DIPLOMARBEIT

Auswahl journalistischer Artikel anhand der Nachrichtenwert-Theorie durch Softwarealgorithmen

ausgeführt am Institut für Softwaretechnik
der Technischen Universität Wien

unter Anleitung von

ao.univ.Prof. Dr. Andreas Rauber

durch

Thomas Christian Pfeiffer

Mat.Nr.: 9325691

Guglgasse 14/103
1110 Wien

Wien, am 3. Oktober 2005

Thomas Christian Pfeiffer

Zusammenfassung

Journalisten entscheiden, welche Artikel interessant sind und welche nicht. Dabei beziehen sie sich auf eine Menge von sogenannten „Nachrichten-Werten“, die eine Entscheidung ermöglichen, ob eine Nachricht weiter verfolgt und dann auch – mehr oder weniger auffällig – publiziert wird. Journalisten haben natürlich nicht eine Liste dieser Nachrichtenwerte an einer Wand des Redaktionsraums hängen, aber sie treffen ihre Entscheidungen unbewusst anhand dieser Kriterien. Die *Nachrichtenwert-Theorie* beschreibt diese Kriterien.

Diese Diplomarbeit beschreibt, wie diese Kriterien mit Software implementiert werden können. Techniken zur Informationsgewinnung bezüglich des Nachrichtenwerts von Artikeln werden ebenso vorgestellt wie das Training eines neuronalen Netzes zur Selektion publizierenswerter Artikel.

Abstract

Journalists decide which articles are newsworthy and which are not. In this process they refer to a set of so-called "news-values". These criteria enable them to determine whether a "story" is followed up and then whether a story makes it into the news, competing against all the other possible items. Of course, journalists do not refer to a list pinned on the wall of their office, but they unconsciously measure a potential news item against these criteria. The *Theory of News-Values* describes these criteria.

This master thesis describes whether these criteria can be implemented in software. It utilizes information retrieval techniques to index articles in terms of news-values, and utilizes neural networks to train classifiers for selecting articles for publication.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Oftmals stellt sich die Frage, warum, in welchem Umfang und an welcher Stelle („Prominenz“) Artikel in Zeitungen, aber auch in Nachrichtensendungen des Rundfunks und des Fernsehens platziert werden. Abgesehen von redaktionellen Einflüssen (Zeitdruck, Artikelgröße, ...) wirken sich die Faktoren der Nachrichtenwert-Theorie auf die Auswahl der Nachrichten aus: Überraschung, Tragweite, Personalisierung und vieles mehr (siehe Kapitel 2.4) stellen jene Faktoren dar, die Nachrichten (aus Sicht des Journalisten) berichtenswert erscheinen lassen.

Ziel dieser Diplomarbeit ist, den Journalisten eine Vorauswahl an Artikeln zu bieten; konkret sollen jene Nachrichten aus dem „Artikelstrom“ einer Nachrichtenagentur wie zB der *APA*[1], *OTS*[2] oder *pressetext.austria*[3] ausgefiltert werden, die publizierenswert erscheinen.

1.1 Vorgehensweise

Diese Arbeit gliedert sich in zwei große Hauptabschnitte: Im ersten Teil werden die publizistisch-sprachwissenschaftlichen Grundlagen erörtert, im zweiten Teil folgen die informatischen Dimensionen sowohl theoretisch wie auch anhand eines neuronalen Netzes, das Schritt für Schritt aufgebaut und verfeinert wird.

In Kapitel 2 werden die Nachrichtenwert-Theorie und vor allem die einzelnen Nachrichtendimensionen und deren Nachrichtenfaktoren vorgestellt. Kapitel 3 erläutert einige Eigenschaften natürlicher Sprache, um sich daran anschließend der Auswahl von Schlüsselwörtern und der elektronischen Repräsentation von Texten zu widmen. Den Abschluss des publizistisch-sprachwissenschaftlichen Teils bildet Kapitel 2.5, das (umgangs-)sprachliche Begriffe für die einzelnen Nachrichtenfaktoren definiert.

Kapitel 4 bringt eine Einführung in neuronale Netze mit Hauptaugenmerk auf *Feedforward-Netze*, die im Rahmen dieser Arbeit verwendet werden. Kapitel 5 beschreibt die Nachrichten-„Daten“ in Hinblick auf Wortverteilungen wie auch auf Ähnlichkeiten der Dokumente untereinander. Kapitel 6 zeigt schließlich, wie die Artikelsuche konkret realisiert werden kann. Dabei wird ein mit SNNS[4] trainiertes neuronales Netz schrittweise verfeinert und der jeweilige Lernerfolg mit unterschiedlichen Parametern grafisch anschaulich präsentiert. Den Abschluss bildet der Test des Netzes und die Analyse der Ergebnisse.

