aufgabe der Spock-Challenge wurde von der Spock-Projektgruppe im SS2007 gelöst, indem der gesamte Korpus in Gruppen von Dokumenten aufgeteilt wurde, die den jeweils gleichen Namen enthalten. Eine solche Dokumentgruppe wurde als Spock-*Instanz* bezeichnet. Jede Instanz wurde einzeln analysiert und geclustert. Die Ergebnisse wurden zusammengefasst und das F-Measure mit einem Evaluationsskript und einem Referenzclustering, welche der Trainingskollektion beilagen, bestimmt.

Analog wurde im Rahmen dieser Arbeit vorgegangen. Für jede Instanz wurde ein Konzeptraumindex berechnet und ein Clustering erstellt. Das F-Measure wird auf die gleiche Weise bestimmt wie im Spock-Projekt, es ist hier mit dem Faktor $\alpha = \frac{1}{3}$ gewichtet, Precision wird also gegenüber Recall bevorzugt. Dies ist begründet durch die Vorgabe von spock.com und der Vergleichbarkeit zu den Ergebnissen des Forschungsprojekts.

Der Spock-Trainingskorpus enthält 44 Instanzen, die über 1000 verschiedene reale Personen referenzieren. An Dokumenten findet sich darin, einfach gesagt, fast alles was das Web zu bieten hat: Firmenseiten, Seiten öffentlicher und wissenschaftlicher Institutionen, Private Seiten, Onlineshops, Blogs, Foren etc. Um ein LSI-Modell für den Spock-Korpus zu trainieren, musste eine repräsentative Auswahl getroffen werden. Hierfür wurde die umfangreichsten Instanz gewählt. Dieser Vorgehensweise lag die Annahme zugrunde, dass sich in dieser Instanz, die mit 2129 Dokumenten ca. 11,7% der Gesamtdokumente abdeckt, die Verteilung des gesamten Korpus hinreichend genau widerspiegelt. Aus dieser Instanz wurde eine Projektion erstellt, die im Experiment zur Konzeptindexberechnung für alle Instanzen herangezogen wurde.

5.2.5 Experimentparameter

Als Clusteringalgorithmus kommt in allen Experimenten *MajorClust* zum Einsatz ([Ste99]). Zur Optimierung des Clustering wird das interne Cluster-Validity-Measure *Expected Density* eingesetzt ([Ste03]). MajorClust ist kein deterministischer Algorithmus, das Resultat kann von Experiment zu Experiment auf den gleichen Testkollektionen leicht abweichen. Diese Eigenschaft lässt sich aber vernachlässigen, da jeweils das Ergebnis mehrerer Tests gemittelt wird.

In den Experimenten wurden verschiedene Parameter variiert um zu evaluieren, wie sich optimale Ergebnisse erzielen lassen. Diese sollen im folgenden erläutert werden.

Vokabularreduktion

Beim Erstellen der TDM wurden Stemming (Snowball-Stemmer) und Stopwordelimination angewendet (vgl. Abschnitt 2.2). Dennoch entstehen beim Indexing sehr viele Terme (s. Tabelle 5.2 S. 46) und der Aufwand zur Analyse der resultierenden TDM wäre zu hoch. Deshalb wurde die Dimension der Dokumentvektoren bereits in einem Vorbereitungsschritt reduziert, indem ein Dokumentfrequenz (df)-Schwellwert vorgegeben und alle Terme mit geringerer df entfernt wurden. Das heißt, aus dem Vokabular wurden seltene Terme entfernt.

Die Anwendung dieser Reduktion ist hinsichtlich des möglichen Informationsverlusts im Zusammenhang mit den semantischen Analysemethoden problematisch, da zu erwarten wäre, dass diese von einem reichhaltigen Termraum profitieren. Theoretisch bedeutet die Reduktion auch einen Informationsverlust für die Standardmethoden, denn seltene Terme haben ein hohes Diskriminierungspotential. Die Referenzergebnisse (Abschnitt 5.3.1) zeigen aber, dass sich durch diesen Reduktionsschritt im Vektorraummodell Performanzvorteile ergeben. Zu erklären ist dieser Effekt mit dem reduzierten Rauschen in den Daten.

Ein zu hoher Reduktionswert sollte vermieden werden, weil damit Dokumente, die nur aus seltenen Termen bestehen, verloren gehen. Die Grenze lag auf den untersuchten Kollektionen bei einer df-Reduktion von 30. Die meisten Experimente wurden mit einer Reduktionsstufe von 20 durchgeführt.

Es wurde ebenfalls der Ansatz verfolgt sehr häufige Terme zu entfernen. Die Idee war, zusätzlich zu den schon während des Indexing entfernten Stopworten weitere Terme mit geringer Diskriminationskraft zu entfernen. Das Ergebnis wurde jedoch in allen Fällen schlechter. Eine Erklärung dafür wäre, dass Konzeptraummethoden aus häufigen Termen starke Konzepte extrahieren können. Bei den Referenzergebnissen verhindert die $tfidf$-Gewichtung, dass häufige Terme einen zu starken Einfluss haben.

Tabelle 5.2 zeigt die Anzahl der indexierten Terme auf den Testkollektionen unreduziert und mit den Reduktionsstufen 10, 20 und 30. Im Falle der Reuters-Experimente exp0 – exp3 sind die Werte gemittelt über 20 Kollektionen, für Spock ist exemplarisch die Termanzahl der umfangreichsten Instanz aufgezeigt.

Die Werte sind grafisch dargestellt in der Abbildung A.1 S. 70. Es ist zu sehen, das schon die Reduktion der Terme mit geringerer df als 10 die Größe des Vokabulars deutlich verringert. Trotz des Informationsverlustes steigt die Retrievalperformanz.

Experiment	Dokumente	Indexterme	Indexterme nach Reduktion		
			df 10	*df* 20	*df* 30
exp0	1000	14105	1900	1035	694
exp1	1000	9593	1441	832	568
exp2	1000	12750	1697	922	621
exp3	1000	8811	1344	782	534
Spock (Instanz 3)	2129	220804	17357	9998	7258

Tabelle 5.2: Durchschnittliche Termanzahl nach Indexierung der Experimentkollektionen und mit verschiedenen Reduktionsstufen

Die Tabelle 5.2 zeigt auch die Charakteristik der Kollektionen. Die aus ähnlichen Themengebieten zusammengestellten Reuters-Experimente exp1 und exp3 weisen ein deutlich kleineres Vokabular auf als die Kollektionen aus verschiedenen Themengebieten. Die Spock-Instanz weist im Gegensatz zu den Reuters-Kollektionen ein sehr umfangreiches Vokabular auf, welches schon auf der ersten Reduktionsstufe deutlich stärker reduziert wird als die Reuters-Vokabulare. Diese Charakteristik kann als typisch für Web-Dokumente angesehen werden, die Daten enthalten sehr viel Rauschen und sind schwerer zu verarbeiten.

Dimension bzw. Anzahl latenter Klassen

Dieser Parameter ist besonders bei LSI für das Resultat ausschlaggebend. Es wurden viele Experimente in verschiedenen Dimensionen ausgewertet um einen optimalen Wert zu finden. Interessant wäre eine Methode, die es erlaubt die optimale Dimension aus der Beschaffenheit der Kollektion oder aus dem Ergebnis der Analyse zu ermitteln. Eine solche Methode ist jedoch bisher nicht bekannt. Dies ist einer der schwierigsten Punkte im Umgang mit latent semantischen Retrievalverfahren, denn mit der falschen Einstellung kann man schlechte Ergebnisse erhalten, wie die Experimente zeigen.

Schwellwert für Ähnlichkeitsgraph (S)

Dies ist eine globale Optimierungsmethode für den Ähnlichkeitsgraph. Beim Erstellen des Graphen werden Werte unterhalb eines bestimmten Schwellwerts ignoriert. Damit wird das Rauschen in den Daten minimiert und der Clusteringalgorithmus unterstützt und beschleunigt. Um herauszufinden, wo dieser Wert anzusetzen ist, wird die Ähnlichkeitswertverteilung auf einer Kollektion untersucht. Problematisch dabei ist jedoch, dass damit Knoten isoliert werden können, wenn alle Kanten unterhalb des Schwellwerts liegen. Diese Knoten würden beim Clustering ignoriert werden und somit die Performanz senken. Der Schwellwert wird bei der Beschreibung der Experimente mit S bezeichnet.

KNNGraph (KNN)

Auch diese Methode ist eine Optimierung des Ähnlichkeitsgraphs. Für jeden Knoten im Graph werden die stärksten Kanten behalten, unabhängig von ihrem Gewicht. Die Anzahl der zu behaltenden Kanten wird als Parameter angegeben. Im Gegensatz zum globalen Schwellwert ist diese Methode besser, da sie eine lokale Optimierung vornimmt. Der gewählte Wert wird bei den Experimenten mit KNN bezeichnet.

5.2.6 Training der Konzeptraummodelle

Für jedes Reuters-Experiment wurden Kollektionen der in Abschnitt 5.2.3 genannten Charakteristiken erstellt und daraus Term-Dokument-Matritzen berechnet. Es wurden Varianten mit tf- und $tfidf$-Gewichten untersucht. Die TDM wurden einer df-Reduktion unterzogen, um die Rechenzeit beim Training zu verringern.

Für das Spock-Experiment wurde wie in Abschnitt 5.2.4 beschrieben die umfangreichste Instanz zum Training eines Modells verwendet. Die Erfahrungen aus den Reuters-Experimenten fanden hier Anwendung, so dass weniger mit verschiedenen Parametern experimentiert wurde.

Erste LSI-Experimente wurden mit Modellen durchgeführt, die mit weniger Dokumenten (500), aber auch geringerer df-Reduktion (5) trainiert wurden. Es stellte sich heraus, dass Modelle, die mit größeren Trainingsmengen und höherer df-Reduktionsrate trainiert wurden, zu besseren Ergebnissen führten. Der Grund hierfür ist wahrscheinlich in der zu geringen Anzahl an Dokumenten zu suchen. Die Trainingskollektion sollte genug Dokumente enthalten, um alle in der gesamten Kollektion vorkommenden wichtigen Konzepte abzubilden. Vor diesem Hintergrund stellt sich die Frage, wie viele Dokumente zum trainieren eines LSI-Modells für eine Kollektion einer bestimmten Größe nötig sind.

Für PLSI-Modelle wurde ebenso vorgegangen. Da jedoch die Laufzeit des Trainings wesentlich höher ist (s. Tabelle 4.1 S. 37) und zudem für jede zu untersuchende Dimension ein einzelnes Modell trainiert werden muss, konnten weniger PLSI-Experimente durchgeführt werden.

5.3 Experimentergebnisse

Dieser Abschnitt zeigt die Resultate, die bei der Evaluation der verschiedenen Modelle erzielt wurden.

5.3.1 Ergebnisse der Reuters-Experimente

Referenzergebnisse Vektorraummodell

Die Abbildungen 5.1 und 5.2 zeigen die besten Ergebnisse, die sich mit tf- und $tfidf$-gewichteten Vektorraummodellen bei entsprechender Parametrisierung erzielen liessen.

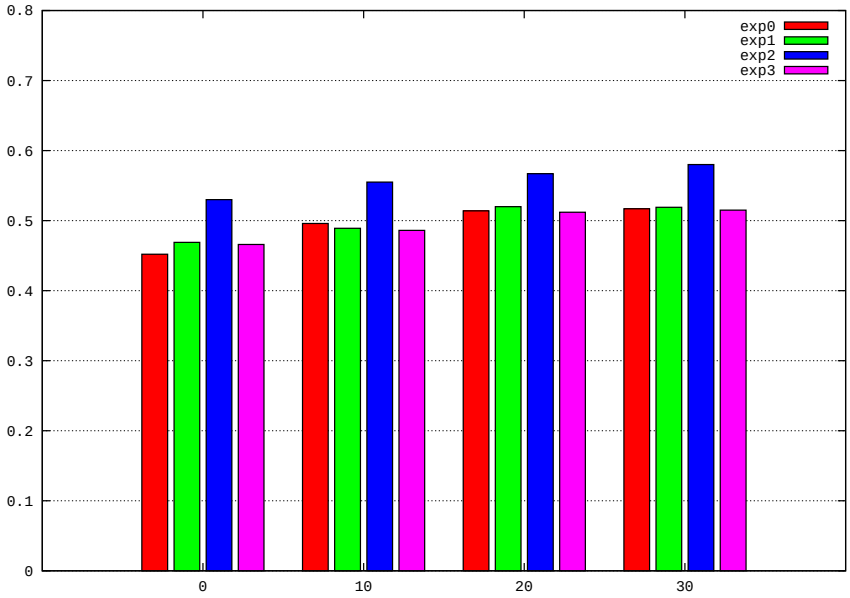

Abbildung 5.1: F-Measures (y-Achse) für Reuters-Kollektionen beim Clustering eines Vektorraummodells mit tf-gewichteten Dokumentvektoren, S 0.3, KNN 50 in den df-Reduktionsstufen 0 – 30 (x-Achse)

Interessant ist die Feststellung, dass die $tfidf$-Gewichtung nicht signifikant bessere Ergebnisse liefert als tf, bei den Kollektionen mit ähnlichen Themen (exp1 und exp3) ist das Resultat sogar deutlich schlechter. Beide Verfahren profitieren von der Reduktion der Terme mit geringer Dokumentfrequenz, $tfidf$ sogar sehr deutlich. In beiden Fällen wurde das beste Resultat mit einem KNN-Wert von 50 erzielt. Der globale Ähnlichkeitsgraph-Schwellwert S liefert für tf mit einem recht hohen Wert von 0.3 ein optimales Ergebnis, für $tfidf$ mit einem geringeren Wert von 0.1. Ein Blick auf die Verteilung der Ähnlichkeitswerte im Vektorraummodell in Abschnitt A.2 S. 71 zeigt, dass

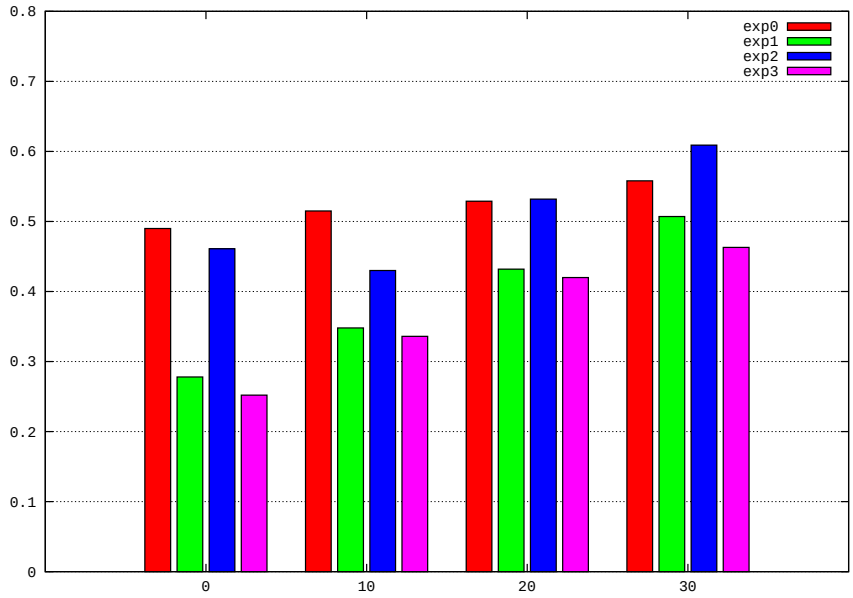

Abbildung 5.2: F-Measures (y-Achse) für Reuters-Kollektionen beim Clustering eines Vektorraummodells mit $tfidf$-gewichteten Dokumentvektoren, S 0.1, KNN 50 in den df-Reduktionsstufen 0 – 30 (x-Achse)

sich bei $tfidf$-Gewichtung die Ähnlichkeiten sehr viel stärker in geringeren Bereichen konzentrieren, das erklärt die Wahl des optimalen Schwellwertes.

