Diplomarbeit
Themenextraktion aus semantischen Netzen

Andreas Heß

28. Februar 2001

Zusammenfassung

Da die Anzahl der Dokumente im Internet oder einem größeren Intranet ständig zunimmt, wird es immer schwieriger, in dieser Informationsflut die relevanten Dokumente zu einer gegebenen Fragestellung zu finden. Die bekannten Volltext-Suchmaschinen sind eine Hilfe, liefern jedoch oft eine unüberschaubar große Anzahl von Treffern.

In dieser Diplomarbeit wird ein Verfahren zur Themenextraktion aus semantischen Netzen erläutert, das es in Verbindung mit der Technik der *Query Expansion* erlaubt, Suchanfragen mit Begriffen aus dem Kontext anzureichern. Zur Traversierung der semantischen Netze werden die aus der künstlichen Intelligenz bekannten Hopfield-Netze benutzt.

Zunächst wird ein allgemeiner Überblick über bisherige Techniken von Volltext-Suchmaschinen und Information-Retrieval-Systemen gegeben. Es folgt ein Überblick über neuronale Netze. Dabei wird ein neuer, effizienter Algorithmus zur Berechnung des stabilen Zustands in Hopfield-Netzen vorgestellt. Anschließend werden die Anwendung von Hopfield-Netzen für die Traversierung von semantischen Netzen und eine Implementierung, die in das Projekt K-Portal der Abteilung IT-Research der Dresdner Bank einfließt, vorgestellt. Die Auswertung der erzielten Ergebnisse auf Basis einer experimentellen *Query Expansion* schließt diese Arbeit ab.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Die Menge an verfügbaren Informationen im Internet oder in einem größeren Intranet nimmt immer weiter zu. Sucht ein Anwender Informationen zu einem bestimmten Thema mit einer der bekannten Volltext-Suchmaschinen, steht er daher heute mehr und mehr vor dem Problem, unter den vielen gefundenen Texten die für ihn relevanten zu finden. Die Ursache für eine kaum überschaubare Zahl von Treffern ist meist eine ungenaue Formulierung des Suchausdrucks. Die Eingabe eines einzelnen Wortes liefert alle Texte als Ergebnis, die das eingegebene Wort enthalten. Auf den Sinnzusammenhang nehmen die Volltext-Suchmaschinen keine Rücksicht. Ideal wäre also, wenn die Suchmaschine selbst den eingegebenen Suchausdruck in einen Kontext stellen könnte und so automatisch den Suchausdruck geeignet erweitern würde.

Derzeit werden neue Ansätze entwickelt, die in diese Richtung weisen. Neben dem Projekt bei der Dresdner Bank, in das dieser Diplomarbeit eingebettet ist, sind hier die Arbeiten von Chen [3] und Chung, Pottenger und Schatz [4] zu nennen. Diese Verfahren ordnen den eingegebenen Begriff mit einem vorher aus den indizierten Texten aufgebauten semantischen Netz in seinen Kontext ein.

1.2 Ziele

Ziel der in dieser Arbeit vorgestellten Verfahren ist es, innerhalb eines semantischen Netzes zu einem oder mehreren vorgegebenen Begriffen thematisch verwandte Begriffe zu finden, um daraus beispielsweise eine bessere Suchanfrage für eine Volltext-Suchmaschine herzuleiten. Dieses Ziel soll mit den Methoden des Konnektionismus, insbesondere mit Hopfield-Netzen, erreicht werden. Die Hoffnung dabei ist, ein besseres Zeitverhalten und/oder bessere Wortklassen als bei graphentheoretischen Algorithmen und den bisherigen auf Hopfield-Netzen basierenden Algorithmen, wie den oben erwähnten experimentellen Systemen von Chen und Chung et al. zu erhalten.

Die Technik, Anfragen für Suchmaschinen mit Begriffen aus dem Kontext zu erweitern, wird *Query Expansion* genannt. Es sind auch weitere Anwendungen für die in dieser Arbeit gezeigten Verfahren denkbar, beispielsweise automatische Indexierung von Texten, wie von Chung in [4] angewendet, oder automatische Textzusammenfassungen.

1.3 Gliederung

In Kapitel 2 werden die konventionellen Techniken des Information Retrieval vorgestellt, angefangen von invertierten Indizes bis zu Clustering und neuen Entwicklungen im Bereich der Metasuchmaschinen.

In Kapitel 3 werden neuronale Netze und im speziellen Hopfield-Netze beschrieben. Außerdem wird eine neue Variante des Algorithmus zum Berechnen von Hopfield-Netzen vorgestellt, der effizienter als die bisherigen Varianten arbeitet.

Semantische Netze sind der Hauptgegenstand der im Rahmen dieser Arbeit betrachteten Algorithmen. Semantische Netze und Algorithmen zur Traversierung semantischer Netze mittels assoziativen neuronalen Netzen werden in Kapitel 4 beschrieben.

Die konkrete Implementierung der betrachteten Algorithmen zur Traversierung von und zur Themenextraktion aus semantischen Netzen wird in Kapitel 5 vorgestellt.

Abschließend werden in Kapitel 6 die erzielten Ergebnisse ausgewertet und es wird ein kurzer Ausblick auf mögliche zukünftige Weiterentwicklungen gegeben.

1.4 Hintergrund

Der Kontext der vorliegenden Arbeit ist das Projekt K-Portal der Abteilung IT-Research der Dresdner Bank in Zusammenarbeit mit den Firmen Temis, Avolon und H.A.S.E GmbH. Das Projekt K-Portal hat sich zum Ziel gesetzt, eine Software zum Betrieb einer Portal-Site zu entwickeln, deren Effizienz die heute gebräuchlichen Systeme durch den Einsatz computerlinguistischer Verfahren übersteigt. Ein Modul des Gesamtsystems ist dabei die im Rahmen dieser Diplomarbeit erstellte Software zur Themenextraktion aus semantischen Netzen. Abbildung 1.1 zeigt die Einbettung in den Anwendungskontext der Suchanfragemodifiktation.

Derzeit existiert bei der Dresdner Bank mit dem System ATHEM[1] bereits eine Software zur Generierung semantischer Netze, sowie ein experimentelles Tool zur Visualisierung und interaktiven Traversierung. Die für diese

[1] ATHEM = Automatischen Themenmediator für das Intranet

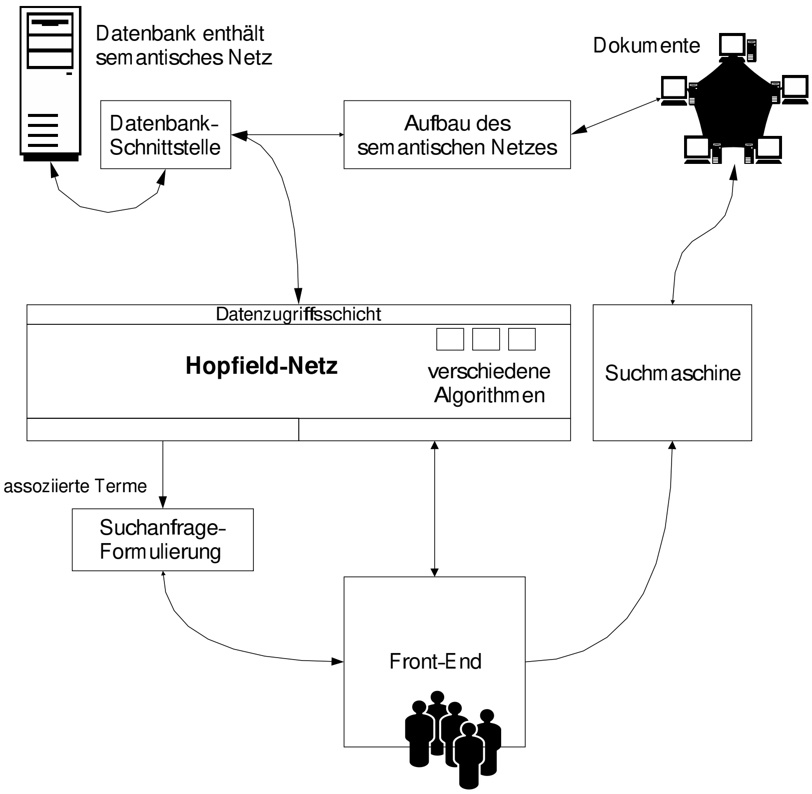

Abbildung 1.1: Anwendung zur Suchanfrage-Erweiterung

Arbeit verwendeten Eingabedaten wurden mit ATHEM generiert. Textbasis war das Intranet der Dresdner Bank, vorwiegend Texte aus dem Umfeld Finanzen, Börse, aber auch Informationstechnologie oder Politik.

1.5 Symbole und Variablen

Im folgenden werden einige in dieser Arbeit verwendeten Variablen und Symbole und ihre Bedeutung aufgelistet. Die hier erwähnten Bezeichner sind für alle nachfolgenden Kapitel relevant und werden im weiteren Verlauf stets in derselben Bedeutung verwandt.

i, j – Knoten in einem semantischen Netz bzw. Neuronen

g_{ij} – Gewicht der Kante von Knoten i zu Knoten j

w_{ik} – Gewicht von Term k in Text i

z_j – Aktivierungszustand von Knoten j

s_j – Summe $\sum_i g_{ij} * z_i$

Θ – Schwellwert aller Neuronen

Θ_j – Schwellwert des Neurons j

Kapitel 2

Konventionelle Suchmaschinen

Bevor auf die neuen Rechercheverfahren mit semantischen Netzen eingegangen wird, die im Rahmen dieser Diplomarbeit weiterentwickelt und implementiert wurden, soll zunächst ein Überblick über die Funktionsweise der traditionellen Suchmaschinen gegeben werden. Im folgenden werden einige Maßzahlen eingeführt, die später in dieser Arbeit referenziert werden. Leser, die mit den grundlegenden Techniken des Information Retrieval vertraut sind, können dieses Kapitel überspringen.

Die Technik des *Information Retrieval* ist viel älter als das WWW und der Begriff der Suchmaschine selbst. Im Bibliothekswesen stand man schon viel früher vor dem Problem, Artikel, Bücher und Schriftstücke anhand von Stichworten aufzufinden und entwickelte die ersten Information-Retrieval-Systeme. Die grundlegende Technik des „invertierten Index“ ist seitdem unverändert geblieben. Die Beschreibung der traditionellen Methoden bis einschließlich Dokument-Clustering stützt sich im wesentlichen auf die Bücher von Salton und McGill [35, 36].

2.1 Grundfunktionen des Information Retrieval

Die Grundfunktionen des Information Retrieval ist das Auffinden von Dokumenten zu einer Anfrage. Die aufgefundenen Dokumente sollen dabei in gewisser Weise „ähnlich“ zu der Anfrage sein. Das Auffinden der Dokumente kann also als ein Ähnlichkeitsvergleich zwischen dem Dokument und der Anfrage mit anschließender Ausgabe von als ausreichend ähnlich erkannten Texten aufgefaßt werden. Dabei gehen meist Zwischenschritte voraus, die eine vom Benutzer angegebene Anfrage zunächst in formale Suchanweisungen umsetzen. Auch wird der Vergleich normalerweise allein schon aus Performancegründen nicht direkt mit den Dokumenten selbst, sondern mit einem Dokumentenindex durchgeführt. Eine Möglichkeit, wie der Ähnlichkeitsvergleich häufig durchgeführt wird, ist über einen sogenannten *invertierten Index*. Dieses Verfahren ist in Abschnitt 2.2 beschrieben. Die automatische Erstellung des Dokumentenindex' wird in Abschnitt 2.3 beschrieben.

Figure 8.1 Conceptual text retrieval. (a) Conceptual information retrieval. (b) Expanded text-retrieval system.

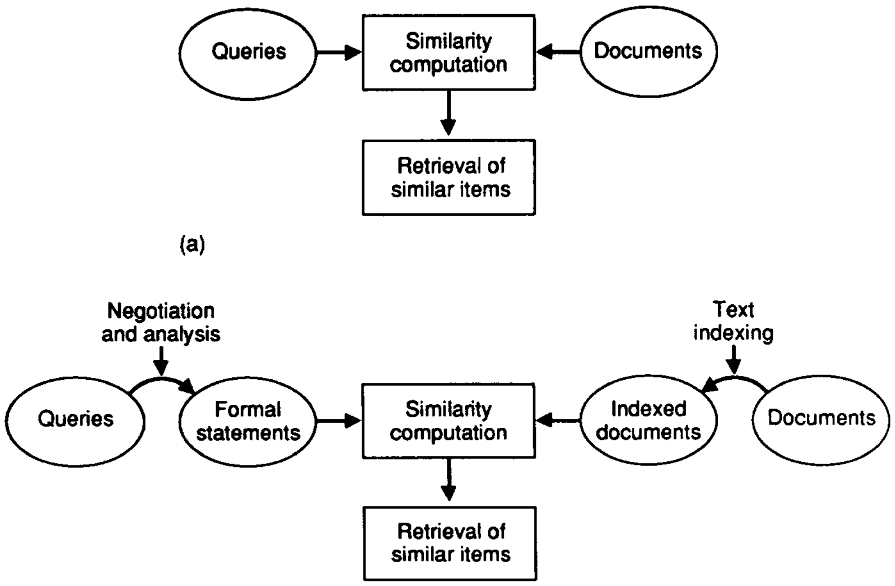

Abbildung 2.1: Anforderungen an ein Information-Retrieval-System (aus [35, S. 231])

2.1.1 Recall und Precision

Die Forderungen an das Information-Retrieval-System sind dabei, daß zum einen viele Dokumente gefunden werden, am besten alle, die zu der Anfrage passen. Auf der anderen Seite sollen aber auch keine irrelevanten Dokumente gefunden werden. Wie gut ein System diese Anforderungen erfüllt, läßt sich über die Maßzahlen *Recall* und *Precision* ausdrücken. Der Recall-Wert drückt dabei die Fähigkeit aus, *alle* relevanten Dokumente zu finden. Der Precision-Wert drückt die Fähigkeit aus, *nur* relevante Dokumente zu finden.[1] Sei a die Anzahl der gefunden Dokumente, die als relevant eingestuft wurden, b die Anzahl der gefunden Dokumente, die als nicht relevant eingestuft wurden und d die Anzahl der als relevant eingestuften Dokumente,

[1] vgl. [36, Seite 172]

die aber nicht gefunden wurden, so lassen sich Recall R und Precision P wie folgt ausdrücken:[2]

$$R = \frac{a}{a+d} \tag{2.1}$$

$$P = \frac{a}{a+b} \tag{2.2}$$

Recall ist also der Anteil der gefundenen Dokumente von allen relevanten Dokumenten, Precision der Anteil der relevanten Dokumente von allen gefundenen Dokumenten.

Precision und Recall sind gegenläufige Ziele, obwohl sich der Benutzer natürlich sowohl einen hohen Recall- als auch einen hohen Precision-Wert wünscht. Wenn aber ein hoher Recall-Wert gewünscht wird, also möglichst viele Dokumente, die vielleicht relevant sind, ausgegeben werden, ist klar, daß dadurch auch das „Rauschen“ steigt. Es schleichen sich dann verhältnismäßig viele irrelevante Dokumente in das Ergebnis ein, die Precision sinkt. Wird eine hohe Precision gewünscht, erkauft man sich dies häufig umgekehrt damit, daß viele ebenfalls relevante Dokumente nicht aufgefunden werden. Bei den heutigen Verhältnissen, in denen die Zahl der Dokumente im Internet oder auch einem großen Intranet beständig zunimmt, ist die Precision jedoch sehr viel wichtiger als der Recall. Wie bereits in der Einleitung erwähnt, besteht das Problem heute nicht mehr darin, überhaupt Dokumente zu finden, sondern unter der Vielzahl von Dokumenten die relevanten herauszufiltern.

Auf die Maßzahlen Recall und Precision wird im folgenden an geeigneter Stelle verwiesen werden.

2.2 Invertierter Index

Aus Gründen der Performance und der Handhabbarkeit wird man einen Ähnlichkeitsvergleich zwischen der Benutzeranfrage und den Dokumenten[3] nicht über den Dokumenten selbst, sondern über einem *Index* durchführen. Zur Erstellung eines Index legt man zunächst eine Reihe von Begriffen fest, die zur Indexierung dienen und nach denen man später suchen kann. Die Festlegung dieses Vokabulars kann auch, wie im nächsten Abschnitt gezeigt, automatisch geschehen. Anschließend betrachtet man alle Dokumente und vermerkt, welche Begriffe sie enthalten. Das Ergebnis ist eine Tabelle (siehe Tabelle 2.1), bei der die Zeilen den Dokumenten und die Spalten den Begriffen entsprechen. Eine 1 in einer Zelle gibt dabei an, daß ein Begriff im Dokument enthalten ist.

Die Tabelle wird nun derart „invertiert“ (eigentlich transponiert), daß jede Zeile einem Begriff entspricht und in einer solchen Zeile abgelesen wer-

[2] vgl. [35, Seite 248]

[3] vgl. Abschnitt 2.1

	Begriff 1	**Begriff 2**	**Begriff 3**	**Begriff 4**
Text 1	1	1	0	1
Text 2	0	1	1	1
Text 3	0	0	1	1
Text 4	0	0	1	1

Tabelle 2.1: Dokumentenindex

den kann, in welchen Dokumenten ein bestimmter Begriff auftaucht. Das Ergebnis ist ein invertierter Index wie in Tabelle 2.2.

	Text 1	**Text 2**	**Text 3**	**Text 4**
Begriff 1	1	0	1	0
Begriff 2	1	1	0	0
Begriff 3	0	1	1	1
Begriff 4	1	1	1	1

Tabelle 2.2: invertierter Index

Für eine einfache Anfrage reicht es nun aus, eine Zeile dieser Tabelle zu lesen. Werden z.B. alle Dokumente gesucht, in denen Begriff 3 vorkommt, liest man in der entsprechenden Zeile alle Einsen aus und findet die Texte 2–4. Fast genau so einfach ist das Suchen nach booleschen Verknüpfungen von Indexbegriffen. Werden Dokumente gesucht, die Begriff 2 UND Begriff 3 enthalten, vergleicht man die beiden Zeilen und gibt nur die Dokumente aus, deren Spalten in beiden Zeilen mit 1 belegt sind. In diesem Beispiel wird also nur Text 2 gefunden. Analog werden für eine ODER-Verknüpfung alle Texte gefunden, in deren Spalte in mindestens einer Zeile eine 1 steht. Für Negationen, wie z.B. „Begriff 2 UND NICHT Begriff 3“, wird die entsprechende Tabellenzeile negiert und anschließend wie bei der UND-Verknüpfung verfahren. Im Beispiel wird also nur Text 1 gefunden.

2.2.1 Erweiterungen des invertierten Index

Wie gezeigt können bereits mit der einfachen Variante des invertierten Index, die bis heute die Basis für alle gängigen Suchmaschinen darstellt, auf effiziente Art und Weise boolesche Suchanfragen bearbeitet werden. Die Ergebnismenge ist jedoch ungeordnet. Es kann allenfalls eine Sortierung nach Aktualität vorgenommen werden, wenn neue Dokumente in der Tabelle immer rechts angefügt werden und die Zeilen von rechts nach links ausgewertet werden. Läßt man allerdings statt 0 oder 1, entsprechend „nicht vorhanden“ oder „vorhanden“, in der Tabelle variable Gewichtungen zu, die beispielsweise über die Frequenz des Auftretens eines Begriffs in einem Dokument errechnet werden können, ist eine bessere Einschätzung der möglichen Relevanz eines gefundenen Dokumentes möglich und relevant erscheinende Dokumente können zuerst gezeigt werden. Wie man solche Gewichte errechnen kann, ist im nächsten Abschnitt (2.3) beschrieben.

Mit einer einfachen Erweiterung sind auch Abfragen nach zwei benachbarten Worten oder zwei im selben Abschnitt vorkommenden Begriffen möglich, wie sie viele gängige Suchmaschinen mit dem Operator NEAR bereitstellen. Dazu muß für jedes Auftreten eines Begriffs in einem Dokument auch die genaue Textstelle vermerkt werden, beispielsweise in dem man die Nummer des Absatzes im Dokument, des Satzes innerhalb des Absatzes und des Wortes innerhalb des Satzes speichert. Dabei kann eine Zelle der Tabelle auch mit mehreren Werten belegt sein, wenn ein Wort in einem Text mehrfach vorkommt. Tabelle 2.3 zeigt, wie der erweiterte invertierte Index dann dargestellt werden kann. Dabei entspricht die jeweils erste Ziffer dem Absatz, die zweite dem Satz und die dritte dem Wort. Betrachtet man die Begriffe 1 und 2 in Text 1, so stellt man fest, daß diese im Text direkt nebeneinander stehen. Eine Abfrage „Begriff 1 BENACHBART ZU Begriff 2“ würde also Text 1 als Ergebnis liefern. Ist ein NEAR-Operator so definiert, daß er ein Dokument als Ergebnis liefert, wenn seine beiden Operanden im gleichen Absatz stehen, so liefert „Begriff 2 NEAR Begriff 4“ nur Text 2 als Ergebnis, während die boolesche Abfrage „Begriff 2 AND Begriff 4“ zusätzlich noch Text 1 geliefert hätte, obwohl die Suchbegriffe dort vielleicht überhaupt nichts miteinander zu tun haben und in verschiedenen Absätzen stehen. Der NEAR- und auch der BENACHBART-Operator sind also Einschränkungen des AND-Operators, deshalb wird aus Gründen der Performance auch normalerweise erst ein einfacheres AND durchgeführt und anschließend werden die Dokumente ausgefiltert, für die die NEAR- bzw. BENACHBART-Bedingung nicht erfüllt ist.

	Text 1	**Text 2**	**Text 3**	**Text 4**
Begriff 1	5;3;2	—	1;3;1 3;4;2	—
Begriff 2	5;3;3	6;2;7 7;1;12 9;3;4	—	—
Begriff 3	—	1;1;5	3;7;1	5;1;1
Begriff 4	7;1;9	7;1;5	2;2;3	5;7;9

Tabelle 2.3: erweiterter invertierter Index

2.3 Automatisches Indexieren

Eine der Hauptaufgaben des Indexierens ist die Erstellung eines geeigneten Indexvokabulars. Dieses Vokabular kann von Hand erstellt werden. Dies wurde in der Anfangsphase des Information Retrieval auch tatsächlich so gehandhabt. Für kleine Bibliotheken oder spezialisierte Dokumentsammlungen mit Texten aus nur einem Fachgebiet ist dieses Vorgehen auch gerade noch praktikabel. Bei der heutigen Informationsflut, der sich die Internet-Suchmaschinen gegenübergestellt sehen ist jedoch eine automatische Indexierung ein absolutes Muß. Außer der Festlegung des Vokabulars soll auch die Vergabe eines Gewichtswertes, der die Bedeutsamkeit eines Indexwortes für das Dokument widerspiegelt (vgl. Abschnitt 2.2.1), automatisch erfolgen. Ein Indexwort, das ein bestimmtes Dokument beschreibt, wird als *Deskriptor* bezeichnet.

Nicht jedes Wort, das in einem Text vorkommt, eignet sich gleich gut als Deskriptor für dieses Dokument. Der Deskriptor soll ja nicht nur für einen guten *Recall* sorgen, sondern das Dokument auch so gut beschreiben, daß eine gute *Precision* erreicht wird. Soll beispielsweise eine Sammlung von technischen Dokumenten über das Innenleben von Computern indexiert werden, wird sich das Wort „Computer“ kaum als Deskriptor eignen, da dieses Wort wahrscheinlich in jedem Text sogar mehrfach vorkommen dürfte. In einer Sammlung von Texten zur Technikgeschichte von der Dampfmaschine bis zur Gegenwart kann dagegen „Computer“ ein guter Deskriptor sein. Bei einer breiten Textbasis, wie sie von den derzeitigen Suchmaschinen verarbeitet wird, ist es außerdem äußerst wichtig, eine gute Einschätzung dafür zu finden, ob ein Suchbegriff in einem Text nur mehr oder weniger zufällig vorkommt (wenn beispielsweise in einem Text über Waschmaschinen von „Computer-“ Steuerung die Rede ist) oder ob der Begriff das Dokument wirklich treffend beschreibt (wie das Wort „Computer“ bspw. einen Vergleichstest von PCs verschiedener Hersteller zwar allgemein, aber dennoch passend beschreibt). Die Frage, ob ein Wort ein guter oder ein schlechter Deskriptor ist, kann also nicht allgemein beantwortet werden, sondern man muß den Deskriptor stets im Kontext der Texte betrachten, die er beschreiben soll. In den folgenden Abschnitten werden verschiedene Methoden vorgestellt, wie eine Gewichtung der Bedeutsamkeit von Indexbegriffen für Dokumente automatisch vorgenommen werden kann.

2.3.1 Termfrequenz inverse Dokumentfrequenz

Man kann davon ausgehen, daß ein Begriff ein Dokument in einem gewissen Maße beschreibt, wenn dieser Begriff häufig im Text vorkommt. Andererseits wird, wie im vorangegangen Abschnitt erläutert, ein Wort denselben Text nicht treffend beschreiben, wenn dieses Wort in sehr vielen Texten auftritt. Diese beiden Grundannahmen reichen bereits aus, um eine heuristische Gewichtungsfunktion zu formulieren. Das Gewicht soll dabei proportional zur Häufigkeit des Auftretens eines Wortes in einem Text und umgekehrt proportional zur Anzahl der Dokumente, in denen dieses Wort vorkommt, sein.[4] Sei k ein Begriff, n die Gesamtzahl aller Dokumente, und d_k die Zahl der Dokumente, in denen k vorkommt. Dann ist

$$v_k = \log_2(\frac{n}{d_k}) = \log_2(n) - \log_2(d_k) + 1 \tag{2.3}$$

eine Maßzahl für die inverse Dokumenthäufigkeit, die in umgekehrt proportionalem Verhältnis zur Anzahl der Dokumente, in denen k vorkommt, steht. Der Logarithmus verhindert dabei, daß die Werte für v_k zu stark auseinanderdriften. Würde nur der Quotient $\frac{n}{d_k}$ verwendet, würden in nur wenigen Dokumenten vorkommende Worte stark überbewertet. Sei i ein Dokument und f_{ik} die Häufigkeit des Auftretens oder die *Frequenz* von Wort k in Text i. Um zu einer guten Gewichtsfunktion zu kommen, multipliziert man nun einfach die inverse Dokumenthäufigkeit mit der Frequenz. Demnach ist

$$w_{ik} = f_{ik} \cdot v_k = f_{ik} \cdot [\log_2(n) - \log_2(d_k) + 1] \tag{2.4}$$

eine geeignete Gewichtsfunktion.

2.3.2 Informationstheoretischer Ansatz

Die relative Häufigkeit von Worten in Texten ist ungleich verteilt. Der Empfänger einer Nachricht hat also in Form dieser Wahrscheinlichkeiten eine gewisse Vorabinformation (*A priori-Wissen*) darüber, was er als nächstes empfangen wird. Nach C.E. Shannon, dem Begründer der Informationstheorie, läßt sich aus der Wahrscheinlichkeit des Auftretens eines Wortes ein *Informationsgehalt* berechnen, der um so größer ist, je kleiner die Wahrscheinlichkeit des Wortes ist. Dies erscheint logisch, wenn man sich den umgekehrten Fall vorstellt: Wären die Häufigkeiten aller Wörter in der Sprache gleich verteilt, so wäre ein Text nur eine homogene Anhäufungen von Wörtern. Nur Abweichungen von dieser Homogenität vermitteln dem Leser eine tatsächliche Information. Sei p_k die Wahrscheinlichkeit, daß Wort k auftritt, so errechnet sich nach Shannon der Informationsgehalt eines Wortes I_k als

$$I_k = -\log_2 p_k \tag{2.5}$$

Betrachten wir ein Dokument i, daß von t Deskriptoren beschrieben wird, so läßt sich daraus eine *Durchschnittsinformation* D_i wie folgt berechnen:

$$D_i = -\sum_{k=1}^{t} p_k \log_2 p_k \tag{2.6}$$

[4]vgl. [36, Seite 68 f.]

Daraus läßt sich ein *Rausch-* oder *Ballast*-Wert ableiten[5], der maximal wird, wenn ein Begriff in jedem Dokument vorkommt. Wie bereits erwähnt, sind solche Begriffe schlecht als Deskriptoren geeignet, die in vielen Dokumenten vorkommen. Sei n die Anzahl der Dokumente, f_{ik} wie oben die Frequenz von Wort k in Text i und t_k die Summe aller Häufigkeiten von k in allen Texten, also $t_k = \sum_{i=1}^{n} f_{ik}$, errechnet sich das Rauschen eines Wortes r_k als:

$$r_k = \sum_{i=1}^{n} \frac{f_{ik}}{t_k} \log_2 \frac{t_k}{f_{ik}} \tag{2.7}$$

Hieraus läßt sich nun der *Informationswert* eines Wortes s_k, auch *Signal* genannt, errechnen, der ähnliche Eigenschaften wie die oben erwähnte inverse Dokumentfrequenz hat.

$$s_k = \log_2(t_k) - r_k \tag{2.8}$$

Um eine Gewichtsfunktion zu erhalten, multipliziert man wie auch bei der inversen Dokumentfrequenz mit der Frequenz des Wortes im Text.

$$w_{ik} = f_{ik} \cdot s_k \tag{2.9}$$

Diese Gewichtsfunktion (2.9) ist aufwendiger zu berechnen als die Funktion über die inverse Dokumenthäufigkeit (2.4). Salton führt in [35, S. 281] aus, daß durch den informationstheoretischen Ansatz gegenüber dem Ansatz über die inverse Dokumenthäufigkeit nur wenig bessere Ergebnisse erzielt werden.

2.3.3 Diskriminanzwert

Auch der Ansatz über den *Diskriminanzwert* nutzt aus, daß es, wie oben definiert, gute und schlechte Indexbegriffe gibt, die entweder die Dokumente gut voneinander unterscheiden oder aber in vielen Dokumenten gleich häufig auftreten. In anderen Worten formuliert bedeutet das, wenn man die Ähnlichkeit von Dokumenten anhand der ihnen zugewiesenen Indexbegriffe beurteilt, daß die Dokumente bei Zuweisung eines schlechten Indexbegriffs einander ähnlicher werden, bei Zuweisung eines guten Indexbegriffs sich dagegen stärker voneinander unterscheiden. Um die Ähnlichkeit mehrerer Dokumente zueinander zu erfassen, führt man eine Maßzahl *Dokumentendichte* Q ein. Der Diskriminanzwert[6] eines Indexbegriffes d_i ist dann die Differenz der Dichte vor (Q_0) und nach (Q_1) der Zuweisung des Indexbegriffes.

$$d_i = Q_0 - Q_1 \tag{2.10}$$

Ein Diskriminanzwert $d_i < 0$, also eine Erhöhung der Dichte, weist darauf hin, daß die Dokumente ähnlicher gemacht werden. Ist $d_i > 0$, verringert sich also die Dichte, eignet sich das Wort dagegen gut als Indexbegriff.