[1] `http://www.apa.at/`
[2] `http://www.ots.at/`
[3] `http://www.pressetext.at/`
[4] *Stuttgart Neural Network Simulator*, `http://www-ra.informatik.uni-tuebingen.de/SNNS/`, [Zell (1995)]

Die Zusammenfassung in Kapitel 7 rekapituliert die einzelnen Schwerpunkte der Arbeit und beleuchtet kurz andere Ansätze, mit deren Hilfe Nachrichten nach ihrer Veröffentlichungswürdigkeit selektiert werden könnten.

Im Anhang werden die die einzelnen Nachrichtenfaktoren repräsentierenden Begriffe definiert, im Anschluss wird die im Rahmen dieser Diplomarbeit erstellte Software vorgestellt, mit deren Hilfe die einzelnen Nachrichten für das Training des neuronalen Netzes aufbereitet werden.

Datenbasis dieser Diplomarbeit sind alle Artikel, die im ersten Quartal 1999 im Ressort „Innenpolitik Österreich“ von der *APA* veröffentlicht wurden.[5] Als „Zielmedium“ ist DER STANDARD[6] vorgesehen; Artikel dieser Zeitung liegen vollständig für das Jahr 1999 elektronisch am Institut für Softwaretechnik vor.

In diesem Zusammenhang möchte ich auch Frau MONIKA DRESSEL von der *APA* herzlich danken, die mir sehr unbürokratisch und rasch die gewünschten Artikel zur Verfügung stellte.

1.2 Grundlegender Ablauf der Artikelauswahl

Die grundlegende Vorgangsweise zur Auswahl von Nachrichten durch Journalisten ist in Abbildung 1.1 dargestellt.

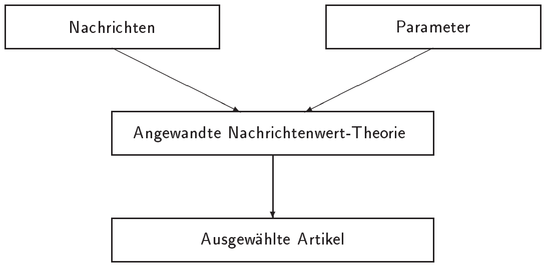


Abbildung 1.1: Grundlegender Ablauf der Artikelauswahl

Auf der einen Seite stehen die von Nachrichtenagenturen (aber auch anderen Medien) veröffentlichten Meldungen. Auf der anderen Seite finden sich Parameter zur Steuerung des Auswahlablaufs: Zeitung, Blattlinie, Ressort, ... Dies führt schließlich zu einer Menge von Artikeln, die veröffentlichenswert erscheinen.

[5] Dabei handelt es sich um 6.112 Artikel!
[6] `http://www.derstandard.at/`

1.3 Güte des Algorithmus

Um festzustellen, ob der mit Hilfe eines neuronalen Netzes implementierte Algorithmus effizient im Sinne einer „realistischen“ Artikelauswahl arbeitet, müssen Bewertungskriterien geschaffen werden. Als einfachste und wirkungsvollste Bewertung bietet sich hierbei ein Vergleich zwischen den durch den Algorithmus gefundenen Artikel und den im Vergleichszeitraum im STANDARD publizierten Artikel an.

Diese Bewertung erfolgt durch die beim Trainieren des neuronalen Netzes ständig errechnete Fehlerrate; beim Test des trainierten Netzes wird vereinfacht formuliert die Differenz zwischen der Anzahl der errechneten veröffentlichten und der Anzahl der durch den STANDARD wirklich veröffentlichten Nachrichten herangezogen (Details werden in Kapitel 6.10.1, Seite 60, diskutiert).

Für das Trainieren des Netzes werden die Daten der Monate Jänner und Februar des Jahres 1999 verwendet.[7] Die Qualität des Algorithmus wird anhand der durch den Algorithmus für veröffentlichenswert befundenen Artikel der ersten Tage des Monats März gemessen – hier zeigt sich, wieviele Artikel gemäß Algorithmus und wieviele durch den STANDARD veröffentlicht würden bzw. wurden.

Auf Kriterien wie Geschwindigkeit, Ressourcenbedarf oder Skalierbarkeit wird keine Rücksicht genommen, da diese im Vergleich zur Effektivität der Artikelauswahl nicht im Vordergrund stehen.

[7] Es handelt sich um 3.559 Artikel.