5.3.2 Ergebnisse LSI

In den Experimenten mit LSI wird untersucht, wie sich die Wahl der Gewichtung der Trainingsmenge und der Gewichtung der projizierten Dokumentvektoren auswirkt und welche Dimension ein optimales Ergebnis liefert. Die LSI Modelle wurden mit tf- und $tfidf$-gewichteten TDM trainiert, jeweils mit einer df-Reduktion von 20. Die Testkollektionen wurden dann ebenfalls mit tf und $tfidf$ gewichtet und in den Konzeptraum projiziert.

Zunächst wird die Verteilung der Ähnlichkeitswerte im LSI-Konzeptraum in verschiedenen Dimensionen analysiert, s. die Abbildungen in Abschnitt A.3 S. 73. Dabei ist auffällig, dass negative Cosinusähnlichkeiten auftreten. Dies ist begründet durch die Singulärwertzerlegung, bei der negative Einträge in der Projektionsmatrix entstehen.

Weiterhin ist interessant, dass sich in geringen Dimensionen die Masse der Ähnlichkeitswerte bei dem Wert 1 konzentriert. Der Effekt ist deutlich stärker, wenn das LSI-Modell mit einer tf-gewichteten TDM trainiert wird. Bei höheren Dimensionen stellt sich eine Verteilung um den Wert 0 ein.

Weiterhin wird die Verteilung der Singulärwerte untersucht, s. die Abildungen in Abschnitt A.5 S. 81. Hier fällt auf, dass sich die Singulärwerte bei einem mit tf-Gewichten trainierten LSI-Modell deutlich von einem mit $tfidf$-Gewichten trainierten Modell unterscheiden. Bei der Singulärwertverteilung lässt sich auch untersuchen, ob ein Zusammenhang zwischen den Werten und der optimalen Dimension besteht. Die Idee ist, dass die Singulärwerte ein Gewicht der Konzepte repräsentieren. Die richtige Dimension könnte dann an einer markanten Stelle in der Verteilungskurve liegen, beispielsweise wo die Singulärwerte einen bestimmten Wert unterschreiten, der Unterschied eines Wertes zum nächsten nicht mehr signifikant ist oder wo der Abfall der Kurve am stärksten ist. Beim Vergleich der Singulärwertverteilungen mit den Clusteringergebnissen (s. Abschnitt 5.3.2) lässt sich jedoch keine dieser Hypothesen bestätigen. Allenfalls liegt der Bereich, in dem die Verteilungskurve nicht mehr signifikant fällt in der Nähe der optimalen Dimension.

Graphparameter bei LSI-Experimenten

Verschiedene Versuche mit Variationen des Ähnlichkeitsschwellwertes S im Bereich 0.1 bis 0.5 haben ergeben, das dieser praktisch keine Auswirkungen auf das Ergebnis hat. Ein solches Resultat ist auch bei der Betrachtung der Cosinusähnlichkeitsverteilung im LSI-Konzeptraum schon absehbar (s. Abschnitt A.3). Für den KNN-Parameter wurde ein Optimum bei 50 ermittelt.

Kollektionen mit 10 Themen

Die Abbildungen 5.3 – 5.6 zeigen die Ergebnisse der LSI-Experimente. Dargestellt sind jeweils die Resultate in den Dimensionen 5 – 40. Hier sind mehrere interessante Feststellungen zu treffen:

Allgemein ist eine sehr niedrige Dimension ausreichend um sehr gute Ergebnisse zu erzielen. Dabei ist entscheidend, wie die Trainingsmenge gewichtet ist. Bei einem Modell, das mit tf-Gewichten trainiert wurde, liegt die optimale Dimension bei 20 – 25, bei einem Modell das mit $tfidf$ trainiert wurde, liegt sie sogar bei 10, was der tatsächlichen Anzahl der Themen entspricht. Die Eigenschaft, dass mit $tfidf$ trainierte Modelle schon in geringeren Dimensionen bessere Ergebnisse liefern, scheint typisch zu sein. Die Gewichtung der Testdaten hat sehr viel weniger Einfluss auf das Ergebnis als die Gewichtung der Trainingsmenge.

Die Resultate in der optimalen Dimension sind, verglichen mit den Referenzergebnissen des Vektorraummodells (Abschnitt 5.3.1 S. 48), signifikant besser.

Die mit $tfidf$ trainierten Modelle liefern bessere Resultate als die tf-trainierten. Dies spricht dafür, das im LSI-Modell die Aussagekraft des Vektorraummodells erhalten

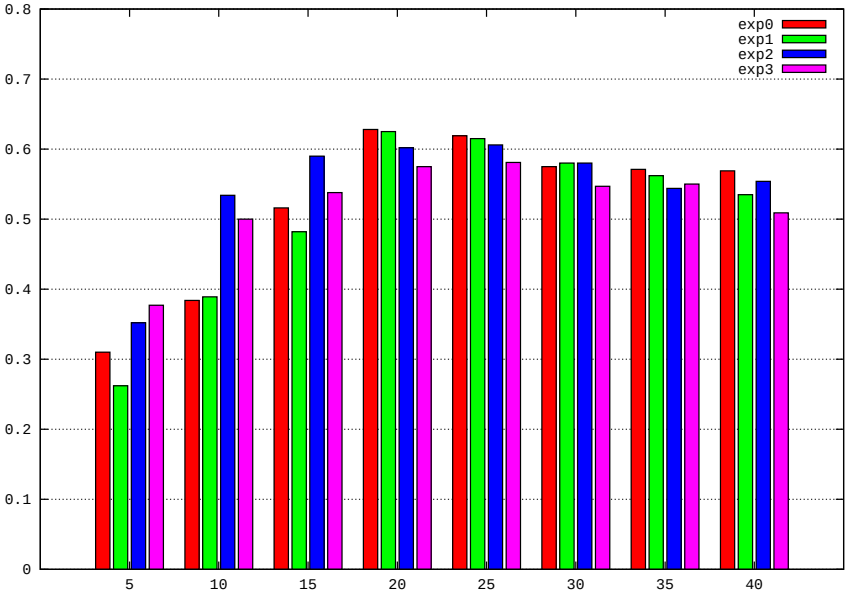

Abbildung 5.3: F-Measures (y-Achse) für Reuters-Kollektionen beim Clustering eines LSI-Modells, trainiert mit einer tf-gewichteten TDM. Die Testkollektionen wurden ebenfalls mit tf gewichtet. Die x-Achse zeigt die Dimensionen.

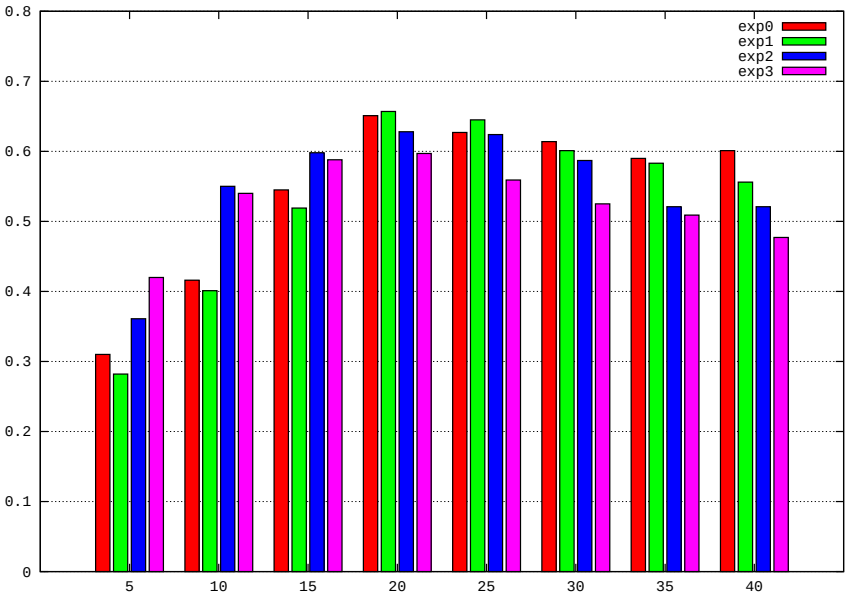

Abbildung 5.4: F-Measures (y-Achse) für Reuters-Kollektionen beim Clustering eines LSI-Modells, trainiert mit einer tf-gewichteten TDM. Die Testkollektionen wurden mit $tfidf$ gewichtet. Die x-Achse zeigt die Dimensionen.

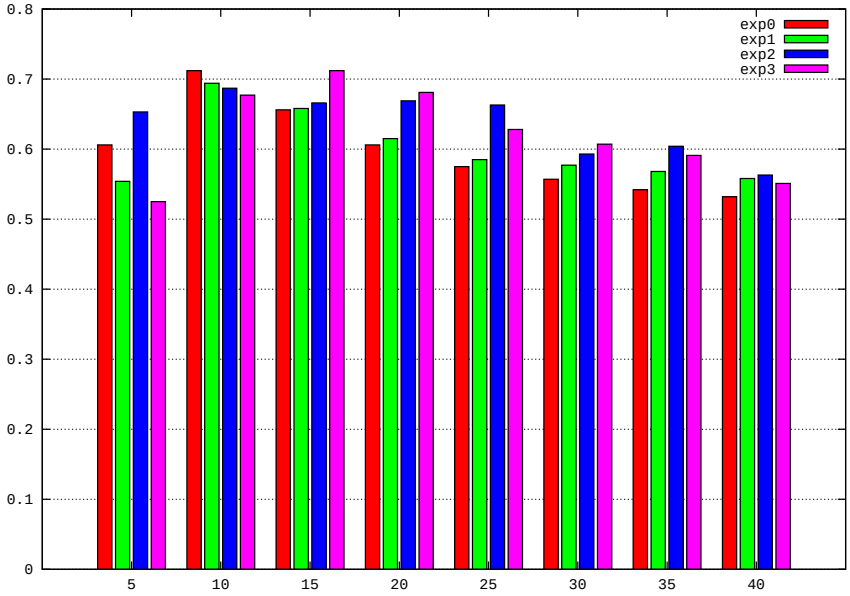

Abbildung 5.5: F-Measures (y-Achse) für Reuters-Kollektionen beim Clustering eines LSI-Modells, trainiert mit einer $tfidf$-gewichteten TDM. Die Testkollektionen wurden mit tf gewichtet. Die x-Achse zeigt die Dimensionen.

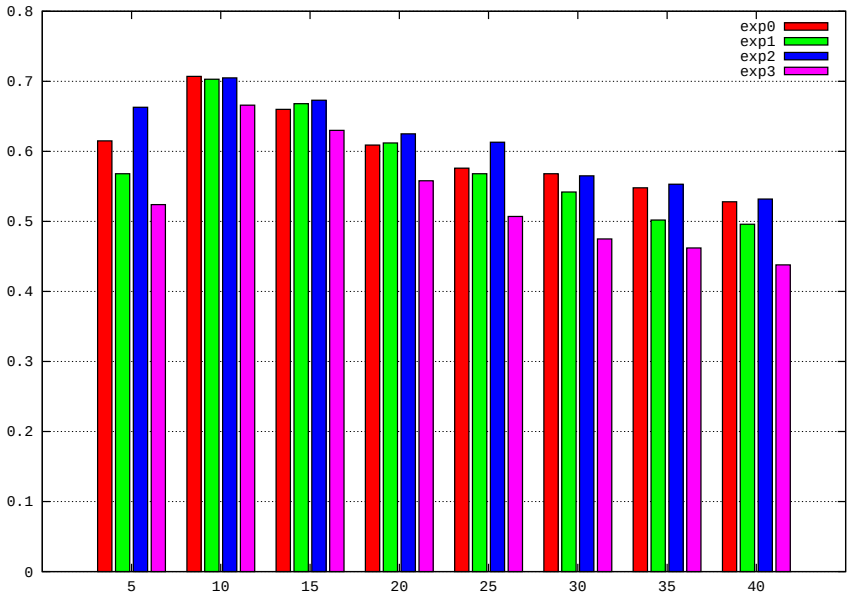

Abbildung 5.6: F-Measures (y-Achse) für Reuters-Kollektionen beim Clustering eines LSI-Modells, trainiert mit einer $tfidf$-gewichteten TDM. Die Testkollektionen wurden mit $tfidf$ gewichtet. Die x-Achse zeigt die Dimensionen.

bleibt. Das durchschnittlich beste Ergebnis liefert eine Kombination aus $tfidf$-trainiertem Modell und tf-gewichteten Testdaten. Dies weist darauf hin, dass die Stärken zweier Gewichtungsmodelle kombiniert werden können.

In zu geringen Dimensionen ist das Ergebnis erwartungsgemäß schlecht, es steht nicht genug Information zur Verfügung. Auch in höheren Dimensionen werden die Ergebnisse wieder schlechter. Dies könnte ein Hinweis darauf sein, dass mit dem Hinzunehmen der höheren Dimensionen die Daten stärker verrauscht werden oder aber, dass das Modell auf die Trainingskollektion überangepasst ist. Daraus könnte man folgern, dass in den geringeren Dimensionen die generellen Konzepte codiert sind, während in den höheren Dimensionen spezialisierte Konzepte hinzukommen, die das Ergebnis auf Daten außerhalb der Trainingsmenge verschlechtern.

Um herauszufinden ob eine geringere Reduktion der Trainingsdaten die Ergebnisse beeinflusst, wurde ein LSI-Modell mit einer $tfidf$-gewichteten TDM und df-Reduktion 10 trainiert. Das Ergebnis ist in Abbildung 5.7 S. 53 zu sehen. Es unterscheidet sich nicht signifikant von dem vergleichbaren Ergebnis eines mit df-Reduktion 20 trainierten Modells (Abb. 5.5 S. 52).