[5] vgl. [36, Seite 70 f.]

[6] nach [35, Seite 282]

Die Dichte einer Dokumentensammlung mit n Dokumenten kann unter Benutzung der paarweisen Ähnlichkeit zweier Dokumente D_i und D_k wie folgt berechnet werden:

$$Q = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{\substack{k=1 \\ i \neq k}}^{n} \mathrm{sim}(D_i, D_k) \tag{2.11}$$

Sei t die Anzahl der Indexierungsbegriffe und g_{ij} wie oben die Gewichtung von Begriff i in Dokument j, kann die Funktion $\mathrm{sim}(D_i, D_k)$ kann dabei definiert werden als:

$$\mathrm{sim}(D_j, D_k) = \sum_{i=1}^{t} w_{ij} \cdot w_{ik} \tag{2.12}$$

Dies entspricht dem inneren Produkt bei Vektoren. In der Tat können sowohl Dokumente als auch Indexbegriffe als Vektoren aufgefaßt werden. Die Tabelle 2.2 wird dabei als eine Matrix A betrachtet:

$$A = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \\ w_{41} & w_{42} & w_{43} & w_{44} \end{bmatrix} \tag{2.13}$$

Zeilenvektoren entsprechen dabei dem Vektor für einen Indexbegriff, Spaltenvektoren entsprechen einem Dokumentvektor.

Betrachtet man Formel 2.12, stellt man fest, daß für Dokumente, denen viele Indexbegriffe zugeordnet sind, stets ein größerer Ähnlichkeitswert errechnet wird als für Dokumente, denen nur wenige Begriffe zugeordnet sind. Dies kann gewollt sein, allerdings führt dies auch zu einer Ungleichbehandlung von großen und kleinen Dokumenten, da große Dokumente ja meist automatisch mehr Indexbegriffe zugeordnet bekommen als kleine. Daher kann man die Ähnlichkeitsfunktion auch wie folgt normalisieren:

$$\mathrm{sim}(D_j, D_k) = \frac{\sum_{i=1}^{t} w_{ij} \cdot w_{ik}}{\sqrt{\sum_{i=1}^{t} (w_{ij})^2 \cdot \sum_{i=1}^{t} (w_{ik})^2}} \tag{2.14}$$

Diese Funktion entspricht dem Cosinus des Winkels zwischen den beiden Dokumentvektoren in einem t-dimensionalen Raum. Außer den beiden Maßzahlen nach Formel 2.12 und 2.14 können noch der Dice-Koeffizient[7],

$$\mathrm{sim}(D_j, D_k) = \frac{\sum_{i=1}^{t} w_{ij} \cdot w_{ik}}{\sum_{i=1}^{t} (w_{ij})^2 + \sum_{i=1}^{t} (w_{ik})^2} \tag{2.15}$$

[7] vgl. Tabelle 10.1 in [35, Seite 318]

der Jaccard-Koeffizient[7]

$$\mathrm{sim}(D_j, D_k) = \frac{\sum_{i=1}^{t} w_{ij} \cdot w_{ik}}{\sum_{i=1}^{t}(w_{ij})^2 + \sum_{i=1}^{t}(w_{ik})^2 - \sum_{i=1}^{t}(w_{ij} \cdot w_{ik})} \tag{2.16}$$

und der Overlap-Koeffizient[8]

$$\mathrm{sim}(D_j, D_k) = \frac{\sum_{i=1}^{t} w_{ij} \cdot w_{ik}}{\min(\sum_{i=1}^{t}(w_{ij})^2, \sum_{i=1}^{t}(w_{ik})^2)} \tag{2.17}$$

als normalisierte Maße für die Vektor-Ähnlichkeit verwendet werden. Rijsbergen schreibt in [33, Seite 25], daß die Normalisierung notwendig ist, um unintuitive Resultate zu vermeiden.

Eine Gewichtfunktion unter Verwendung der hier angegebenen Ähnlichkeitsfunktionen lautet dann wie folgt:

$$w_{ij} = f_{ij} \cdot d_i \tag{2.18}$$

Dabei ist f_{ij} die Frequenz und d_i der Diskriminanzwert nach Formel 2.10.

Die in diesem Abschnitt angegebenen Ähnlichkeitsfunktionen können nicht nur für die Bestimmung von Deskriptorgewichten verwendet werden, sondern sind auch zum Dokument-Clustering (siehe Abschnitt 2.4) notwendig.

2.3.4 Probabilistischer Ansatz

Eine weitere Möglichkeit, geeignete Gewichtswerte zu ermitteln, ist durch einen probabilistischen Ansatz. Die bisher vorgestellten Methoden bestimmen das Indextermgewicht nur über Häufigkeitsbetrachtungen. Die Diskriminanzwert-Methode beispielsweise versucht, die Gewichte so zu vergeben, daß eine möglichst große Distanz zwischen benachbarten Dokumenten entsteht, wobei benachbart im Sinne der oben angegebenen Ähnlichkeitsfunktionen zu verstehen ist. Ziel des probabilistischen Ansatzes ist dagegen, eine große Distanz zwischen relevanten und nicht relevanten Dokumenten zu erreichen. Salton stellt in [35] Gleichungen zur probabilistischen Termgewichtung auf, schreibt aber, daß das Problem an diesem Ansatz ist, im Vorfeld zwischen relevanten und nicht relevanten Dokumenten zu unterscheiden. Diese Unterscheidung kann durch ein Feedback nach Durchführung einer Abfrage erreicht werden, jedoch ist nicht klar, ob bei wenigen durchgeführten Abfragen die Unterscheidung zwischen relevanten und nicht relevanten Dokumenten bereits stabil genug ist.

[8] vgl. [33, Seite 25]

2.4 Dokument-Clustering

Wie bereits in Abschnitt 2.3.3 skizziert kann ein Dokument als ein Vektor aufgefaßt werden, dessen Komponenten die Termgewichte sind. Über die Gleichungen 2.14 bis 2.17 läßt sich eine Ähnlichkeit zwischen zwei Dokumenten im Dokumentenraum beschreiben. Dies ist die Basis für automatische Klassifizierung von Dokumenten und sogenanntes Dokument-Clustering. Bei diesen Verfahren werden die Dokumente in Gruppen (Cluster) zusammengefaßt, die jeweils einem Thema zugeordnet sind, so wie in einer Bibliothek die Bücher nach Themen geordnet sind.

Die Idee ist, dem Benutzer die Navigation dadurch zu erleichtern, daß zu einem Thema gehörende Dokumente auch zusammen präsentiert (und ggf. auch zusammen abgespeichert) werden. Grundlage ist die von Rijsbergen in [33] formulierte „Cluster Hypothesis“, die besagt, daß (im Dokumentenraum) nahe beieinanderliegende Dokumente wahrscheinlich für dieselben Anfragen relevant sind.

2.4.1 Hierarchisches Clustering

Oft wird das Clustering hierarchisch vorgenommen. Auf der untersten Ebene wird eine kleine Zahl von Dokumenten zu einem Cluster zusammengefaßt. Auf der nächsthöheren Ebene faßt man mehrere dieser Cluster zu einem Supercluster zusammen. Durch eine solche Hierarchie lassen sich Suchanfragen effizient beantworten. Eine Anfrage wird mit Repräsentanten der Cluster der höchsten Ebene verglichen, und nur die Cluster mit der höchsten Ähnlichkeit werden weiter betrachtet. Innerhalb dieser wird ebenfalls ein Ähnlichkeitsvergleich durchgeführt, bis man letztendlich auf der untersten Ebene die Dokumente findet. Bei einer Suche mit hierarchischen Clustern kann auf die klassische Suche mit invertiertem Index verzichtet werden. Für die Ähnlichkeitsvergleiche wählt man als Repräsentanten oft statt des Vektors eines tatsächlichen Dokuments einen errechneten *Centroiden*, der im geometrischen Zentrum des Clusters liegt und somit das „Durchschnittsdokument“ dieses Clusters ist. Entsprechend läßt sich der Centroidvektor auch einfach als Durchschnittswert berechnen. Sei m die Anzahl der Dokumente im Cluster, C_k die k-te Komponente des Centroidvektors C und w_{ik} wie oben das Gewicht von Term k in Dokument i, so gilt:[9]

$$C_k = \frac{1}{m} \sum_{i=1}^{m} w_{ik} \tag{2.19}$$

[9] vgl. [36, Seite 231]

Es kann zweckmäßig sein, den Centroidvektor durch Division durch seine Länge auf einen Wertebereich für seine Komponenten von 0 bis 1 zu bringen.[10] Die Länge L_C des Centroidvektors ist (bei t Komponenten)

$$L_C = \sqrt{\sum_{i=1}^{t} C_i^2} \tag{2.20}$$

Zur Generierung einer hierarchischen Clusterstruktur gibt Salton in [35] einen einfachen Algorithmus (siehe Algorithmus 1) an.

Algorithmus 1 Generierung einer hierarchischen Clusterstruktur

```
Berechne alle paarweisen Dokumentähnlichkeiten
Plaziere zunächst jedes Dokument in einen eigenen Cluster
while noch mehr als 1 Cluster vorhanden do
  Fasse Cluster, deren paarweise Ähnlichkeit über einem Schwellwert Θ
  liegt, zu einem neuen Cluster zusammen
  Berechne zu dem neuen Cluster neue paarweise Ähnlichkeiten
end while
```

Wählt man als Schwellwert Θ die größte auftretende paarweise Ähnlichkeit, so spricht man von *single link clustering*. Andere Möglichkeiten sind denkbar.[11]

2.4.2 Klassifizierung

Automatische Klassifizierung funktioniert ähnlich wie Clustering. Der wesentliche Unterschied ist allerdings, daß die Klassen, die zur Einteilung der Dokumente verwendet werden, vorgegeben sind und sich nicht erst aus den Dokumenten ergeben. Beispiele für Klassifizierungsaufgaben sind beispielsweise die Einteilung der Meldung einer Nachrichtenagentur in die Bereiche Politik, Wirtschaft, Sport und Panorama oder, wie oben schon erwähnt, die Eingruppierung eines Buches in die Themenbereiche einer Bibliothek. Eine moderne Anwendung ist die Zuordnung einer eingehenden E-Mail zu einem Geschäftsbereich.

Die der Klassifizierung zugrundeliegende Ähnlichkeitsberechnung kann analog zu der Berechnung beim Clustering verstanden werden. Zur Berechnung der Centroiden der Klassen, die für den Vergleich herangezogen werden, müssen zunächst ausreichend viele Beispieldokumente bereitgestellt werden. Wird dann im laufenden Betrieb ein *Feedback* durchgeführt, bei dem man falsch zugeordnete Dokumente von Hand in die richtigen Klassen einteilt, verbessert sich die Leistung des Systems.

[10] vgl. [36, Seite 231]

[11] Salton beschreibt die Clustergenerierung in [35] ab Seite 330.

Das Problem der Klassifizierung kann auch als Lernaufgabe angesehen und mit neuronalen Netzen und Lernalgorithmen gelöst werden. Nach dem bisherigen Erkenntnisstand sind die klassischen Clustering-Methoden allerdings tendentiell besser geeignet.

2.5 Neuere Entwicklungen

2.5.1 Metasuchmaschinen

In den vergangenen Jahren hat der Umfang des WWW so stark zugenommen, daß es für die einzelnen Suchmaschinen immer schwieriger wird, auch nur einen Teil des gesamten Web zu indizieren. Lawrence und Giles [20] schätzten in einer im April 1998 veröffentlichten Studie eine untere Grenze für die Größe des WWW auf 320 Millionen Seiten. Die 1998 umfangreichste Suchmaschine, HotBot, erfaßte 34% dieser Seiten. Einer neueren Studie derselben Autoren [21] aus dem Jahre 1999 zufolge stieg der Umfang inzwischen auf 800 Millionen Seiten. Die nunmehr umfangreichste Suchmaschine, Northern Light, erfaßt nur noch 16% des WWW. Dieser Mißstand führte zur Entwicklung von Metasuchmaschinen, die ihrerseits verschiedene Suchmaschinen und Kataloge parallel abfragen und die Ergebnisse gesammelt präsentieren. Da die von den Suchmaschinen erfaßten Bereiche sich nur zu einem Teil überschneiden, läßt sich durch Kombination der sechs größten Suchmaschinen der abgedeckte Bereich um den Faktor 3,5 erhöhen.[12]

Nicht jedes Formular zur Abfrage mehrerer Suchmaschinen ist jedoch eine echte Metasuchmaschine. Auf der Tagung „Internet Society“ im Juli 1998 in Genf wurden die Forderungen an Metasuchmaschinen erstmals formuliert:[13]

Parallele Suche – Die verschiedenen Suchmaschinen müssen parallel abgefragt werden

Ergebniszusammenführung – Die Ergebnisse der einzelnen Abfragen müssen zusammengefaßt und in einem einheitlichen Format dargestellt werden.

Doubletten-Eliminierung – Von mehreren Suchmaschinen gelieferte Ergebnisse sollen nicht mehrfach erscheinen.

Boolesche Operatoren – Mindestens AND und OR-Operatoren müssen für die Abfrage zur Verfügung stehen.

Kein Informationsverlust – Die Informationen, die eine Suchmaschine liefert, wie z.B. eine Kurzbeschreibung der Fundstelle, müssen auch angezeigt werden.

---
[12] vgl. [20]
[13] vgl. [37]

Vereinheitlichte Syntax – Die spezifischen Eigenschaften der verwendeten Suchmaschinen dürfen für die Bedienung der Metasuchmaschine keine Rolle spielen (*search engine hiding*).

Vollständige Suche – Die Metasuchmaschine soll so lange suchen, wie eine der verwendeten Suchmaschinen noch Fundstellen liefert.

Es braucht eigentlich nicht erwähnt zu werden, daß einige der als Metasuchmaschinen ausgewiesenen Dienste nicht alle der hier aufgeführten Kriterien erfüllen. Nach [37] erfüllten 1998 nur der auf deutschsprachige Webseiten spezialisierte, vom regionalen Rechenzentrum Niedersachsen an der Uni Hannover entwickelte und betriebene Dienst MetaGer[14] und Highway61 [15] alle sieben Kriterien. Die populäre Suchmaschine Metacrawler [16] erfüllt die ersten sechs Kriterien.

Eine Metasuchmaschine dient, wie erklärt wurde, also vor allem dazu, mehr Dokumente zu finden. Oder anders ausgedrückt, durch eine Metasuchmaschine läßt sich der *Recall*-Wert verbessern. Öfter wird man jedoch vor dem Problem stehen, unter den vielen, nach einer Suchanfrage präsentierten Dokumenten, die wirklich relevanten herauszufiltern. Der *Precision*-Wert wird durch Metasuchmaschinen allerdings nicht deutlich gesteigert. Letzteres kann man mit *Query-Expansion*-Techniken erreichen, wie sie im Abschnitt 2.5.4 beschrieben werden und deren Verbesserung auch Ziel dieser Arbeit ist.

2.5.2 Suchmaschinen 3. Ordnung

Um gleichzeitig möglichst viele Dokumente zu erfassen und auch die Präzision zu erhöhen, arbeiten die Betreiber von MetaGer an der Uni Hannover an einer sogenannten Level-3-Suchmaschine. Auf eine Benutzerabfrage hin wird eine Metasuchmaschine nach Dokumenten zu dem gewünschten Thema befragt. Die Dokumente werden durch einen „Relevanzfilter“ geleitet und in einer Datenbank abgelegt. Danach kann der Benutzer spezialisierte Anfragen zu einem Thema an die Datenbank richten.[17]

2.5.3 Gewichtung über Linkstruktur

Die Suchmaschine Google geht einen neuen Weg zur Gewichtung der Ergebnisse. Zwar ist auch Google eine konventionelle Indexsuchmaschine, allerdings werden die Ergebnisse nicht nach traditionellen Gesichtpunkten gewichtet, sondern der „Page Rank“ einer Seite richtet sich nach der Anzahl

[14] `meta.rrzn.uni-hannover.de`
[15] `www.highway61.com`
[16] `www.metacrawler.com`
[17] vgl. [37]

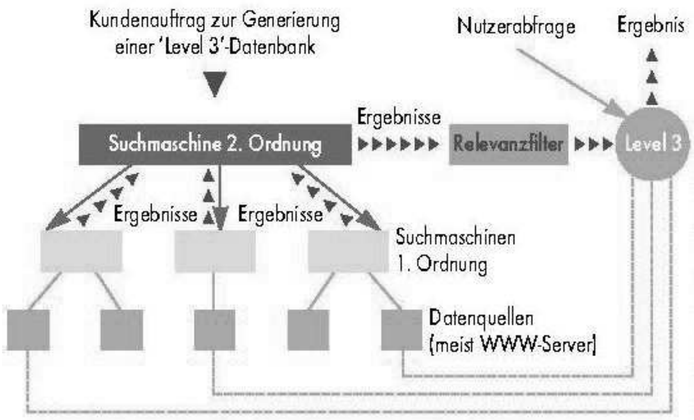

Abbildung 2.2: Suchmaschine 3. Ordnung (aus [37])

der Links, die auf diese Seite verweisen. Die Links selbst sind auch unterschiedlich gewichtet. So zählt ein Link, der von einer Seite ausgeht, die selbst einen hohen Page Rank hat stärker als einer, der von einer schwach bewerteten Seite ausgeht. Man geht davon aus, daß solche „populäre" Seiten auch relevantere Inhalte aufweisen. Kritiker wenden allerdings ein, daß durch diese Bewertung nur die ohnehin stark besuchten Seiten gestärkt und unabhängige Anbieter von Information benachteiligt würden.

Ein ähnliches Verfahren zur Gewichtung verwendet auch die Suchmaschine Direct Hit. Hier werden allerdings nicht die Hyperlinks gezählt, sondern wie oft die Benutzer der Suchmaschine die Seite angeklickt haben. Bei dieser Art des Feedbacks werden allerdings auch die ohnehin schon weit vorne stehenden Einträge bevorzugt.[18]

2.5.4 Suchanfrage-Modifikationen

Eine Möglichkeit, die Qualität der automatischen Indexierung zu verbessern, ist, die Wörter in den Dokumenten nicht als solche zu verwenden, sondern sie zunächst entsprechend zu modifizieren (*Query Modification* bzw. *Query Expansion*). Salton schlägt in [35] dazu zwei verschiedene Verfahren für hochfrequente (oft vorkommende) und niederfrequente (selten vorkommende) Wörter vor.

Phrasengenerierung

Hochfrequente Wörter, die zu allgemein sind, als daß sie sich für die Indexierung eignen, werden nicht als solche, sondern im Satzzusammenhang als Indexbegriff verwandt. Wie bereits oben erwähnt, dürfte das Wort „Computer"

[18] vgl. [34]

sich kaum als Indexterm eignen, insbesondere, wenn sich in der Dokumentensammlung anteilsmäßig gesehen viele Texte befinden, die sich in irgendeiner Weise mit dem Thema „Computer" befassen. Dieses Wort wird daher nicht als solches als Indexterm verwandt, sondern Phrasen wie bspw. „Computer-Software" oder „Computer-Wissenschaft" werden generiert. Durch dieses Vorgehen wird die *Precision* gesteigert.

Thesaurusgruppen

Niederfrequente Wörter werden dagegen mit Hilfe eines Thesaurus um sinngleiche Wörter angereichert, die dann ebenfalls als Indexbegriff dienen. Alternativ, je nach Architektur der verwendeten Suchmaschine, kann auch die Thesaurusklasse als solche dem Dokument als Indexterm zugeordnet werden. Dadurch wird der *Recall* gesteigert.

Anwendung von Phrasengenerierung und Thesaurus

Die Anwendung eines Thesaurus kann nicht nur bei der Indexierung, sondern auch bei der Suche sinnvoll sein. Im Rahmen der *Query Expansion* können zur Steigerung des *Recall* über einen Thesaurus ermittelte Begriffe über eine OR-Verknüpfung zur ursprünglichen Anfrage hinzugefügt werden, falls ein niederfrequenter Begriff eingegeben wurde.

Die Phrasengenerierung zur Steigerung der *Precision* dagegen kann zwar bei der Indexierung verwendet werden, wirft aber bei der Suche Probleme auf. Zur Phrasengenerierung muß der Kontext herangezogen werden. Bei der Suche kann es aber schwierig sein, aus dem eingegebenen Suchwort den Kontext zu rekonstruieren. Die im Rahmen dieser Diplomarbeit vorgestellten Verfahren mit semantischen Netzen bieten sich als ein Ausweg an.

Relevanz-Feedback

Eine andere, sehr effektive Methode, die Anfrage umzuformulieren, um die Precision zu erhöhen, ist das Relevanz-Feedback.[19] Dabei wird, nach einer ersten Anfrage, dem Benutzer die Möglichkeit gegeben, die bisher präsentierten Resultate als relevant bzw. irrelevant einzustufen. Aufgrund dieser Feedback-Information wird die Abfrage dann um Begriffe aus den relevanten Dokument angereichert.

Wie die Bezeichnung des Verfahrens aber schon ausdrückt, kann damit lediglich eine Abfrage *nach* der Durchführung für einen zweiten Durchgang verfeinert werden. Um bereits bei der ersten Abfrage bessere Ergebnisse zu erzielen, ist es möglich, ein *ad hoc* oder *blindes* Feedback durchzuführen. Dabei werden die von der Suchmaschine selbst als relevant eingeschätzten, also

[19] Wie z.B. von Salton und McGill in [36, Kapitel 4] beschrieben.

mit einem hohen Ranking versehenen Dokumente für das Feedback herangezogen. Da hierbei der Nutzer nicht wirklich eingreift, muß man eigentlich mehr von einem Pseudo-Feedback sprechen.[20]

Kontextbezogene Anfrage-Reformulierung

Generell soll das Hinzufügen von Termen zu einer Suchanfrage immer so geschehen, daß dabei der Kontext der Anfrage eingegrenzt wird. Dies ist auch beim Relevanz-Feedback der Fall, wobei *nach* einer ersten Anfrage aus bereits gefundenen Dokumenten Begriffe extrahiert und der ursprünglichen Anfrage hinzugefügt werden. Gewünscht wäre allerdings ein Hinzufügen von Kontext-Information bereits *vor* der ersten Anfrage.

Während beim Relevanz-Feedback nur das Thema entscheidend ist, ist bei einem anderen Ansatz, der dabei verfolgt wurde, die *Kategorie* des Textes das entscheidende Kriterium.[21] Hinzugefügt werden dabei Terme, die für eine bestimmte Art von Dokument charakteristisch sind. So soll dem Benutzer die Möglichkeit gegeben werden, seine Suche beispielsweise auf private Homepages oder wissenschaftliche Publikationen zu beschränken. Glover et al. zeigen dabei in [8], daß ausgefeiltere, über statistische Verfahren ermittelte, Ergänzungen bessere Ergebnisse liefern als die jeweils „naive“ Ergänzung („Homepage“). In ihrem Forschungsbericht zeigen sie Methoden zum Erlernen von guten Anfrage-Modifikationen.

Personalisierte Suche

Durch eine personalisierte Suche, wie sie von Lawrence in [19] vorgeschlagen wird, ist es möglich, den Kontext eines Suchbegriffs auch bei Mehrdeutigkeiten zu „erraten“. Dabei werden die persönlichen Suchgewohnheiten eines Benutzers herangezogen, um den Kontext zu ermitteln. Aus den bisherigen Anfragen wird geschlossen, was der Benutzer will. („guessing, what the user wants“[22])

Semantische Netze

Ein neuer Weg, für eine Abfrage den Kontext eines Begriffs zu ermitteln, ist über semantische Netze. Vereinfachte semantische Netze, bei denen die Assoziationen zwischen den Begriffen nicht qualifiziert sind, lassen sich relativ leicht über statistische oder aufwendiger über linguistische Methoden aufbauen. Solche semantischen Netze geben bei geeigneter Traversierung Auskunft über den Kontext eines Wortes und eignen sich somit auch für den Einsatz zur Anfrage-Erweiterung. Die Extraktion des Kontextes zu einem

[20] Mitra et al. beschreiben in [28] fortgeschrittene Techniken für automatische Anfrage-Reformulierung.

[21] vgl. [19] und [8]

[22] [19], Abschnitt 2.4

Begriff aus einem semantischen Netz steht im Mittelpunkt dieser Diplomarbeit und wird in den folgenden Kapiteln näher erläutert.

Unter der Bezeichnung „*Topic Maps*“ existiert ein Standard [13] zur Repräsentation von semantischen Netzen mit qualifizierten Assoziationen in XML.[23] Diese Topic Maps können auch direkt zur Navigation benutzt werden und treten dann an die Stelle klassischer *inverted index*-Systeme.[24]

[23] Siehe auch Abschnitt 4.1

[24] Einen aktuellen Überblick liefert der Artikel [7]

Kapitel 3

Neuronale Netze

Dieses Kapitel beinhaltet im 1. Teil einen Überblick über die Grundfunktionen neuronaler Netze. Im 2. Teil werden Hopfield-Netze näher betrachtet und eine neue Variante des Algorithmus zur Berechnung von Hopfield-Netzen gezeigt, der effizienter als die bisherigen Algorithmen arbeitet. Die Anwendung der Hopfield-Netze zur Themenextraktion aus semantischen Netzen wird im nächsten Kapitel beschrieben.

3.1 Grundfunktionen neuronaler Netze

3.1.1 Biologische und formale Neuronen

Neuronale Netze sind aus dem Versuch heraus entstanden, die Funktionen des menschlichen Gehirns nachzuahmen. Innerhalb der Informatik werden sie in den Bereich der subsymbolischen künstlichen Intelligenz eingeordnet, da sich ihren Funktionseinheiten, den Neuronen, keine Symbole zuordnen lassen sondern sich eine solche Zuordnung allenfalls aus der Gesamtheit eines neuronalen Netzes ergibt.

Vereinfacht gesprochen lösen in den Neuronen im Gehirn elektrische Impulse, die über sogenannte *Dendriten* in den Zellkörper gelangen, bei einer genügend starken Aktivierung einen Impuls am *Axon*, dem „Ausgang“ der Zelle, aus. Das Axon ist über *Synapsen* mit Dendriten anderer Nervenzellen verbunden. In den Synapsen findet ein elektrochemischer Übergang statt. Dadurch wird erreicht, daß sich Impulse nur in eine Richtung, vom Axon zum Dendrit, ausbreiten können. Außerdem kann in den Synapsen die Stärke der Weiterleitung reguliert werden.

Das erste Modell eines Neurons wurde 1943 von McCulloch und Pitts in [27] aufgestellt. Sie fassen dementsprechend ein formales Neuron als einen Addierer mit Schwellwert auf, der die Summe der gewichteten Aktivierungen an seinen Eingängen bildet und bei Erreichen eines Schwellwertes (im folgenden mit Θ bezeichnet) an seinem Ausgang einen Impuls abgibt.[1] McCulloch und Pitts zeigten, daß sich mit Neuronen die booleschen AND- und OR-Verknüpfungen darstellen lassen. Allein durch Einstellen des Schwellwertes läßt sich ein Neuron bei gleicher Architektur zwischen AND (bei n Eingängen mit jeweils gleichem Gewicht g sei $\Theta = g$) und OR ($\Theta = 1$). Läßt man zur Gewichtung der Eingänge auch negative Werte zu, kann auch die Negation dargestellt werden.

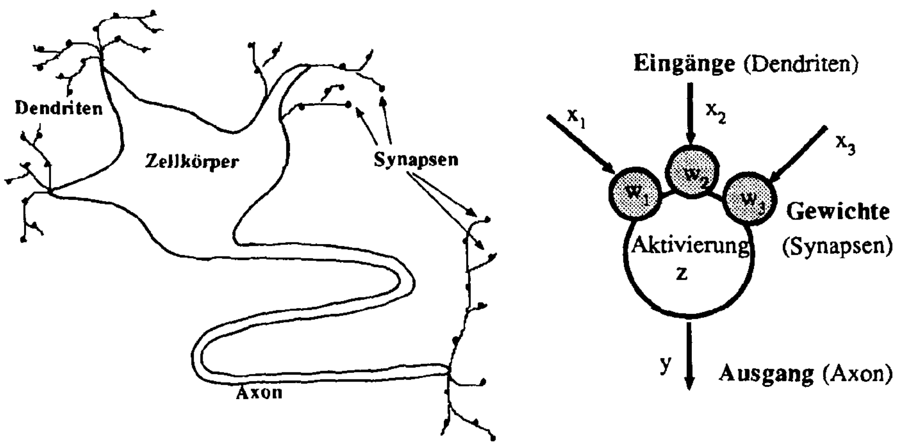

Abbildung 3.1: biologisches und formales Neuron (aus[2, S. 36])

Der Schwellwert Θ kann auch über einen Eingang simuliert werden. Fügt man zu einem formalen Neuron einen zusätzlichen Eingang hinzu, der immer aktiviert ist, läßt sich der Schwellwert über die Gewichtung dieses Eingangs einstellen. Der tatsächliche Schwellwert kann für jedes Neuron dann konstant als Null definiert werden. Dieses Vorgehen kann in der praktischen Anwendung Vorteile haben, wenn z.B. über einen Lernalgorithmus der Schwellwert dynamisch eingestellt werden soll.

3.1.2 Aktivierungsregeln

Neben der bereits besprochenen Aktivierungsregel, nach der ein Neuron aktiviert wird (oder „feuert"), wenn die Summe der Eingänge einen Schwellwert übersteigt, sind auch andere Ausgabefunktionen denkbar. Im Modell von McCulloch und Pitts sind die Ein- und Ausgänge entweder logisch 0 oder 1, es kann jedoch auch ein anderer Wertebereich verwendet werden, beispielsweise ein kontinuierlicher Übergang von -1 bis +1.

Die von McCulloch und Pitts vorgeschlagenen Ausgabefunktion sind Treppenfunktionen.