2 Die Nachrichtenwert-Theorie

2.1 Theoretische Konzepte zur Nachrichtenauswahl

Schon seit den fünfziger Jahren wird die Nachrichtenauswahl der Massenmedien theoretisch und empirisch in einer Vielzahl von Studien untersucht. Im Wesentlichen kann man diese Untersuchungen zu drei Forschungstraditionen zusammenfassen: die Gatekeeper-Theorie, die „New Bias“-Forschung und die Nachrichtenwert-Theorie. Diese Theorien weisen natürlich Querverbindungen und Überschneidungen auf, sodass eine eindeutige Zuordnung nicht immer möglich ist.

- Die **Gatekeeper-Theorie** besagt, dass Journalisten als „Filter“ agieren und in dieser Eigenschaft aufgrund verschiedenster Kriterien (indivuelle Prädispositionen und/oder institutionelle Faktoren) entscheiden, ob Nachrichten erwähnenswert sind oder nicht. Dieser Prozess läuft auf mehreren Ebenen ab (beispielsweise beim Reporter, beim Redakteur oder beim Herausgeber) und dient letztlich der Begrenzung der Informationsmenge. (Eine Entscheidung für einen bestimmten Nachrichteninhalt führt natürlich zu einer Unterdrückung anderer Themen.) Ergänzend – und in Erweiterung der Gatekeeper-Theorie – muss erwähnt werden, dass die Nachrichten den Filter nicht unverändert passieren, sondern modifiziert werden und dass manche Teilaspekte ausführlicher behandelt werden als andere (vgl. [Wimmer (1996), S. 40f.]). Eine detaillierte Darstellung der Gatekeeper-Theorie findet sich zum Beispiel in [Staab (1990), S. 12ff.].
- Ziel der **„New Bias“-Forschung** ist es, Unausgewogenheiten, Einseitigkeiten und politische Tendenzen in der Medienberichterstattung zu messen und über deren Ursachen Aufschluss zu erhalten. Man untersucht dabei im Wesentlichen den Zusammenhang zwischen den politischen Einstellungen von Journalisten und ihrer Nachrichtenauswahl bzw. ihrer Berichterstattung mit Hilfe von Inhaltsanalysen oder mittels experimenteller Untersuchungen. Die „New Bias“-Forschung wird beispielsweise in [Staab (1990), S. 27ff.] ausführlich dargestellt.
- Die **Nachrichtenwert-Theorie** wird in diesem Kapitel ausführlich diskutiert. Kurz zusammengefasst kann gesagt werden, dass Journalisten zu publizierende Nachrichten aufgrund bestimmter Kriterien, die Annahmen über das Interesse der Rezipienten darstellen, auswählen. Diese Auswahl wird durch die Nachrichtenwert-Theorie beschrieben.

Die Grundlage all dieser Forschungsansätze sind die Untersuchungen von WINFRIED SCHULZ, wonach Nachrichten keinesfalls Realität widerspiegeln, sondern vielmehr eine Interpretation der Umwelt seien – die (politische) Realität wird also durch die „Medienrealität“ konstituiert (vgl. [Erbring (1989), S. 157]). Wichtig hierbei ist allerdings, dass Rezipienten Nachrichten als wirklich, als wahr, akzeptieren, wodurch Nachrichten für den Rezipienten wiederum zu Realität werden (vgl. [Wimmer (1996), S. 40]).

2.2 Motive für eine Theorie der Nachrichtenwerte

„Wichtig ist [. . .], dass die Medien überhaupt eine Gebrauchswertigkeit für das Publikum aufweisen“ und dass diese Gebrauchswertigkeit „eine optimale Verwertung kommunikationsindustriellen Kapitals“[1] garantiere ([Holzer (1973), S. 72]). Die Medien müssten sich daher auf die Gebrauchswertansprüche, die das Publikum an sie richtet, einlassen; nur auf dieser Basis könnten die Medien überhaupt ihre ökonomische (und ideologische) Funktion erfüllen.

Sah Holzer im Jahr 1973 die Gebrauchswertansprüche des Publikums noch im „Verlangen nach subjektiv wirklichen Lösungen wirklicher Lebensprobleme“, so kann dies mittlerweile erweitert werden auf den Wunsch nach Unterhaltung im weitesten Sinne. Der Begriff der Unterhaltung möge hier also auch den Begriff der Information beinhalten. Medien müssen sich daher fragen: Was will das Publikum? Und im Speziellen: Welche Ereignisse (Nachrichten) müssen (wie aufbereitet) gebracht werden (um möglichst hohe Einschaltziffern bzw. Auflagezahlen zu gewährleisten)?