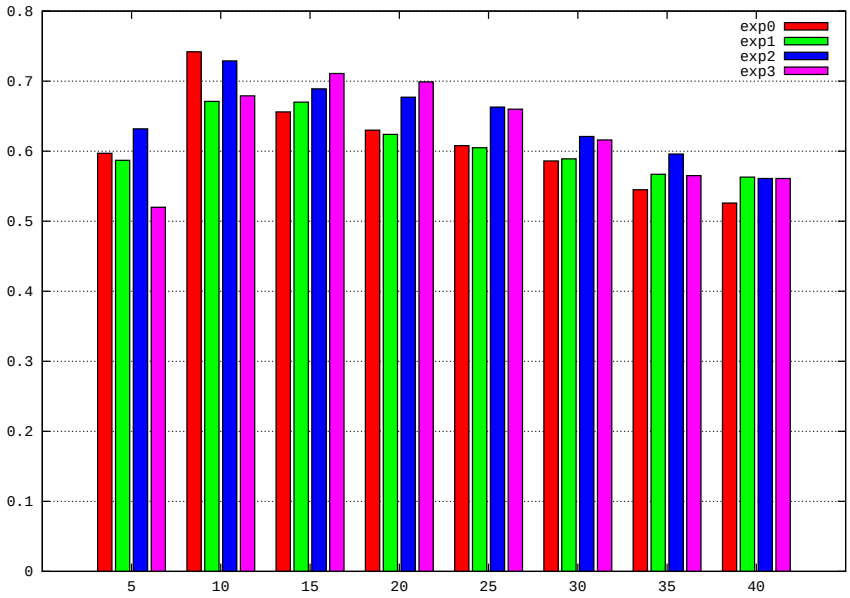

Abbildung 5.7: F-Measures (y-Achse) für Reuters-Kollektionen beim Clustering eines LSI-Modells, trainiert mit einer $tfidf$-gewichteten TDM mit df-Reduktion 10. Die Testkollektionen wurden mit tf gewichtet. Die x-Achse zeigt die Dimensionen.

Umfangreichere Kollektionen

Um zu untersuchen, wie die auf 1000 Dokumenten trainierten LSI-Modelle auf umfangreichere Kollektionen skalieren, wurden mit der bekannten Verteilung Kollektionen von 2000 und 3000 Dokumenten erstellt. Die ebenfalls bekannten, auf 1000 Dokumenten basierenden LSI-Modelle, wurden genutzt um einen Konzeptraumindex zu erstellen und die größeren Kollektionen zu clustern.

Die Abbildungen 5.8 und 5.9 zeigen die Referenzergebnisse für die Kollektionen mit 2000 und 3000 Dokumenten. Erwartungsgemäß werden die Ergebnisse mit einer höheren Dokumentanzahl schwächer.

Die Abbildungen 5.10 und 5.11 zeigen, dass das LSI-Modell auf umfangreicheren Kollektionen gute Ergebnisse liefert. Dies ist für die praktische Anwendbarkeit von entscheidender Bedeutung. Auch die Verschlechterung des Ergebnisses in höheren Dimensionen ist hier nicht so stark ausgeprägt, was gegen die o.g. Theorie der Überanpassung spricht.

Auch für die Experimente mit umfangreicheren Kollektionen wurde getestet, ob Trainingsmengen mit geringerer *df*-Reduktion bessere Ergebnisse liefern. Die Ergebnisse unterschieden sich jedoch nicht signifikant von den hier gezeigten Ergebnissen und werden daher nicht extra aufgeführt.

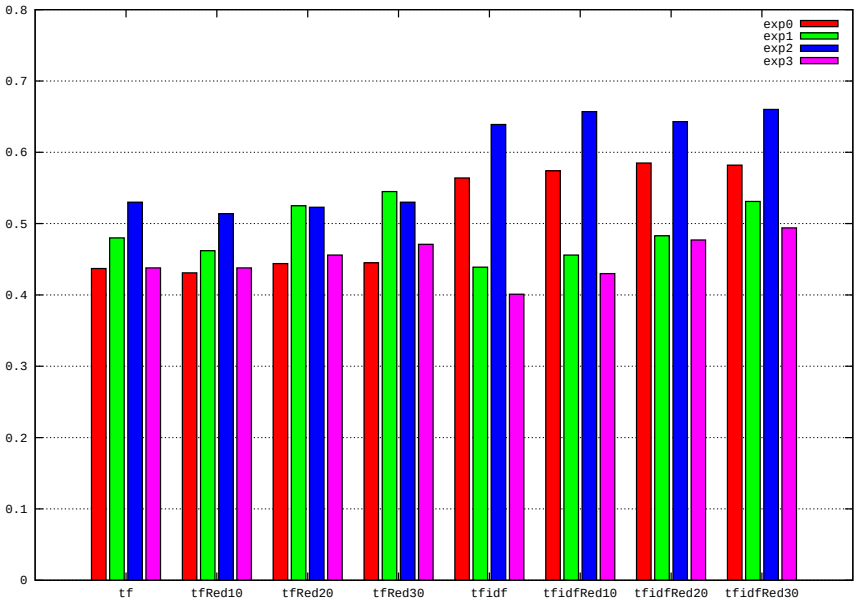

Abbildung 5.8: F-Measures (y-Achse) für Reuters-Koll. mit 2000 Dok. beim Clustering eines Vektorraummodells mit tf (links) und $tfidf$ (rechts)-gewichteten Dokumentvektoren, S 0.3, KNN 50, jeweils in den *df*-Reduktionsstufen 0 – 30 (x-Achse)

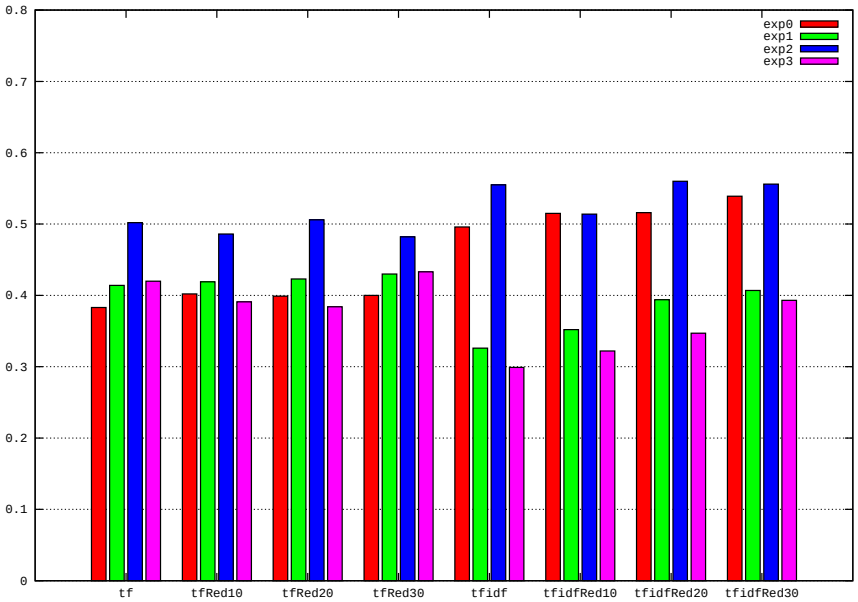

Abbildung 5.9: F-Measures (y-Achse) für Reuters-Koll. mit 3000 Dok. beim Clustering eines Vektorraummodells mit tf (links) und $tfidf$ (rechts)-gewichteten Dokumentvektoren, S 0.3, KNN 50, jeweils in den df-Reduktionsstufen 0 – 30 (x-Achse)

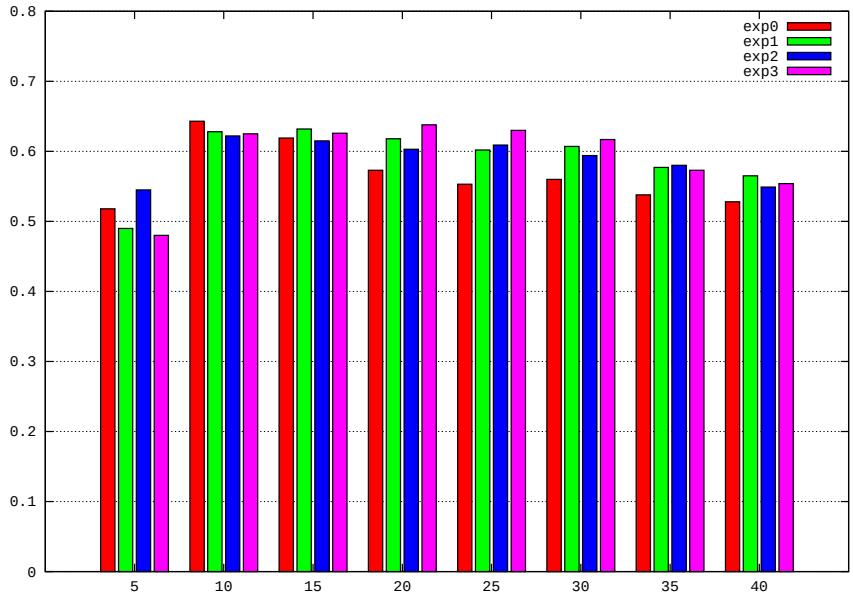

Abbildung 5.10: F-Measures (y-Achse) für Reuters-Koll. mit 2000 Dok. beim Clustering eines LSI-Modells, trainiert mit $tfidf$-Gewichten mit df-Reduktion 20. Die Testkollektionen wurden mit tf gewichtet. Die x-Achse zeigt die Dimensionen.

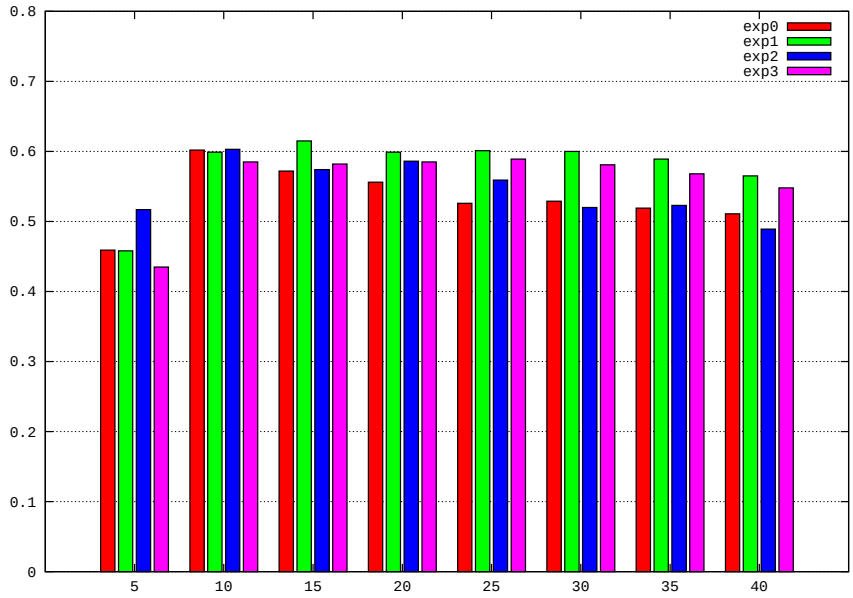

Abbildung 5.11: F-Measures (y-Achse) für Reuters-Koll. mit 3000 Dok. beim Clustering eines LSI-Modells, trainiert mit $tfidf$-Gewichten mit df-Reduktion 20. Die Testkollektionen wurden mit tf gewichtet. Die x-Achse zeigt die Dimensionen.

Kollektionen mit 20 Themen

Um zu untersuchen, wie gut LSI auf Kollektionen mit größerer Bandbreite an Themen arbeitet, wurden die Versuche mit LSI auf Reuters-Kollektionen mit 20 ausgewählten Themengebieten ausgeweitet. Es werden die gleichen vier Kollektionscharakteristika untersucht wie bei den 10-Themen-Experimenten. Wie bei den vorherigen Experimenten wurden Testkollektionen mit 1000 Dokumenten erstellt und davon unabhängig LSI-Modelle mit ebenfalls 1000 Dokumenten und der gleichen Themencharakteristik trainiert.

Die Abbildungen 5.12 und 5.13 zeigen die Referenzergebnisse. Die Resultate sind erwartungsgemäß schwächer alls bei 10 Themen. Die $tfidf$-Gewichte zeigen hier in allen Reduktionsstufen leichte Vorteile gegenüber tf.

Wie bei den Experimenten mit 10 Themen wurden allen Kombinationen aus Gewichtung der Trainingsmenge und der Testmenge evaluiert. Die Abbildungen 5.14 und 5.15 zeigen die Ergebnisse einmal mit tf-gewichteter Trainingsmenge und einmal mit $tfidf$-gewichteter Trainingsmenge, in beiden Fällen wurden tf-gewichtete Dokumentvektoren projiziert. Auf die Darstellung der Ergebnisse $tfidf$-gewichteter projizierter Dokumentvektoren wird verzeichtet, da sie sich nicht signifikant unterscheiden.

Auffällig ist hier, dass die Ergebnisse des tf-trainierten LSI Modells sich kaum von den Referenz-F-Measures unterscheiden, die Ergebnisse des $tfidf$-trainierten Modells aber

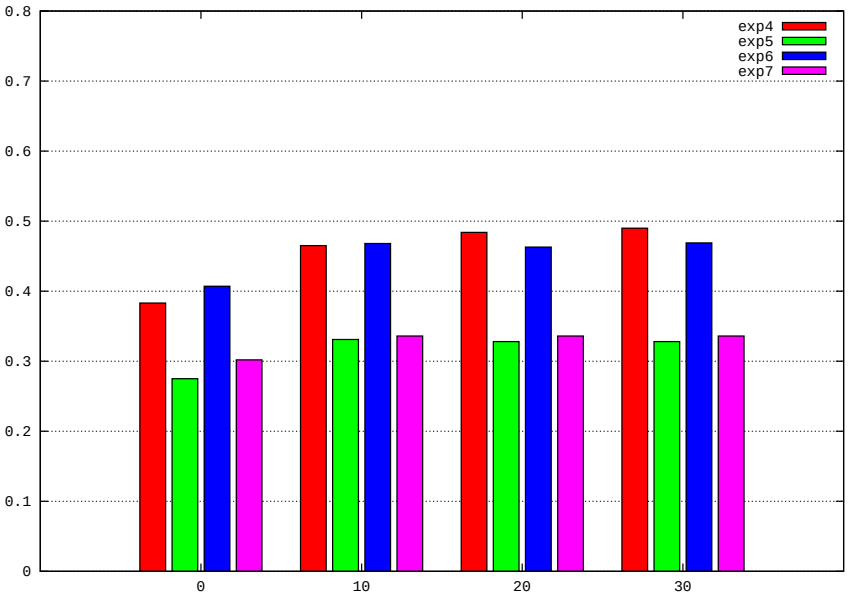

Abbildung 5.12: F-Measures (y-Achse) für Reuters-Koll. mit 20 Themen beim Clustering eines Vektorraummodells mit tf-gewichteten Dokumentvektoren, S 0.3, KNN 25 in den df-Reduktionsstufen 0 – 30 (x-Achse)

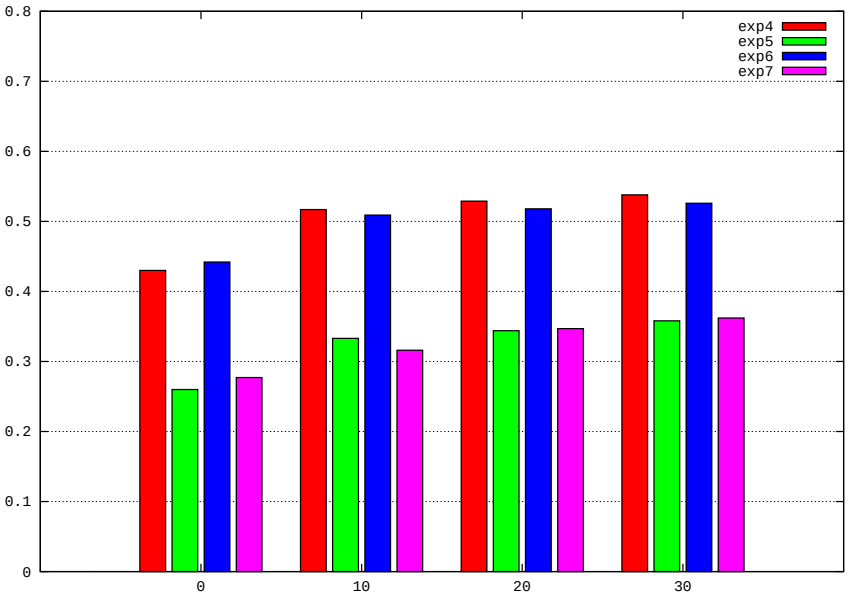

Abbildung 5.13: F-Measures (y-Achse) für Reuters-Koll. mit 20 Themen beim Clustering eines Vektorraummodells mit $tfidf$-gewichteten Dokumentvektoren, S 0.1, KNN 50 in den df-Reduktionsstufen 0 – 30 (x-Achse)

deutlich besser sind. Die schon in den vorherigen Beispielen gezeigte Charakteristik, dass $tfidf$-trainierte Modelle schon bei geringeren Dimensionen starke Ergebnisse liefern, bestätigt sich. Die Verschlechterung der Ergebnisse in höheren Dimensionen ist hier deutlich schwächer ausgeprägt. Weitere Experimente bis Dimension 100 haben ergeben, dass die Resultate nur sehr langsam schlechter werden. Der Bereich an Dimensionen in dem eine gute Retrievalperformanz möglich ist, ist also deutlich größer. Es gibt jedoch weiterhin eine optimale Dimension.