[1] Zu neuronalen Netzen existieren zahlreiche Lehrbücher. Die hier verwendeten sind [2] und [18], sowie die Vorlesungsskripte [31], [32] und [39].

$$f(x) := \begin{cases} 0 & \text{falls } x \leq 0 \\ 1 & \text{sonst} \end{cases} \tag{3.1}$$

$$f(x) := \begin{cases} -1 & \text{falls } x \leq 0 \\ 1 & \text{sonst} \end{cases} \tag{3.2}$$

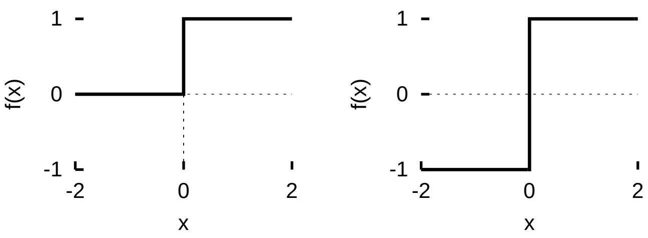

Abbildung 3.2: Sprungfunktionen (3.1 – 3.2)

Möglich sind auch lineare Funktionen. Will man verhindern, daß die Funktionswerte unbegrenzt wachsen können und damit die Aktivierungen der Neuronen möglicherweise zu stark werden und sich „aufschaukeln“, kann man abschnittsweise definierte, begrenzt lineare Funktionen verwenden.

$$f(x) := x \tag{3.3}$$

$$f(x) := \begin{cases} 0 & \text{falls } x \leq -t \\ 1/2 + (1/2t)x & \text{falls } -t < x < t \\ 1 & \text{falls } x \geq t \end{cases} \tag{3.4}$$

$$f(x) := \begin{cases} -1 & \text{falls } x \leq -t \\ (1/t)x & \text{falls } -t < x < t \\ 1 & \text{falls } x \geq t \end{cases} \tag{3.5}$$

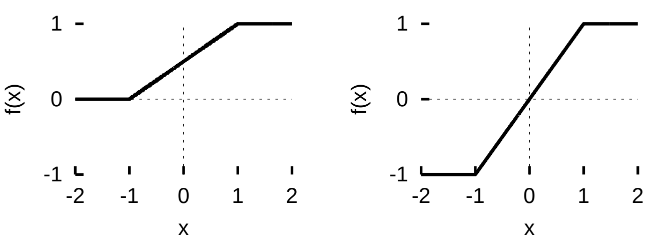

Abbildung 3.3: begrenzt lineare Aktivierungsfunktionen (3.4 – 3.5), t = 1

Benötigt man stetige Funktionen, verwendet man *sigmoide Funktionen*, auch *Quetschfunktionen* genannt. Beispiele hierfür sind die Fermi-Funktion (3.6), der hyperbolische Tangens (3.7) und die Cosinus-Quetschfunktion (3.8). Der Fermi-Funktion kommt eine besondere Bedeutung zu, da sie beim in Abschnitt 4.2.3 besprochenen *Simulated Annealing*-Verfahren mit einem variierenden Parameter T verwendet wird.

$$f(x) := \frac{1}{1 + e^{-\frac{x-\Theta}{T}}} \tag{3.6}$$

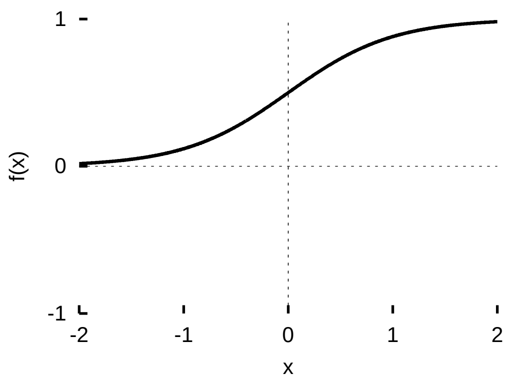

Abbildung 3.4: Fermi-Funktion (3.6), $\Theta = 0$, T = 0.5

$$f(x) := \begin{cases} -1 & \text{falls } x \leq -t \\ \tanh(\frac{x * \pi}{t}) & \text{falls } -t < x < t \\ 1 & \text{falls } x \geq t \end{cases} \tag{3.7}$$

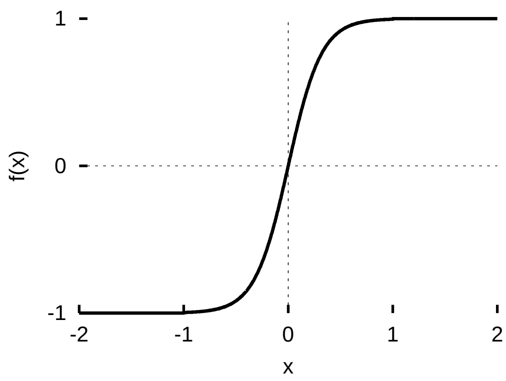

Abbildung 3.5: hyperbolischer Tangens (3.7), t = 1

$$f(x) := \begin{cases} 0 & \text{falls } x \leq -t \\ 1/2 * (1 + \cos(\frac{x*\pi}{2t} - \frac{\pi}{2})) & \text{falls } -t < x < t \\ 1 & \text{falls } x \geq t \end{cases} \tag{3.8}$$

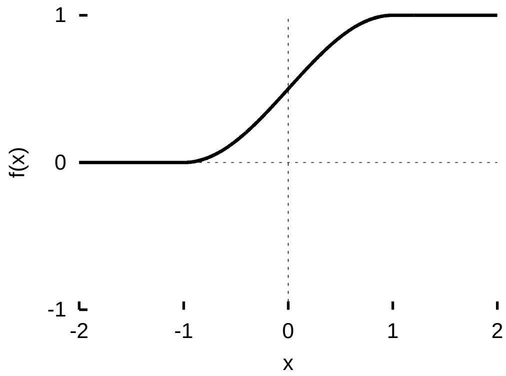

Abbildung 3.6: Cosinus-Quetschfunktion (3.8), t = 1

Der Metropolis-Algorithmus, der in Abschnitt 4.2.2 besprochen wird, verwendet eine abschnittsweise definierte Übergangsfunktion, die oberhalb eines Schwellwertes Θ stets 1 ist und darunter exponentiell abfällt.

$$f(x) := \begin{cases} e^{\frac{x-\Theta}{T}} & \text{falls } x < \Theta \\ 1 & \text{falls } x \geq \Theta \end{cases} \tag{3.9}$$

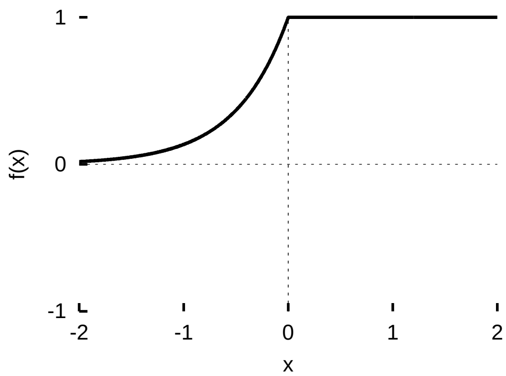

Abbildung 3.7: Metropolis-Funktion (3.9), T = 0.5

3.1.3 Feed-Forward-Netze

Neben der Fähigkeit einfacher Neuronen, die logischen Funktionen AND, OR und NOT nachzubilden, können bereits einzelne Neuronen einfache Klassifizierungsaufgaben erfüllen. Betrachtet man die Eingabe an den Eingängen der Neuronen als Vektoren, so kann man die Gewichte so einstellen, daß das Neuron zwischen zwei unterscheidbaren Gruppen von Vektoren unterscheiden kann, wenn die Klassen *linear separierbar* sind. Dies bedeutet, daß bei graphischer Interpretation der Vektoren als Punkte das Neuron genau dann die Fähigkeit zur Klassifizierung besitzt, wenn sich die beiden Punktwolken durch eine Gerade trennen lassen.[2] Ein Beispiel für Punkte in der Ebene zeigt Abbildung 3.8. Für dreidimensionale Vektoren muß die Trennung entsprechend durch eine Ebene, für n-dimensionale Vektoren durch eine Hyperebene möglich sein.

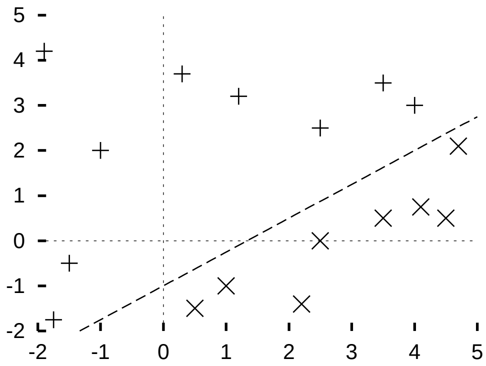

Abbildung 3.8: linear separierbare Musterklassen

Betrachtet man die logische XOR-Funktion, so stellt man fest, daß sie nicht linear separierbar ist. Um auch solche Klassifikationsaufgaben und logische Funktionen darzustellen, die nicht linear separierbar sind, muß man mehrere Neuronen in einem *neuronalen Netz* zusammenschalten.

Schichtenmodell

Diese Schaltung erfolgt meist in Schichten. Dabei sind die Eingänge von Neuronen einer Schicht nur mit den Ausgängen von Neuronen der jeweils vorangehenden Schicht verbunden. Verbindungen von Neuronen innerhalb einer Schicht untereinander sind nicht erlaubt, ebensowenig wie Rückkopplungen. Die Aktivierungen der Neuronen können sich also nur in einer Richtung ausbreiten. Die Eingabe erfolgt über dedizierte Eingabeneuronen in der

[2] vgl. [2, S. 44f.]

ersten Schicht, die Ausgabe an dedizierten Ausgabeneuronen in der letzten Schicht.

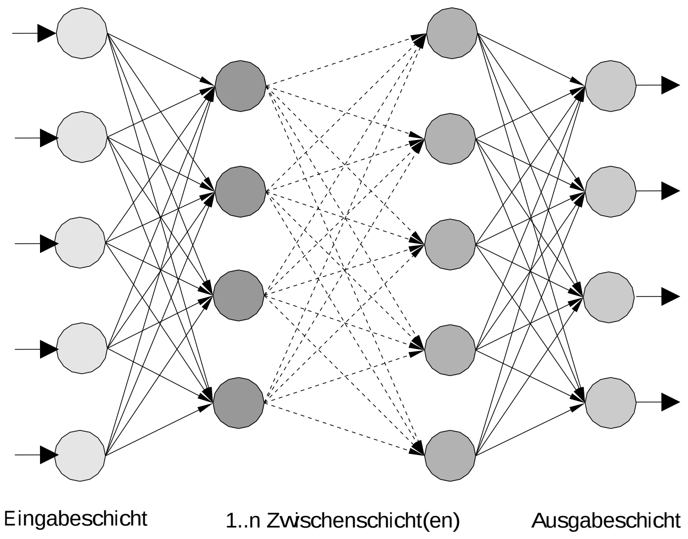

Abbildung 3.9: Aufbau mehrschichtiger Feed-Forward-Netze

Ein Beispiel für ein solches mehrschichtiges Netz, das eine XOR-Funktion nachbildet, zeigt Abbildung 3.10.

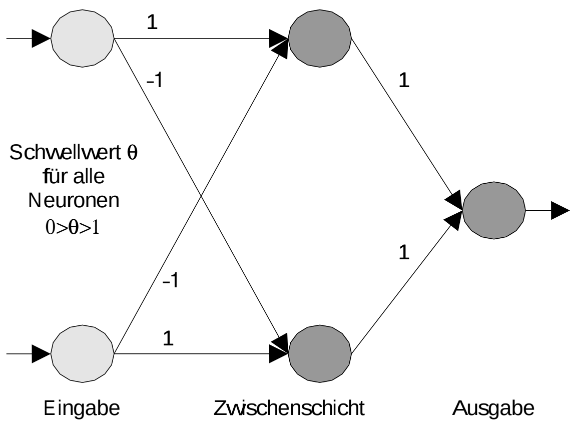

Abbildung 3.10: Neuronales Netz für logisches XOR

In der Praxis wird man neuronale Netze nicht nur für Nachbildungen von logischen Funktionen benutzen wollen. Für die meisten anderen Klassifikationsaufgaben ist jedoch die Einstellung der Gewichte und Schwellwerte nicht trivial. Üblicherweise stellt man daher die Gewichte nicht direkt ein, sondern verwendet einen Lernalgorithmus. Eine bekannte Lernregel ist die *Hebb'sche*

Lernregel. Die Lernalgorithmen basieren entweder darauf, daß dem Netz Paare von Eingabe- und Ausgabevektoren präsentiert werden *(Überwachtes Lernen)*, oder man überläßt dem Netz selbst, die Eingabedaten geeignet zu klassifizieren. Auch für dieses *unüberwachte Lernen* existieren geeignete Algorithmen. Die verschiedenen Lernregeln werden hier allerdings nicht näher betrachtet, da sie für die Anwendung im Rahmen der Themenextraktion aus semantischen Netzen irrelevant sind.[3]

3.2 Das Hopfield-Modell

Im Gegensatz zu den bisher betrachteten Feed-Forward- oder vorwärtsgekoppelten neuronalen Netzen gestatten *rückgekoppelte* oder *Feed-Back-Netze* eine Ausbreitung der Aktivierung nicht nur in eine Richtung. Insbesondere ist auch das auf den Physiker John J. Hopfield zurückgehende Hopfield-Modell[4] ein rückgekoppeltes Netz. Hopfield stellte einen Zusammenhang zwischen magnetischen Anomalien in Spingläsern (bestimmten seltenen Erden) und rückgekoppelten neuronalen Netzen fest.[5] Hopfield definierte eine für das Systemverhalten ausschlaggebende Zielfunktion und interpretierte sie als Energie. Die Charakteristika des von Hopfield verwendeten Netzes lassen sich wie folgt als Forderungen an ein neuronales Netz zusammenfassen:

1. **Vollständige Vernetzung** – Alle Neuronen sind paarweise miteinander verbunden. Es existiert keine geschichtete Struktur.

2. **Keine direkten Rückkopplungen** – Kein Neuron ist mit sich selbst rückgekoppelt.

3. **Symmetrische Verbindungen** – Die Verbindungen zwischen den Neuronen haben symmetrische Gewichte ($g_{ij} = g_{ji}$).

4. **Binäre Neuronen** – Es werden gewöhnliche McCulloch-Pitts-Neuronen mit den Zuständen 0 und 1 sowie einem Schwellwert Θ verwendet.

5. **Keine dedizierten Ein-/Ausgabe-Neuronen** – Jedes Neuron ist sowohl Ein- als auch Ausgabeneuron, es gibt keine verdeckten Neuronen.

Die Forderungen 2 und 3 nach Freiheit von direkten Rückkopplungen und symmetrischen Verbindungen sind dabei wichtiger als die Forderungen

[3]Lehrbücher über neuronale Netze, wie auch [18] und [2], gehen detailliert auf die diversen Lernalgorithmen ein.

[4]vgl. [11]

[5]s. [2, S. 179]

1 und 4 nach vollständiger Vernetzung und binären Neuronen. Im folgenden wird gezeigt, daß man auf die beiden letzteren auch verzichten kann, ohne daß die grundlegenden Eigenschaften von Hopfield-Netzen davon berührt werden.

3.2.1 Konvergenz von Hopfield-Netzen

Da nach Forderung 5 jedes Neuron sowohl Ein- als auch Ausgabeneuron ist und insbesondere die Neuronen auch nicht in Schichten organisiert sind, ist bei Hopfield-Netzen eine Stabilisierung des Gesamtnetzes nach initialer Aktivierung bestimmter Neuronen (und Berechnung der Propagierung der Aktivierungen) von Interesse. Der Beweis dafür, daß ein solches Netz überhaupt zu einem stabilen Zustand hin konvergiert, ergibt sich dabei aus der hopfieldschen Energiefunktion. Seien E die Gesamtenergie, z_i der Aktivierungszustand von Neuron i und g_{ij} das Gewicht der Verbindung von Neuron i zu Neuron j, so lautet die Energiefunktion[6]:

$$E := -\sum_i \sum_{i<j} g_{ij} \cdot z_i \cdot z_j - \sum_i \Theta_i \cdot z_i \tag{3.10}$$

$$E := -\frac{1}{2} \sum_i \sum_j g_{ij} \cdot z_i \cdot z_j - \sum_i \Theta_i \cdot z_i \tag{3.11}$$

Die beiden Schreibweisen (3.10) und (3.11) sind äquivalent. Während in (3.10) durch die Bedingung $i < j$ eine doppelte Zählung von Verbindungen verhindert wird, wird das gleiche Ergebnis in (3.11) durch den Faktor 1/2 erreicht.

Eine Änderung eines Aktivitätszustandes eines Neurons soll nur dann stattfinden, wenn sich die Energie dadurch vermindert oder zumindest gleich bleibt. Dabei läßt sich „lokal“, also auf jedes einzelne Neuron bezogen, entscheiden, wie eine Änderung des Aktivitätszustandes die Gesamtenergie beeinflußt. Der Beitrag eines Neurons i zur Gesamtenergie läßt sich berechnen als[7]:

$$\Delta E_i = \sum_j g_{ij} \cdot z_j + \Theta_i \tag{3.12}$$

Betrachtet man nun die eben erwähnte Bedingung, daß sich durch eine Zustandsänderung die Gesamtenergie vermindern oder gleichbleiben soll, so ergibt sich:

$$z_i = \begin{cases} 1 & \text{falls } \Delta E_i \geq 0 \\ 0 & \text{sonst} \end{cases} \tag{3.13}$$

[6] vgl. [18] und [32]

[7] vgl. [18, S. 115]

Gleichung (3.13) entspricht nun genau der Schwellwertbedingung für McCulloch-Pitts-Neuronen. Da $z_i = 0 \vee z_i = 1$, und damit insbesondere auch $0 \leq z_i \leq 1$, ist E durch eine Konstante nach unten beschränkt.[8] Es gilt:

$$E \geq -\frac{1}{2} \sum_i \sum_{i \neq j} g_{ij} \tag{3.14}$$

Damit ist gezeigt, daß E stets zu einem Energieminium hin konvergiert. E ist eine sogenannte *Liapunovfunktion.*[9] Weiterhin gilt somit, daß sich das Hopfield-Netz nach einer endlichen Zahl von Zustandsübergängen stabilisiert.

Dies gilt auch dann noch, wenn auf die Forderung nach vollständiger Vernetzung verzichtet wird, da fehlende Kanten als Kanten mit Gewicht 0 aufgefaßt werden können.[10]

Insbesondere wird aber nichts darüber ausgesagt, ob der erreichte stabile Zustand auch der Zustand mit der kleinsten Energie $E = E_{min} = -(1/2) \sum_i \sum_{i \neq j} g_{ij}$ ist. Im allgemeinen wird das Hopfield-Netz eher zu einem *lokalen Minimum* als zum globalen Minimum konvergieren.

Es ist auch die Forderung nach binären Neuronen für die Konvergenz des Netzes nicht notwendig, es genügt vielmehr, daß $0 \leq z_i \leq 1$, denn dann gilt immer noch Gleichung (3.14). Daher können auch andere Übergangsfunktionen verwendet werden, insbesondere auch einige der in Abschnitt 3.1.2 beschriebenen, wie die Fermi-Funktion (3.6). Hopfield zeigt in [12], daß Netze aus Neuronen mit stetigen Übergangsfunktionen die gleichen Grundeigenschaften wie Hopfield-Netze mit binären Neuronen haben.

3.2.2 Effizientes Berechnen des Hopfield-Netzes

Die Berechnung des Hopfield-Netzes erfolgt normalerweise wie in dem im folgenden in Pseudocode angegebenen und im Einklang mit Gleichung (3.13) arbeitenden Algorithmus 2. Man berechnet für jedes Neuron die Summe der Gewichte der Kanten, die zu benachbarten aktivierten Neuronen führen. Ist diese Summe größer als der Schwellwert des Neurons, so wird das Neuron aktiviert. Die Reihenfolge, in der die Neuronen abgearbeitet werden, ist beliebig. Der Algorithmus ist beendet, wenn sich das Hopfield-Netz stabilisiert hat[11]. Man beachte, daß zu einem Zeitpunkt nur jeweils ein Neuron betrachtet wird. Gegenüber einer parallelen Abarbeitung aller Neuronen ist das Modell somit geringfügig verändert,[12] allerdings geht man so dem Pro-

[8] vgl. [32, S. 75]
[9] vgl. [32] und [2]
[10] vgl. [11]
[11] vgl. [31]
[12] [2, S. 180]

blem aus dem Weg, daß bei paralleler Berechnung das System zwischen zwei Zuständen oszillieren kann.[13]

Algorithmus 2 Berechnung eines Hopfield-Netzes

repeat
  Wähle ein beliebiges Neuron j.
  $s_j := \sum_i g_{ij} * z_i$ {Berechne die Summe $s_j = \sum_i g_{ij} * z_i$ aus den Produkten der Gewichte der Kanten g_{ij} und der Aktivierungszustände z_i der entsprechenden adjazenten Neuronen i.}
  $z_j := f(s_j)$ {Wende eine geeignete Aktivierungsfunktion (vgl. Abschnitt 3.1.2) an.}
until Zustand bei allen Neuronen stabil

Bei der Wahl des zu bearbeitenden Neurons j kann eine Verbesserung der Effizienz ansetzen, da j beliebig ist. Man nehme als Grundzustand ein Hopfield-Netz an, bei dem kein Neuron aktiviert ist. Weiterhin sei der Schwellwert $\Theta \geq 0$. Aus Algorithmus 2 ergibt sich dann, daß ein Neuron j nur dann seinen Zustand ändern kann, wenn ein zu j adjazentes Neuron ebenfalls seinen Zustand geändert hat, da sonst der Schwellwert nie überschritten wird. Man braucht also generell nur solche Neuronen zu betrachten, bei denen sich im vorherigen Iterationsschritt der Zustand benachbarter Neuronen geändert hat. Dieses Vorgehen ähnelt der Idee des in [5] beschriebenen Rete-Algorithmus von Forgy zur Effizienzsteigerung bei Pattern-Matching-Problemen. Während beim Rete-Algorithmus bei Änderungen im *working memory* die Konfliktmenge aktualisiert wird, kann beim Algorithmus zur Berechnung des Hopfield-Netzes nach einer Änderung des Zustands eines Neurons die Liste der im nächsten Schritt zu bearbeitenden Neuronen aktualisiert werden. Dieses Vorgehen ist in Algorithmus 3 beschrieben. Zur Verwaltung der jeweils zu betrachtetenden Neuronen führt man zwei Listen ein. Eine der Listen wird zu Beginn mit den zu den anfangs aktivierten Neuronen benachbarten Neuronen gefüllt. Danach werden die neuen Zustände aller Neuronen in der Liste bestimmt. Falls bei einem Neuron eine Zustandsänderung auftritt, müssen die zu diesem Neuron benachbarten Neuronen im nächsten Iterationsschritt betrachtet werden und werden dann in die neue Liste eingefügt. Sind alle Neuronen der ersten Liste berechnet worden, wird die Berechnung der Zustände für die Neuronen der neuen Liste durchgeführt. Wenn sich das Netz stabilisiert, treten keine Zustandsänderungen mehr auf und die Liste wird leer. Der Algorithmus ist dann beendet.

Da bei diesem Vorgehen in der Regel weitaus weniger Neuronen als in Algorithmus 2 betrachtet werden müssen, ist es deutlich effizienter als ein Verfahren mit zufälliger Neuronen-Auswahl.

[13] [18, Seite 166]

Algorithmus 3 Effiziente Berechnung eines Hopfield-Netzes

Liste A := alle zu den initial aktivierten Neuronen benachbarten Neuronen
Liste B := leer
while Liste A nicht leer **do**
  for all Neuronen j in A **do** {Berechne die Summe s_j und den neuen Aktivierungszustand wie in Algorithmus 2}
    $s_j := \sum_i g_{ij} * z_i$ {Berechne die Summe $s_j = \sum_i g_{ij} * z_i$ aus den Produkten der Gewichte der Kanten g_{ij} und der Aktivierungszustände z_i der entsprechenden adjazenten Neuronen i.}
    $z_j := f(s_j)$ {Wende eine geeignete Aktivierungsfunktion (vgl. Abschnitt 3.1.2) an.}
    if Aktivierungszustand z_j um mehr als einen kleinen Betrag ϵ geändert **then**
      Liste B := Liste B + alle zu j benachbarten Neuronen
    end if
  end for
  Liste A := Liste B
  Liste B := leer
end while

Zu beachten ist, daß der Schwellwert Θ, wie bereits oben erwähnt, größer 0 sein muß.[14] Andernfalls kann es Neuronen im Hopfield-Netz geben, die bei der ersten Iteration auch ohne äußere Stimulation aktiviert werden. Durch Algorithmus 3 würden diese Neuronen nicht aktiviert, wenn sie nicht von außen stimuliert werden. Algorithmus 3 ist in diesem Punkt nicht äquivalent zu Algorithmus 2, der auch in diesem Fall korrekt reagiert. Falls Schwellwerte $\Theta < 0$ nötig sind und Algorithmus 3 verwendet werden soll, müssen vor der Initialisierung von Liste A alle Neuronen mit $\Theta < 0$ daraufhin überprüft werden, ob sie wegen ihres niedrigen Schwellwertes feuern. Falls ein solches Neuron feuert, wird es ebenfalls in Liste A aufgenommen.

3.2.3 Anwendungsgebiete von Hopfield-Netzen

Neben der Anwendung von Hopfield-Netzen zur Traversierung von semantischen Netzen, die im nächsten Kapitel beschrieben wird, gibt es die zwei klassischen Anwendungsgebiete Mustererkennung und Optimierungsaufgaben.

[14] Falls für die Aktivierungszustände der Neuronen Werte < 0 (beispielsweise zwischen -1 und $+1$) zugelassen werden, muß der Schwellwert Θ größer als der kleinste vorkommende Aktivierungszustand sein.

Assoziative Speicher und Mustererkennung

Rückgekoppelte Netze im allgemeinen und Hopfield-Netze im speziellen können als assoziative Speicher betrachtet und genutzt werden. Bei geeigneter Einstellung der Gewichte entsprechen die stabilen Zustände des Hopfield-Netzes erlernten Mustern. Da die stabilen Zustände lokale Energieminima sind, wird durch Iteration des Netzes ein unvollständiges oder verrauschtes Muster „repariert“ und es stellt sich das erlernte Muster ein. Somit ist ein Einsatz zur Mustererkennung, z.B. in OCR[15]-Systemen, möglich.

Optimierungsaufgaben

Da die Iteration eines Hopfield-Netzes einer Suche nach einem Energieminimum entspricht, ist auch ein Einsatz bei Optimierungsproblemen möglich. Durch eine geeignete Problemkodierung ist es beispielsweise möglich, das *Problem des Handlungsreisenden* und andere NP-vollständige Probleme zu approximieren.[16] Das Verhalten des Netzes, zu lokalen statt zu globalen Minima zu konvergieren, was bei der Anwendung zur Mustererkennung natürlich erwünscht ist, stellt für diese Anwendung allerdings einen Nachteil dar. Einen Ausweg verspricht das bereits erwähnte und in Abschnitt 4.2.3 beschriebene *Simulated Annealing*-Verfahren. Dennoch muß man sich vor Augen halten, daß wie bei den meisten heuristischen Verfahren im allgemeinen zwar eine möglicherweise sehr gute, aber suboptimale Lösung gefunden wird.

[15] *Optical Character Recognition*, = Schrifterkennung

[16] s. [2, S. 183 ff.]

Kapitel 4

Hopfield-Netze im Information Retrieval

4.1 Semantische Netze

Eine im Rahmen der Künstlichen Intelligenz gebräuchliche Form der symbolischen Wissensrepräsentation ist das semantische Netz. Ein semantisches Netz eignet sich dazu, Beziehungen und Sinnzusammenhänge zwischen Konzepten zum Ausdruck zu bringen. Anwendungen hierfür finden sich im Bereich des Sprachverstehens, aber auch im Bereich des Verlagswesens[1] und im Bereich des Information Retrieval.

Ein semantisches Netz ist ein Graph, in dem die Knoten begriffliche Einheiten (*Entitäten*) wie Objekte oder abstrakte Konzepte und die Kanten Beziehungen zwischen diesen begrifflichen Einheiten symbolisieren. Die Kanten sind üblicherweise gerichtet und qualifizieren die Art der Beziehung. Mögliche Arten von Beziehungen zwischen begrifflichen Einheiten sind beispielsweise *hat ein*, *ist ein*, *besteht aus* oder *ist ein Merkmal von*. Es sind auch andere Qualifizierungen möglich, allerdings wird oft die Anzahl der verwendeten Beziehungstypen je nach Anwendung auf die Grundformen eingeschränkt.

Man extrahiert Fakten aus dem semantischen Netz, in dem man von den Entitäten ausgehend den Kanten folgt. In einem semantischen Netz können sowohl lexikonartige, generelle Fakten wie bspw. „Ein Haus hat ein Dach.“ oder „Ein Stuhl ist ein Möbel.“ festgehalten werden, aber auch Fakten, die die momentane Situation beschreiben. („Der Mann liest ein Buch, das auf dem Tisch liegt.“)

Zur Repräsentation von semantischen Netzen in XML existiert inzwischen ein Standard [13].[2] Solche *Topic Maps* mit umfangreichem Lexikonwissen eigenen sich sehr gut zur Navigation in bestimmten Themenbereichen.

---

[1] siehe [38]

[2] vgl. auch [7], [30], [6] und [40]

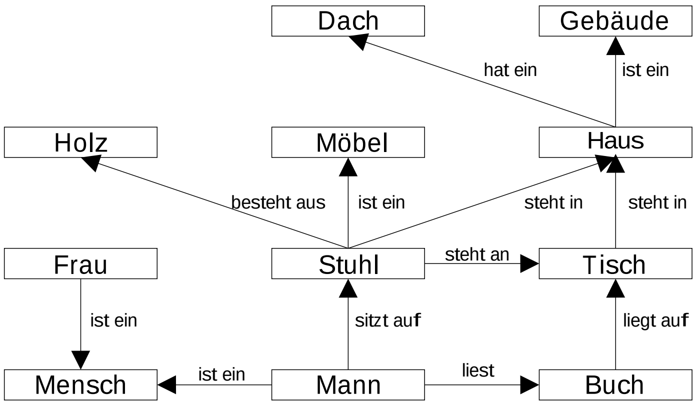

Abbildung 4.1: Einfaches semantisches Netz

Allerdings haben sie wegen der qualifizierten Kanten den Nachteil, daß sie sich für die im Rahmen dieser Arbeit gewünschten Anwendungen schlechter automatisch traversieren lassen als semantische Netze mit unqualifizierten, aber gewichteten Kanten. Wenn im folgenden von semantischen Netzen die Rede ist, sind nicht qualifizierte, sondern *quantifizierte* semantische Netzen gemeint. Bei dieser Art von Netzen sind die Kanten nicht mit Bezeichnern, sondern lediglich mit Gewichtungen versehen, die die Stärke eines Zusammenhangs zwischen zwei Entitäten kennzeichnen. Weiterhin sind die Kanten im Gegensatz zu den qualifizierten Kanten ungerichtet. Für die im folgenden beschriebenen Einsatzbereiche ist eine solche Information aber ausreichend.

4.1.1 Aufbau quantifizierter semantischer Netze

Quantifizierte semantische Netze lassen sich mit statistischen Methoden des Information Retrieval aufbauen.

Dekang Lin beschreibt in [24] und [23], wie mit Maßen für die Ähnlichkeit zweier Worte semantische Zusammenhänge aus einem Textkorpus extrahiert werden können. Lin verwendet ein auf dem informationstheoretischen Ansatz[3] beruhendes eigenes Ähnlichkeitsmaß, erwähnt aber, daß sich auch die bekannten Ähnlichkeitsmaße, wie sie in Kapitel 2 in Formeln (2.14)–(2.17) beschrieben sind, nutzen lassen. Im folgenden werden die in der ATHEM-Software verwendeten Ähnlichkeitsmaße beschrieben.[4]

Berechnung der Kantengewichte in der ATHEM-Software

Betrachtet man das Maß der inversen Dokumentfrequenz (siehe Formel (2.3) in Abschnitt 2.3.1), so läßt sich ein der Gewichtsfunktion (2.4) ähnliches

[3] siehe Formeln (2.5)–(2.9)
[4] vgl. [9]

Maß definieren, das aber die Frequenz eines Terms k in allen Dokumenten betrachtet. Sei $f(k)$ die Frequenz von Term k in allen Dokumenten, n die Gesamtzahl der Dokumente und $d(k)$ die Zahl der Dokumente, in denen k vorkommt, so sei das Maß Termfrequenz · inverse Dokumentfrequenz, kurz *tfidf*:

$$t(k) = 1 + f(k)\log(n/d(k)) \tag{4.1}$$

Um ein Netz von Beziehungen zwischen Termen aufzubauen, ist auch ein Maß nötig, das die relative Häufigkeit des gemeinsamen Auftretens zweier Terme beschreibt. Zwei Terme werden als gemeinsam auftretend bezeichnet, wenn sie innerhalb eines gewissen Textfensters bzw. eines Höchstabstands auftreten. Sei $f(i,j)$ die Anzahl der gemeinsamen Vorkommen der Terme i und j und $f(i)$ wie oben die Termfrequenz:

$$s(i,j) = \min(\frac{f(i,j)}{f(i)}, \frac{f(i,j)}{f(j)}) \tag{4.2}$$

Zusätzlich zu diesem Gewichtungsterm werden der Erwartungswert und die Standardabweichung herangezogen.

$$E(i,j) = \frac{t(i) + t(j)}{2} \tag{4.3}$$

$$\sigma(i,j) = \sqrt{\mid E(i,j) - t(i) \mid^2 + \mid E(i,j) - t(j) \mid^2} \tag{4.4}$$

Aus diesen statistischen Werten errechnet sich direkt die Gewichtung der Kanten zwischen den Termen. Damit nicht zwischen allen Termen Kanten entstehen, müssen die einzelnen Terme eine Mindestfrequenz ϑ_f und die Termpaare eine Mindestfrequenz ϑ_p überschreiten. Die Kanten beschreiben einen *syntagmatischen* Zusammenhang zwischen den Entitäten. Das bedeutet, daß sich ein Zusammenhang ausschließlich über Nähe der Worte zueinander im Text definiert, d.h. die Worte in einem *Syntagma* stehen.

$$g(i,j) = \begin{cases} s(i,j) \cdot (E(i,j) - \sigma(i,j)) & \text{falls } \begin{cases} f(i) > \vartheta_f & \wedge \\ f(j) > \vartheta_f & \wedge \\ f(i,j) > \vartheta_p & \end{cases} \\ 0 & \text{sonst} \end{cases} \tag{4.5}$$

Ein Beispiel für ein nach diesen Regeln erzeugtes semantisches Netz zeigt Abbildung 4.2. Wie man sieht, können in solchen semantischen Netzen auch Mehrdeutigkeiten auftreten. Das Wort „Lizenz“ beispielsweise bezieht sich zum einen offensichtlich auf Softwarelizenzen (vgl. „Windows“ und „Solaris“), zum anderen auf die Vergabe der UMTS-Mobilfunk-Lizenzen. Ähnlich verhält es sich mit dem Wort „Forum“. In der ATHEM-Software, die zur Erzeugung dieses Netzes verwendet wurde, werden neben syntagmatischen auch *paradigmatische* Zusammenhänge berechnet. Ein paradigmatischer Zusammenhang besteht zwischen zwei Entitäten, wenn sie zwar nicht im Text zusammen auftauchen, aber sich auf eine Klasse mit gleichen oder ähnlichen Eigenschaften beziehen. Wenn zwischen zwei Entitäten ein paradigmatischer Zusammenhang besteht, sind sie in einem Syntagma zueinander austauschbar.[5]

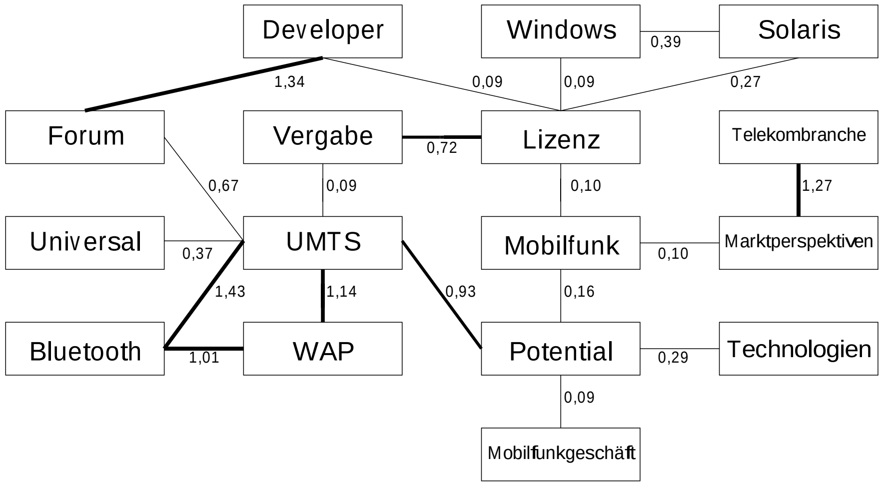

Abbildung 4.2: Ausschnitt aus einem semantischen Netz, automatisch erzeugt durch die ATHEM-Software

4.2 Algorithmen zur Themenextraktion

Zur Traversierung von und zur Themenextraktion aus semantischen Netzen bieten sich zum einen konventionelle graphentheoretische Algorithmen und zum anderen auf neuronalen Netzen basierende Algorithmen an. Graphentheoretische Algorithmen haben dabei den Nachteil, daß sie oft harte Abgrenzungen zwischen relevanten und nicht relevanten Teilen des Graphen finden müssen und oft einfach nur die direkte Umgebung des zuerst aktivierten Knotens betrachten.

Neuronale Netze haben dagegen von sich aus die Eigenschaften, die eine intuitive und von weichen Abgrenzungen geprägte Traversierung des Graphen erlauben. Dazu wird das semantische Netz als ein neuronales Netz, normalerweise als Hopfield-Netz, betrachtet. Von der Verwendung neuronaler Algorithmen erhofft man sich eine intuitivere Begriffsauswahl und ein besseres Laufzeitverhalten als bei konventionellen Algorithmen.

[5] vgl. [42]

Das Hopfield-Modell ist unter den verschiedenen Modellen neuronaler Netze das, welches am besten für die Repräsentation semantischer Netze geeignet ist. Wesentliche Eigenschaften der quantifizierten semantischen Netze finden sich auch im Hopfield-Modell wieder und umgekehrt. Im Gegensatz zu den Perzeptron-Netzen und anderen vorwärtsgerichteten Modellen neuronaler Netze sind Hopfield-Netze – ebenso wie semantische Netze – nicht geschichtet aufgebaut. Die ungerichteten Kanten des semantischen Netzes entsprechen den geforderten symmetrischen Verbindungen im Hopfield-Modell. Wie in Abschnitt 3.2 gezeigt, kann auf die Forderung der vollständigen Vernetzung, die im semantischen Netz nicht gegeben ist, verzichtet werden. Auch das Ergebnis der Berechnung, das im HopfieldModell einem stabilen Zustand entspricht, entspricht der gewünschten Ausgabe: Die aktivierten Neuronen entsprechen Knoten des semantischen Netzes, die in einem kontextuellen Zusammenhang zu den eingangs aktivierten Knoten bzw. Neuronen stehen.

4.2.1 Algorithmen mit festem Schwellwert

Die Wahl des Schwellwertes Θ für die einzelnen Neuronen, der ja nicht durch das zugrundeliegende semantische Netz vorgegeben ist, ist von kritischer Bedeutung für das Ergebnis der Berechnung. Wählt man den Schwellwert zu groß, kann dies dazu führen, daß überhaupt keine Neuronen aktiviert werden. Wählt man den Schwellwert zu klein, werden große Teile des Netzes aktiviert, die möglicherweise gar keine relevante semantische Beziehung zu dem ursprünglich aktivierten Knoten mehr haben. Ein Ausweg kann sein, den Schwellwert nicht von vornherein festzulegen.

Iterative Suche

Ein sehr einfaches, aber auch ineffizientes Verfahren für die Schwellwertwahl ist die iterative Suche. Dabei gilt für alle Neuronen des Netzes jeweils der gleiche Schwellwert Θ. Statt eines festen Schwellwertes gibt man aber nur einen Startwert Θ_0 an. Es bietet sich hier der Mittelwert zwischen den maximalen und minimalen Gewichtswerten g_{min} und g_{max} an. Weiterhin gibt man eine minimale und maximale Zahl von aktivierten Neuronen n_{min} und n_{max}, die man im stabilisierten Zustand wünscht, vor. Man rechnet nun das Netz einmal mit Schwellwert $\Theta = \Theta_0$ durch und vergleicht die Anzahl der tatsächlich aktivierten Neuronen n mit n_{min} und n_{max}. Ist $n < n_{min}$ so verringert man die Schwelle Θ um einen Wert δ, ist $n > n_{max}$ so erhöht man die Schwelle Θ um einen Wert δ, gleichzeitig verringert man δ, um sich einem endgültigen Schwellwert in immer feineren Schritten anzunähern. Danach setzt man die Aktivierungen auf die Ursprungswerte zurück und wiederholt die Berechnung so lange bis entweder n im gewünschten Bereich liegt, δ eine untere Grenze ϵ durchbrochen hat oder eine maximale Anzahl von Iterationen durchlaufen wurde. Man beachte, daß aufgrund der Beschaffenheit des Netzes nicht immer ein Schwellwert existiert, so daß genau eine bestimmte Zahl von Neuronen nach der Berechnung des Netzes aktiviert ist.

Algorithmus 4 Iterative Suche

```
n_min :=Mindestzahl von aktivierten Neuronen
n_max :=Höchtszahl von aktivierten Neuronen
Θ := Θ_0
repeat
   Berechne Hopfield-Netz mit Algorithmus 3
   n :=Anzahl aktivierte Neuronen
   if not n_min ≤ n ≤ n_max then
      if n > n_max then
         Θ := Θ + δ
      else
         Θ := Θ − δ
      end if
      verringere δ geeignet
   end if
until n_min ≤ n ≤ n_max oder δ < ϵ
```

Dieses Verfahren zur Schwellwertberechnung ist nicht sonderlich effizient, da meist mehrere Berechnungsdurchläufe durchgeführt werden müssen, um sich einem vernünftigen Schwellwert anzunähern.

Zweistufige Berechnung

Ein anderer Weg, um mit festen Schwellwerten zu brauchbaren Ergebnissen zu kommen, kann eine zweistufige Berechnung sein, wie sie in Algorithmus 5 vorgestellt wird. In der ersten Stufe wird ein sehr geringer Schwellwert – auch 0 ist möglich – gewählt. Damit wird erlaubt, daß sich die Aktivierungen zunächst relativ ungehemmt über das Netz ausbreiten können und somit eine ausreichend große Zahl von möglicherweise relevanten Neuronen betrachtet wird. Es muß eine maximale Zahl von Iterationen vorgegeben werden, nach der die erste Phase abbricht, da sich sonst wegen des zu geringen Schwellwertes zu viele Neuronen aktivieren. Als eine Iteration kann ein Durchlaufen der **while**-Schleife in Algorithmus 3 verstanden werden. Für jeden dieser Iterationsschritte erhöht sich die maximale Entfernung der betrachteten Neuronen von den eingangs aktivierten Neuronen um eins, da im Falle einer Änderung des Aktivierungszustandes im nächsten Schritt die Nachbarn der aktuell berechneten Knoten betrachtet werden. Die maximale Zahl von Iterationen kann somit als eine maximale Entfernung betrachtet werden.

Um nun aus den vielen in diesem Radius aktivierten Neuronen stark verbundene und damit vermutlich relevante zu finden, startet man eine zweite Phase, in der ein höherer Schwellwert eingestellt wird. Die Schwelle Θ sollte nun so hoch gewählt werden, daß ein ungebremstes Ausbreiten der Aktivierung über das Netz unterbunden wird. Das Hopfield-Netz stabilisiert sich dann nach wenigen Durchläufen.

Nachteile der festen Schwellwertwahl

Die Nachteile der Algorithmen mit festem Schwellwert sind offensichtlich. Bei einer festen Wahl der Parameter sind die Ergebnisse meist nicht brauchbar, weil entweder zu viele oder keine Neuronen aktiviert werden. Eine iterative Suche ist äußerst ineffizient. Aus diesen Gründen wurden nach ersten Experimenten diese Varianten als ungeeignet verworfen.

n-fach verbundene Subgraphen

Einzig die Variante mit zweistufigem Schwellwert ist gangbar. Die zweite Stufe entspricht bei geeigneter Schwellwertwahl einem graphentheoretischen Algorithmus zur Suche von n-fach verbundenen Subgraphen. Beachtet man die Kantengewichte nicht und setzt $g_{ij} = 1$, summiert also ohne Beachtung der Gewichte $s_i = \sum_j z_j$, so bleiben nach Stabilisierung des Netzes Subgraphen des Verbindungsgrades Θ aktiviert.

Für die Anwendung im Information Retrieval ist dieses Ergebnis jedoch auch nur bedingt geeignet. Zum einen bleibt die Gewichtung der Kanten unberücksichtigt, zum anderen hängt das Ergebnis sehr stark vom Vernetzungsgrad ab. Da in einem semantischen Netz sowohl Regionen mit hohem als auch solche mit niedrigem Vernetzungsgrad existieren können, müßte der Schwellwert entsprechend angepaßt werden. Weiterhin hat sich die Laufzeit des Algorithmus in ersten Experimenten als deutlich höher als bei den anderen Algorithmen herausgestellt, so daß auch dieses Verfahren nicht weiter betrachtet wird.

Andere graphentheoretische Ansätze

Die weiter unten beschriebenen Algorithmen zur Suche von maximalen Cliquen nach Arun Jagota und das davon abgeleitete Verfahren mit lokaler Hemmung arbeiten auch mit konstanten Schwellwerten. Da diese beiden Verfahren allerdings nicht das gegebene Netz als solches betrachten, sondern hemmende Kanten einfügen, treffen die hier geschilderten Nachteile auf diese Verfahren nicht zu. Wenn im weiteren Verlauf der Arbeit von Verfahren mit festen Schwellwerten die Rede ist, sind die in diesem Abschnitt beschriebenen Verfahren gemeint, nicht der Algorithmus nach Arun Jagota für maximale Cliquen und nicht das Verfahren mit lokaler Hemmung.

Algorithmus 5 Zweistufige Schwellwertwahl

```
n_max := maximale Zahl der Iterationen für Phase 1 {Entspricht maximaler Entfernung von den initial aktivierten Neuronen, siehe Text}
n := 0 {Iterationszähler}
Θ := Schwellwert für Phase 1
Liste A := alle zu den initial aktivierten Neuronen benachbarten Neuronen
Liste B := leer
while Liste A nicht leer und n < n_max do
  for all Neuronen j in A do {vgl. Algorithmus 3}
    s_j := Σ_i g_ij * z_i
    if s_j > Θ then {Summe s_j größer als Schwellwert Θ}
      z_j := 1 {Aktiviere das Neuron.}
    else
      z_j := 0 {Deaktiviere das Neuron.}
    end if
    if Aktivierungszustand z_j geändert then
      Liste B := Liste B + alle zu j benachbarten Neuronen
    end if
  end for
  Liste A := Liste B
  Liste B := leer
  n := n + 1
end while
Θ := Schwellwert für Phase 2
while Liste A nicht leer do
  for all Neuronen j in A do {vgl. Algorithmus 3}
    s_j := Σ_i g_ij * z_i
    if s_j > Θ then {Summe s_j größer als Schwellwert Θ}
      z_j := 1 {Aktiviere das Neuron.}
    else
      z_j := 0 {Deaktiviere das Neuron.}
    end if
    if Aktivierungszustand z_j geändert then
      Liste B := Liste B + alle zu j benachbarten Neuronen
    end if
  end for
  Liste A := Liste B
  Liste B := leer
end while
```

4.2.2 Algorithmen mit kontinuierlicher Aktivierungsfunktion

Verfahren mit sigmoider Aktivierungsfunktion

Hsinchun Chen hat in [3] bereits 1993 den Vorschlag gemacht, zur Traversierung von semantischen Netzen ein Hopfield-Netz mit einer sigmoiden Übergangsfunktion zu verwenden. Er verwendete die Fermi-Funktion (siehe Gleichung (3.6)) mit einem festen Wert für Θ und T. Zur Berechnung des Netzes verwendete er einen parallelen Algorithmus. Um die Probleme bei Verwendung von paralleler Aktivierung zu umgehen (Oszillation des Systems um Energieminimum, s. Abschnitt 3.2.2), wurde in der Implementierung der serialisierte Algorithmus 3 verwendet. Weiterhin erlaubte Chen nicht-symmetrische Kantengewichte, während das hier verwendete semantische Netz nur symmetrische Kanten enthält. Ein Hopfield-Netz mit einer Fermi-Funktion als Aktivierungsregel wurde auch 1998 von Chung, Pottenger und Schatz an der University of Illinois in einem in [4] beschriebenen experimentellen Textindizierungssystem verwendet.

Gegenüber Verfahren mit festem Schwellwert haben Aktivierungsregeln mit sigmoiden Funktionen für die Anwendung auf semantischen Netzen klare Vorteile. Wird der Wertebereich der Funktion auf einen Wert kleiner als 1 begrenzt und sind auch die Kantengewichte auf einen Bereich zwischen 0 und 1 normiert, ist die Ausdehnung der Aktivierung durch den Algorithmus selbst begrenzt, da die Aktivierungsstärke mit zunehmender Entfernung vom Ausgangspunkt aus abnimmt. Das System reagiert generell unempfindlicher gegen kleine Änderungen der Parameter als bei einem festen Schwellwert.

Metropolis-Algorithmus

Neben sigmoiden Übergangsfunktionen können auch andere Funktionen verwendet werden, ohne daß sich die grundlegenden Eigenschaften wie Konvergenz des Netzes ändern. Ein solche Funktion ist die Übergangsfunktion des auf eine Veröffentlichung von Metropolis, Rosenbluth und Teller zurückgehenden Metropolis-Algorithmus (siehe Gleichung (3.9)). In der im Rahmen dieser Arbeit erstellten Implementierung unterscheidet sich der Metropolis-Algorithmus für die Anwendung im Information Retrieval nur durch diese Übergangsfunktion vom oben beschriebenen Verfahren von Chen mit einer sigmoiden Funktion. Die Metropolis-Übergangsfunktion kann wie die Fermi-Funktion mit variierendem Parameter T im nachfolgend beschriebenen Simulated Annealing Verfahren verwendet werden.

4.2.3 Simulated Annealing

Wie in Abschnitt 3.2.1 besprochen, erfolgt beim ursprünglichen Hopfield-Modell der Übergang eines Neurons in einen anderen Aktivierungszustand nach Gleichung (3.13) genau dann, wenn sich die Energie des Gesamtnetzes vermindert oder zumindest gleich bleibt. Betrachtet man allerdings ein Hopfield-Netz mit einer kontinuierlichen Übergangsfunktion wie der Fermi-Funktion (3.6), so fällt auf, daß ein Zustandsübergang auch dann erfolgen kann, wenn sich die Gesamtenergie des Netzes erhöht. Ist nämlich die Summe ($s_i = \sum_j g_{ij} \cdot z_j$) $< \Theta$, so ist dennoch im allgemeinen $f(s_i) > 0$. Es findet also dennoch eine – zwar geringe – Aktivierung des Neurons statt, obwohl die Schwellwertbedingung nicht erfüllt ist. Diese Aktivierung bei Unterschreitung von Θ ist um so größer, je größer T (siehe Gleichung (3.6)) ist. Für sehr große $T \to \infty$ gilt sogar für *beliebige* x $f(x) \approx \frac{1}{2}$. Ist $T = 0$, so erfolgt im Grenzwert der Übergang zur Treppenfunktion. Umgekehrt erfolgt ein Zustandsübergang nicht immer vollständig, d.h. im allgemeinen ist $f(s_i) < 1$, auch wenn $s_i > \Theta$.

Interpretiert man dies physikalisch, so kann man eine Analogie zur Erhitzung eines Materials über den Schmelzpunkt und einem anschließenden langsamen Ausglühen feststellen. Das Material geht dabei von einem Zustand höherer Energie zu einem Zustand geringer Energie über (z.B. geordneter Kristall), wobei das Aufheizen den Übergang von einem lokalen Minimum zu einem tiefer liegenden Energieminimum ermöglicht.[6] Bei der anschließenden Abkühlung stabilisiert sich das Material dann auf einem niedrigeren Energieniveau. Der Parameter T entspricht dabei der Temperatur. Ähnlich kann auch bei Hopfield-Netzen durch ein zunächst hohes und dann abnehmendes T ein Übergang aus einem lokalen Minimum heraus mit dem Ziel, ein tiefer liegendes Minimum zu erreichen, erfolgen. Man sprich daher von „simuliertem Ausglühen“ bzw. „simulated annealing“. Wie in Abschnitt 3.2.1 gezeigt, konvergiert der Hopfield-Algorithmus auch für hohe T. Man spricht dann auch von einem *thermalen Gleichgewicht*.[7]

Vor einer Verringerung der Temperatur kann gewartet werden, bis sich ein thermales Gleichgewicht eingestellt hat. Man spricht dann von *homogenem* Simulated Annealing. Erfolgt die Temperaturverringerung dagegen kontinuierlich, spricht man entsprechend von *inhomogenem* Simulated Annealing.

Für die Traversierung des semantischen Netzes wurden sowohl homogenes als auch inhomogenes Simulated Annealing, sowie inhomogenes Simulated Annealing mit der Metropolis-Funktion implementiert. Man beachte, daß der Schwellwert Θ bei Verwendung von Simulated Annealing höher eingestellt werden kann als bei Verwendung der Fermi-Funktion ohne Simulated Annealing, da durch die anfangs hohe Temperatur T ja durchaus auch ein Zustandsübergang erfolgen kann, wenn die Schwelle nicht überschritten wird. Der erwünschte Effekt ist dabei ähnlich wie bei der zweistufigen Schwellwertwahl der, daß sich die Aktivierung zunächst über einen Teil des

[6] [2, S. 201]

[7] vgl. [18]

Netzes ausbreiten soll, bevor sich dann durch Verminderung der Temperatur die relevanten Aktivierungen herausbilden.

4.2.4 Fixierung der Eingabe

Insbesondere bei den Algorithmen mit kontinuierlicher Aktivierungsfunktion und dem Simulated Annealing Verfahren ist es möglich, bestimmte Teile des Netzes zu *fixieren*, d.h. aus dem weiteren Reaktivierungsprozeß herauszunehmen.[8] Bei den fixierten Neuronen wird die Aktivierung nicht neu berechnet, sondern es wird ein konstanter Wert eingesetzt. Die Konvergenz des Algorithmus wird dadurch nicht beeinträchtigt.

Fixiert man die eingangs aktivierten Neuronen auf eine konstante Aktivierung von 1, ist die Ausbreitung der Aktivierungen stärker an die Eingabe gebunden. Neuronen in der Nachbarschaft der Startneuronen profitieren von der konstanten Aktivierung, während ohne Fixierung insbesondere bei schwach vernetzten Startneuronen diese möglicherweise in späteren Iterationsschritten nicht mehr reaktiviert werden.

Es ist auch umgekehrt möglich, eine Fixierung auf 0 vorzunehmen, um zu verhindern, daß sich die Aktivierung über bestimmte Neuronen hinaus ausbreitet. Dies kann sinnvoll sein, um bestimmte Teilbereiche auszuschließen. Betrachtet man erneut Abbildung 4.2, kann es beispielsweise sinnvoll sein, die Knoten „Lizenz“ oder „UMTS“ und „Mobilfunk“ auf 0 zu fixieren, wenn „Lizenz“ im Kontext von Software gemeint ist. Eine solche Fixierung könnte interaktiv vom Benutzer ausgelöst werden, wenn sich nach einem ersten Berechnungsschritt herausstellt, daß sich durch ein mehrdeutiges Wort die Aktivierung in eine nicht gewünschte Richtung ausbreitet.

4.2.5 Maximal-Clique-Algorithmen

Wie in Abschnitt 3.2.3 erwähnt, können Hopfield-Netze auch zur Approximation von NP-vollständigen Optimierungproblemen verwendet werden. Ein solches Problem ist neben dem Problem des Handlungsreisenden auch das sogenannte *Cliquenproblem*, die Suche der größten Clique in einem Graphen. Arun Jagota hat sich mit diesen Problemen befaßt und zeigt in [14] eine Variante des Hopfield-Netzes, deren stabile Zustände nicht-erweiterbaren (maximalen) Cliquen eines Graphen entsprechen. Auf semantische Netze übertragen können maximale Cliquen als die Begriffe zu einem zusammenhängenden Themengebiet interpretiert werden.

In einem Graphen $G = (V, E)$ mit Kanten E und Knoten V ist eine Clique ein *vollständiger induzierter Subgraph*. Sei V' eine Teilmenge von V, E' eine Teilmenge von V' und $E|V'$ die Menge aller Kanten, die beide Endpunkte in V' haben. Dann ist jeder Graph $G' = (V', E')$ ein *Subgraph* von

[8] vgl. [18, S. 124]; Fixierung bei Simulated Annealing führt zu einem Modell ähnlich der *Boltzmann-Maschine*.

G und insbesondere der Graph $G|V' = (V', E|V')$ ein *induzierter Subgraph* von G. Ein Graph heißt *vollständig*, wenn alle Paare von Knoten durch eine Kante miteinander verbunden sind, d.h. der Graph hat als Kanten alle zweielementigen Teilmengen einer n-elementigen Menge V. Ein Subgraph $G' = (V', E|V')$ heißt *maximale* oder *nicht erweiterbare* Clique, wenn kein zu G' adjazenter Knoten v mit der Eigenschaft $G'' = (V' \cup \{v\}, E|(V' \cup \{v\}))$ und G'' ist Clique existiert. Weiterhin sei der Grad eines Knotens $v \in V$, d.h. die Anzahl der mit v inzidenten Kanten, bezeichnet mit $\deg v$, und $\max(\deg v)$ der maximale Grad eines Knotens. [9]

Um nun aus einem Graphen $G = (V, E)$ eine maximale Clique zu extrahieren, überführt man G derart in ein Hopfield-Netz, daß man für jedes $v \in V$ ein Neuron j und für jedes $e \in E$ eine synaptische Verbindung mit Gewichtung $g_{ij} = 1$ setzt. G und entsprechend das Hopfield-Netz sind an dieser Stelle im allgemeinen nicht vollständig vernetzt. Man ergänzt nun zusätzliche hemmende synaptische Verbindungen mit einem Gewicht $g_{ij} = \rho < -\max(\deg v)$. Man beachte, daß eventuell vorhandene Gewichte im ursprünglichen Graphen – oder, entsprechend dem Anwendungsfeld, im semantischen Netz – bei der Suche nach Cliquen irrelevant sind und daher nicht betrachtet werden müssen. Abbildung 4.4 zeigt den entstehenden Graphen. Man beachte, daß Graphen, in denen gemäß dem beschriebenen Verfahren Kanten ergänzt wurden, nach der Ergänzung vollständig vernetzt sind.

Nach Berechnung des dadurch entstandenen Hopfield-Netzes entsprechen im stabilen Zustand die aktivierten Neuronen den Knoten genau einer maximalen Clique des erzeugenden Graphen, wenn vor Beginn der Iteration mindestens ein Neuron aktiviert war. Wenn genau ein Neuron aktiviert war, liegt der dem Neuron entsprechende Knoten in der maximalen Clique. Waren mehrere Neuronen aktiviert, die in einer maximalen Clique liegen, so wird genau diese Clique aktiviert. Waren mehrere Neuronen aktiviert, die nicht in einer gemeinsamen Clique liegen, so wird genau eine Clique aktiviert, die anderen initial aktivierten Neuronen sind deaktiviert.

Während der Iteration des Hopfield-Netzes geschieht dabei folgendes:[10]

1. Es werden solange Neuronen deaktiviert, bis die noch aktivierten Neuronen in einer Clique liegen.

2. Es werden solange Neuronen aktiviert, bis die Clique maximal ist.

Da die Anordnung der hemmenden Verbindungen invers zu den ursprünglichen Verbindungen ist und alle das gleiche Gewicht haben, läßt sich unter Verwendung eines Zählers der aktivierten Neuronen der Algorithmus zur Berechnung des Hopfield-Netzes effizienter gestalten. Algorithmus 3 zur effizienten Berechnung des Hopfield-Netzes wurde daher für die Berechnung

[9] Zu den graphentheoretischen Ausführungen vgl. [17, S. 14f, S. 163].

[10] vgl. Theorem 1 in [14]

von maximalen Cliquen abgewandelt, das Ergebnis ist Algorithmus 6. Man beachte, daß sich zur Berechnung maximaler Cliquen als Aktivierungsfunktion der Neuronen nur die Sprungfunktion anbietet.