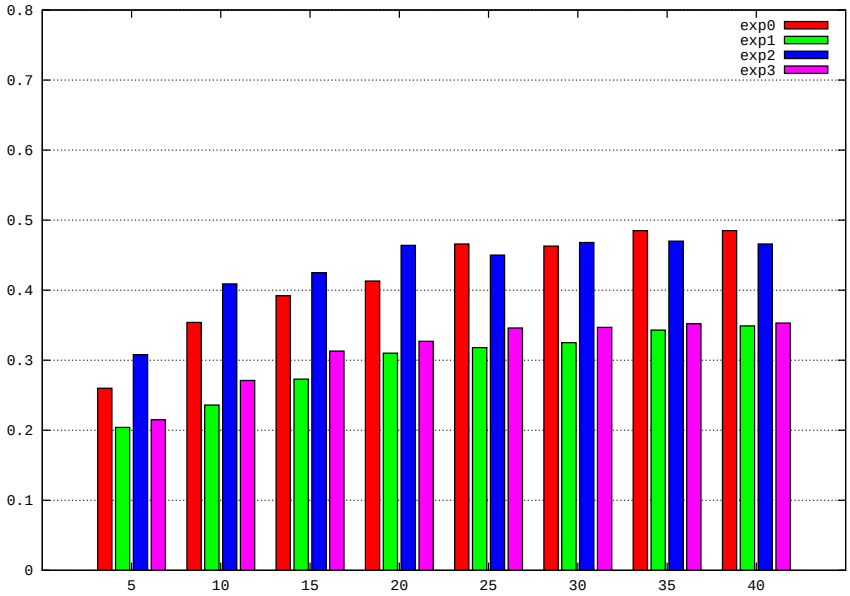

Abbildung 5.14: F-Measures (y-Achse) für Reuters-Koll. mit 20 Themen beim Clustering eines LSI-Modells trainiert mit einer tf-gewichteten TDM. Die Testkollektionen wurden mit tf gewichtet. Die x-Achse zeigt die Dimensionen.

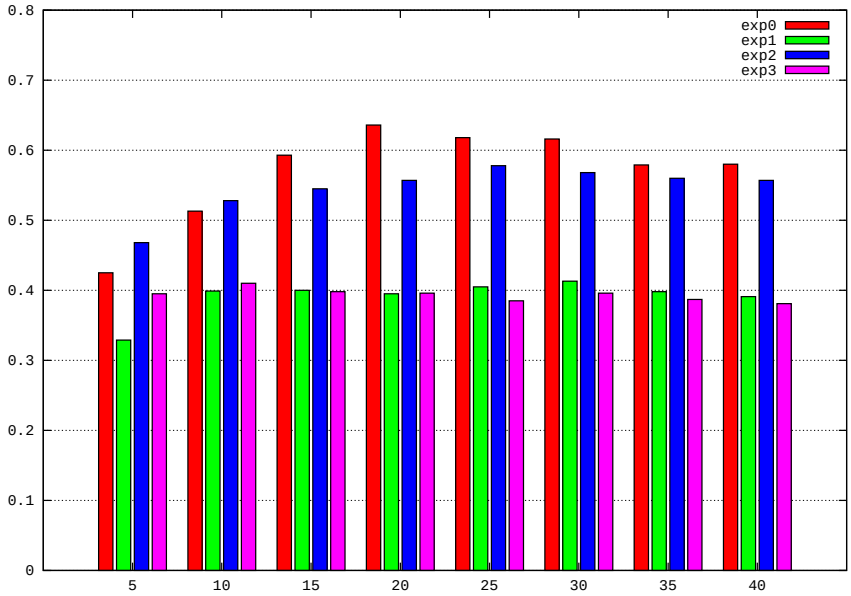

Abbildung 5.15: F-Measures (y-Achse) für Reuters-Koll. mit 20 Themen beim Clustering eines LSI-Modells trainiert mit einer $tfidf$-gewichteten TDM. Die Testkollektionen wurden mit tf gewichtet. Die x-Achse zeigt die Dimensionen.

5.3.3 Ergebnisse PLSI

Ein PLSI-Modell ließ sich nur mit tf-gewichteten TDM trainieren und das Training dauert um ein Vielfaches länger als das Training eines LSI-Modells (s. Tabelle 4.1 S. 37), daher wurden hier weitaus weniger Experimente durchgeführt. Die benutzte Implementation (s. Abschnitt 4.1.1) bietet keine Funktionalität um eine Anfrage oder ein neues Dokument in den Konzeptraum zu bringen. Der in Abschnitt 3.3.6 vorgeschlagene Ansatz, Konzept-Dokument-Wahrscheinlichkeiten per EM zu berechnen erschien zu aufwendig, daher wurde ein eigener Ansatz implementiert:

Die Parameter $P(z)$ und $P(w|z)$ eines trainierten Modells werden verwendet um für einen Dokumentvektor mit den Termgewichten $n(d,w)$ die Darstellung $P(d|z)$ zu berechnen:

$$P(d|z) = P(z) \sum_{w} n(d,w) P(w|z) \forall z \tag{5.1}$$

Für die Dokumentvektoren, die auf diese Weise in den PLSI-Raum überführt werden, kann als Termgewicht außer tf auch wieder $tfidf$ benutzt werden. Der berechnete Wert entspricht nicht mehr einer definierten Wahrscheinlichkeit, eignet sich aber dennoch zur Evaluation im Konzeptraum.

Die Analyse der Ähnlichkeitsverteilung im Konzeptraum (Abschnitt A.4 S. 79) zeigt eine Streuung über den gesamten Bereich. Im Gegensatz zu LSI entstehen hier keine negativen Ähnlichkeiten. Die Ähnlichkeitsverteilung lässt auch vermuten, dass der Graphschwellwert S wenig Einfluss auf das Ergebnis hat, wa sich in verschiedenen Experimenten mit S zwischen 0.2 und 0.5 bestätigte. Das Modell zeigte sich auch recht robust gegenüber Änderungen des KNN-Wertes, erst mit zu geringen Werten unter 30 verschlechterte sich die Retrievalperformanz deutlich.

Die Abbildung 5.16 und 5.17 zeigen die Ergebnisse auf den Reuters-Kollektionen mit 10 Themen für PLSI-Modelle trainiert mit 30 – 50 latenten Klassen. Unter 30 latenten Klassen sind die Ergebnisse deutlich schwächer, für höhere Werte konnten aufgrund der Laufzeit des EM-Algorithmus, die von der Anzahl latenter Klassen abhängt, keine Modelle trainiert werden. Aus dem gleichen Grund konnten auch keine Modelle für Versuche mit 20 Themen trainiert werden.

Es zeigt sich, dass die Ergebnisse mit $tfidf$-gewichteten Dokumentvektoren bei den Experimenten mit ähnlichen Themengebieten teilweise signifikant besser sind, während die Performanz auf den Kollektionen mit verschiedenen Themen recht ähnlich ist. Auch mit dem PLSI-Modell lassen sich also durch Kombination verschiedener Gewichtungsverfahren bessere Ergebnisse erzielen.

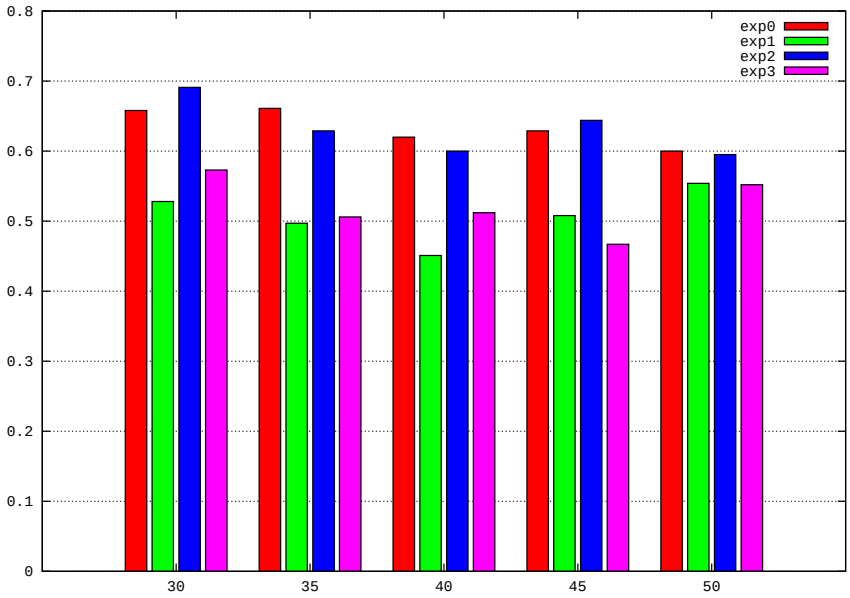

Abbildung 5.16: F-Measures (y-Achse) für Reuters-Koll. beim Clustering eines PLSI-Modells, trainiert mit einer tf-gewichteten TDM. Die Testkollektion wurden mit tf gewichtet. Die x-Achse zeigt die Dimensionen.

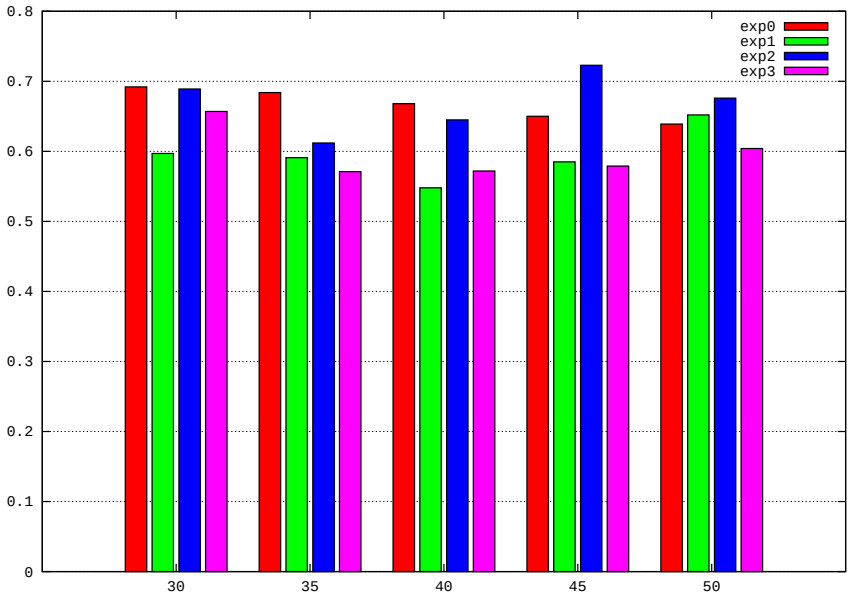

Abbildung 5.17: F-Measures (y-Achse) für Reuters-Koll. beim Clustering eines PLSI-Modells, trainiert mit einer tf-gewichteten TDM. Die Testkollektion wurden mit $tfidf$ gewichtet. Die x-Achse zeigt die Dimensionen.

Das durchschnittlich beste Ergebnis wird mit 30 latenten Klassen erzielt, der Effekt der Abschwächung bei höheren Werten ist hier nicht beobachtbar. Das erklärt sich dadurch, dass das Modell beim Training mit der jeweils gewünschten Anzahl latenter Klassen ein optimales Ergebnis annähert. Damit relativiert sich auch das bei LSI bestehende Problem der optimalen Dimension.

Umfangreichere Kollektionen

Ebenso wie die LSI-Modellen wurden auch die PLSI-Modelle hinsichtlich ihrer Fähigkeit untersucht, auf umfangreichere Dokumentkollektionen zu skalieren. Die Abbildungen 5.18 und 5.19 zeigen die Ergebnisse. Diese sind, wie schon bei LSI, besser als die Referenzergebnisse, es gelten hier also die gleichen Aussagen hinsichtlich der Anwendbarkeit, die bereits bei der Betrachtung von LSI getriffen wurden.

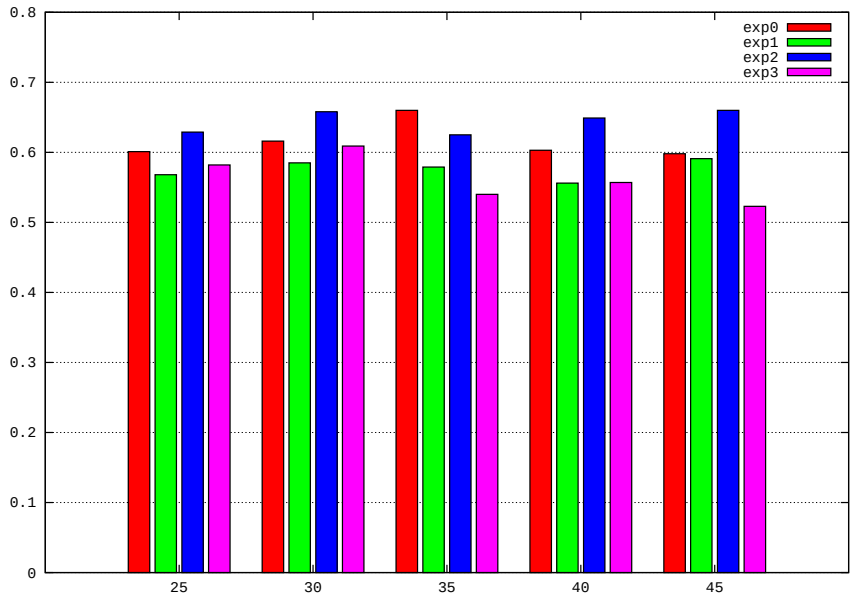

Abbildung 5.18: F-Measures (y-Achse) für Reuters-Koll. mit 2000 Dokumenten beim Clustering eines PLSI-Modells, trainiert mit tf-Gewichten mit df-Reduktion 20. Die Testkollektionen wurden mit $tfidf$ gewichtet. Die x-Achse zeigt die Dimensionen.