Algorithmus 6 Effiziente Berechnung maximaler Cliquen mit Hopfield-Netzen

Liste A := alle zu den initial aktivierten Neuronen benachbarten Neuronen
Liste B := leer
c_0 := Anzahl der aktivierten Neuronen
while Liste A nicht leer **do**
  for all Neuronen j in A **do**
    $c_j := \sum_i g_{ij} \cdot z_i$
    $s_j := c_j + (c_0 - c_j) \cdot \rho$
    $z_j := \begin{cases} 0 & \text{falls } s_j \leq 0 \\ 1 & \text{sonst} \end{cases}$ {Aktivierungsregel (3.1)}
    if Aktivierungszustand z_j geändert **then**
      $c_0 := \begin{cases} c_0 + 1 & \text{falls } z_j = 1 \\ c_0 - 1 & \text{falls } z_0 = 0 \end{cases}$
      Liste B := Liste B + alle zu j benachbarten Neuronen
    end if
  end for
  Liste A := Liste B
  Liste B := leer
end while

4.2.6 Verfahren mit lokaler Hemmung

In Anlehnung an den Algorithmus zur Suche maximaler Cliquen läßt sich ein Berechnungsmodell mit relativer statt absoluter Hemmung ableiten. Mit dem Ziel, Teile des Graphen zu finden, die stark verbunden sind, aber nicht den Kriterien Clique oder n-fach verbundener Subgraph genügen, wurde Jagotas Modell dahingehend abgewandelt, daß hemmende Kanten nur zwischen Knoten, die eine Entfernung von 2 zueinander haben, eingefügt werden. Weiterhin wurden statt der zur Suche maximaler Cliquen geeigneten angenäherten absoluten Hemmung über ein Gewicht ρ einen Wert $\varrho \approx -g$, der etwa den gleichen Betrag wie die positiven Gewichte hat, verwendet. Der genaue Wert ist nicht entscheidend, ϱ sollte lediglich in etwa dem durchschnittlichen Betrag der positiven Gewichte entsprechen. Der gewählte Wert ϱ ist für das gesamte Netz gleich. Um einen Graphen in das Berechnungsmodell mit lokaler Behinderung zu überführen, ersetzt man seine Kanten nach folgender Vorschrift:

$$g'_{ij} := \begin{cases} g_{ij} & \text{falls } g_{ij} > 0 \\ \varrho & \text{falls } g_{ij} = 0 \wedge \exists (v \in V) \text{ mit } v \text{ adjazent zu } i \text{ und } j \\ 0 & \text{sonst} \end{cases} \tag{4.6}$$

Abbildung 4.5 zeigt die ergänzten hemmenden Kanten gestrichelt. Im Gegensatz zum Verfahren nach Jagota muß der Graph nach der Ergänzung der hemmenden Kanten nicht notwendigerweise vollständig vernetzt sein.

Die eigentliche Berechnung des Netzes kann dann mit festem Schwellwert (d.h. mit einer Sprungfunktion als Übergangsfunktion) berechnet werden. Anders als beim Algorithmus nach Jagota genügt der stabile Zustand keinen besonderen graphentheoretischen Kriterien. Von daher kann es im Gegensatz zu diesen Algorithmus für maximale Cliquen durchaus sinnvoll sein, die Gewichte des ursprünglichen Netzes beizubehalten.

Das Einfügen der hemmenden Kanten muß nicht im Vorhinein geschehen, sondern kann, wie auch in der im Rahmen dieser Arbeit erstellten Implementierung, während der Berechnung der Aktivierung eines Neurons durchgeführt werden. An dieser Stelle müssen dann zusätzlich zu den direkten Nachbarn alle Knoten mit Abstand 2 traversiert und in die Summe hemmend einbezogen werden, wenn nicht eine direkte Kante besteht.

In ersten Experimenten hat sich gezeigt, daß das Verfahren mit lokaler Hemmung gegenüber den anderen Algorithmen eine längere Laufzeit hat, aber durchaus brauchbare Ergebnisse liefert. Eine genauere Auswertung findet sich – wie auch zu allen anderen besprochenen Verfahren – in Kapitel 6.

4.3 Anwendungen von Hopfield-Netzen im Information Retrieval

4.3.1 Textindexierung

Für die Anwendung von semantischen Netzen und Hopfield-Netzen im Information Retrieval eröffnen sich vielfältige Möglichkeiten. Wie bereits in Abschnitt 4.2.2 erwähnt, wurde von Chung, Pottenger und Schatz in [4] ein System mit einem auf semantischen Netzen beruhenden Hopfield-Netz zur Textindexierung vorgestellt. Dem System wurde eine Kurzzusammenfassung eines Textes präsentiert und es lieferte mit dem Hopfield-Netz gefundene assoziierte Terme als Vorschläge für Indexterme. Damit sollte zum einen erreicht werden, daß im Gegensatz zu Freitext-Indextermen effizientere Indexterme aus einem kontrollierten Vokabular, nämlich dem semantischen Netz, verwendet werden können. Weiterhin sollten durch die Anreicherung von Termen aus dem semantischen Netz automatisch Indexbegriffe herangezogen werden, die im Text der Zusammenfassung so nicht auftauchten, aber

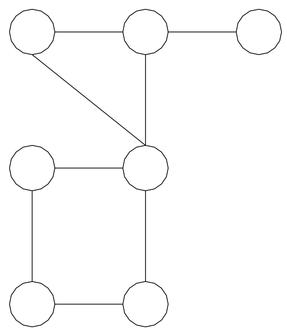

Abbildung 4.3: Ein einfacher Graph

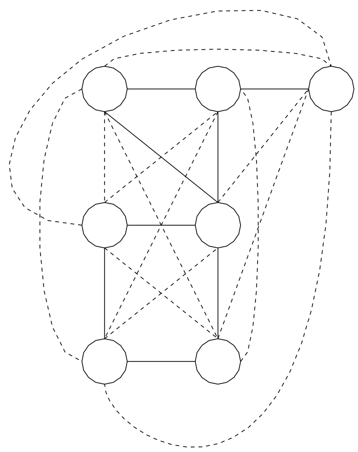

Abbildung 4.4: Durch den Maximal-Clique Algorithmus ergänzte Kanten

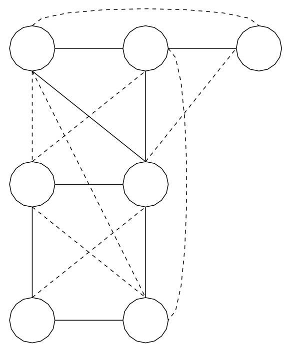

Abbildung 4.5: Durch das Verfahren mit lokaler Hemmung ergänzte Kanten

dennoch einen kontextuellen Bezug zum Text hatten. Das System funktionierte, hatte jedoch Probleme mit ambivalenten Termen, die in mehreren Kontexten stehen konnten. Es konnte dann zu einer Aktivierung von nicht relevanten Termen aus einem jeweils anderen Kontext kommen. Es wurde daher nicht ein automatischer, sondern ein interaktiver Einsatz des Systems empfohlen.

4.3.2 Query Expansion

Ein weiteres mögliches Einsatzfeld für Hopfield-Netze ist die Anreicherung von Suchanfragen. Dem Benutzer einer Suchmaschine kann über die Traversierung semantischer Netze die Möglichkeit gegeben werden, seine Anfrage interaktiv oder automatisch mit Begriffen aus dem Kontext anzureichern. Das Ziel ist dabei eine Steigerung der *Precision*[11], falls die Terme in einschränkender Weise (mit boolescher AND-Verknüpfung) hinzugefügt werden, oder eine Steigerung des *Recall*, wenn die Terme mit boolescher OR-Verknüpfung hinzugefügt werden. Das semantische Netz wird dabei ähnlich einem Thesaurus verwendet. Die interaktive Verwendung in einem Information Retrieval System war auch Chens [3] experimentelle Anwendung. Ebenso ist der interaktive Einsatz zur Query Expansion im K-Portal-Projekt geplant.

4.3.3 Automatische Textzusammenfassung

Eine mögliche zukünftige Anwendung könnte auch die automatische Textzusammenfassung sein. Beim Lauf des Systems können Terme aus dem jeweils betrachteten Textabschnitt im Hopfield-Netz aktiviert werden und so der Kontext bestimmt werden. Diese Information kann zur Zusammenfassung herangezogen werden. Derzeit ist mir allerdings kein System zur Textzusammenfassung auf Basis von Hopfield-Netzen bekannt.

[11] vgl. Abschnitt 2.1.1

Kapitel 5

Implementierung

Die durch das Projekt bestimmten Anforderungen an das Modul Themenextraktion sehen eine weitgehend plattformunabhängige, mindestens unter Linux, Sun Solaris und Windows lauffähige, Implementierung in C++ vor. Das semantische Netz ist in einer Datenbank gespeichert, für die eine objektorientierte Zugriffsschnittstelle existiert. Als Datenbanksystem wird DB/2 von IBM verwendet.[1]

Das Modul Themenextraktion wird im Rahmen des Projekts K-Portal von verschiedenen anderen Modulen aufgerufen. Daher ist eine mehrschichtige Architektur notwendig, in der die Benutzerinteraktion und die Interaktion mit anderen Modulen von der eigentlichen Berechnung getrennt ist. Bei der Implementierung verfolgte ich einen objektorientierten Ansatz, wobei die Standard Template Library Verwendung fand.

Nach einem kurzen Überblick über die Klassenstruktur wird im folgenden auf die einzelnen Klassen genauer eingegangen. Abbildung 5.1 zeigt die Beziehungen der Klassen zueinander. *TopicMap* ist das zentrale Objekt, das die benötigten Teile des semantischen Netzes hält.

Die Klassen *Token* und *TokenAssocs* finden sich auch in der Datenbank als Entitäten. In Abbildung 5.2 ist ein Ausschnitt des Datenbankschemas dargestellt. Neben den gezeigten Entitäten enthält die Datenbank auch Verweise auf die Dokumente, in denen die entsprechenden Tokens vorkommen. Diese Teile der Datenbank sind jedoch für die Anwendung „Themenextraktion aus semantischen Netzen" nicht notwendig und werden daher nicht weiter betrachtet. Der Ausschnitt des Datenmodells, der im Abbildung 5.2 dargestellt ist, reicht aus, um eine in sich konsistente Datenbank aufzubauen, die für den Betrieb der hier beschriebenen Verfahren ausreicht. Die Tabelle *TokenAssocs* wird über die in Abschnitt 4.1.1 erläuterten Berechnungsformeln aus der Tabelle *TokenPairs* erzeugt, zur Laufzeit des Moduls Themenextraktion werden allerdings nur die Tabellen *TokenAssocs* und *Tokens* benötigt. Um während der Berechnung eines neuen semantischen Net-

[1] vgl. [29]

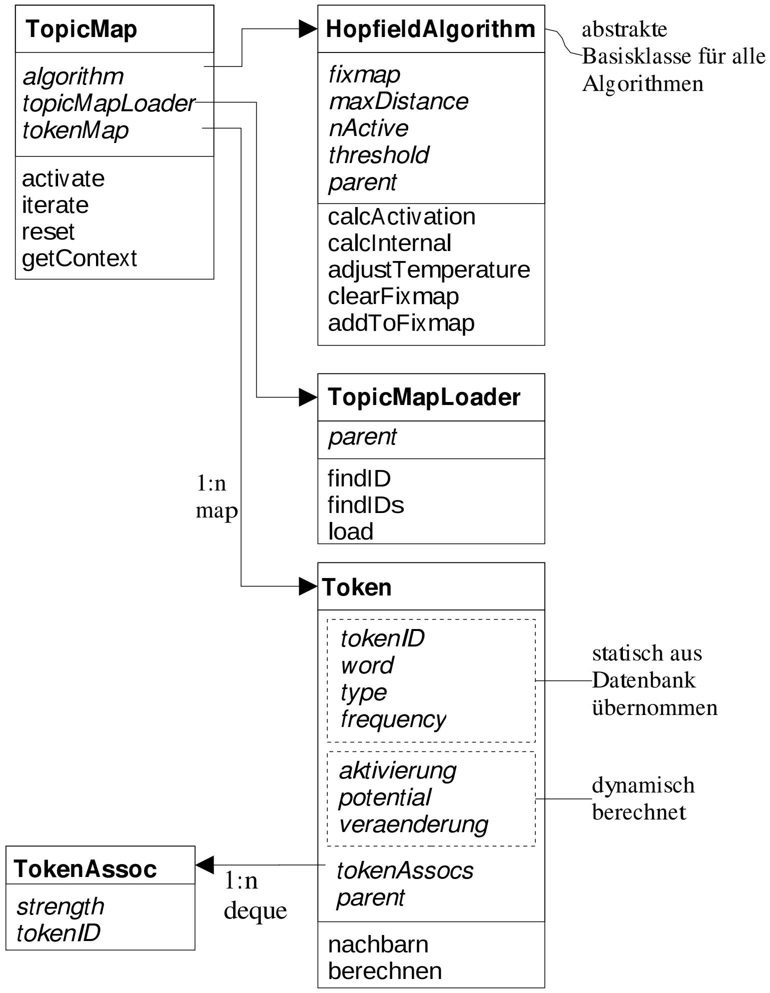

Abbildung 5.1: Objektbeziehungen

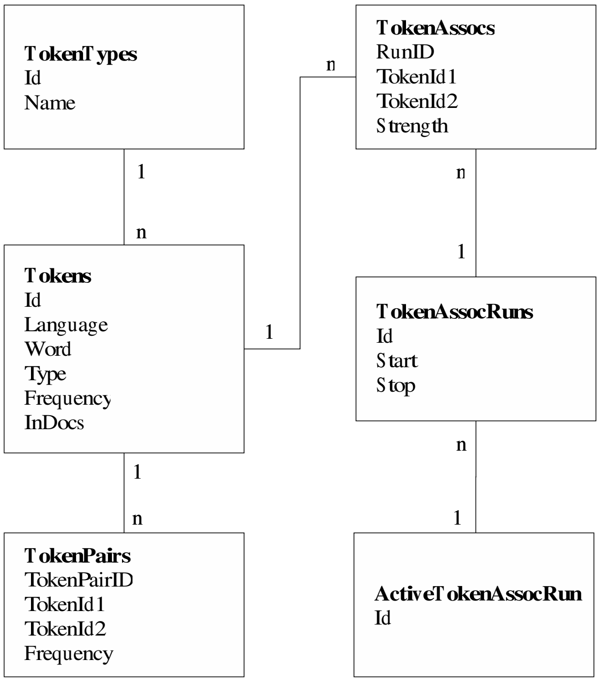

Abbildung 5.2: Ausschnitt aus dem Datenbankschema

zes – d.h. während der Neuberechnung der Tabelle *TokenAssocs* – weiterhin konsistente Daten für den Betrieb zur Verfügung zu haben, wird für jede Neuberechnung ein Eintrag in *TokenAssocRuns* erzeugt. Sobald eine neue Berechnung abgeschlossen ist, wird die *Id* des neuen Laufs in die Tabelle *ActiveTokenAssocRun*, die nur eine Zeile enthält, eingetragen.

Zur Laufzeit des Algorithmus werden über einen *TopicMapLoader* die jeweils benötigten Tokens geladen und ein Objekt erzeugt. Innerhalb der *TopicMap* sind die Tokens in einer assoziativen Liste (Klasse *map* aus der Standard Template Library) gespeichert, auf die einzelnen Tokens kann über die ID-Nummer zugegriffen werden. Insbesondere bei der Initialisierung des Hopfield-Netzes, wenn die ID also noch nicht bekannt ist, ist es auch notwendig, über die eingegebene Zeichenkette auf ein Token zugreifen zu können. Diese Suche ist im *TopicMapLoader* untergebracht, da an dieser Stelle stets ein Datenbankzugriff zur Suche nötig ist. Zu einem Begriff kann es mehrere Tokens mit unterschiedlichen Sprachen oder Wortklassen geben, es werden dann alle Tokens, die abgehende Assoziationen zu anderen Tokens haben, geladen und aktiviert.

TopicMapLoader ist eine abstrakte Klasse, es gibt neben einer Subklasse für den Datenbankzugriff auch eine zweite, die erlaubt, das semantische Netz aus einer Datei einzulesen. Die Klasse *HopfieldAlgorithm* ist eine abstrakte Klasse für Algorithmen zur Berechnung der Aktivierung eines Neurons.

Für jeden vorgestellten Algorithmus existiert eine Subklasse zu *HopfieldAlgorithm*, die jeweils spezifische Erweiterungen enthält und die Methode *calcInternal* überlädt.

5.1 Klasse *TopicMap*

Die Klasse *TopicMap* bietet einen Rahmen für das semantische Netz, die Datenzugriffsschicht und die Berechnungsalgorithmen. Die Attribute *algorithm* und *topicMapLoader* sind Zeiger auf Objekte der Klasse *HopfieldAlgorithm* und *TopicMapLoader*. Diese beiden Klassen sind abstrakte Basisklassen für Algorithmen zur Berechnung der Aktivierung eines Neurons bzw. zum Laden von Tokens aus der Datenbank oder einem File.

Das Attribut *tokenMap* ist eine Instanziierung von `map<int, Token>` und stellt somit eine assoziative Liste zum Zugriff auf die Tokens über die entsprechenden ID-Nummern dar. Diese *map* enthält zur Laufzeit jeweils den relevanten Teil des semantischen Netzes. Über die Klasse *TopicMapLoader* werden während der Berechnung Tokens zur *map* hinzugefügt. Tokens werden nie aus der *map* entfernt, so daß es sich empfiehlt, für jede Sitzung eine neue *TopicMap* anzulegen, da sonst der Hauptspeicherbedarf zu groß wird, aber für mehrere Abfragen innerhalb einer Sitzung die selbe *TopicMap* zu verwenden, damit die Zahl der notwendigen Datenbankzugriffe und somit die Laufzeit verringert wird.

5.1.1 Methode *iterate*

Die Methode *iterate* implementiert Algorithmus 3 (effizientes Berechnen eines Hopfield-Netzes). Die im Algorithmus erwähnten Listen sind als Satz von IDs (`set<int>`) implementiert. Die Berechnung der Summe und der Aktivierung (in Algorithmus 3 alles innerhalb der **while**-Schleife) ist allerdings in in die Methode *calcActivation* bzw. *calcInternal* von *HopfieldAlgorithm* ausgelagert, da diese Befehlsfolge für die einzelnen Algorithmen (siehe Kapitel 4) unterschiedlich ist. In *iterate* wird lediglich für jedes zu berechnende Neuron die *berechnen*-Methode des korrespondierenden Tokens aufgerufen, die wiederum die eigentliche Berechnung an die bereits erwähnte *calcActivation*-Methode der Klasse *HopfieldAlgorithm* weiterreicht.

5.1.2 Methode *getContext*

Die Methode *getContext* stellt die Aufrufschnittstelle bereit. Übergeben wird eine Zeichenkette, die durch Leerzeichen getrennt die zu aktivierenden Begriffe enthält. Zusätzlich ist vor jedem Begriff ein führendes „+“ oder „–“-Zeichen erlaubt. Dies bewirkt eine Fixierung des entsprechenden Tokens auf einen Potential- und Aktivierungs-Wert von 1 bzw. 0. Die Behandlung der Fixierung geschieht in der Klasse *HopfieldAlgorithm*. Der Eingabestring wird analysiert, für jeden extrahierten Begriff werden die entsprechenden Tokens aktiviert. Anschließend wird der stabile Zustand des Hopfield-Netzes über *iterate* berechnet. Die Ergebnisse werden in einer STL-multimap als Paare von Aktivierungsstärke und ID zurückgegeben.

5.2 Von *HopfieldAlgorithm* abgeleitete Klassen

Die Klasse *HopfieldAlgorithm* stellt die abstrakte Basisklasse für alle Algorithmen zur Berechnung des Hopfield-Netzes dar. Die Klassenhierarchie ist in Abbildung 5.3 dargestellt. Die Methode *calcInternal* muß in jedem Fall überladen werden, sie implementiert die Berechnung der Aktivierungsfunktion. Die Klasse *ChenAlgorithm*, die das Verfahren nach Chen mit der Fermi-Funktion als Aktivierungsregel implementiert, stellt dabei die (nicht abstrakte) Basisklasse für alle Algorithmen mit sigmoider Aktivierungsfunktion dar, die Klassen für Verfahren mit Simulated Annealing erben von *ChenAlgorithm*. Die Methode *adjustTemperature* wird nach Stabilisierung des Netzes von *TopicMap::iterate* aus aufgerufen und ist für homogenes Simulated Annealing (Klasse *HAnnealingAlgorithm*) notwendig, um nach Erreichen eines ersten thermalen Gleichgewichts die Temperatur zu senken. Bei inhomogenem Simulated Annealing (Klasse *AnnealingAlgorithm*) erfolgt die Anpassung der Temperatur kontinuierlich nach jeder Berechnung. Die Methode *adjustTemperature* wird auch beim Verfahren mit zweistufiger Schwellwertwahl (Klasse *TwoStepAlgorithm*) verwendet, um den Schwellwert für die zweite Stufe zu erhöhen. Klasse *SubgraphAlgorithm* implementiert den Hopfield-Algorithmus mit festem Schwellwert, *TwoStepAlgorithm* kann die Berechnungsregel von dieser Klasse erben. Die Klassen *JagotaAlgorithm* und *LocalInhibitionAlgorithm* implementieren das Verfahren zur Suche maximaler Cliquen nach Arun Jagota bzw. den Algorithmus mit lokaler Hemmung.

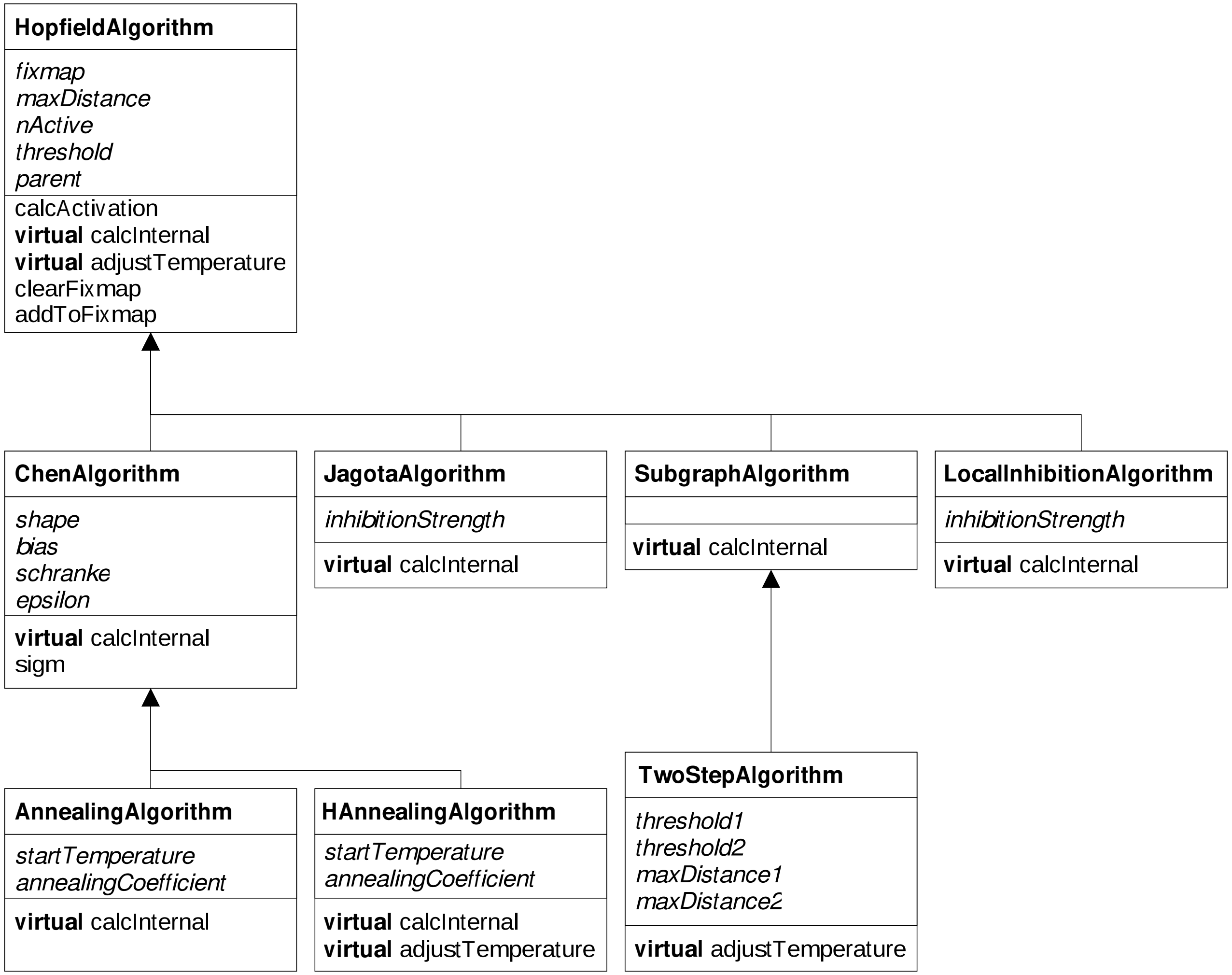

Abbildung 5.3: Hierarchiediagramm für von *HopfieldAlgorithm* abgeleitete Klassen

Kapitel 6

Evaluierung

Da zum gegenwärtigen Zeitpunkt die andere Module des Projekts K-Portal noch nicht fertiggestellt sind, ist eine Evaluierung der Funktion nicht einfach. Die Ergebnisse einer Themenextraktion und einer Traversierung eines semantischen Netzes können nicht losgelöst von der Generierung des semantischen Netzes und der gewünschten Anwendung betrachtet werden. Ebenso ist es schwierig, mehr als nur „intuitive“ Aussagen über die Qualität der Ergebnisse einer Traversierung des semantischen Netzes zu machen, da sich dies zahlenmäßig nur schwer fassen läßt. Eine Möglichkeit der Evaluierung von Information-Retrieval-Systemen sind aber die in Abschnitt 2.1.1 vorgestellten Maßzahlen *Precision* und *Recall*. Eine vollständige Precision-/Recall-Analyse würde allerdings im Aufwand den Rahmen dieser Arbeit übersteigen. Insbesondere ist für den Recall-Wert notwendig, zu wissen, wie viele Dokumente und wie viele für eine gegebene Anfrage relevante Dokumente sich in der Textsammlung befinden. Für ein großes Intranet wie das der Dresdner Bank, das sich zudem sehr schnell ändert, ist diese Frage nur sehr schwer zu beantworten. Da man für die Ermittlung der Precision jedoch die Anzahl der Dokumente in der Sammlung nicht kennen muß, und da der Precision-Wert ohnehin als wichtiger als der Recall-Wert betrachtet werden kann[1], wird im folgenden der Recall-Wert nicht betrachtet.

Wie bereits erwähnt, ist eine Anwendung des semantischen Netzes die automatische oder interaktive *Query Expansion*. Im folgenden wird daher vor allem dieser Anwendungsaspekt berücksichtigt.

6.1 Auswertung der Algorithmen

Bevor weiter auf ein der Praxis angenähertes Testszenario eingegangen wird, soll zunächst die direkte Ausgabe der verwendeten Algorithmen betrachtet und auf ihre Tauglichkeit hin überprüft werden. Aufgrund der Vielzahl der vorgestellten Algorithmen ist es zunächst nötig, für die weitere Evaluierung

[1] siehe Abschnitt 2.1.1

eine Vorauswahl zu treffen. In ersten Experimenten nach der Implementierung stellten sich einige Charakteristika der Algorithmen heraus, die diese Auswahl erleichtern.

Die Tests wurden durchgeführt mit einem Pentium II 266 MHz mit 96 MB Hauptspeicher und Windows 2000 als Client-Rechner, der über eine 100 MBit/s schnelle TCP/IP Netzwerkverbindung mit einem Dual-Pentium III 800 MHz Datenbankserver mit IBM DB/2 unter Linux kommuniziert. Alle Zeitangaben beziehen sich auf diese Systemkonstellation. Zur interaktiven Traversierung des semantischen Netzes wurden die in Kapitel 5 beschriebenen Klassen mit einem experimentellen Windows-Dialogprogramm kombiniert und mit Visual C++ 6.0 compiliert. Zum Datenbankzugriff wird die bereits erwähnte objektorientierte Schnittstelle von Heiner Petith benutzt.

Es wurden zwei Berechnungsläufe des semantischen Netzes benutzt, die auf dem selben Textkorpus beruhen. Der zweite Lauf wurde mit anderen Parametern gestartet und hat einen deutlich höheren Vernetzungsgrad. Durch die Berechnungsformel für die paradigmatischen Kanten können nicht-normierte Gewichte größer als 1 entstehen. Da die Konvergenz des Hopfield-Netzes nicht mehr in jedem Fall gewährleistet ist, wenn die Gewichtswerte größer als die entsprechenden Schwellwerte der Übergangsfunktionen sind ($g_{ij} > \Theta$), wurden für die Traversierung des Netzes die g_{ij} auf einen Maximalwert von 1 begrenzt.

6.1.1 Metropolis-Übergangsfunktion

Es hat sich in Experimenten gezeigt, daß der Metropolis-Algorithmus bei gleicher Einstellung der Parameter sehr ähnliche Ergebnisse liefert wie das Verfahren von Chen mit sigmoider Übergangsfunktion (Abschnitt 4.2.2). Auch bei inhomogenem Simulated Annealing (Abschnitt 4.2.3) unterscheiden sich die Ergebnisse zwischen den beiden verwendeten Übergangsfunktionen praktisch nicht. Auf eine gesonderte Betrachtung der Metropolis-Algorithmen wurde daher verzichtet.

6.1.2 Graphentheoretische Algorithmen

Maximale Cliquen

Der Algorithmus zur Ermittlung maximaler Cliquen funktioniert zufriedenstellend. Seine Qualität für die Verwendung im Information Retrieval hängt jedoch unter anderem sehr stark vom Grad der Vernetzung des zugrundeliegenden semantischen Netzes ab. Weiterhin kann es ein Problem sein, daß ein Begriffs-Knoten im allgemeinen in mehreren maximalen Cliquen liegt. Bei Aktivierung nur eines Tokens und Anwendung des Algorithmus ist nicht klar, welche Clique nach Konvergenz des Hopfield-Netzes aktiviert ist. Weiterhin ist zu beachten, daß zwar eine *maximale*, aber nicht notwendigerweise die größte Clique gefunden wird.

Durch Hinzuziehen eines zweiten, benachbarten Begriffs bei der initialen Aktivierung kann eine Verbesserung erzielt werden. Es wird dann eine Clique aktiviert, in der beide Begriffe liegen. Insbesondere in stark vernetzten Bereichen kann so oft eine sinnvolle Gruppe von Worten zu einem Thema extrahiert werden.

Wie die Ergebnis-Tabellen zeigen, funktioniert der Algorithmus bei stark vernetzten Graphen besser, da die in diesen Graphen enthaltenen Cliquen im allgemeinen größer sind. Tabelle 6.1 zeigt am Beispiel des Begriffs „Bausparen", wie sich die Clique durch Hinzufügen eines zweiten Begriffs verschiebt. Tabelle 6.2 zeigt am Beispiel des Begriffs „Leitzinsen" den Einfluß des unterschiedlichen Vernetzungsgrades der beiden Berechnungsläufe auf das Ergebnis. Man beachte, daß die sprachverarbeitenden Routinen, die zur Berechnung des Netzes eingesetzt wurden, zulassen, daß auch Abkürzungen in mehreren unterschiedlichen Schreibweisen und verschiedene Wortformen aufgenommen werden, wie bei Lauf 2 geschehen und in Tabelle 6.2 ersichtlich.