Zu allen PLSI-Ergebnissen muss erwähnt werden, dass im Vergleich zu LSI wesentlich weniger Zeit für die Optimierung zur Verfügung stand. Die Werte sind trotzdem sehr gut und es ist zu erwarten, dass mit mehr Aufwand noch bessere Ergebnisse erzielt werden können.

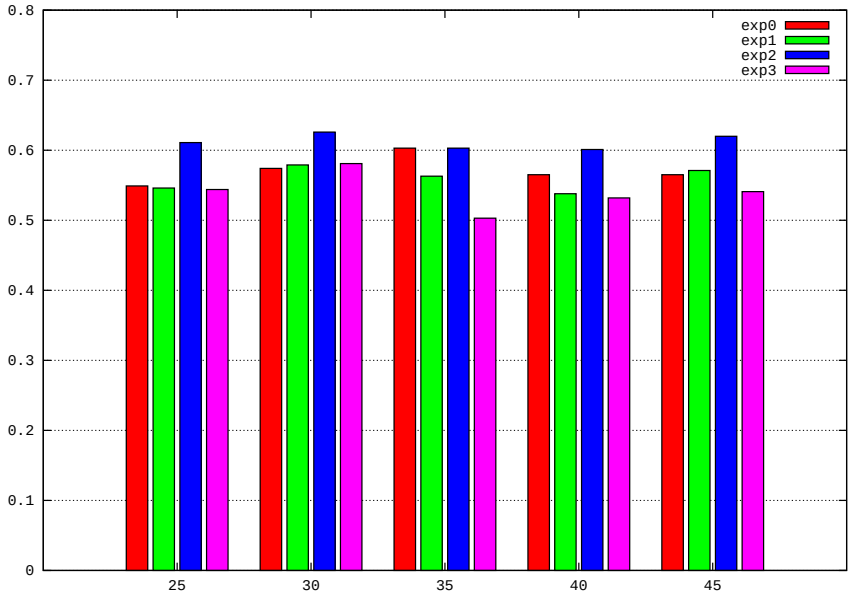

Abbildung 5.19: F-Measures (y-Achse) für Reuters-Koll. mit 3000 Dokumenten beim Clustering eines PLSI-Modells, trainiert mit tf-Gewichten mit df-Reduktion 20. Die Testkollektionen wurden mit $tfidf$ gewichtet. Die x-Achse zeigt die Dimensionen.

5.3.4 Spock-Experiment

Das Referenzergebnis für Spock (Abb. 5.20) zeigt deutliche Vorteile bei $tfidf$-Gewichtung der Dokumentvektoren. Zu beachten ist hier der negative Einfluss der df-Reduktion. Auf einer heterogenen Kollektion ist die Diskriminierungsinformation seltener Terme wichtig.

Überraschend ist das starke Ergebnis des LSI-Experiments (Abb. 5.21). Die als Trainingsmenge gewählte umfangreichste Instanz, gewichtet mit $tfidf$ und df-reduziert mit 20, liefert offenbar ausreichend Konzeptinformation für den ganzen Korpus. Die Dimension bei der die besten Ergebnisse erzielt werden ist allerdings sehr hoch, maximal möglich ist 2129, dieser Wert entspricht der Anzahl der Dokumente in der Trainingsmenge. Gegenüber den knapp 10000-dimensionalen Term-Dokumentvektoren ist es trotzdem eine Reduktion um den Faktor 5.

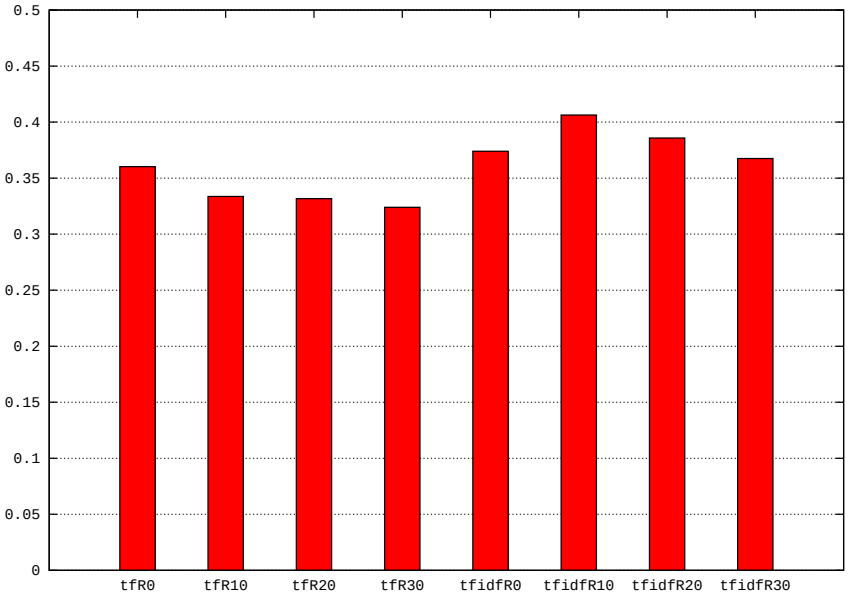

Abbildung 5.20: Spock Referenz-F-Measures für tf- und $tfidf$-gewichtete Dokumentvektoren, jeweils bei den Reduktionsstufen 0, 10, 20 und 30. Für die tf-gewichteten Versuche wurde der Threshold bei 0.3 angesetzt, für die $tfidf$-gewichteten bei 0.1, KNN in allen Fällen bei 50.

Auf dem Spock-Korpus wurde auch ein PLSI-Experiment durchgeführt. Die Trainingsmenge wurde aus 1750 Dokumenten der schon bei LSI gewählten Spock-Instanz zusammengestellt, der Grund hierfür ist die Notwendigkeit einer gleich dimensionierten Validationsmenge, wofür die zweitgrößte Spock-Instanz gewählt wurde, die etwa den genannten Umfang hat. Das Modell wurde mit 50 latenten Klassen trainiert, die Trainingszeit belief sich auf über 17 Stunden (s. Tabelle 4.1 S. 37). Das Ergebnis des Clusterings war ein

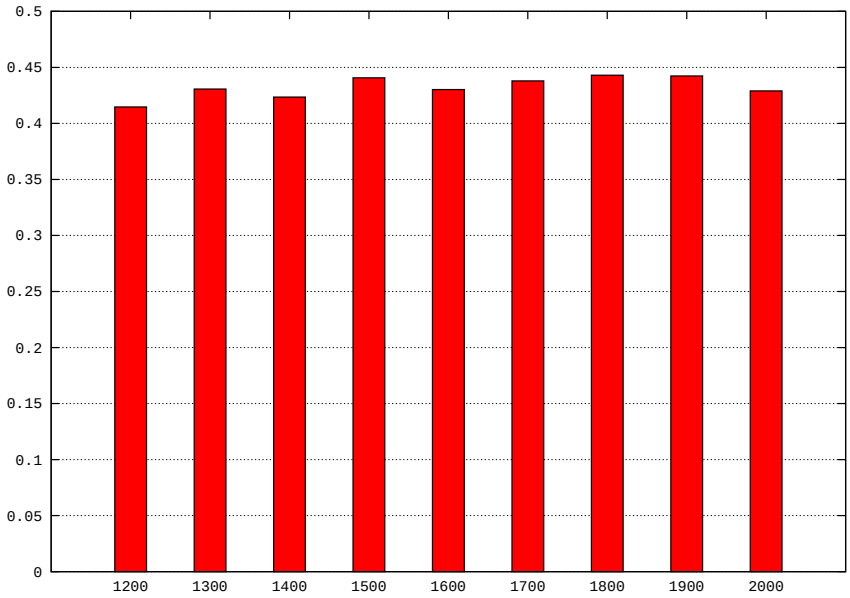

Abbildung 5.21: Spock F-Measures für ein LSI-Modell, trainiert aus den $tfidf$-gewichteten Dokumentvektoren der umfangreichsten Spock-Instanz. Dargestellt sind die Dimensionen 1200-2000.

F-Measure unter 0.1, also so schlecht, dass es hier nicht grafisch dargestellt werden muss. Dieser Misserfolg hat verschiedene Gründe, der wichtigste ist jedoch wahrscheinlich die viel zu geringe Anzahl der latenten Klassen. Einen Wert zu wählen, der im Bereich der besten Ergebnisse von LSI liegt, hätte den Zeitrahmen eines Experiments deutlich gesprengt. Die weiteren Parameter, die beim Training eines PLSI-Modells variiert werden können, konnten nicht einmal in den Reuters-Experimenten vollständig untersucht werden. Dieses Beispiel zeigt, dass PLSI deutlich schwieriger zu benutzen und damit auch noch ein Stück praxisferner ist, als LSI.

5.3.5 Weiterführende Untersuchungen

Die Experimente könnten hinsichtlich der folgenden Fragen fortgeführt werden:

- Wie wirken sich andere Termgewichtungen auf das Training eines LSI-Modells aus?
- Wie gut generalisiert das LSI-Modell auf den Spock-Testkorpus und bringt es vielleicht weitere Nutzinformation in das bestehende Spock-System?

Darüber hinaus wäre es interessant, das Modell auf anderen Korpora mit erweiterten Möglichkeiten zu testen, besonders auf solchen, die Anfragen und erwünschte Ergebnissmengen definieren.

6 Zusammenfassung

In dieser Arbeit werden Methoden zur semantischen Textanalyse vorgestellt und deren Anwendung zur Erstellung eines Konzeptraumindex für Dokumentkollektionen beschrieben. Die Einleitung beschreibt das allgemeine Problem der Indexerstellung zur Unterstützung der Suche und Verwaltung einer Dokumentsammlung. Kapitel zwei stellt ein klassisches automatisches Indexierungsverfahren vor und motiviert die weitere Entwicklung, indem Grenzen des Systems gezeigt werden, die mit neuen Verfahren überwunden werden können. Das folgende Kapitel stellt zwei Methoden vor, die in der Lage sind, Konzepte hinter den Texten zu erkennen und die Dokumente entsprechend dieser Konzepte zu indexieren. Diese Verfahren, Latent Semantic Indexing und Probabilistic Latent Semantic Indexing, werden diskutiert und einander gegenübergestellt. Im weiteren Verlauf wird ein Framework vorgestellt, welches die beiden Verfahren implementiert und für Experimente genutzt wird.

Die Experimente beziehen sich auf ein spezielles Problem des Information Retrieval, die automatische unüberwachte Kategorisierung einer Dokumentkollektion. Die Performanz von Dokumentmodellen auf Basis eines Konzeptindex wird auf verschiedenen Dokumentsammlungen mit dem Standardmodell verglichen. Es zeigt sich, dass die Konzeptraummodelle in der Lage sind, auf bestimmten Kollektionen deutlich bessere Ergebnisse zu liefern. Es werden auch die Probleme angesprochen, dabei besonders die schwierige Wahl der richtigen Parameter um optimale Ergebnisse zu erzielen und die hohe Laufzeit des Modelltrainings. Es wird gezeigt, wie der Aufwand mit einer Datenreduktion vor dem Training verringert werden kann und dass sich der Informationsverlust bei diesem Prozess nicht sehr negativ auswirkt. Die Experimente zeigen somit, dass eine praktische Anwendbarkeit der Modelle durchaus gegeben ist. Auf kleineren, homogenen Dokumentsammlungen reicht ein repräsentativer geringer Teil der Dokumente aus um ein Modell zu trainieren, das in der Lage ist, gute Clusteringergebnisse zu liefern. Dies wird mit den Experimenten auf dem Reuters-Korpus gezeigt. Die Ergebnisse sind übertragbar, z.B. auf den Dokumentbestand kleiner Unternehmen oder spezialisierter wissenschaftlicher Institute oder auch eine private Dokumentsammlung, beispielsweise die Artikel im eigenen Weblog.

Ein interessantes Ergebnis ist das unerwartet gute Abschneiden von LSI auf dem Spock-Challenge-Trainingskorpus. Üblicherweise wird angenommen, dass die Konzeptraumverfahren auf einer sehr heterogene Kollektion wie dem WWW versagen. Der Spock-Korpus ist sicher damit nicht vergleichbar, sollte aber ein ähnlich schwieriges Problem abbilden.

Es sieht so aus, als wäre LSI in der Lage, bei bestimmten Problemstellungen auch auf solchen Kollektionen sehr gute Ergebnisse zu bringen. Eine vorstellbare Anwendung ergibt sich in einem Bereich, der im Mittelpunkt des Interesses vieler Gewerbetreibender im Internet steht – der kontextsensitiven Werbung. Automatische Systeme, welche semantische Informationen nutzen, können Anzeigen gezielter auswählen. Daher ist zu vermuten, das Techniken wie LSI und PLSI durchaus auch im Interesse der großen Suchmaschinenbetreiber stehen, wenn auch nicht zum Nutzen der Websuche.

Obwohl das Problem des Berechnungsaufwandes aufgrund seiner Natur auch mit verbesserter Computertechnik nicht zu beheben ist, lässt es sich dennoch entschärfen. Die Auswahl einer guten repräsentativen Trainingsmenge ist wichtiger, als einfach nur möglichst viele Dokumente zum Training zu verwenden. Das zweite wichtige Problem, das Finden der richtigen Dimension für optimale Ergebnisse, relativiert sich mit dem Schritt zu PLSI, wo ein optimales Ergebnis für eine gewünschte Dimension berechnet wird, solange diese nicht zu gering gewählt ist. Aktuelle Arbeiten wie [Bas05] zeigen auch für LSI Möglichkeiten, diese Schwierigkeit zu umgehen und betrachten die Problematik aus einer interessanten neuen Perspektive. Die Forschung im Bereich der semantischen Analyseverfahren bietet noch viele Möglichkeiten und die Methoden selbst werden sicher stärkere Verbreitung finden.

Literaturverzeichnis

[Bas05] Bast, Holger und Majumdar, Debapriyo: Why spectral retrieval works, in: *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, New York, NY, USA, S. 11–18

[Ber94] Berry, Michael W.; Dumais, Susan T. und O'Brien, Gavin W.: Using Linear Algebra for Intelligent Information Retrieval, Techn. Ber. UT-CS-94-270 (1994), URL `citeseer.ist.psu.edu/berry95using.html`

[Ble03] Blei, David M.; Ng, Andrew Y. und Jordan, Michael I.: Latent Dirichlet Allocation. *Journal of Machine Learning Research* (2003), Bd. 3:S. 993–1022, URL `http://www.jmlr.org/papers/v3/blei03a.html`

[Bor04] Borman, Sean: The Expectation Maximization Algorithm – A short tutorial (2004), introduces the Expectation Maximization (EM) algorithm and fleshes out the basic mathematical results, inclusing a proof of convergence. The Generalized EM algorithm is also introduced.