Eingabe	Lauf[2]	Ergebnisse
Bausparen	1	Bausparen Träume Wünsche
	2	Bausparen Lebensversicherung Todesfall Nürnberger JF
Bausparen VL[3]	1	Bausparen VL Förderung
	2	Bausparen VL Förderung Prämienbegünstigte

Tabelle 6.1: Maximale Cliquen für „*Bausparen*"

Verfahren mit lokaler Hemmung

Beim von Jagotas Algorithmus zur Bestimmung maximaler Cliquen abgeleiteten Verfahren mit lokaler Hemmung ist das Ziel, Bereiche starker Vernetzung, die aber nicht den strengen Kriterien Clique oder n-fach verbundener

[2] Berechnungslauf des semantischen Netzes; siehe Abschnitt 6.1

[3] VL = Abkürzung für „Vermögenswirksame Leistungen"

Eingabe	Lauf	Ergebnisse
Leitzinsen	1	Leitzinsen Anhebung
	2	Leitzinsen Anhebung Zinserhöhung Zinserhöhungen FED Fed EZB Notenbank Straffung Zinssenkung

Tabelle 6.2: Maximale Cliquen für „*Leitzinsen*"

Subgraph genügen, zu finden. Wie die Ergebnistabellen im Anhang zeigen, wurde dieses Ziel erreicht. Allerdings ist die Laufzeit des Algorithmus recht hoch, in Ausnahmefällen sogar fast 2 Minuten. Weiterhin kann, wie auch beim nachfolgend beschriebenen Subgraph-Algorithmus, eine stark verbundene Gruppe aktiviert werden, die nicht direkt in Zusammenhang mit dem eigentlich gesuchten Begriff steht. So werden beispielsweise bei „Lebensversicherung" eine Reihe von Städtenamen aktiviert. Die Verbindung ist dabei über „Nürnberger", einem Anbieter von Lebensversicherungen, gegeben. Die meisten der durch den Algorithmus genannten Städte sind jedoch nicht Sitz von Versicherungsunternehmen.

n-fach verbundene Subgraphen

Wie bereits in Abschnitt 4.2.1 ist das Ergebnis des Algorithmus, der zusammenhängende Subgraphen findet, nicht besonders geeignet für die Anwendung im Information Retrieval, da auch innerhalb eines semantischen Netzen Regionen mit stark unterschiedlichen Zusammenhangsgrad existieren. So liefert für „Bausparen" der Subgraph-Algorithmus mit einer Reichweite von 2 und einem Schwellwert von 6 als Ergebnis 28 Tokens, während für dieselben Einstellungen zum Begriff „UMTS" nichts gefunden wird. Erst wenn man den Schwellwert auf 4 senkt, werden überhaupt Tokens aktiviert. Der gefundene 4-fach zusammenhängende Subgraph enthält jedoch das Wort „UMTS" überhaupt nicht und steht in keinem Zusammenhang mit UMTS, obwohl die Tokens über das semantische Netz nur 2 Kanten von dort entfernt sind; „Universal" ist direkt mit „UMTS" verknüpft. Tabelle 6.3 zeigt das Ergebnis der Berechnung, Abbildung 6.1 einen Ausschnitt des semantischen Netzes. Wegen dieser Nachteile wurde der Algorithmus nicht weiter

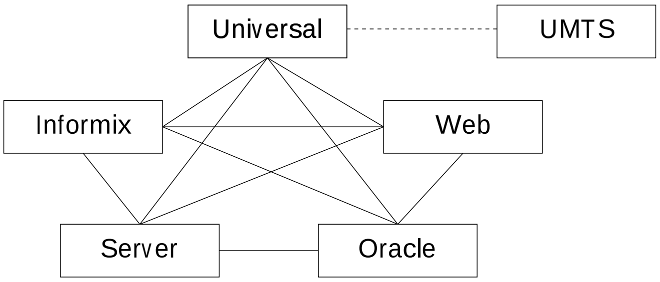

Abbildung 6.1: Ausschnitt des semantischen Netzes bei „*UMTS*“ und „*Universal*“

betrachtet, daher taucht er auch bei den Ergebnistabellen im Anhang nicht auf.

Eingabe	Lauf	Ergebnisse
UMTS	1	Oracle Informix Server Web Universal

Tabelle 6.3: 4-fach zusammenhängender Subgraph für „*UMTS*“

Der Algorithmus mit festem Schwellwert und iterativer Suche wurde wegen der in Abschnitt 4.2.1 beschriebenen Probleme von vornherein ausgeschlossen.

6.1.3 Verfahren mit sigmoider Aktivierungsfunktion

Das Verfahren von Chen mit der Fermi-Funktion als Aktivierungsregel wurde mit den Parametern $\Theta = 1.3$ und $T = 0.25$ getestet. Der maximale Suchradius wurde auf 2 begrenzt, es werden also nur Neuronen bzw. Tokens mit einer maximalen Entfernung von 2 vom Ausgangspunkt aus gesehen aktiviert. Getestet wurde zum einen ohne Fixierung, zum anderen mit Fixierung des Ursprungsbegriffs auf 1. Die Ergebnisterme wurden nach der Aktivierungsstärke, die sich bei Konvergenz des Netzes einstellte, sortiert. Die Ergebnisse mit Fixierung erreichten wie erwartet eine höhere Lokalität um den Ursprungsbegriff herum. Oft ergab sich für die höchstplazierten Begriffe die gleiche Reihenfolge, die sich bei Betrachtung der direkten Nachbarn in der Reihenfolge der Kantenstärke ergibt. Beim Verfahren ohne Fixierung ist die Aktivierungsstärke der Terme nach der Konvergenz generell schwächer und über die verschiedenen Terme gleichmäßiger. Beim Verfahren ohne Fixierung sind unter den 10 aktivierungsstärksten Termen oft Begriffe mit Entfernung 2 vom Ursprungsbegriff, während bei Fixierung Begriffe mit Entfernung 1 üblicherweise stärker gewichtet sind.

In den Ergebnistabellen sind die direkten Nachbarn, die durch Hopfield-Netz mit und ohne Fixierung aktivierten Tokens sowie die durch das nachfolgend diskutierte Simulated Annealing-Verfahren aktivierten Tokens gegenübergestellt. Da teilweise sehr viele Tokens aktiviert wurden, sind nur die 10 am stärksten gewichteten Begriffe mit ihrer Aktivierungsstärke angegeben. Die ausgewählten Begriffe sind Begriffe aus den Themengebieten Finanzen und Informationstechnik. Wegen der Menge der Daten werden die Ergebnisse im Detail nicht an dieser Stelle, sondern in Anhang A in den Tabellen A.3 ff. wiedergegeben.

Simulated Annealing

Das Simulated Annealing-Verfahren wurde in der inhomogenen Variante mit den Parametern $\Theta = 2$ und $T = 0.8$ getestet. Bei jeder Aktivierung wurde der Wert T mit dem Abkühlungskoeffizienten 0.8 multipliziert, die Abkühlung ist also verhältnismäßig stark. Der Suchradius wurde auf 4 begrenzt, allerdings wurden in den Tests keine Tokens mit einer Entfernung größer als 2 aktiviert. Das Simulated Annealing-Verfahren wurde generell mit Fixierung getestet, da ohne Fixierung durch den stark abfallenden Parameter T teils überhaupt keine Tokens mehr aktiviert waren. Auch diese Ergebnisse stehen in den Tabellen A.3 ff.

Laufzeitverhalten

Die Antwortzeiten der getesteten Algorithmen mit kontinuierlicher Aktivierungsfunktion sind auf dem oben beschriebenen Testsystem als für eine interaktive Anwendung ausreichend zu beurteilen. Die Zeiten schwanken je nach gewähltem Algorithmus und Begriff in den gezeigten Tests zwischen 2,9 und 5,8 Sekunden, mit einem „Ausreißer“ bei 9,0 Sekunden. Der Datenbankzugriff scheint bei einen großen Anteil der Rechenzeit in Anspruch zu nehmen, wie ein Vergleich mit einer Programmversion, die das semantische Netz aus Dateien einliest, zeigt. In der Praxis ist dennoch die Verwendung einer Datenbank vorgesehen und auch sinnvoll. Es werden daher im folgenden nur die Zeiten, die mit der Programmversion mit Datenbankzugriff ermittelt wurden, betrachtet.

Das Hopfield-Netz konvergierte nach maximal 5 Iterationen, wobei ein Durchlauf der **while**-Schleife in Algorithmus 3 als eine Iteration gewertet wird.

Im Vergleich zu den Angaben zum Laufzeitverhalten, die Hsinchun Chen in [3] zu seiner Implementation macht, sind die im Rahmen dieser Arbeit ermittelten Werte deutlich besser. Chens Implementierung brauchte mindestens 9, höchstens 28 und durchschnittlich 18,8 Iterationen und eine Zeit von mindestens 6, höchstens 36 und durchschnittlich 24,5 Sekunden. Chen verwendete verschiedene, aus 1 – 10 Begriffen bestehende Aktivierungen. Für aus nur einem Begriff bestehende Aktivierungen ergeben sich aber keine deutlich besseren Werte. Die genauen Zahlen sind in Tabelle A.1 in Anhang A angegeben. Aus mehreren Begriffen bestehende Aktivierungen werden für die im Rahmen dieser Arbeit erstellte Implementierung in Abschnitt 6.1.4 im Zusammenhang mit Fixierung gesondert betrachtet.

Die Ergebnisse sind allerdings nicht direkt vergleichbar, da Chen in [3] keine Angaben zum verwendeten Testsystem und zu den verwendeten Datenstrukturen macht.

6.1.4 Verfahren mit Fixierung auf Null

In Abschnitt 4.2.4 wird beschrieben, daß es bei mehrdeutigen Worten sinnvoll sein kann, Tokens derart zu fixieren, daß sich die Aktivierung nicht über sie hinaus ausbreiten kann. Damit lassen sich bei Worten, die in mehreren Kontexten vorkommen, die relevanten Tokens genauer einschränken. Im vorliegenden semantischen Netz gibt es einige ambivalente Worte. Für den Test wurde die Abkürzung „DDR" ausgewählt. In der Textsammlung, auf der das semantische Netz basiert, gibt es zum einen Informationen über DDR-RAM[4] und zum anderen Texte, die sich mit Grundstücksrecht auf dem Gebiet der ehemaligen DDR befassen. Weiterhin ist eine semantische Verknüpfung zum Investmenthaus „Dresdner Kleinwort Benson" aus dem Netz abzuleiten, die wohl entweder von einer Stellungnahme des Hauses zu Aktien eines Energiekonzerns, der in Ostdeutschland Braunkohle abbaut, oder zu einer Stellungnahme zu Aktien von „Developers Diversified Realty", Wertpapierkürzel „DDR", herrührt. Im Test wurde versucht, zum einen nur Begriffe aus dem Bereich der Informationstechnik und zum anderen nur Begriffe aus dem Bereich des Grundstücksrechts zu extrahieren. Die direkten semantischen Nachbarn des Begriffs „DDR" sind „Büros", „SDRAM", „DRAM", „Kleinwort", „Grundstücke" und „Recht". Für die Suche nach Computer-spezifischen Termen wurden „Grundstücke", „Kleinwort", „Büros" und „Recht" auf Null fixiert, für die Suche zu Termen über Grundstücksrecht „DRAM", „SDRAM" und „Kleinwort". Die so geartete Themenextraktion funktionierte durchaus gut, die genauen Ergebnisse sind in Anhang A ab Tabelle A.7 wiedergegeben. Ein Nachteil bei der Fixierung mehrere Tokens ist, daß sich die Laufzeit, wie auch in den Tabellen ersichtlich, deutlich gegenüber dem einfachen Verfahren verschlechtert.

Für die zukünftige Entwicklung ist denkbar, die Erkennung von ambivalenten Tokens zu automatisieren. Sind mehrere Kontexte streng voneinander getrennt, sollte es theoretisch reichen, den ursprünglich gewählten Begriff zu

[4] DDR = Abk. für *Double Data Rate*, doppelte Datenrate

deaktivieren und einen Begriff aus dem gewünschten Kontext zu aktivieren. Allerdings sind die verschiedenen Themenbereiche oft zusätzlich querverbunden, so daß eine automatische Erkennung nicht einfach ist.

6.1.5 Vorläufige Bewertung der Ergebnisse

Alle drei auf Basis von Hopfield-Netzen betrachteten Verfahren liefern Begriffe, die man meist intuitiv als in Beziehung zum Ursprungsbegriff stehend betrachten würde. Eine zahlenmäßige Erfassung der Qualität der Begriffe darüber hinaus ist ohne Einbeziehung des Anwendungskontextes schwierig.

Im nachfolgend diskutierten Testszenario „Query Expansion“ wird die Anwendung des semantischen Netzes zur Suchanfrage-Erweiterung näher diskutiert, hier ist eine zahlenmäßige Bewertung einfacher, da zumindest die *Precision* der gelieferten Dokumente eine Aussage über die Qualität der generierten Suchanfrage bietet.

6.2 Testszenario Query Expansion

Wie die Vorstudie zum Projekt K-Portal [10] ergeben hat, werden innerhalb der bestehenden Suchlösung in der Dresdner Bank von den Benutzern zu über 65% Einwortabfragen verwendet. Dieses Ergebnis deckt sich auch mit Untersuchungen, die für Suchmaschinen im WWW durchgeführt wurden. In [15] werden drei Studien, die verschiedene Suchmaschinen berücksichtigen, gegenübergestellt. Der Anteil der Abfragen mit nur einem Wort variieren darin von 26% für AltaVista[5] über 31% für Excite[6] bis 55% für Fireball[7]. Weiterhin ist der Gebrauch von booleschen Operatoren in der Abfrage nicht weit verbreitet. In der Dresdner Bank werden immerhin zu 21% die Operatoren *and* und *not* eingesetzt, allerdings nicht immer mit der korrekten Syntax. Bei Excite dagegen machen nur 9% und bei Fireball sogar nur 3% der Benutzer Gebrauch von der Möglichkeit boolescher Abfragen.

Eines der Hauptanwendungsgebiete der Themenextraktion aus semantischen Netzen im Projekt K-Portal wird also die Anreicherung von Einwortabfragen um weitere Terme aus dem Kontext sein. Ziel dieses Testszenarios ist es, die Tauglichkeit der in dieser Arbeit vorgestellten Algorithmen für diese Aufgabe zu überprüfen.

Für ausgewählte Suchbegriffe werden im folgenden die Suchergebnisse einer Einwortabfrage und die Ergebnisse von über das semantische Netz ermittelten angereicherten Abfragen verglichen. Zur Anreicherung wurden die zwei neben dem Ausgangsbegriff am stärksten gewichteten Begriffe der verschiedenen Algorithmen hinzugefügt und mit booleschem AND verknüpft.

[5] `www.altavista.com`, dt. Version unter `www.altavista.de`
[6] `www.excite.com`, dt. Version unter `www.excite.de`; vgl. auch [16]
[7] `www.fireball.de`

Wie die Ergebnistabellen in Anhang A zeigen, sind die Abfragen für das Verfahren nach Chen mit Fixierung und die Betrachtung der nächsten Nachbarn im semantischen Netz fast immer identisch. Der so gewählte Suchterm wurde in die hausinterne Suchmaschine der Dresdner Bank eingegeben und die ersten 10 gefundenen Dokumente wurden betrachtet. Diese Vereinfachung ist durchaus legitim und praxisnah, da die meisten Benutzer ohnehin keine weiteren Ergebnisse betrachten. So hat die Benutzerstudie ergeben, daß 54,1% der Benutzer nur die ersten drei Ergebnisse ansehen und 80% nur die erste Seite der Suchergebnisse, nämlich die ersten 10 Dokumente, in Betracht ziehen.

Im folgenden werden nur Aussagen über die *Precision*[8] innerhalb der ersten 10 Dokumente getroffen. Insbesondere eine Aussage über den *Recall*-Wert ist nicht möglich, da nicht überblickt werden kann, wie viele relevante Dokumente sich noch im Intranet der Dresdner Bank befinden. Eine höhere Präzision dürfte ohnehin für die meisten Anfragen das vorrangige Ziel sein, da die Zahl der Dokumente in einem größeren Intranet oder auch im Internet in einem hohen Maße wächst, und es von daher nicht mehr das Problem ist, überhaupt ein Dokument zu finden, sondern vielmehr, unter den vielen gefundenen Dokumenten die wirklich relevanten herauszufiltern.

Die Ergebnisse der Tests, die im Intranet der Dresdner Bank durchgeführt wurden, sind zahlenmäßig in Tabelle A.14 im Anhang erfaßt.

Drucken

Als erste Abfrage wurde „Drucken“ ausgewählt. Dies ist ein allgemeiner Begriff, wie er aber von einem „naiven“ Benutzer dennoch in der Hoffnung, beispielsweise Informationen zum Drucken aus bestimmten Anwendungen heraus zu bekommen, eingegeben werden könnte.

Für die einfache Abfrage wurden unter den ersten 10 Dokumenten nur ein relevanter Eintrag über die Menüführung eines bestimmten Druckermodells gefunden. Die drei höchstplazierten Dokumente handelten dagegen von einem Tool zur Gestaltung von graphischen Benutzeroberflächen und einer Java-Entwicklungsumgebung. Auch die anderen Dokumente waren nur als irrelevant einzustufen. Zudem war das oben erwähnte einzige relevante Dokument nicht abrufbar, da es inzwischen gelöscht oder zu einer anderen Adresse verschoben wurde.

Die zu „Drucken“ am stärksten gewichteten Nachbarbegriffe im semantischen Netz sind „Links“ und „Schaltfläche“. Insbesondere „Links“ erscheint auf den ersten Blick als Anfrage-Anreicherung wenig intuitiv. Nichtsdestotrotz lieferte die Anfrage „drucken AND links AND schaltfläche“ innerhalb der ersten 10 Dokumente praktisch nur relevante Dokumente. Einige befaßten sich mit dem Drucken aus bestimmten Anwendungen heraus. So ist

[8] zur Definition von *Precision* und *Recall* siehe Abschnitt 2.1.1

beispielsweise Treffer 3 eine Einführung in die Software MS Outlook, in der auch das Drucken von Terminplänen und Mails beschrieben wird. Treffer 5 liefert das Dokument „Die Druckerbibel“, in der alle Aspekte des Druckens eingehend beleuchtet werden.

Das Simulated Annealing Verfahren liefert als stärkste Begriffe „Links“ und „Bericht“. Die Anfrage „drucken AND links AND bericht“ lieferte auch eine deutliche Verbesserung gegenüber der Einwortabfrage, gegenüber der vorherigen Anfrage jedoch weniger relevante Dokumente. Bei den gefundenen relevanten Dokumenten handelte es sich überwiegend um Anleitungen für spezielle in der Dresdner Bank verwendete Applikation. Allgemeine Treffer wie die „Druckerbibel“ wurden nicht gefunden. Offenbar ist der Einfluß des Suchbegriffs „Bericht“ eher negativ, denn statt Informationen über das Drucken von Berichten erhält man vielmehr beispielsweise das für diese Anfrage irrelevante Dokument „Berichterstattung Eigenkompetenzprüfung“.

Das vom Chen-Algorithmus ohne Fixierung vorgeschlagene Wort „Icon-Leiste“, das eigentlich noch halbwegs intuitiv erscheint, taucht überhaupt nicht zusammen mit „Drucken“ auf. Es wurden keine Dokumente gefunden. Dies weist darauf hin, daß zwischen „Drucken“ und „Icon-Leiste“ offenbar ein paradigmatischer, aber kein syntagmatischer Zusammenhang besteht. Wie in Abschnitt 4.1.1 erwähnt, wurden im hier verwendeten semantischen Netz auch paradigmatische Zusammenhänge berechnet. Diese sind allerdings für eine restriktive Query Expansion eher hinderlich.

Lebensversicherung

Bei der Suchanfrage „Lebensversicherung“ lieferte bereits die unmodifizierte Anfrage verhältnismäßig viele relevante Dokumente, vorwiegend Informationen zum Vertrieb. Durch Query Expansion konnte die Precision daher auch nur schwer gesteigert werden. Nach Anreicherung mit „Kapitalzahlung“, dem am stärksten gewichteten Nachbarn, wurden fast ausschließlich Antragsformulare für Lebensversicherungen gefunden. Ein ähnliches Ergebnis liefert die Suchmaschine, wenn mit dem zweitplazierten Begriff „Todesfall“ angereichert wird. Die vom Chen-Algorithmus vorgeschlagene Erweiterung „Profil“ und „DIT“ verschlechterte das Ergebnis sogar, von den drei Ergebnissen war nur eines relevant. Das Simulated Annealing Verfahren schlägt „Nürnberger“ vor. Da die Dresdner Bank jedoch keine Lebensversicherungen der Nürnberger Versicherung vermittelt, wurde hierfür nichts gefunden, obwohl ein Zusammenhang zwischen den vom semantischen Netz ermittelten Begriffen besteht.

Bausparen

Auch für die Anfrage „Bausparen“ liefert bereits die unmodifizierte Anfrage 6 relevante Dokumente. Durch die Anreicherung mit „Freiheit“ konnte die

Precision allerdings nochmals gesteigert werden, so daß nur noch relevante Dokumente innerhalb der ersten 10 Dokumente gefunden wurden. Die Anreicherung „Kombi“, die vom Chen-Algorithmus ohne Fixierung vorgeschlagen wurde, brachte dagegen eine leichte Verschlechterung, da offenbar nicht nur Bausparangebote von der Dresdner Bank als „Kombi“ beworben werden und somit auch andere Dokumente gefunden werden. Die vom Simulated Annealing-Verfahren vorgeschlagene Anreicherung „Wünsche“ und „Thema“ schränkte die Ergebnismenge so stark ein, daß nur noch 8 Dokumente gefunden wurden, von denen 5 als relevant einzustufen waren. Die Anreicherungen „bausparen AND zielerreichung AND freiheit“ sowie „bausparen AND kombi AND träume“ liefert keine Treffer, da sie schon eine zu starke Einschränkung vornehmen.

Bausparen und vermögenswirksame Leistungen

Einen Test über das Thema Bausparen führte ich auch im Internet mit der deutschen Variante der Suchmaschine AltaVista[9] durch. In Anlehnung an den in Abschnitt 6.1.2 geschilderten Test des Algorithmus für maximale Cliquen wurde zunächst nach „bausparen AND vl“ und anschließend nach „bausparen AND vl AND förderung“ und nach „bausparen AND vl AND förderung AND prämienbegünstigte“ gesucht.[10] Durch die erste Erweiterung mit „Förderung“ ließ sich der Anteil relevanter Dokumente innerhalb der ersten 10 gezeigten Treffen von 4 auf 6 erhöhen. Eine weitere Anreicherung mit „Prämienbegünstigte“ verzerrte jedoch das Ergebnis, weil eine zum Thema Bausparen irrelevante Seite über prämienbegünstigte Investmentfonds der Dresdner Bank innerhalb der ersten 10 Treffer insgesamt 9 Mal von verschiedenen Servern als Treffer präsentiert wird. Schließt man diese Seiten etwa durch eine weitere Ergänzung mit „AND NOT dresdner“ aus, werden nur noch 5 Seiten gefunden, von denen zwei doppelte Treffer sind und eine nicht erreichbar ist. Die verbleibenden zwei Treffer sind allerdings stark relevant und enthalten wissenswerte Informationen über Bausparen und vermögenswirksame Leistungen, wie zum Beispiel Einkommenshöchstgrenzen für die Wohnungsbauprämie und die Arbeitnehmer-Sparzulage.

XSL

Wieder im Intranet der Dresdner Bank wurde die Anfrage zu Extensible Style Sheets, XSL, getestet. Die ursprüngliche Anfrage lieferte 3 relevante Treffer, die allerdings die weit unten plazierten Treffer 8–10 waren. Die mit den direkten Nachbarn „XHTML“ und „Language“ angereicherte Anfrage lieferte 5 Treffer, die auch besser plaziert waren. Die Anreicherung mit

[9] `www.altavista.de`

[10] Durch die Kleinschreibung wird gemäß der Syntax der Suchmaschine erreicht, daß case-insensitiv gesucht wird.

„CSS“ und „SGML“, vom Chen-Algorithmus vorgeschlagen, lieferte 9 relevante Treffer unter den ersten 10. Die Anreicherung nach dem Simulated Annealing Verfahren wurde nicht getestet.

Mannesmann

Für die Suche nach „Mannesmann“ wurden als Ergebnis Dokumente über die feindliche Übernahme von Mannesmann durch Vodafone erwartet. Tatsächlich wurden für die nicht angereicherte Anfrage jedoch nur 2 relevante Treffer gefunden. Die Anreicherung mit den vom Chen-Algorithmus vorgeschlagenen Termen „Aktion“ und „Fusion“ brachte nur 7 Treffer insgesamt, davon 2 relevante. Unter den irrelevanten Treffern befanden sich, offenbar durch den Suchbegriff „Fusion“ beeinflußt, Dokumente über die geplante Fusion der Dresdner Bank mit der Deutschen Bank. Die Anreicherung mit „Vodafone“ und „Arcor“, den am stärksten gewichteten direkten Nachbarn im semantischen Netz und gleichzeitig den vom Chen-Algorithmus mit Fixierung vorgeschlagenen Begriffe, lieferten mit 7 relevanten Treffern unter den ersten 10 eine deutliche Verbesserung.

6.2.1 Ergebnisse des Testszenarios Query Expansion

In den meisten Fällen im Test konnte durch die Anreicherung der Suchanfrage die Precision der Suchergebnisse gegenüber einer Einwortabfrage deutlich gesteigert werden. Insbesondere bei allgemeinen Suchbegriffen wie „Drukken“, bei denen sonst kaum etwas brauchbares gefunden wird, bringt die Query Expansion eine deutliche Verbesserung.

Die ausgefeilteren, auf Hopfield-Netzen basierenden Verfahren zur Traversierung des semantischen Netzes scheinen jedoch meist keine Verbesserung gegenüber der Betrachtung der direkten Nachbarschaft im semantischen Netz zu haben. Insbesondere die Traversierung mit Fermi-Funktion als Aktivierungsregel für die Neuronen schneidet schlecht ab, weil sie zu weit vom ursprünglichen Suchbegriff entfernte Begriffe stark gewichtet. Die Anreicherung der Anfrage mit vom Simulated Annealing Verfahren vorgeschlagenen Begriffen dagegen schränkt die Ergebnismenge offenbar zu stark ein.

Ein Grund für dieses eher schlechte Abschneiden der Traversierung mit Fermi-Funktion zur Query Expansion kann jedoch die Mischung syntagmatischer und paradigmatischer Kanten im verwendeten semantischen Netz sein. Für eine in diesem Testszenario durchgeführte einschränkende Anfrageerweiterung sind paradigmatische semantische Verbindungen nicht geeignet, da die Suchmaschinen nur auf Vorkommen im selben Text reagieren, was bei paradigmatischen Verknüpfungen jedoch explizit nicht der Fall sein muß und meist auch nicht ist.

Für zukünftige Entwicklungen sollte daher unbedingt eine Trennung zwischen syntagmatischen und paradigmatischen Kanten im semantischen Netz möglich sein, so daß für eine restriktive Query Expansion nur die syntagmatischen und für andere Anwendungen wie Thesaurusgruppen-Generierung oder erweiternde Query Expansion nur die paradigmatischen Verknüpfungen herangezogen werden können.

Ein weiterer wichtiger Grund ist aber auch darin zu sehen, daß die Restriktion in der „naiven" Query Expansion mit AND nicht gewichtet ist. Dies ist eigentlich absolut notwendig, da sonst nicht zwischen zusätzlichen Kontexten und Kernbegriffen unterschieden wird. Ein Beispiel für diesen negativen Effekt ist die Anreicherung von „Drucken" mit „Bericht", wie im vorherigen Abschnitt geschildert.

Die eingestellten Parameter und Schwellwerte für die verschiedenen Algorithmen sind möglicherweise auch nicht optimal. In Experimenten hat sich gezeigt, daß durch Variation der Parameter durchaus sehr unterschiedliche Ergebnisse erzielen lassen.[11] Die optimalen Werte zu finden ist dabei eine rein experimentelle Aufgabe, und wahrscheinlich müssen die Werte für jedes Netz neu angeglichen werden, um eine annähernd optimale Leistung zu erreichen. Das Problem dabei ist, daß eine allgemeingültige Vorschrift, wie die Schwellwerte einzustellen sind, nicht existiert.

Weiterhin muß man erwähnen, daß die verwendete Art der Query Expansion, bei der einfach zusätzlich Begriffe mit AND hinzugefügt wurden, natürlich recht naiv ist. Ausgefeiltere Techniken unter Verwendung des NEAR-Operators der Suchmaschine, Klammerung und Verwendung von mehr als nur zwei zusätzlichen Begriffen sind denkbar und liefern möglicherweise bessere Ergebnisse.

Auch dürfen die positiven Ergebnisse der Themenextraktion des von der Graphentheorie entlehnten Verfahrens zur maximalen Cliquensuche nicht unberücksichtigt bleiben. Möglicherweise ist auch eine Verknüpfung verschiedener Verfahren sinnvoll, in der eine breite Traversierung des Netzes mit Hopfield-Algorithmen mit kontinuierlicher Aktivierungsfunktion einer Einschränkung durch Cliquensuche vorausgeht. Die Möglichkeiten sind jedoch derart vielfältig, daß sie den Rahmen dieser Diplomarbeit sprengen würden.

[11] Eine Ausnahme von dieser Regel ist, wenn – insbesondere bei den Verfahren mit kontinuierlicher Aktivierungsfunktion – im Vergleich zu den Gewichtswerten hohe Werte für Θ eingestellt werden. Dann bleibt die Ergebnismenge weitgehend konstant, und auch mit Simulated Annealing stellen sich keine anderen Ergebnisse ein. Eine ähnliche Beobachtung hat im übrigen auch schon McCulloch schon gemacht, wenn auch auf andere Ebene. Er erwähnt in [25], daß auch, wenn die Schwelle natürlicher Neuronen sehr hoch ist, wie unter Anästhesie, in wichtigen Bereichen des menschlichen Gehirns immer noch die selben Ausgaben erscheinen, so daß beispielsweise die Atmung noch funktioniert. Aber dies nur am Rande.

6.3 Bewertung der Ergebnisse

Zusammenfassend läßt sich sagen, daß die Themenextraktion aus semantischen Netzen mittels Algorithmen auf Basis des Hopfield-Modells möglich ist und intuitive Ergebnisse liefert.

Effiziente Berechnung von Hopfield-Netzen

Die Performance dieser Verfahren konnte gegenüber früher durchgeführten Arbeiten wie der von Chen [3] in dieser Diplomarbeit deutlich gesteigert werden. Dies läßt sich auf die Verwendung des in Abschnitt 3.2.2 dieser Arbeit vorgestellten Algorithmus 3 zur effizienteren Berechnung des stabilen Zustandes von Hopfield-Netzen zurückführen. Dieser Algorithmus kann über das Einsatzfeld Themenextraktion hinaus auch für andere Anwendungen nützlich sein, bei denen mit Hopfield-Netzen gearbeitet wird. So könnte sich beispielsweise die auf Hopfield-Netzen und Simulated Annealing basierende Approximation des Problems des Handlungsreisenden oder des Cliquenproblems[12] durch die Verwendung dieses von Forgys Rete-Algorithmus abgeleiteten Verfahrens beschleunigen lassen.

Query Expansion

Weiterhin wurde gezeigt, daß sich durch die in Abschnitt 2.5.4 vorgestellte und in Abschnitt 6.2 getestete Technik der Suchanfrage-Modifikation oder *Query Expansion* die *Precision*[13] einer Suchanfrage an klassische Indexsuchmaschinen steigern läßt.

Die Verwendung von auf Hopfield-Netzen basierenden Verfahren zur Traversierung eines semantischen Netzes, aus dem die zur Anfrageerweiterung verwendeten Begriffe entnommen wurden, erwies sich jedoch im durchgeführten Test als gegenüber der Verwendung von im semantischen Netz direkt benachbarten Begriffen als nicht vorteilhaft. Dies läßt jedoch noch nicht den allgemeinen Schluß zu, daß die Verfahren nicht geeignet sind. Insbesondere ist es ein großes Problem, daß im Test keine Gewichtung der Terme bei der Query Expansion vorgenommen werden konnte. Außerdem ist das für diese Untersuchung verwendete semantische Netz keineswegs optimal. Durch eine Trennung von syntagmatischen und paradigmatischen semantischen Verknüpfungen kann mit großer Wahrscheinlichkeit eine Verbesserung der Leistungen erzielt werden. Weiterhin wäre eine Verwendung in Kombination mit dem ebenfalls in Abschnitt 2.5.4 beschriebenen Relevanz-Feedback-Verfahren denkbar.

[12] Das Problem des Handlungsreisenden und das sogenannte Cliquenproblem, die Suche der größten Clique in einem Graphen, sind NP-vollständige Probleme, die sich mit Hopfield-Netzen approximieren lassen.

[13] siehe Abschnitt 2.1.1

6.4 Ausblick

Neben der ersten Anwendung im Rahmen der Query Expansion wird es in Zukunft auch neue Anwendungsfelder für die Kombination von Hopfield- und semantischen Netzen geben. Wie bereits erwähnt wurden Hopfield-Netze auch schon in einem experimentellen, interaktiven System zur Textindexierung verwendet. Das Potential der Hopfield-Netze im Information Retrieval dürfte damit jedoch noch nicht ausgereizt sein. Ebenfalls bereits kurz erwähnt wurde die Möglichkeit des Einsatzes auf dem Gebiet der automatischen Textzusammenfassung (*Summarization*); eine Anwendung, die sicher eine genauere Betrachtung verdient und weiter verfolgt werden wird.

Anhang A

Ergebnisse

A.1 Performance-Vergleich

Algorithmus	Anz. Iterationen			Zeit in Sekunden		
	min.	max.	∅	min.	max.	∅
Chen (alle Ergebnisse)[1]	9	28	18,8	6	36	24,5
Chen (nur 1 Token aktiviert[1])	11	18	15	10	26	17,8
Fermi-Fkt.[2, 3]	1	5	3,7	2,9	3,2	2,9
Fermi-Fkt. mit Fixierung[3]	1	5	2,4	5,7	9,0	6,0
Simulated Annealing[3]	1	2	1,8	5,7	5,8	5,8
Lokale Hemmung[3]	2	10	4,3	4,1	118,6	23,7
maximale Clique[3, 4]	2	2	2	2,9	4,0	3,3

Tabelle A.1: Performance-Vergleich

[1] von Chen selbst durchgeführte Tests mit seiner Implementierung
[2] eigene Implementierung nach Chen, siehe Text
[3] Testergebnisse basieren auf Tests mit 11 verschiedenen Begriffen
[4] eigene Implementierung nach Jagota, siehe Text

A.2 Ergebnisse im Detail

A.2.1 Ergebnistabellen für Verfahren mit kontinuierlicher Aktivierungsfunktion

Wie in den nachfolgenden Tabellen ersichtlich, sind die von den Algorithmen ermittelten Worte aus dem Kontext des Ausgangsbegriffs meist intuitiv als korrekt einzustufen. Es gibt jedoch einige wenige Ausnahmen, die jedoch üblicherweise im semantischen Netz begründet sind, nicht in den Traversierungsalgorithmen. Meist sind ambivalente Begriffe schuld an diesem Problem. So wird beispielsweise bei „Geldmarkt“ unter Umständen „Bahn“ aktiviert, obwohl man diesen Begriff sicher nicht mit „Geldmarkt“ assoziieren würde. Da die Deutsche Bahn jedoch zum Sondervermögen des Bundes gehört und sowohl der Ausdruck „Bundesanleihen“ den Bund und den Geldmarkt verknüpft als auch der Begriff „Sondervermögen“ eine Verknüpfung zwischen Finanzaspekten und der Deutschen Bahn herstellt, wird der Begriff „Bahn“ dennoch aktiviert.

Tabelle A.2: Ergebnisse für Bausparen und Leitzinsen

Bausparen											
direkte Nachbarn 5,3s, 24 Tokens			Fermi-Fkt. 2,9s, 32 Tokens, 4 Iterationen			Fixierung 9,0s, 32 Tokens, 2 Iterationen			Simulated Annealing 5,7s, 18 Tokens, 2 Iterationen		
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.
1	Bausparen	0	0,005	Kombi	1	1	Bausparen	0	1	Bausparen	0
0,696	Freiheit	1	0,005	Träume	1	0,082	Freiheit	1	0,055	Wünsche	1
0,433	Zielerreichung	1	0,005	Zielerreichung	1	0,030	Zielerreichung	1	0,022	Thema	1
0,357	Fit	1	0,005	Aktuell	1	0,023	Fit	1	0,012	Förderung	1
0,319	VL	1	0,005	Netzes	2	0,020	VL	1	0,004	Versicherungen	1
0,282	Lebensversicherung	1	0,005	Vol	1	0,017	Lebensversicherung	1	0,001	Lebensversicherung	1
0,280	Vermögensaufbau	1	0,005	Lebensversicherungen	1	0,017	Vermögensaufbau	1	0,000	Controlling	1
0,259	Baufinanzierung	1	0,005	Fit	1	0,015	Baufinanzierung	1	0,000	Tochterinstitute	1
0,235	Controlling	1	0,005	Eigentum	2	0,014	Controlling	1	0,000	Sparverträge	1
0,223	Versicherungen	1	0,005	Baufinanzierung	1	0,013	Versicherungen	1	0,000	Argumente	1

Leitzinsen											
direkte Nachbarn 3,2s, 9 Tokens			Fermi-Fkt. 3,0s, 109 Tokens, 4 Iterationen			Fixierung 5,8s, 109 Tokens, 2 Iterationen			Simulated Annealing 5,8s, 16 Tokens, 2 Iterationen		
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.
1	Leitzinsen	0	0,005	Mietzinses	2	1	Leitzinsen	0	1	Leitzinsen	0
1	Basispunkte	1	0,005	Betriebsgewinns	2	0,246	Basispunkte	1	0,125	Anhebung	1
0,757	Anhebung	1	0,005	Kapitalmarktzinses	2	0,102	Anhebung	1	0,053	Erhöhung	1
0,604	FED	1	0,005	Zinsanhebung	2	0,070	Erhöhung	1	0,023	EZB	1
0,584	Erhöhung	1	0,005	Marktteilnehmern	2	0,058	FED	1	0,004	Zweifel	1
0,456	EZB	1	0,005	Reposatz	2	0,041	EZB	1	0,001	Mal	1
0,225	Notenbank	1	0,005	Geldpolitik	2	0,015	Bundesanleihen	2	0,000	Notenbank	1
0,184	Zweifel	1	0,005	Treasuries	2	0,014	Treasuries	2	0,000	FED	1
0,133	Mal	1	0,005	Grundfreibetrages	2	0,013	Notenbank	1	0,000	Heidelberg	2
—	—	—	0,005	Eingangssteuersatzes	2	0,013	Staatsanleihen	2	0,000	Effizienz	2

Tabelle A.3: Ergebnisse für Datenbanken und Drucken

Datenbanken											
direkte Nachbarn 3,0s, 37 Tokens			Fermi-Fkt. 2,9s, 37 Tokens, 1 Iteration			Fixierung 5,8s, 37 Tokens, 1 Iteration			Simulated Annealing 5,7s, 17 Tokens, 1 Iteration		
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.
1	Datenbanken	0	1	Datenbanken	0	1	Datenbanken	0	1	Datenbanken	0
0,295	Ressource	1	0,018	Ressource	1	0,018	Ressource	1	0,046	Ratings	1
0,240	Informix	1	0,014	Informix	1	0,014	Informix	1	0,028	Tabellen	1
0,222	Sybase	1	0,013	Sybase	1	0,013	Sybase	1	0,010	Inhalte	1
0,176	Dateien	1	0,011	Dateien	1	0,011	Dateien	1	0,005	Ressource	1
0,176	Tabellen	1	0,011	Tabellen	1	0,011	Tabellen	1	0,001	Ausgaben	1
0,175	Oracle	1	0,011	Oracle	1	0,011	Oracle	1	0,000	Anbieter	1
0,156	Zugriff	1	0,010	Zugriff	1	0,010	Zugriff	1	0,000	Datenbestände	1
0,152	Middleware	1	0,010	Middleware	1	0,010	Middleware	1	0,000	Dateien	1
0,135	Restore	1	0,009	Restore	1	0,009	Restore	1	0,000	Katalog	1

Drucken											
direkte Nachbarn 3,0s, 69 Tokens			Fermi-Fkt. 3,0s, 142 Tokens, 4 Iterationen			Fixierung 5,8s, 142 Tokens, 2 Iterationen			Simulated Annealing 5,8s, 17 Tokens, 2 Iterationen		
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.
1	Drucken	0	0,005	Maturities	1	1	Drucken	0	1	Drucken	0
0,596	Links	1	0,005	Icon-Leiste	2	0,056	Links	1	0,100	Links	1
0,417	Schaltfläche	1	0,005	Maskenfelder	1	0,028	Schaltfläche	1	0,025	Bericht	1
0,373	Kopfzeile	1	0,005	Datensatzfelder	1	0,024	Kopfzeile	1	0,010	Vielzahl	1
0,364	Symbol	1	0,005	Funktionsbyte	1	0,023	Symbol	1	0,003	Gruppe	1
0,356	Funktionsbyte	1	0,005	CICSPRO	1	0,022	Funktionsbyte	1	0,001	Sammeln	1
0,270	Dialogfenster	1	0,005	SQL-Anweisungen	1	0,016	Dialogfenster	1	0,000	Maske	1
0,230	Strg	1	0,005	Host-Muster	1	0,014	Strg	1	0,000	Emulation	1
0,223	AP	1	0,005	STD	1	0,013	AP	1	0,000	Symbol	1
0,209	Statistiken	1	0,005	TEST-DAT	1	0,013	Statistiken	1	0,000	Strg	1

XSL											
direkte Nachbarn 3,0s, 6 Tokens			Fermi-Fkt. 2,9s, 10 Tokens, 4 Iterationen			Fixierung 5,7s, 10 Tokens, 2 Iterationen			Simulated Annealing 5,8s, 14 Tokens, 2 Iterationen		
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.
1	XSL	0	0,0054863	XHTML	1	1	XSL	0	1	XSL	0
0,667	XHTML	1	0,0054863	XSL	0	0,074	XHTML	1	0,057	XML	1
0,301	Language	1	0,0054863	CSS	1	0,019	Language	1	0,006	XHTML	1
0,239	Style	1	0,0054863	SGML	2	0,014	Style	1	0,000	Grundlage	2
0,201	XML	1	0,0054863	Language	1	0,012	XML	1	0,000	OI	2
0,176	CSS	1	0,0054863	Spezifikation	2	0,011	CSS	1	0,000	Beitrag	2
—	—	—	0,0054863	Style	1	0,006	SGML	2	0,000	Weiterentwicklung	2
—	—	—	0,0054863	HTML	2	0,006	Geräten	2	0,000	Einsatz	2
—	—	—	0,0054863	Geräten	2	0,006	HTML	2	0,000	Unterschied	2
—	—	—	0,0054863	XML	1	0,006	Spezifikation	2	0,000	Unterstützung	2

Geldmarkt											
direkte Nachbarn 3,0s, 24 Tokens			Fermi-Fkt. 2,9s, 49 Tokens, 3 Iterationen			Fixierung 5,8s, 49 Tokens, 2 Iterationen			Simulated Annealing 5,8s, 19 Tokens, 1 Iteration		
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.
1	Geldmarkt	0	0.007	Geldmarkt	0	1	Geldmarkt	0	1	Geldmarkt	0
0,579	Finanzanlagen	1	0.007	Einkommensbesteuerung	2	0.053	Finanzanlagen	1	0.046	Sonstige	1
0,565	Sondervermögen	1	0.007	Bundes	2	0.050	Sondervermögen	1	0.026	Verzinsung	1
0,487	Wertpapieren	1	0.007	Bund	2	0.037	Wertpapieren	1	0.014	Renten	1
0,474	Rendite	1	0.007	Bahn	2	0.035	Rendite	1	0.029	Zinssatz	1
0,442	Wiederanlage	1	0.006	Wertpapier-Sondervermögen	2	0.031	Wiederanlage	1	0.001	Wertpapier	1
0,341	Futures	1	0.006	Kapitalanlagegesellschaft	2	0.021	Futures	1	0.001	Wertpapieren	1
0,323	Rentenfonds	1	0.006	Anteilscheine	2	0.020	Rentenfonds	1	0.000	Niveau	1
0,285	Papieren	1	0.006	Grundstücks	2	0.017	Papieren	1	0.000	Finanzanlagen	1
0,275	Portefeuille	1	0.006	Sondervermögens	2	0.016	Portefeuille	1	0.000	Rentenanlagen	1

Tabelle A.4: Ergebnisse für XSL und Geldmarkt

Mannesmann											
direkte Nachbarn 3,0s, 57 Tokens			Fermi-Fkt. 3,0s, 114 Tokens, 4 Iterationen			Fixierung 5,8s, 97 Tokens, 3 Iterationen			Simulated Annealing 5,8s, 16 Tokens, 2 Iterationen		
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.
1	Mannesmann	0	0,006	Aktion	1	1	Mannesmann	0	1	Mannesmann	0
1	Vodafone	1	0,006	Fusion	1	0,323	Vodafone	1	0,056	Siemens	1
0,836	Arcor	1	0,006	Telekom	1	0,169	Orange	1	0,023	Aussichten	1
0,708	Orange	1	0,006	Siemens	1	0,135	Arcor	1	0,010	Mio	1
0,505	Einigung	1	0,006	Milliarden	1	0,073	Einigung	1	0,004	Angebot	1
0,380	Kapitalerhöhung	1	0,006	Aktiengesellschaft	1	0,039	Otelo	1	0,001	Aktienkurs	1
0,365	Otelo	1	0,006	Deutsche	1	0,025	Kapitalerhöhung	1	0,000	Aktionäre	1
0,289	Aktion	1	0,006	Gewinn	1	0,022	Synergien	1	0,000	Strategie	1
0,230	Aktionär	1	0,006	Aktionäre	1	0,017	Aktion	1	0,000	Übernahme	1
0,229	Atecs	1	0,006	Engineering	1	0,017	Zusammenschlusses	1	0,000	Ausscheiden	1

Lebensversicherung											
direkte Nachbarn 5,6s, 75 Tokens			Fermi-Fkt. 3,2s, 273 Tokens, 4 Iterationen			Fixierung 5,9s, 273 Tokens, 3 Iterationen			Simulated Annealing 5,8s, 17 Tokens, 2 Iteration		
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.
1	Kapitalzahlung	1	0,00588392	PROFIL	2	1	Lebensversicherung	0	1	Lebensversicherung	0
1	Lebensversicherung	0	0,00585349	DIT	2	0,312	Kapitalzahlung	1	0,078	Nürnberger	1
0,738	Nürnberger	1	0,00578288	Leben	2	0,126	Todesfall	1	0,046	AG	1
0,700	Kapital	1	0,00575982	Freiraum	2	0,119	Versicherungssumme	1	0,010	TDM	1
0,681	Versicherungssumme	1	0,00575845	Frauen	2	0,108	OPTION	1	0,003	Varianten	1
0,567	Todesfall	1	0,00572891	Männer	2	0,097	Kapital	1	0,001	Risiko	1
0,499	PLUS	1	0,00564315	Frau	2	0,096	Nürnberger	1	0,000	Ansprüchen	1
0,484	OPTION	1	0,005637	Phasen	2	0,073	PLUS	1	0,000	Abschluß	1
0,475	Dez	1	0,0056203	Kombination	2	0,056	Beiträge	1	0,000	Klassische	1
0,450	Versicherungsschutz	1	0,00561213	Erstausgabe	2	0,036	Dez	1	0,000	Alter	1

Tabelle A.5: Ergebnisse für Mannesmann und Lebensversicherung

UMTS											
direkte Nachbarn 3,2s, 10 Tokens			Fermi-Fkt. 3,1s, 64 Tokens, 5 Iterationen			Fixierung 5,9s, 64 Tokens, 5 Iterationen			Simulated Annealing 5,7s, 15 Tokens, 2 Iterationen		
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.
1	UMTS	0	1.000	WAP	1	1	UMTS	0	1	UMTS	0
1	Bluetooth	1	1.000	UMTS	0	1.000	WAP	1	0,079	Schl	1
1	WAP	1	0,999	Bluetooth	1	0,999	Bluetooth	1	0,050	Potential	1
0,934	GSM	1	0,996	Schl	1	0,996	Schl	1	0,038	Forum	1
0,744	Schl	1	0,900	Forum	1	0,900	Forum	1	0,007	Universal	1
0,674	Forum	1	0,256	PDA	2	0,256	PDA	2	0,000	Vergabe	1
0,376	Universal	1	0,231	Wireless	2	0,231	Wireless	2	0,000	GSM	1
0,243	Bandbreite	1	0,229	Gem	2	0,229	Gem	2	0,000	Bandbreite	1
0,113	Potential	1	0,188	GSM	1	0,188	GSM	1	0,000	WAP	1
0,094	Vergabe	1	0,168	Special	2	0,168	Special	2	0,000	Folie	2

Winword											
direkte Nachbarn 2,9s, 6 Tokens			Fermi-Fkt. 2,9s, 24 Tokens, 4 Iterationen			Fixierung 5,7s, 24 Tokens, 2 Iterationen			Simulated Annealing 5,8s, 17 Tokens, 2 Iteration		
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.
0,005	OPTION	2	1	Winword	0	1	Winword	0	1	Winword	0
1	Musterbrief	1	0,005	Kundenbroschüre	2	0,231	Musterbrief	1	0,058	Excel	1
1	Winword	0	0,005	PVSA	2	0,028	Anklicken	1	0,045	Musterbrief	1
0,412	Anklicken	1	0,005	Beraterinformation	2	0,013	Standardanzeige	2	0,043	Anklicken	1
0,214	Excel	1	0,005	Plakat	2	0,013	Excel	1	0,001	Style	1
0,171	Style	1	0,005	Standardanzeige	2	0,011	Style	1	0,000	Arbeitsbuch	2
0,113	MS	1	0,005	Eigennutzer	2	0,010	Plakat	2	0,000	Access	2
—	—	—	0,005	Initiator	2	0,009	MS	1	0,000	Umsteiger	2
—	—	—	0,005	PLUS	2	0,008	Mailing	2	0,000	Fortgeschrittene	2
—	—	—	0,005	Beratungsprozeß	2	0,008	Selektionslisten	2	0,000	Outlook	2

Tabelle A.6: Ergebnisse für UMTS und Winword

DDR											
direkte Nachbarn 3,4s, 7 Tokens			Fermi-Fkt. 2,9s, 16 Tokens, 4 Iterationen			Fixierung 5,7s, 16 Tokens, 2 Iterationen			Simulated Annealing 5,7s, 10 Tokens, 2 Iterationen		
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.
1	DDR	0	0,005	Mbit	2	1	DDR	0	1	DDR	0
1	Büros	1	0,005	SDRAM	1	0,231	Büros	1	0,124	Büros	1
0,617	SDRAM	1	0,005	DRAM	1	0,061	SDRAM	1	0,046	Recht	1
0,362	DRAM	1	0,005	EDO	2	0,028	DRAM	1	0,012	Kleinwort	1
0,198	Kleinwort	1	0,005	DDR	0	0,012	Kleinwort	1	0,004	Grundstücke	1
0,191	Grundstücke	1	0,005	Bit	2	0,012	Grundstücke	1	0,002	DRAM	1
0,064	Recht	1	0,005	Grundstücke	1	0,007	Recht	1	0,001	SDRAM	1
—	—	—	0,005	MB	2	0,007	Mbit	2	0,000	Verteilung	2
—	—	—	0,005	MHz	2	0,007	Eröffnung	2	0,000	Niederlassungen	2
—	—	—	0,005	Kleinwort	1	0,007	EDO	2	0,000	Eröffnung	2

DDR, Kontext RAM						DDR, Kontext Grundstücke					
Fermi-Fkt. m. Fixierung 17,1s, 9 Tokens, 2 Iterationen			Simulated Annealing 17,8s, 1 Token, 3 Iterationen			Fermi-Fkt. m. Fixierung 14,3s, 7 Tokens, 2 Iterationen			Simulated Annealing 16,3s, 12 Tokens, 6 Iterationen		
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.
1	DDR	0	1	DDR	0	1	DDR	0	1	Mineralgewinnungsrecht	2
0,061	SDRAM	1	—	—	—	0,231	Büros	1	1	Eigentümers	2
0,028	DRAM	1	—	—	—	0,012	Grundstücke	1	1	Erbbaurecht	2
0,007	Mbit	2	—	—	—	0,007	Recht	1	1	DDR	0
0,007	EDO	2	—	—	—	0,007	Eröffnung	2	1	Grundbuch	2
0,006	MHz	2	—	—	—	0,006	Niederlassungen	2	1	Eigentümer	2
0,006	MB	2	—	—	—	0,006	Verteilung	2	1	Grundstücke	1
0,006	Bit	2	—	—	—	—	—	—	1	Grundstücks	2
0,006	PC	2	—	—	—	—	—	—	1	Gebäude	2
—	—	—	—	—	—	—	—	—	1	Grundstück	2

Tabelle A.7: Ergebnisse für DDR, verschiedene Kontexte

A.2.2 Ergebnistabellen für graphentheoretische Verfahren

Bausparen								
direkte Nachbarn 5,3s, 24 Tokens			maximale Clique 2,9s, 3 Tokens, 2 Iterationen			lokale Hemmung 7,0s, 5 Tokens, 2 Iterationen		
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.
1	Bausparen	0	0,378	Bausparen	0	1	Träume	1
0,696	Freiheit	1	0,356	Träume	1	1	Baufinanzierung	1
0,433	Zielerreichung	1	0,345	Wünsche	1	1	Bausparen	0
0,357	Fit	1	—	—	—	1	Thema	1
0,319	VL	1	—	—	—	1	Wünsche	1
0,282	Lebensversicherung	1	—	—	—	—	—	—
0,280	Vermögensaufbau	1	—	—	—	—	—	—
0,259	Baufinanzierung	1	—	—	—	—	—	—
0,235	Controlling	1	—	—	—	—	—	—
0,223	Versicherungen	1	—	—	—	—	—	—

Leitzinsen								
direkte Nachbarn 3,2s, 9 Tokens			maximale Clique 3,0s, 2 Tokens, 2 Iterationen			lokale Hemmung 5,8s, 6 Tokens, 3 Iterationen		
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.
1	Leitzinsen	0	0,757	Leitzinsen	0	1	Leitzinsen	0
1	Basispunkte	1	0,757	Anhebung	1	1	Basispunkte	1
0,757	Anhebung	1	—	—	—	1	EURIBOR	2
0,604	FED	1	—	—	—	1	Erhöhung	1
0,584	Erhöhung	1	—	—	—	1	Anhebung	1
0,456	EZB	1	—	—	—	1	Förderung	2
0,225	Notenbank	1	—	—	—	—	—	—
0,184	Zweifel	1	—	—	—	—	—	—
0,133	Mal	1	—	—	—	—	—	—

Tabelle A.8: Ergebnisse für Bausparen und Leitzinsen

Datenbanken								
direkte Nachbarn 3,0s, 37 Tokens			maximale Clique 3,0s, 2 Tokens, 2 Iterationen			lokale Hemmung 5,8s, 37 Tokens, 1 Iteration		
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.
1	Datenbanken	0	0,056	Datenbanken	0	1	Bond	2
0,295	Ressource	1	0,056	Ratings	1	1	NET	2
0,240	Informix	1	—	—	—	1	Werbungskosten	2
0,222	Sybase	1	—	—	—	1	SNiFF	2
0,176	Dateien	1	—	—	—	1	Stückzinsen	2
0,176	Tabellen	1	—	—	—	1	Sun	2
0,175	Oracle	1	—	—	—	1	Oracle	1
0,156	Zugriff	1	—	—	—	1	Java	2
0,152	Middleware	1	—	—	—	1	Cancel	2
0,135	Restore	1	—	—	—	1	Continuus	1

Drucken								
direkte Nachbarn 3,0s, 69 Tokens			maximale Clique 3,9s, 4 Tokens, 2 Iterationen			lokale Hemmung 29,6s, 7 Tokens, 4 Iterationen		
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.
1	Drucken	0	0,999	Links	1	1	Einfügen	1
0,596	Links	1	0,845	Drucken	0	1	Befehl	1
0,417	Schaltfläche	1	0,731	Einfügen	1	1	Kopieren	1
0,373	Kopfzeile	1	0,496	Löschen	1	1	Bearbeiten	2
0,364	Symbol	1	—	—	—	1	Menü	1
0,356	Funktionsbyte	1	—	—	—	1	Löschen	1
0,270	Dialogfenster	1	—	—	—	1	Drucken	0
0,230	Strg	1	—	—	—	—	—	—
0,223	AP	1	—	—	—	—	—	—
0,209	Statistiken	1	—	—	—	—	—	—

Tabelle A.9: Ergebnisse für Datenbanken und Drucken

XSL								
direkte Nachbarn 3,0s, 6 Tokens			maximale Clique 3,5s, 4 Tokens, 2 Iterationen			lokale Hemmung 4,0s, 7 Tokens, 3 Iterationen		
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.
1	XSL	0	1.411	Language	1	1	XSL	0
0,667	XHTML	1	1.022	XML	1	1	CSS	1
0,301	Language	1	0,771	Style	1	1	Markup	2
0,239	Style	1	0,741	XSL	0	1	Language	1
0,201	XML	1	—	—	—	1	Style	1
0,176	CSS	1	—	—	—	1	HTML	2
—	—	—	—	—	—	1	XML	1

Geldmarkt								
direkte Nachbarn 3,0s, 6 Tokens			maximale Clique 3,0s, 2 Tokens, 2 Iterationen			lokale Hemmung 23,6s, 10 Tokens, 4 Iterationen		
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.
1	Geldmarkt	0	0,061	Geldmarkt	0	1	Einkünfte	2
0,579	Finanzanlagen	1	0,061	Sonstige	1	1	Rendite	1
0,565	Sondervermögen	1	—	—	—	1	Wertpapieren	1
0,487	Wertpapieren	1	—	—	—	1	Zinsen	2
0,474	Rendite	1	—	—	—	1	Erträge	2
0,442	Wiederanlage	1	—	—	—	1	Kosten	2
0,341	Futures	1	—	—	—	1	Anlage	2
0,323	Rentenfonds	1	—	—	—	1	Sonstige	1
0,285	Papieren	1	—	—	—	1	TDM	2
0,275	Portefeuille	1	—	—	—	1	Summe	2

Tabelle A.10: Ergebnisse für XSL und Geldmarkt

Mannesmann								
direkte Nachbarn 3,0s, 57 Tokens			maximale Clique 3,8s, 3 Tokens, 2 Iterationen			lokale Hemmung 26,8s, 8 Tokens, 6 Iterationen		
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.
1	Mannesmann	0	0,405	Mannesmann	0	1	Celanese	2
1	Vodafone	1	0,345	Fusion	1	1	Aventis	2
0,836	Arcor	1	0,321	Siemens	1	1	Hoechst	2
0,708	Orange	1	—	—	—	1	Anschaffungskosten	2
0,505	Einigung	1	—	—	—	1	Abspaltung	2
0,380	Kapitalerhöhung	1	—	—	—	1	Aktie	1
0,365	Otelo	1	—	—	—	1	Aktionäre	1
0,289	Aktion	1	—	—	—	1	Aktien	2
0,230	Aktionär	1	—	—	—	—	—	—
0,229	Atecs	1	—	—	—	—	—	—

Lebensversicherung								
direkte Nachbarn 5,6s, 75 Tokens			maximale Clique 3,2s, 3 Tokens, 2 Iterationen			lokale Hemmung 21,6s, 11 Tokens, 5 Iteration		
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.
1	Kapitalzahlung	1	0,168	Lebensversicherung	0	1	Berlinische	1
1	Lebensversicherung	0	0,165	Aktiengesellschaft	1	1	LV	1
0,738	Nürnberger	1	0,110	AG	1	1	Lebensversicherung	0
0,700	Kapital	1	—	—	—	1	Nürnberger	1
0,681	Versicherungssumme	1	—	—	—	1	Niederlassung	2
0,567	Todesfall	1	—	—	—	1	Stuttgart	2
0,499	PLUS	1	—	—	—	1	Nürnberg	2
0,484	OPTION	1	—	—	—	1	München	2
0,475	Dez	1	—	—	—	1	Hannover	2
0,450	Versicherungsschutz	1	—	—	—	1	Hamburg	2

Tabelle A.11: Ergebnisse für Mannesmann und Lebensversicherung

UMTS								
direkte Nachbarn 3,2s, 10 Tokens			maximale Clique 3,1s, 3 Tokens, 2 Iterationen			lokale Hemmung 5,7s, 4 Tokens, 3 Iterationen		
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.
1	UMTS	0	1.113	UMTS	0	1	UMTS	0
1	Bluetooth	1	1.112	WAP	1	1	Bluetooth	1
1	WAP	1	0,226	Potential	1	1	WAP	1
0,934	GSM	1	—	—	—	1	Schl	1
0,744	Schl	1	—	—	—	—	—	—
0,674	Forum	1	—	—	—	—	—	—
0,376	Universal	1	—	—	—	—	—	—
0,243	Bandbreite	1	—	—	—	—	—	—
0,113	Potential	1	—	—	—	—	—	—
0,094	Vergabe	1	—	—	—	—	—	—

Winword								
direkte Nachbarn 2,9s, 6 Tokens			maximale Clique 3,2s, 3 Tokens, 2 Iterationen			lokale Hemmung 5,8s, 13 Tokens, 4 Iteration		
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.
1	Musterbrief	1	1.214	Excel	1	1	MS	1
1	Winword	0	1.113	MS	1	1	Microsoft	2
0,412	Anklicken	1	0,326	Winword	0	1	Icons	2
0,214	Excel	1	—	—	—	1	Symbole	2
0,171	Style	1	—	—	—	1	Office	2
0,113	MS	1	—	—	—	1	Anklicken	1
—	—	—	—	—	—	1	Navigationsleiste	2
—	—	—	—	—	—	1	Word	2
—	—	—	—	—	—	1	Windows	2
—	—	—	—	—	—	1	Outlook	2

Tabelle A.12: Ergebnisse für UMTS und Winword

DDR											
direkte Nachbarn 3,4s, 7 Tokens			maximale Clique 3,2s, 2 Tokens, 2 Iterationen			lokale Hemmung 5,7s, 10 Tokens, 2 Iterationen					
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.			
1	DDR	0	0,064	DDR	0	1	Büros	1			
1	Büros	1	0,064	Recht	1	1	Recht	1			
0,617	SDRAM	1	—	—	—	1	Niederlassungen	2			
0,362	DRAM	1	—	—	—	1	Verteilung	2			
0,198	Kleinwort	1	—	—	—	1	Personal	2			
0,191	Grundstücke	1	—	—	—	1	KS	2			
0,064	Recht	1	—	—	—	1	Konzernstab	2			

DDR, Kontext RAM						DDR, Kontext Grundstücke					
maximale Clique 18,0s, 3 Tokens, 2 Iterationen			lokale Hemmung 17,3s, 5 Tokens, 4 Iterationen			maximale Clique 11,4s, 2 Tokens, 2 Iterationen			lokale Hemmung 14,6s, 8 Tokens, 3 Iterationen		
Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.	Gew.	Ausgabe	Ab.
1.394	SDRAM	1	1	SDRAM	1	1	DDR	0	1	DDR	0
1.139	DRAM	1	1	DRAM	1	0,064	Recht	1	1	Büros	1
1	DDR	0	1	DDR	0	—	—	—	1	Recht	1
—	—	—	1	Bit	2	—	—	—	1	Niederlassungen	2
—	—	—	1	MB	2	—	—	—	1	Verteilung	2
—	—	—	1	MHz	2	—	—	—	1	Personal	2
—	—	—	—	—	—	—	—	—	1	KS	2
—	—	—	—	—	—	—	—	—	1	Konzernstab	2

Tabelle A.13: Ergebnisse für DDR, verschiedene Kontexte

A.3 Testszenario Query Expansion

Die Zahlen in Tabelle A.14 geben die *Precision* der ersten 10 Treffer der Suchanfrage an, also die Zahl der als relevant erachteteten Dokumente geteilt durch 10. Wurden weniger als 10 Treffer gefunden, ist die Zahl als Bruch mit der Anzahl der Treffer im Nenner angegeben, sowie als Dezimalzahl auf die erste Nachkommastelle gerundet.