[BY99] Baeza-Yates, Ricardo A. und Ribeiro-Neto, Berthier A.: *Modern Information Retrieval*, ACM Press / Addison-Wesley (1999), URL `citeseer.ist.psu.edu/baeza-yates99modern.html`

[Dee90] Deerwester, Scott; Dumais, Susan T.; Furnas, George W.; Landauer, Thomas K. und Harshman, Richard: Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science* (1990), Bd. 41(06):S. 391–407

[Fer03] Ferber, Reginald: *Information Retrieval*, Dpunkt Verlag, Heidelberg (2003), URL `http://information-retrieval.de/irb/ir.html`

[Hof98] Hofmann, Thomas und Puzicha, Jan: Unsupervised Learning from Dyadic Data, Techn. Ber. TR-98-042, International Computer Science Insitute, Berkeley, CA (1998), URL `citeseer.ist.psu.edu/article/hofmann98unsupervised.html`

[Hof99] Hofmann, Thomas: Probabilistic Latent Semantic Analysis, in: *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, Morgan Kaufmann, San Francisco, CA, S. 289–29

[Lan98] Landauer, Thomas K.; Foltz, Peter W. und Laham, Darrell: An Introduction to Latent Semantic Analysis. *Discourse Processes* (1998), Bd. 25:S. 259–284

[Nea98] Neal, R. M. und Hinton, G. E.: A new view of the EM algorithm that justifies incremental, sparse and other variants, in: M. I. Jordan (Herausgeber) *Learning in Graphical Models*, Kluwer Academic Publishers (1998), S. 355–368, URL `citeseer.ist.psu.edu/neal93new.html`

[Pap98] Papadimitriou, Christos H.; Tamaki, Hisao; Raghavan, Prabhakar und Vempala, Santosh: Latent Semantic Indexing: A Probabilistic Analysis, S. 159–168, URL `citeseer.ist.psu.edu/papadimitriou98latent.html`

[Sal68] Salton, G. und Lesk, M. E.: Computer Evaluation of Indexing and Text Processing. *J. ACM* (1968), Bd. 15(1):S. 8–36

[Sal75] Salton, G.; Wong, A. und Yao, C. S.: A Vector Space Model for Automatic Indexing. *Communications of the ACM* (1975), Bd. 18(11):S. 229–237

[Sch03] Schein, Andrew I.; Popescul, Alexandrin und Ungar, Lyle H.: PennAspect: Two-Way Aspect Model Implementation, Techn. Ber. MS-CIS-01-25, University of Pennsylvania Department of Computer and Information Science, Philadelphia, PA (2003), URL `http://www.cis.upenn.edu/datamining/software_dist/PennAspect/MS-CIS-01-25.ps`

[Ste99] Stein, Benno und Niggemann, Oliver: On the Nature of Structure and its Identification, in: Peter Widmayer; Gabriele Neyer und Stefan Eidenbenz (Herausgeber) *Graph-Theoretic Concepts in Computer Science*, Bd. 1665 LNCS von *Lecture Notes in Computer Science*, Springer, Berlin Heidelberg New York, S. 122–134

[Ste03] Stein, Benno; Meyer zu Eissen, Sven und Wissbrock, Frank: On Cluster Validity and the Information Need of Users, in: M. H. Hanza (Herausgeber) *Proceedings of the 3rd IASTED International Conference on Artificial Intelligence and Applications (AIA 03), Benalmádena, Spain*, ACTA Press, Anaheim, Calgary, Zurich, S. 216–221

[Ste07] Stein, Benno: Information Retrieval WS2007/08, in: *Lectures in Web Technology (Advanced)*, Bauhaus-Universität Weimar (2007), URL `http://www.uni-weimar.de/cms/medien/webis/teaching/lecture-notes.html#information-retrieval`

[Wei05] Weikum, Gerhard: Advanced IR Models (2005), lectures (Chapter 4) IRDM WS 2005, Max-Planck-Institut fuer Informatik.

A Analyse der Daten

A.1 Verteilung der Dokumentfrequenzen

Die Abbildung A.1 zeigt die Verteilung der Dokumentfrequenzen für die vier Reutersexperimente.

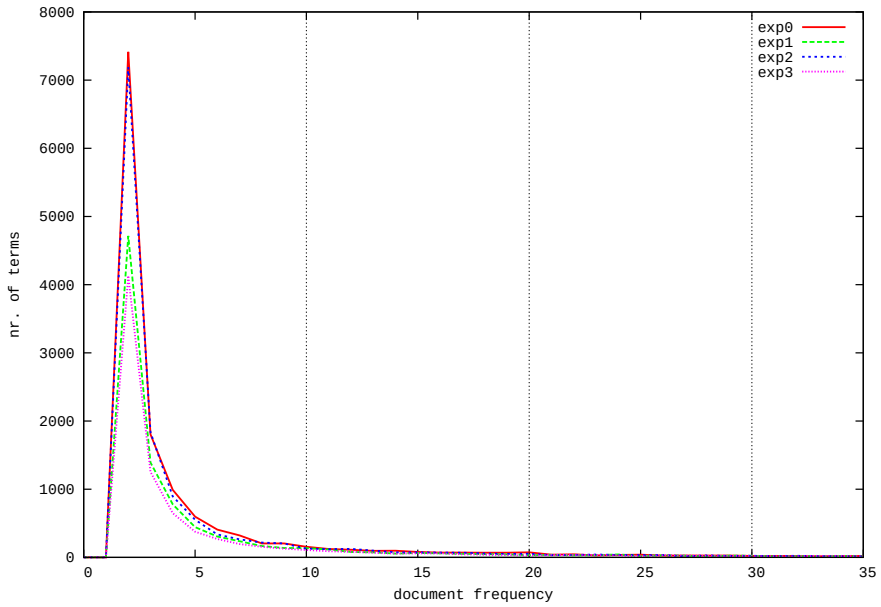

Abbildung A.1: Verteilung der Dokumentfrequenz für die Reuters-Experimente exp0-exp3. An der y-Achse ist die Anzahl der Terme abgetragen, die x-Achse zeigt die Dokumentfrequenz im Bereich 0 – 35, eingetragen sind auch die Reduktionsstufen 10, 20 und 30.

A.2 Cosinus-Ähnlichkeitswerte im Vektorraummodell

Die Abbildungen A.2 bis A.5 zeigen die Verteilung der Cosinusähnlichkeitswerte für die Reuters-Experimentkollektionen im Vektorraummodell, jeweils links mit tf-Gewichtung, rechts mit $tfidf$-Gewichtung auf verschiedenen df-Reduktionsstufen (0, 10, 20 und 30). Für jede Kollektionscharakteristik wurde eine Testkollektion analysiert, die anderen zeigen ähnliche Werte.

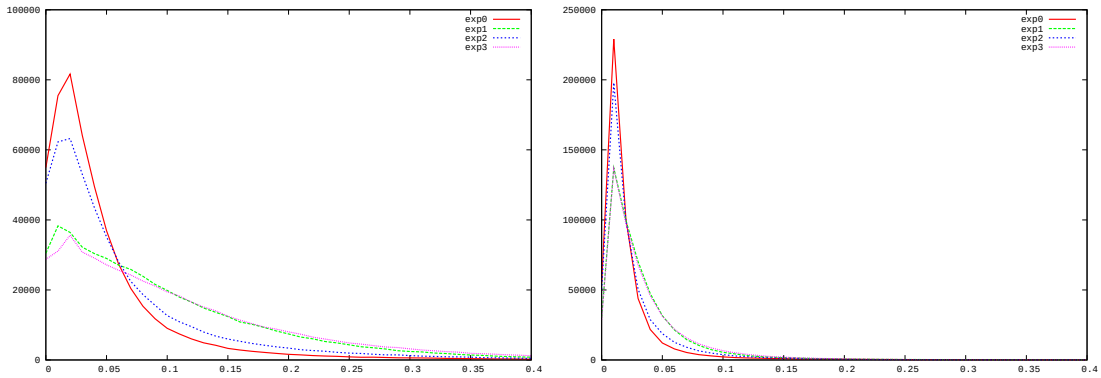

Abbildung A.2: Verteilung der Cosinusähnlichkeiten im Vektorraummodell, links tf, rechts $tfidf$ mit df-Reduktion 0

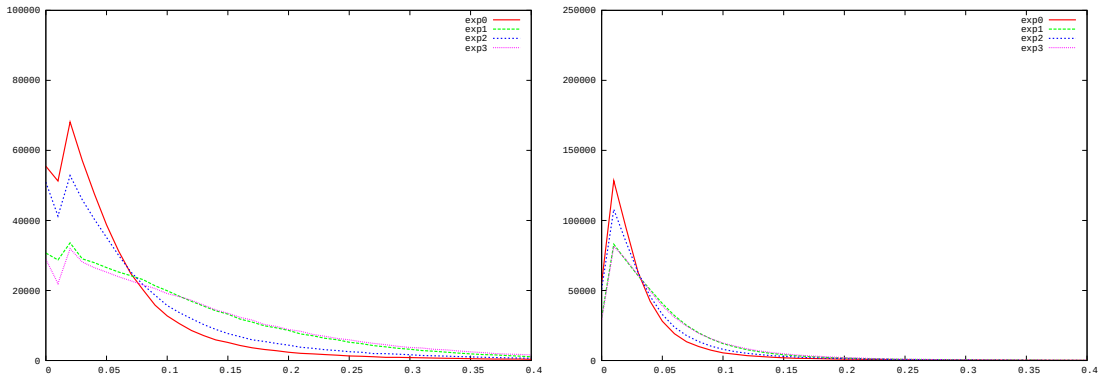

Abbildung A.3: Verteilung der Cosinusähnlichkeiten im Vektorraummodell, links tf, rechts $tfidf$ mit df-Reduktion 10

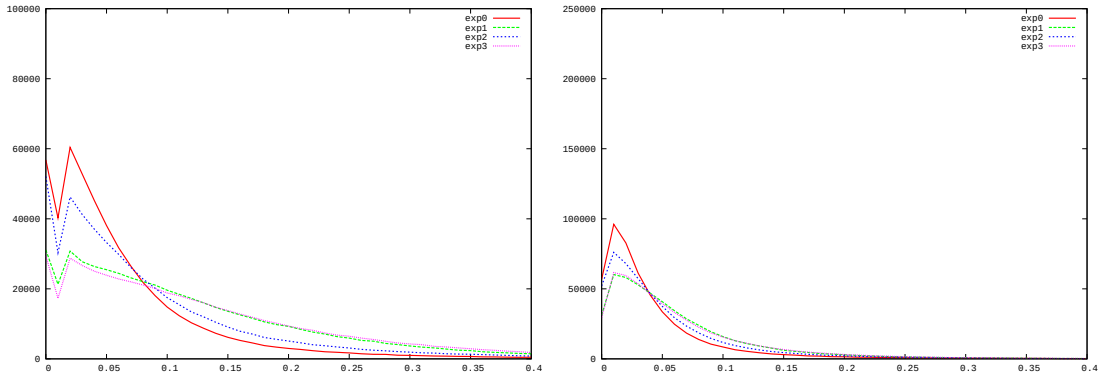

Abbildung A.4: Verteilung der Cosinusähnlichkeiten im Vektorraummodell, links tf, rechts $tfidf$ mit df-Reduktion 20

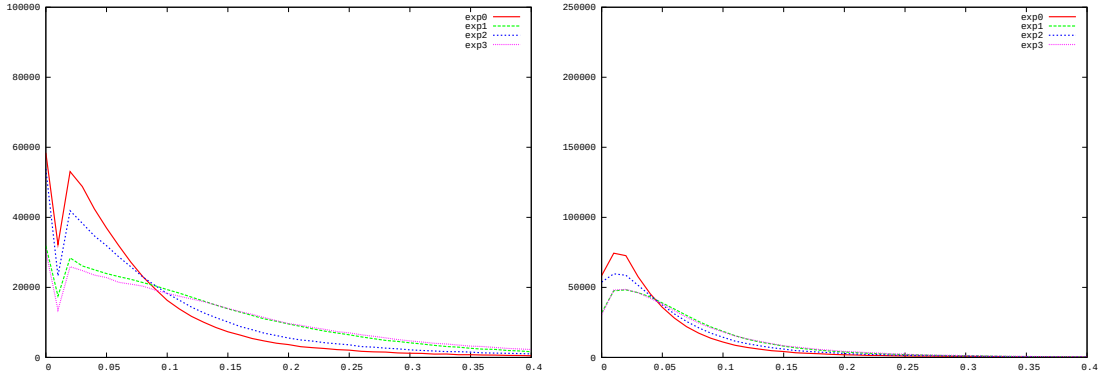

Abbildung A.5: Verteilung der Cosinusähnlichkeiten im Vektorraummodell, links tf, rechts $tfidf$ mit df-Reduktion 30

A.3 Cosinus-Ähnlichkeitswerte im LSI-Raum

Die folgenden Abbildungen zeigen die Verteilung der Cosinus-Ähnlichkeitswerte für die Reuters-Experimentkollektionen im semantischen LSI-Raum. Für jede Kollektionscharakteristik wurde eine Kollektion analysiert, die anderen zeigen ähnliche Werte. Die linken Abbildungen wurden aus tf-gewichteten TDM erzeugt, die rechten aus $tfidf$-gewichteten. Reduktionsstufe für die TDM war 20.

LSI trainiert mit tf

Cosinus-Ähnlichkeitswerte für projizierte Kollektionen in den Dimensionen 5, 10, 20, 40 und 80. Das LSI-Modell wurde mit einer tf-gewichteten TDM trainiert. Die linken Darstellungen zeigen das Resultat aus projizierten tf-gewichteten Dokumentvektoren, rechts wurden $tfidf$-gewichtete Dokumentvektoren projiziert.

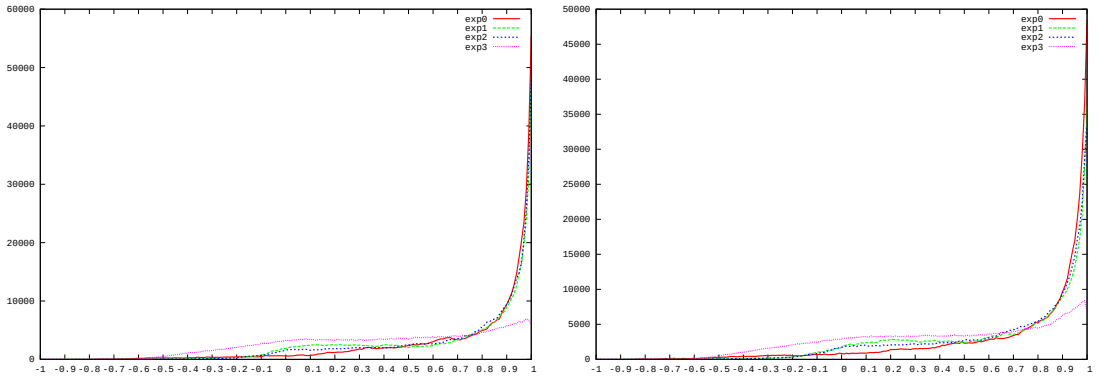

Abbildung A.6: Verteilung der Cosinusähnlichkeiten einer in den LSI-Konzeptraum projizierten Kollektion. Das LSI-Modell wurde mit tf-Gewichten trainiert. Dargestellt ist Dimension 5.

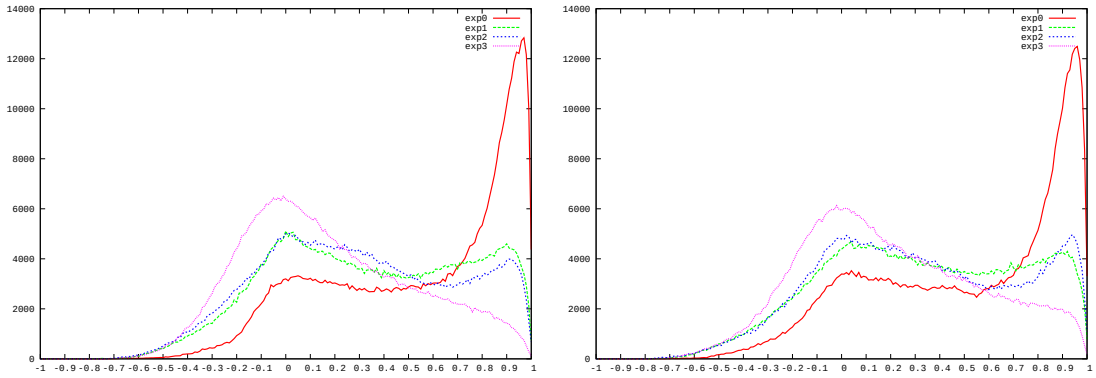

Abbildung A.7: Verteilung der Cosinusähnlichkeiten einer in den LSI-Konzeptraum projizierten Kollektion. Das LSI-Modell wurde mit tf-Gewichten trainiert. Dargestellt ist Dimension 10.

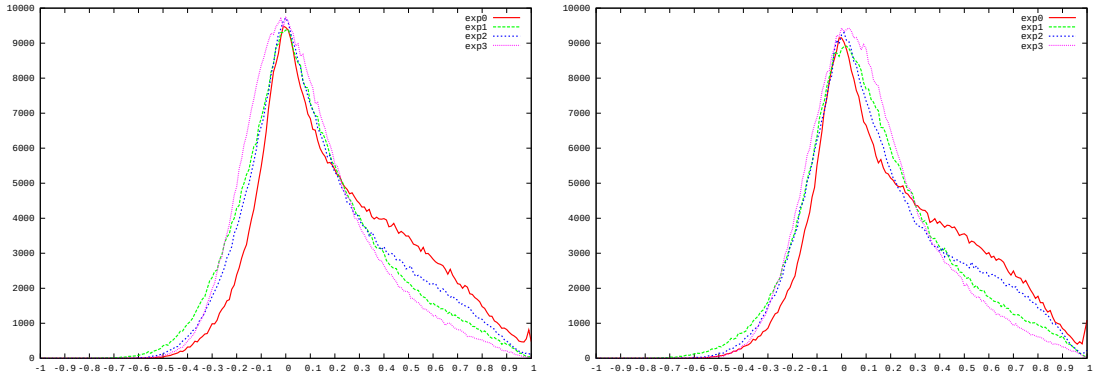

Abbildung A.8: Verteilung der Cosinusähnlichkeiten einer in den LSI-Konzeptraum projizierten Kollektion. Das LSI-Modell wurde mit tf-Gewichten trainiert. Dargestellt ist Dimension 20.

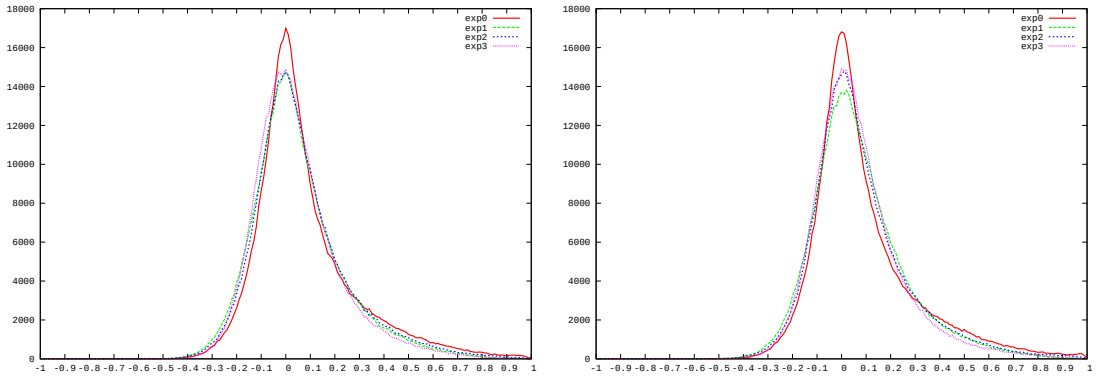

Abbildung A.9: Verteilung der Cosinusähnlichkeiten einer in den LSI-Konzeptraum projizierten Kollektion. Das LSI-Modell wurde mit tf-Gewichten trainiert. Dargestellt ist Dimension 40.

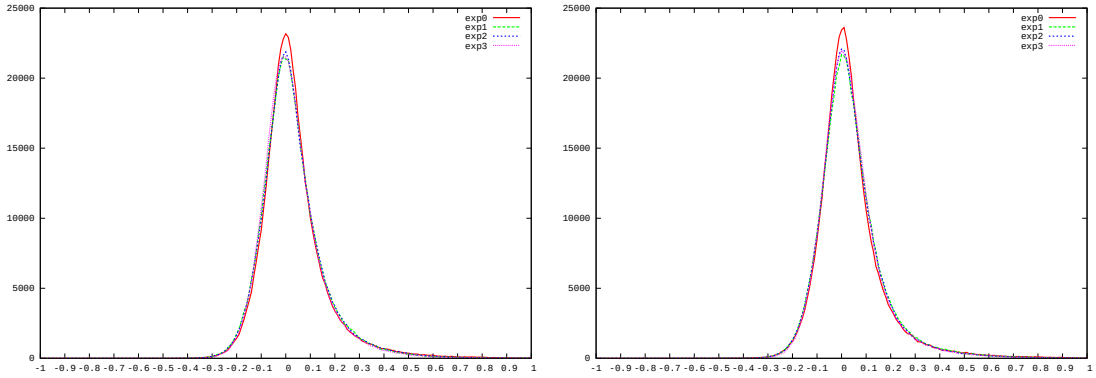

Abbildung A.10: Verteilung der Cosinusähnlichkeiten einer in den LSI-Konzeptraum projizierten Kollektion. Das LSI-Modell wurde mit tf-Gewichten trainiert. Dargestellt ist Dimension 80.

LSI trainiert mit $tfidf$

Cosinus-Ähnlichkeitswerte für projizierte Kollektionen in den Dimensionen 5, 10, 20, 40 und 80. Das LSI-Modell wurde mit einer $tfidf$-gewichteten TDM trainiert. Die linken Darstellungen zeigen das Resultat aus projizierten tf-gewichteten Dokumentvektoren, rechts wurden $tfidf$-gewichtete Dokumentvektoren projiziert.

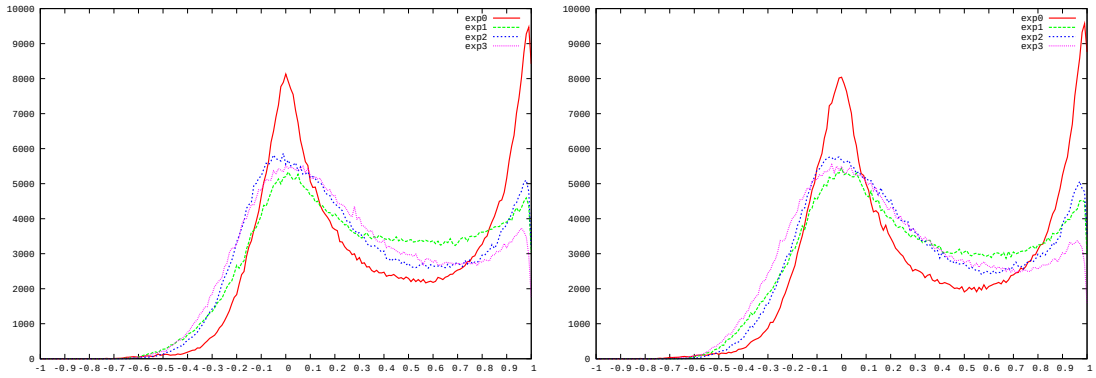

Abbildung A.11: Verteilung der Cosinusähnlichkeiten einer in den LSI-Konzeptraum projizierten Kollektion. Das LSI-Modell wurde mit $tfidf$-Gewichten trainiert. Dargestellt ist Dimension 5.

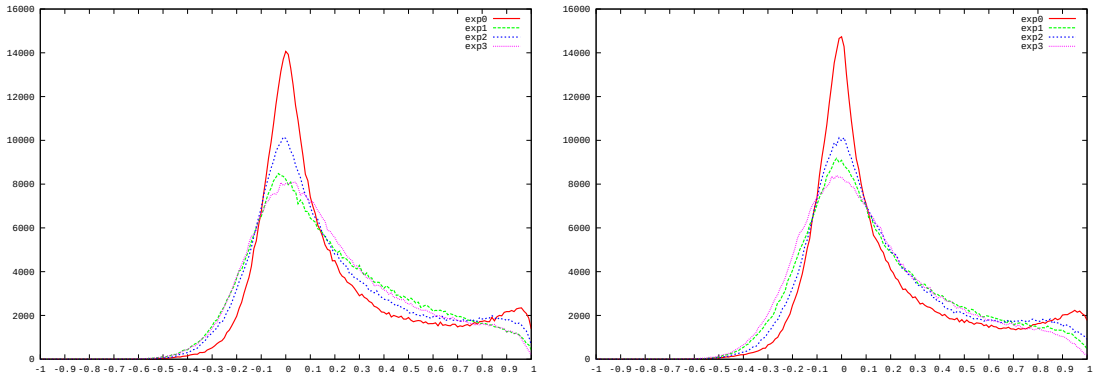

Abbildung A.12: Verteilung der Cosinusähnlichkeiten einer in den LSI-Konzeptraum projizierten Kollektion. Das LSI-Modell wurde mit $tfidf$-Gewichten trainiert. Dargestellt ist Dimension 10.

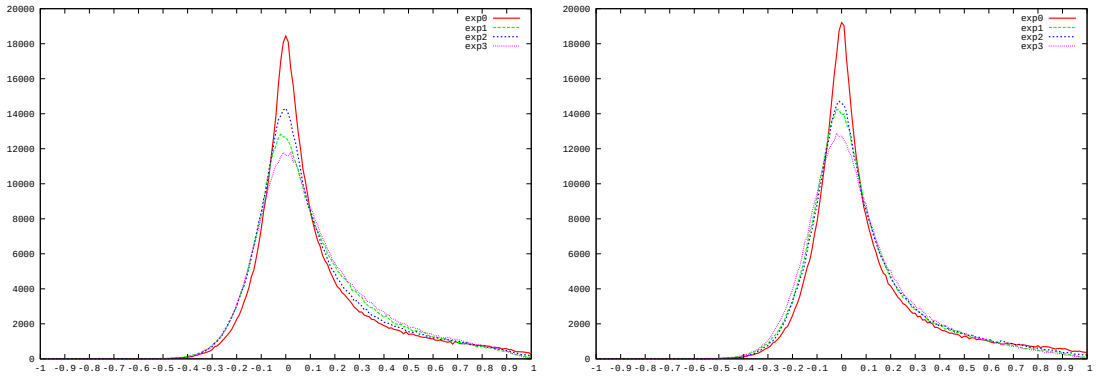

Abbildung A.13: Verteilung der Cosinusähnlichkeiten einer in den LSI-Konzeptraum projizierten Kollektion. Das LSI-Modell wurde mit $tfidf$-Gewichten trainiert. Dargestellt ist Dimension 20.

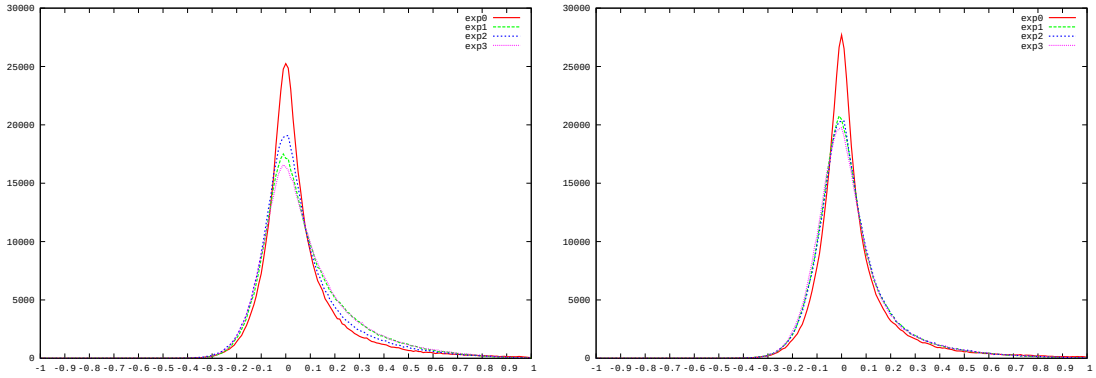

Abbildung A.14: Verteilung der Cosinusähnlichkeiten einer in den LSI-Konzeptraum projizierten Kollektion. Das LSI-Modell wurde mit $tfidf$-Gewichten trainiert. Dargestellt ist Dimension 40.

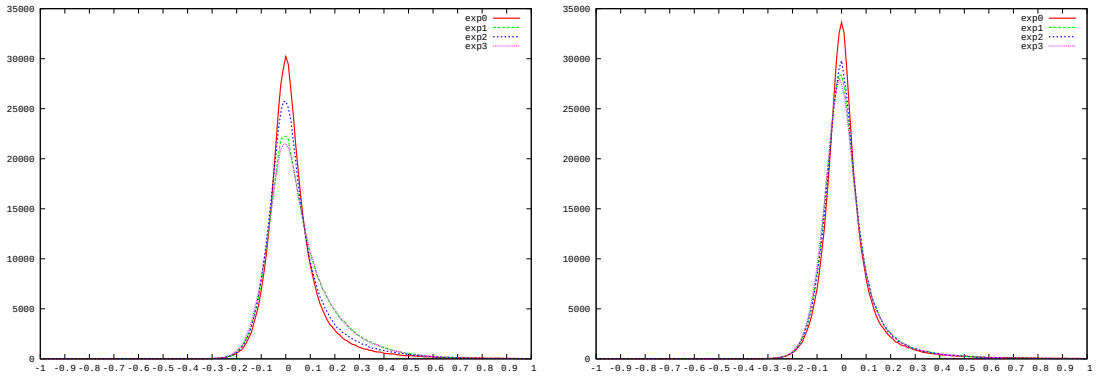

Abbildung A.15: Verteilung der Cosinusähnlichkeiten einer in den LSI-Konzeptraum projizierten Kollektion. Das LSI-Modell wurde mit $tfidf$-Gewichten trainiert. Dargestellt ist Dimension 80.