Da das Verfahren mit Fermi-Funktion Fixierung fast immer die benachbarten Begriffe, die am stärksten gewichtet sind, auch am stärksten gewichtet, wurde auf eine gesonderte Betrachtung dieses Verfahrens verzichtet.

Einwortabfrage		Nachbarn	
Anfrage	Precision	Anfrage	Precision
Drucken	$0,1$	Links, Schaltfläche	$1,0$
Lebensversicherung	$0,8$	Kapitalzahlung	$1,0$
Bausparen	$0,6$	Freiheit	$1,0$
Mannesmann	$0,2$	Vodafone, Arcor	$0,7$
XSL	$0,3$	XHTML, Language	$0,5$

Tabelle A.14: Precision-Werte für Query Expansion

	Fermi-Fkt.		Simulated Annealing	
	Anfrage	Precision	Anfrage	Precision
Drucken	Icon-Leiste	—[5]	Links, Bericht	$0,7$
Lebensversicherung	Profil, DIT	$1/3 = 0,3$	Nürnberger	—[5]
Bausparen	Kombi	$0,5$	Wünsche, Thema	$5/8 = 0,6$
Mannesmann	Aktion, Fusion	$2/7 = 0,3$	Siemens, Aussichten	—[5]
XSL	CSS, SGML	$0,9$	—[6]	—[6]

Tabelle A.15: Precision-Werte für Query Expansion

[5] Keine Dokumente gefunden.
[6] Nicht getestet.

Anhang B

Das experimentelle Windows-Frontend

Zum Test der Implementierung der Themenextraktionsverfahren entstand eine experimentelle Oberfläche unter Windows, die ein interaktives Traversieren des semantischen Netzes mit den in dieser Arbeit entwickelten und implementierten Verfahren erlaubt. Das semantische Netz kann aus einer DB/2-Datenbank oder aus Dateien im CSV-Format[1] geladen werden.

B.1 Bedienung

Die Bedienung des Demo-Programms „Themenextraktion aus semantischen Netzen" erfolgt dialogfeldbasiert. Abbildung B.1 zeigt den Hauptdialog. Um das semantische Netz zu traversieren, gibt man im einfachsten Fall im Texteingabefeld den Ausgangsbegriff ein, wählt in der Auswahlbox den zu verwendenden Algorithmus aus und klickt auf „Suchen". Die Ausgabe erscheint im rechten Teil des Fensters. Die Begriffe sind nach ihrer Gewichtung (Aktivierung nach Stabilisierung des Hopfield-Netzes) sortiert. Abbildung B.1 zeigt das Hauptfenster des Programms mit einer einfachen Suchanfrage.

Ein Doppelklick in die Ausgabeliste fügt den gewählten Begriff in die Eingabezeile ein, ein Klick auf die Schaltfläche „Übernehmen" stellt die oben angezeigte alte Anfrage wieder ins Eingabefeld.

Es können mehrere Begriffe durch einzelne Leerzeichen getrennt eingegeben werden. Eingangsbegriffe können mit einem führenden „+" auf eine konstante Aktivierung von 1 fixiert werden. Mit einem führenden „–" werden Begriffe auf 0 fixiert. Der Sinn der Fixierung von Begriffen (bzw. Neuronen) ist in Abschnitt 4.2.4 näher erklärt. In Abbildung B.1 wurde der Begriff „Datenbanken" fixiert.

[1] CSV = *comma seperated value*, durch Komma getrennte Daten

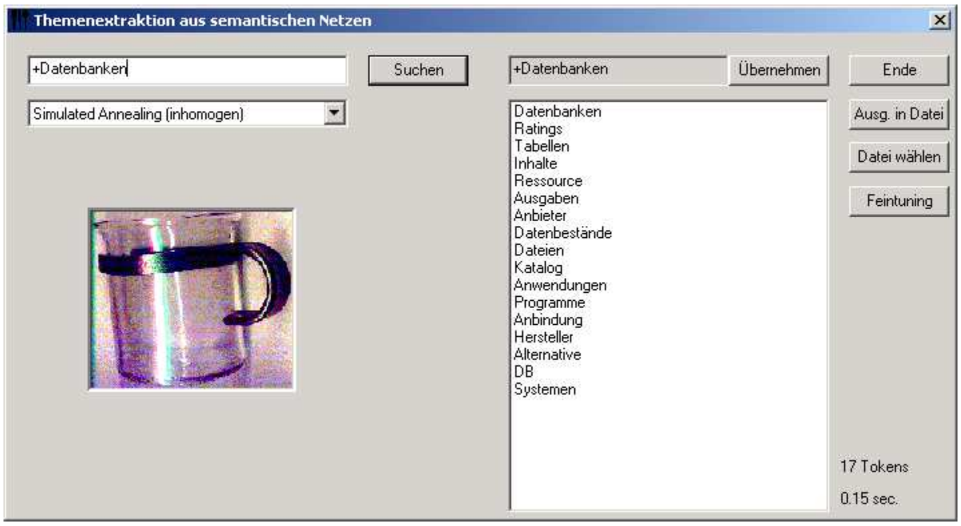

Abbildung B.1: Das experimentelle Windows-Frontend

Nach der Ausgabe werden rechts unten im Fenster die Zahl der insgesamt aktivierten Begriffe und die für die Berechnung benötigte Zeit angezeigt.

B.1.1 Ausgabe in Datei

Die Ausgabe kann auch in eine Datei geschrieben werden. Zusätzlich werden dann zu jedem Begriff die Aktivierungsstärke und die Entfernung vom Ausgangsbegriff ausgegeben. Zur Ausgabe in eine Datei muß man zunächst die Ausgabedatei auswählen. Nach einem Klick auf die Schaltfläche „Datei wählen" öffnet sich ein Auswahldialog. Wird eine bestehende Datei ausgewählt, wird diese nicht überschrieben, die Ausgabe wird immer ans Ende angefügt. Auf diese Weise können mehrere Berechnungsläufe in einer Datei protokolliert werden. Vor jeder Begriffsliste wird ein Kopf eingefügt, der Name und Parameter des Algorithmus sowie die zur Berechnung benötigte Zeit und andere Daten zeigt. Die Ausgabe erfolgt bei einem Klick auf die Schaltfläche „Ausg. in Datei". Ein Beispiel für die vom Programm erzeugte Ausgabe zeigt Abbildung B.2. Die Aktivierungsstärke wird, wie im Beispiel auch, in Exponentialschreibweise dargestellt, wenn sie in Dezimalschreibweise aus Platzgründen nicht darstellbar wäre. Im Kopf werden auch die Parameter gelistet, die für den gewählten Algorithmus keine Bedeutung haben, wie im Beispiel der Wert „Behinderung".

B.1.2 Feintuning

Nach einem Klick auf die Schaltfläche „Feintuning" werden im linken Teil des Fensters weitere Eingabefelder angezeigt, mit denen sich die Parameter

```
----
+Drucken, Simulated Annealing
Temperatur, 0.8
Abkuehlung, 0.8
Bias, 2
Behinderung, 3
Schranke, 1
max. Anzahl, 10
Schwelle, 0
Suchradius, 4
max. Iterationen, -1
Ablauf:
Iterationen, 2
Tokens aktiviert, 17
Zeit, 5.818
0, 1, Drucken, 0
1, 0.100319, Links, 1
2, 0.0253758, Bericht, 1
3, 0.00998286, Vielzahl, 1
4, 0.00301724, Gruppe, 1
5, 0.000610948, Sammeln, 1
6, 0.00018595, Maske, 1
7, 1.24168e-005, Emulation, 1
8, 5.07023e-006, Symbol, 1
9, 6.94056e-008, Strg, 1
```

Abbildung B.2: Beispiel für eine Ausgabedatei der Algorithmen im Detail einstellen lassen. Es sind nur jeweils die Felder aktiviert, die für den gewählten Algorithmus von Bedeutung sind. Die Daten können für jeden Algorithmus getrennt eingestellt werden. Abbildung B.3 zeigt das Programmfenster mit den erweiterten Einstellmöglichkeiten. Ein Klick auf die Schaltfläche „Verstecken“ führt zurück zur einfachen Ansicht (wie in Abbildung B.1).

Bedeutung der Parameter

Im folgenden sind die Bedeutungen der einzelnen Parameter näher erläutert:

max. Iterationen – Anzahl von Iterationen, nach denen zwangsweise abgebrochen wird, oder -1, falls kein vorzeitiger Abbruch erfolgen soll.

konvergiert nach ... Iterationen – Nur Ausgabe: Anzahl Iterationen, nach denen sich der stabile Zustand eingestellt hat, oder -1, wenn die Berechnung nach *max. Iterationen* abgebrochen wurde.

Temperatur – Für Verfahren mit kontinuierlicher Übergangsfunktion der Parameter T der Funktion.

Abkühlung – Für Simulated Annealing-Verfahren der Faktor, mit dem die *Temperatur* vor jedem Berechnungsschritt (bei inhomogenem Simulated Annealing) bzw. bei Erreichen des thermalen Gleichgewichts (bei homogenem Simulated Annealing) multipliziert wird.

Bias – Für Verfahren mit kontinuierlicher Übergangsfunktion der Parameter Θ der Funktion.

Behinderung – Für das Jagota-Verfahren für maximale Cliquen und das Verfahren mit lokaler Behinderung die Gewichtung der hemmenden Kanten.

Schranke – Für Verfahren mit kontinuierlicher Übergangsfunktion Faktor, mit dem die Übergangsfunktion multipliziert wird. Ist *Schranke* < 1, ist die Aktivierung des Netzes stärker lokal. Werte > 1 können ein „Aufschaukeln“ der Aktivierung bewirken, die Konvergenzkriterien für Hopfield-Netze sind dann nicht mehr erfüllt.

max. Anzahl – Anzahl der Begriffe, die ausgegeben werden. Wirkt sich nur auf die Ausgabe und die Ausgabe in eine Datei aus, nicht auf die Berechnung.

Schwelle – Für Algorithmen mit festem Schwellwert (Subgraph-Algorithmen, Clique-Algorithmus, Verfahren mit lokaler Hemmung) der Schwellwert. Für alle anderen Algorithmen ein zusätzlicher Schwellwert, der überschritten werden muß, damit ein Begriff ausgegeben wird, sich aber nicht auf die Berechnung auswirkt.

Suchradius – Maximale Länge des Weges zwischen den Ausgangsbegriffen und dem noch betrachteten Teil des semantischen Netzes. Wenn der *Suchradius* und die *Temperatur* zu hoch und *Bias* zu niedrig eingestellt werden, kann sich die Aktivierung über weite Teile des semantischen Netzes ausbreiten. Es werden dann unter Umständen äußerst viele Tokens aktiviert und die Berechnung des stabilen Zustandes kann *sehr* lange dauern. Ein Wert größer als 4 ist in den seltensten Fällen sinnvoll, da Begriffe, die im semantischen Netz weit voneinander entfernt sind, praktisch keinen kontextuellen Bezug mehr zueinander haben.

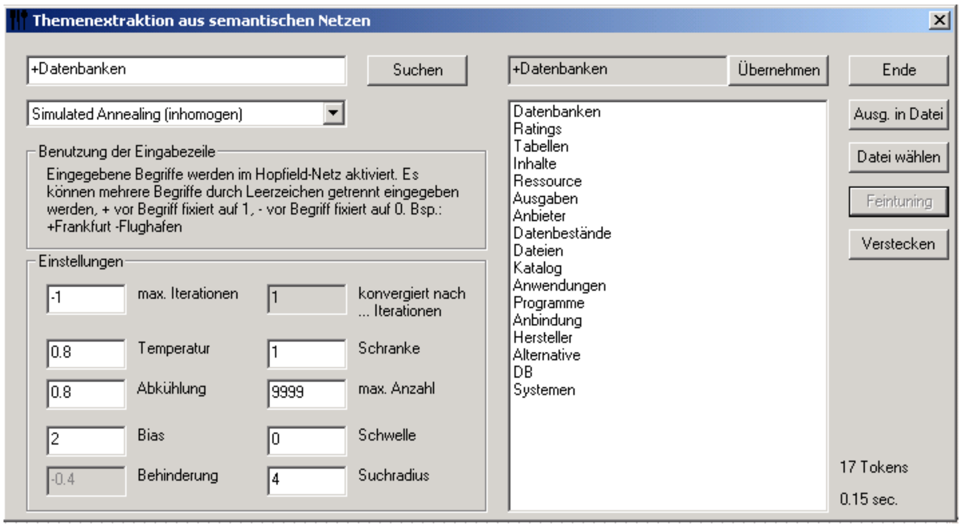

Abbildung B.3: Das experimentelle Windows-Frontend

B.2 Konfiguration

Die Konfiguration erfolgt über eine Datei `h3demo.ini`, die sich in dem Verzeichnis befinden muß, von dem aus das Programm gestartet wird. Die Datei folgt einer relativ einfachen Syntax: Die Datei ist in Sektionen aufgeteilt, deren Namen in eckige Klammern eingeschlossen sind. Es werden die Sektionen `global` und `h3demo` verwendet. Innerhalb der Sektionen können Zuweisungen der Form *`Name = Wert`* erfolgen. Leerzeilen werden ignoriert, Kommentarzeilen können mit einem Semikolon eingeleitet werden. Die möglichen Einstellungen sind im folgenden erklärt. Generell gilt, daß alle Angaben, die sich auf Datenbankeinstellungen beziehen, wie z.B. Benutzername und Paßwort, weggelassen werden können, wenn das semantische Netz aus Dateien gelesen wird. Umgekehrt können alle Angaben, die sich auf Dateien beziehen, weggelassen werden, wenn aus der Datenbank gelesen wird. Nur `databaseName` und `databaseSchema` stehen in der Sektion `global`, alle anderen Angaben stehen in der Sektion `h3demo`.

databaseName – Name der zu verwendeten Datenbank, aus der das semantische Netz gelesen wird.

databaseSchema – Verwendetes Datenbankschema.

readFromFile – Wenn diese Angabe auf `true` gesetzt wird, wird das semantische Netz aus Dateien statt aus der einer DB/2-Datenbank gelesen. Wird die Angabe auf `false` gesetzt wird, wird die Datenbank verwendet. Wird die Angabe weggelassen, wird `false` impliziert.

fileTokens – Datei, aus der die Tabelle „Tokens“ gelesen wird, mit vollständiger Pfadangabe.

fileAssocs – Datei, aus der die Tabelle „TokenAssocs“ gelesen wird, mit vollständiger Pfadangabe.

databaseUser – Benutzername für die Datenbank.

databasePasswd – Paßwort für die Datenbank.

runId – Wenn ein bestimmter Berechnungslauf (*TokenAssocRun* des semantischen Netzes verwendet werden soll, kann er hier angegeben werden. Wird diese Angabe weggelassen oder auf -1 gesetzt, wird der zu verwendende Berechnungslauf über die Tabelle *activeTokenAssocRun* ermittelt.

B.3 Format der Eingabedateien

Wie bereits mehrfach erwähnt wurde, kann das semantische Netz anstatt aus einer DB/2-Datenbank[2] auch aus zwei Dateien gelesen werden, die die Daten der Tabellen „Tokens“ und „TokenAssocs“ im CSV-Format enthalten. Die Daten in der Tokens-Datei (normalerweise als `tokens.csv` bezeichnet) folgen dem Format:

```
Id, Language, "Word", Type, Frequency, InDocs
```

Die Bedeutung der einzelnen Felder ist dabei:

Id – Eindeutige Nummer des Tokens (In der Datenbank Primärschlüssel der Tabelle).

[2]In Kapitel 5 ist das Datenmodell beschrieben.

```
; Ini-File für h3demo
; Themenextraktion aus semantischen Netzen

[global]

databaseName = DREBA
databaseSchema = MINING

[h3demo]

; Aus Datei lesen?
readFromFile = true

; Dateiname fuer Tabelle Tokens
fileTokens = c:\daten\not_normalized\tokens.csv

; Dateiname fuer Tabelle TokenAssocs
fileAssocs = c:\daten\not_normalized\tokenassocs.csv

; Username fuer Datenbank
databaseUser = andreas

; Passwort fuer Datenbank
databasePasswd = dremi17

; Zu verwendende RunId
; (Kann weggelassen oder auf -1 gesetzt werden,
; dann wird die runId aus der Tabelle
; activeTokenAssocRun gelesen)
runId = 2
```

Abbildung B.4: Beispiel für eine Konfigurationsdatei

Language – Schlüssel der Sprache des Wortes. Wird für diese Anwendung nicht verwendet. In der Datenbank Fremdschlüssel einer Tabelle *Languages*, die aber hier ebenfalls nicht verwendet wird.

Word – Ein Token, d.h. ein Wort, in Anführungszeichen eingefaßt.

Type – Wort-Typ. Hier nicht verwendet.

Frequency – Absolute Häufigkeit des Vorkommens des Wortes. Wird hier nicht verwendet.

InDocs – Nicht verwendet.

Die nicht benutzten Felder sind aus Gründen der Kompatibilität mit der Datenbank aufgenommen und werden ignoriert. Auf diese Art ist es möglich, von einer bestehenden Datenbank einen Export in CSV-Dateien vorzunehmen und diese zu verwenden. Weil auf diese Art keine Informationen verloren gehen, können die Dateien später in eine andere Datenbank importiert werden.

Die Daten der TokenAssocs-Datei (normalerweise als `tokenassocs.csv` bezeichnet) folgen dem Format:

```
RunID, TokenId1, TokenId2, Strength
```

Die Bedeutung der einzelnen Felder ist dabei:

RunID – Gibt in der Datenbank den Berechnungslauf an, der diese Assoziation berechnet hat. Wird für semantische Netze, die aus Dateien eingelesen werden, ignoriert.

TokenId1 – Nummer des Tokens, von dem die semantische Verbindung ausgeht.

TokenId2 – Nummer des Tokens, an dem die semantische Verbindung endet. Die Verbindungen müssen symmetrisch sein, d.h. für jede Verbindung von A nach B muß eine Verbindung von B nach A in der gleichen Stärke existieren.

Strength – Stärke der Verbindung. Der Nachkommateil wird mit einem Punkt statt mit Komma abgetrennt. Es sind auch Werte größer als 1 erlaubt, allerdings werden diese beim Einlesen auf 1 begrenzt.

Die Daten in der TokenAssocs-Datei müssen aufsteigend nach TokenId1 sortiert sein. Die Abbildungen B.5 und B.6 zeigen beispielhaft Ausschnitte aus den Dateien.

```
50552, 0, "Spareinlagen", 0, 455, 0
50556, 0, "Vormerkungen", 0, 222, 0
50559, 0, "Stückzinsen", 0, 1151, 0
50560, 0, "Rückmeldungen", 0, 62, 0
50566, 0, "Jahresvergleich", 0, 83, 0
50567, 0, "Umsatzvolumen", 0, 528, 0
50569, 0, "Trägersystem", 0, 38, 0
50573, 0, "Signals", 0, 41, 0
50598, 0, "Handling", 0, 222, 0
50604, 0, "Wertes", 0, 644, 0
50606, 0, "Betragsfeld", 0, 82, 0
50621, 0, "Bereitstellen", 0, 375, 0
50624, 0, "Teamleitern", 0, 59, 0
50711, 0, "Nacharbeiten", 0, 51, 0
50721, 0, "Informationsbeschaffung", 0, 101, 0
50729, 0, "Bundesländer", 0, 223, 0
50743, 0, "Mehrzahl", 0, 200, 0
50744, 0, "Eingrenzung", 0, 48, 0
```

Abbildung B.5: Ausschnitt aus tokens.csv

```
0, 162729, 19073, 0.06082440
0, 162729, 37626, 0.06531020
0, 162730, 119979, 2.43983006
0, 162730, 26161, 0.23587300
0, 162730, 31336, 0.68263698
0, 162730, 67099, 0.13841601
0, 162731, 162678, 2.23129010
0, 162731, 57527, 1.48699999
0, 162731, 26161, 0.25262401
0, 162731, 61204, 0.72832203
0, 162759, 167313, 8.27058983
0, 162759, 36389, 0.88800699
```

Abbildung B.6: Ausschnitt aus tokenassocs.csv

Algorithmenverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] Boyer, Paul D.: Concept Mapping – Frequently Asked Questions / University of Wisconsin-Parkside. 1997. – Forschungsbericht

[2] Brause, Rüdiger: *Neuronale Netze*. Stuttgart : B.G. Teubner, 1991

[3] Chen, Hsinchun: Machine Learning for Information Retrieval: Neural Networks, Symbolic Learning, and Genetic Algorithms / MIS Department, College of Business and Public Administration, University of Arizona. Tucson, Arizona, 1993. – Forschungsbericht

[4] Chung, Yi-Ming ; Pottenger, William M. ; Schatz, Bruce R.: Automatic Subject Indexing using an Associative Neural Network / CANIS – Community Architectures for Network Information Systems Laboratory, University of Illinois at Urbana-Champaign. 1998. – Forschungsbericht

[5] Forgy, C.L.: Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. In: *Artificial Intelligence* 19 (1982), Nr. 17–37

[6] Freese, Eric: Using Topic Maps for the representation, management and discovery of knowledge / ISOGEN International. Dallas, USA, 2000. – Forschungsbericht

[7] Gerick, Thomas: Sinnsuche – Retrieval: Methoden, Trends, Produkte. In: *iX* (2000), Dezember, S. 124–128

[8] Glover, Eric ; Flake, Gary ; Lawrence, Steve ; Birmingham, William P. ; Kruger, Andries ; Giles, C. L. ; Pennock, David: Improving Category Specific Web Search by Learning Query Modifications. In: *Symposium on Applications and the Internet, SAINT*. San Diego, CA, January 8–12 2001

[9] Grund, Wolfgang: *ATHEM2 – Berechnung Kantengewichte*. Hünfelden: H.A.S.E GmbH, November 2000. – intern

[10] Hilbig, Jens: *Vorstudie K-Portal*. Dezember 2000. – intern

[11] Hopfield, John J.: Neural networks and physical systems with emergent collective computational abilities. In: *Proceedings of the National Academy of Sciences* Bd. 79, 1982, S. 2554–2558

[12] Hopfield, John J.: Neurons with graded response have collective computational properties like those of two-state neurons. In: *Proceedings of the National Academy of Sciences* Bd. 81, 1984, S. 3088–3092

[13] ISO/IEC. *13250, Topic Navigation Maps.* September 1998

[14] Jagota, Arun: The Hopfield-style Network as a Maximal-Clique Graph Machine / Dept Of Computer Science, State University Of New York At Buffalo. 1990. – Forschungsbericht

[15] Jansen, Bernard J. ; Pooch, Udo: A Review of Web Searching Studies and a Framework for Future Research / University of Maryland, Asian Division. 2000. – Forschungsbericht

[16] Jansen, Bernard J. ; Spink, A. ; Saracevic, T.: Real life, real users, and real needs: A study and analysis of user queries on the web. In: *Information Processing and Management* 36 (2000), Nr. 2, S. 207–222

[17] Jungnickel, Dieter: *Graphen, Netzwerke und Algorithmen.* Mannheim/Wien/Zürich : Wissenschaftsverlag, 1990

[18] Kratzer, Klaus P.: *Neuronale Netze.* Hanser, 1993

[19] Lawrence, Steve: Context in Web Search. In: *IEEE Data Engineering Bulletin* 23 (2000), Nr. 3, S. 25–32

[20] Lawrence, Steve ; Giles, C. L.: Searching the World Wide Web. In: *Science* 280 (1998), Nr. 5360, S. 98–100

[21] Lawrence, Steve ; Giles, C. L.: Accessibility of Information on the Web. In: *Nature* 400 (1999), Nr. 6740, S. 107–109

[22] Lebeth, Kai: *K-Portal.* Projektpräsentation. Dezember 2000. – intern

[23] Lin, Dekang: Automatic Retrieval and Clustering of Similar Words. In: *COLING-ACL98.* Winnipeg, Manitoba, Canada, 1998

[24] Lin, Dekang: Extracting Collocations from Text Corpora. In: *First Workshop on Computational Terminology.* Winnipeg, Manitoba, Canada, 1998

[25] McCulloch, Warren S.: What Is A Number, that a Man May Know It, and a Man, that He May Know a Number? In: *General Semantics Bulletin.* Lakeville, Connecticut, 1961. – dt. Übersetzung erschienen in [26]

[26] McCulloch, Warren S.: *Verkörperungen des Geistes – Computerkultur Bd. 7.* Wien : Springer, 2000

[27] McCulloch, Warren S. ; Pitts, Walter H.: A Logical Calculus of the Ideas Immanent in Nervous Activity. In: *Bulletin of Mathematical Biophysics* 5 (1945), S. 115–133. – dt. Übersetzung erschienen in [26]

[28] Mitra, M. ; Singhal, A. ; Buckley, C.: Improving Automatic Query Expansion. In: *Proceedings of the 21 st Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, 206214.*

[29] Petith, Heiner: *Software-Entwicklung im Projekt K-Portal.* Dezember 2000. – intern

[30] Rath, Hans H.: Mozart oder Kugel – Mit Topic Maps intelligente Informationsnetze aufbauen. In: *iX* (1999), Dezember, S. 149–155

[31] Richter, Michael M.: Einführung in die künstliche Intelligenz / Universität Kaiserslautern. 1991. – Vorlesungsskript

[32] Richter, Michael M.: Konnektionismus / Universität Kaiserslautern. 1991. – Vorlesungsskript

[33] Rijsbergen, C.J. van: *Information Retrieval.* Second Edition. London : Butterworths, 1979

[34] Rötzer, Florian: Direct Hit – Eine bessere Suchmaschine? In: *Telepolis* (1998)

[35] Salton, Gerard: *Automatic Text Processing.* Addison-Wesley, 1989

[36] Salton, Gerard ; McGill, Michael J.: *Information Retrieval – Grundlegendes für Informationswissenschaftler.* McGraw-Hill, 1987

[37] Sander-Beuermann, Dr. W.: Schatzsucher – Die Internet-Suchmaschinen der Zukunft. In: *c't – Magazin für Computertechnik* 13 (1998), S. 178–184

[38] Schmidt, Ingrid ; Müller, Carolin: Zaubernetz – Inhaltsstrukturen und Topic Maps als Potenzial neuer Informationstechnik. In: *iX* (2000), November, S. 100–107

[39] Schneider, Henner: Grundlagen neuronaler Netze / Fachhochschule Darmstadt. 1997. – Vorlesungsskript

[40] Sigel, Alexander: Towards knowledge organization with Topic Maps / Informationszentrum Sozialwissenschaften. Bonn, 2000. – Forschungsbericht

[41] Warth, Dora: Künstliche Intelligenz: Spracherkennung und Sprachverstehen / Johannes Gutenberg-Universität Mainz, Fachbereich Angewandte Sprach- und Kulturwissenschaft in Germersheim. 1997. – Referat zum Hauptseminar P̈sycholinguistik: Mentale Prozesse in der Sprachverarbeitung;̈ Seminarleiter: Prof. Dr. Dieter Huber. Im Internet unter `www.fask.uni-mainz.de/user/warth/Ki.html`

[42] Warth, Dora: Einführung in die Linguistik / Johannes Gutenberg-Universität Mainz, Fachbereich Angewandte Sprach- und Kulturwissenschaft in Germersheim. 1999. – Einführender Text nach einer Vorlesung von Prof. Dr. Dieter Huber. Im Internet bei `www.fask.uni-mainz.de/inst/iaspk/Linguistik/Welcome.html`

[43] Winkler, Hartmut: Suchmaschinen. In: *Telepolis* (1997)

[44] Wolfram, Dietmar: A Query-Level Examination of End User Searching Behaviour on the Excite Search Engine. In: *Dimensions of a Global Information Science – Proceedings of the 28th Annual Conference* Canadian Association for Information Science, 2000