A.4 Cosinus-Ähnlichkeitswerte im PLSI-Raum

Die folgenden Abbildungen zeigen die Verteilung der Cosinus-Ähnlichkeitswerte für die Reuters-Experimentkollektionen im semantischen PLSI-Raum. Für jede Kollektionscharakteristik wurde eine Kollektion analysiert, die anderen zeigen ähnliche Werte. Reduktionsstufe für die Trainings-TDM war 20.

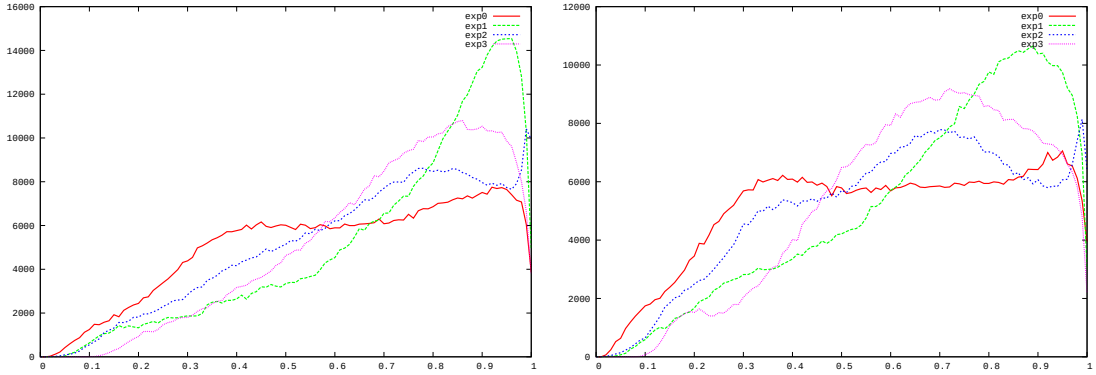

Abbildung A.16: Verteilung der Cosinusähnlichkeiten einer Kollektion im PLSI-Konzeptraum. Das PLSI-Modell wurde mit tf-Gewichten trainiert. Dargestellt ist Dimension 10.

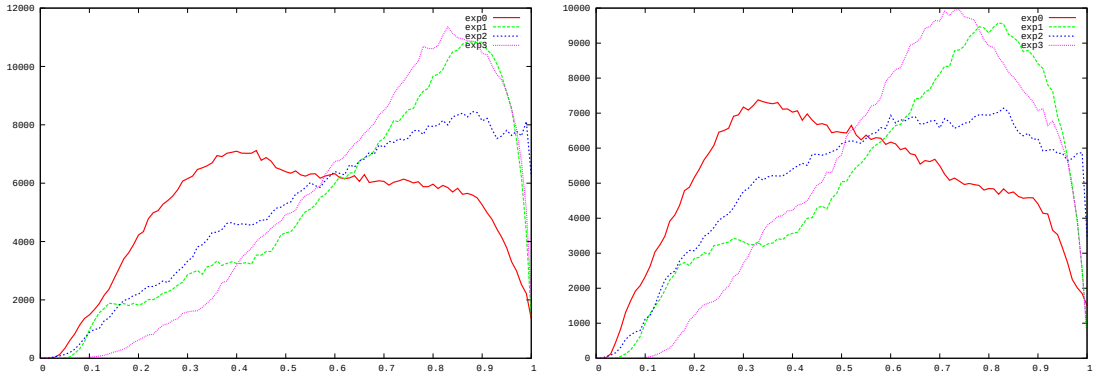

Abbildung A.17: Verteilung der Cosinusähnlichkeiten einer Kollektion im PLSI-Konzeptraum. Das PLSI-Modell wurde mit tf-Gewichten trainiert. Dargestellt ist Dimension 20.

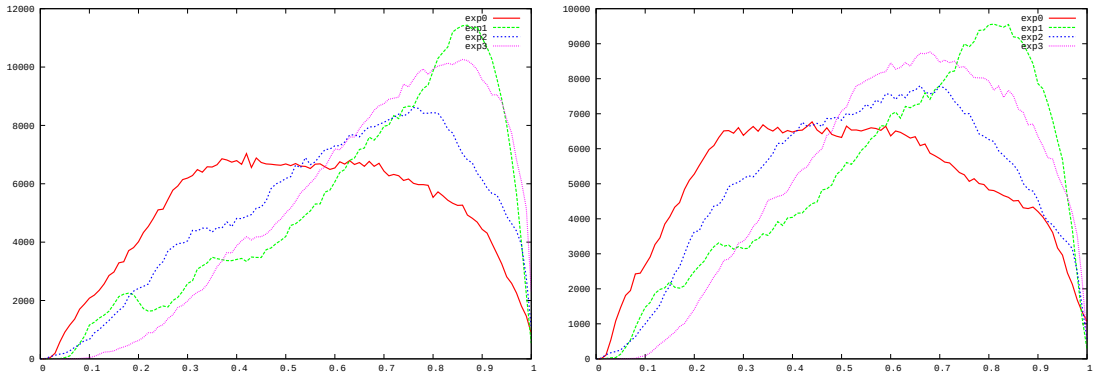

Abbildung A.18: Verteilung der Cosinusähnlichkeiten einer Kollektion im PLSI-Konzeptraum. Das PLSI-Modell wurde mit tf-Gewichten trainiert. Dargestellt ist Dimension 30.

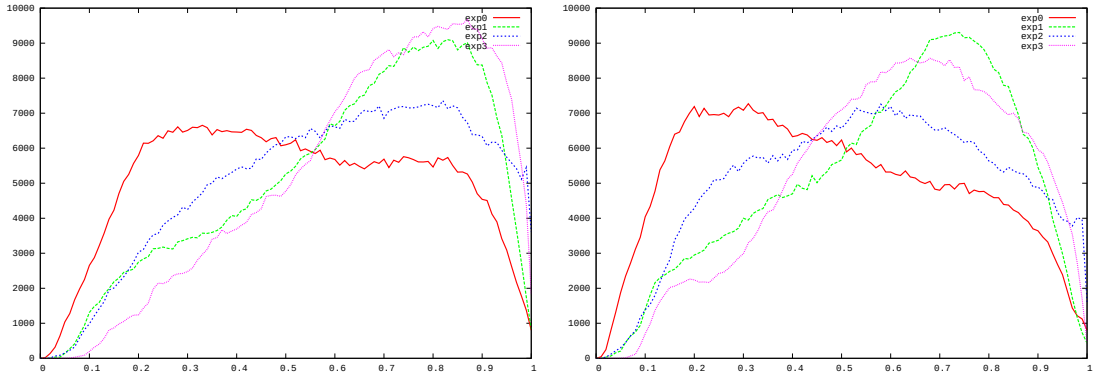

Abbildung A.19: Verteilung der Cosinusähnlichkeiten einer Kollektion im PLSI-Konzeptraum. Das PLSI-Modell wurde mit tf-Gewichten trainiert. Dargestellt ist Dimension 40.

A.5 Verteilung der Singulärwerte

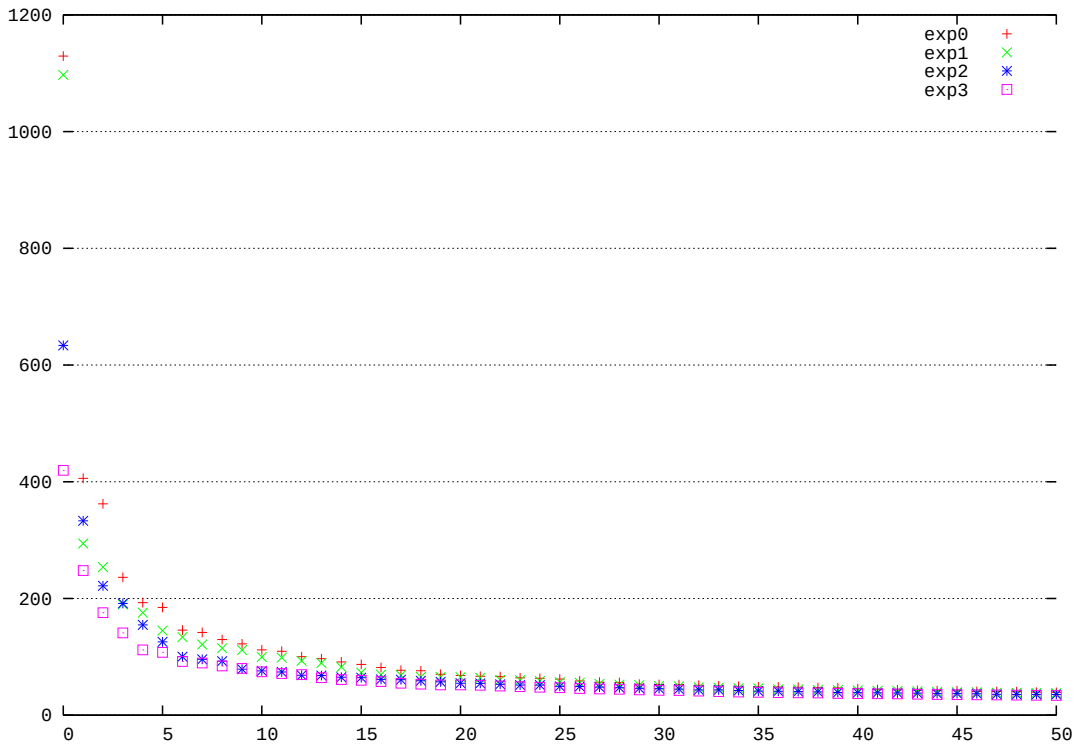

Abbildung A.20: Die ersten 50 Singulärwerte einer SVD einer tf-gewichteten TDM

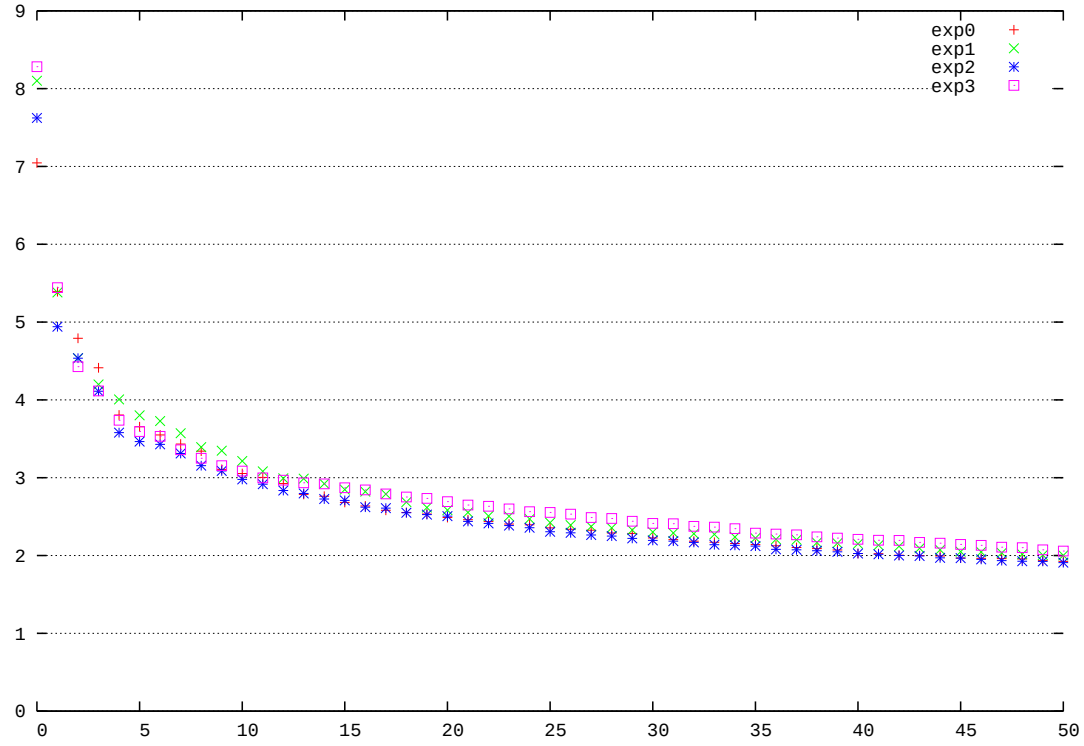

Abbildung A.21: Die ersten 50 Singulärwerte einer SVD einer $tfidf$-gewichteten TDM

B Zusammenstellung der Reuters-Experimente

Gewählte Reuters-Klassen:

- CCAT (Corporate/Industrial): C11 (Strategy/Plans), C21 (Production/Services), C33 (Contracts/Orders)
- ECAT (Economics): E11 (Economic Performance), E12 (Monetary/Economic), E211 (Expenditure/Revenue), E311 (Industrial Production), E511 (Balance of Payments), E71 (Leading Indicators)
- GCAT (Government/Social): G15 (European Community), GSCI (Science and Technology), GSPO (Sports)
- MCAT (Markets): M12 (Bond Markets), M14 (Commodity Markets), M11 (Equity Markets), M131 (Interbank Markets)

	Dokumente			Dokumente	
	exp0	exp2		exp1	exp3
C11	100	20	E11	100	20
C21	100	150	E12	100	150
C33	100	70	E211	100	70
E11	100	230	E311	100	230
E71	100	70	E511	100	70
G15	100	30	E71	100	30
GSCI	100	50	M11	100	50
GSPO	100	100	M12	100	100
M12	100	200	M131	100	200
M14	100	80	M14	100	80

Tabelle B.1: Kategorieauswahl und Anzahl der Dokumente für die Reuters-Experimente exp0 – exp3

C Notation und Abkürzungen

Zeichen	Bedeutung
d, d_i	Dokument
D	Menge von Dokumenten
q, q_i	Anfrage (Query)
Q	Menge von Anfragen
t, t_i	Term (Indexterm)
T	Menge von Termen
w, w_i	Wort
W	Menge von Worten
tf	Termfrequenz
df	Dokumentfrequenz
$tfidf$	Termfrequenz-Inversdokumentfrequenz
$\mathbf{d}$	Dokumentrepräsentation
$\mathbf{D}$	Menge von Dokumentrepräsentation
$\mathbf{q}$	Repräsentation einer Anfrage (formalisierte Anfrage)
$\mathbf{Q}$	Menge formalisierter Anfragen
$\mathbf{v}$	Vektor (Spaltenvektor)
$\mathbf{v}^T$	Transponierter Vektor (Zeilenvektor)
$\mathbf{v}_i$	i-te Komponente eines Vektors
$\mathrm{A}, \mathrm{A}_{m,n}$	Matrix
m, n	Zeilen, Spalten einer Matrix
A'	modifizierte Matrix
$\mathcal{R}$	Retrievalmodell
$\rho_{\mathcal{R}}$	Retrieval-Funktion
$d \in D$	d ist enthalten in D
AB	Matrixmultiplikation
TDM	Term-Dokument-Matrix
IR	Information Retrieval
LSI	Latent Semantic Indexing
PLSI	Probabilistic Latent Semantic Indexing
LSA	Latent Semantic Analysis

Abbildungsverzeichnis

Tabellenverzeichnis