Schon auf den ersten Blick ist erkennbar, dass Neues mehr interessiert als Altes, Veränderung mehr als Bestand, Normverletzung mehr als Normalität, Gefahr mehr als Sicherheit, Nahes mehr als Fernes, Prominente mehr als Unbekannte usw. „Hinter diesen Werten verbergen sich sozial und kulturell bedingte Konventionen [. . .], die freilich nicht beliebig von Journalisten erfunden werden können, sondern letztlich vom Publikum ausgehen“[2] ([Erbring (1989), S. 158]).

Einen Erklärungsansatz, welche Kriterien – also *Nachrichtenwerte* – Ereignisse erfüllen müssen, um zu Nachrichten zu werden, liefert die Nachrichtenwert-Theorie.[3]

2.3 Historische Entwicklung

Schon 1922 schuf Walter Lippmann in seinem Buch „Public Opinion“ die erkenntnistheoretische Grundlage der Nachrichtenwert-Theorie. Er erkannte, „dass die Wirklichkeit aufgrund ihrer Komplexität nicht adäquat erkannt werden könne, sondern Realitätsauffassung grundsätzlich nach Stereotypen erfolge“ ([Staab (1990), S. 40]). Medien unterliegen – ebenso wie die menschliche Erkenntnis – einem Dilemma: Nachrichten widerspiegeln nicht die Realität, sondern sind das Ergebnis (subjektiver) Selektionsentscheidungen, die nicht auf Regeln, sondern auf Konventionen beruhen. Lippmann wirft daher die bereits oben erwähnte Frage nach den Kriterien, die ein Ereignis erfüllen müsse, um zu einer Nachricht zu werden, auf. Dabei wird erstmals der Begriff des *Nachrichtenwerts* („news value“) geprägt.

[1] Alle wörtlichen Zitate wurden in die neue Rechtschreibung transkribiert.
[2] Auch wenn solche Verzerrungen der „Realität“ zu bedauern sind – Kritik an diesen Werten wünscht sich letztlich ein anderes Publikum.
[3] Eine komplette Zusammenfassung der Nachrichtenwert-Theorie findet sich in [Staab (1990)].

In weiterer Folge spaltete sich die Nachrichtenwert-Theorie in eine *amerikanische* und in eine *europäische Forschungstradition*. Eine genaue Unterscheidung dieser beiden Traditionen würde den Rahmen dieser Arbeit überschreiten, weswegen hier nur auf die europäische Tradition detailliert eingegangen werden soll.[4]

Einar Östgaard begründete 1965 die europäische Forschungstradition, indem er die Ursachen für die Verzerrungen im Nachrichtenfluss systematisierte. Er unterschied dabei zwischen *externen* und *internen Faktoren*:

- **Externe Nachrichtenfaktoren** beeinflussen den Nachrichtenfluss von außen. Konkret sind damit Maßnahmen von Regierungen (Zensur), von Nachrichtenagenturen (politisch motiviertes „Nachrichten-Management“) oder von Eigentümern (wirtschaftliche Interessen) gemeint.
- **Interne Nachrichtenfaktoren** sind jene Aspekte von Nachrichten, die sie für den Rezipienten interessant machen. Östgaard unterscheidet dabei zwischen Simplifikation (einfache Nachrichten werden komplexen vorgezogen), Identifikation (Bekanntes interessiert mehr als Unbekanntes) und Sensationalismus (dramatische Sachverhalte erlangen mehr Aufmerksamkeit).

Ebenfalls 1965 wurde von Johan Galtung und Mari Holmboe Ruge ein weit über das Konzept von Östgaard hinausgehender Ansatz entworfen. Sie formulierten in einem ersten Schritt zwölf Nachrichtenwerte[5] und darauf aufbauend fünf Hypothesen über deren Zusammenwirken. Schulz kritisierte diesen Ansatz sowohl in der Methodik als auch in der zugrunde liegenden empirischen Studie; auch das theoretische Grundkonzept weise Unschärfen auf (vgl. [Staab (1990), S. 63f.]). In weiterer Folge bemängelt Schulz ein erkenntnistheoretisches Problem der damaligen Nachrichtenwert-Theorie: die als „Falsifikationsversuch“ angelegten Vergleiche von „faktischer Realität“ und „Medienrealität“. Solche Falsifikationsversuche „müssten [. . .] prinzipiell scheitern, da über das faktische Geschehen kein intersubjektiv gültiger Konsens zu erzielen sei und somit nur verschiedene Interpretationen der faktischen Realität miteinander verglichen werden könnten“ ([Staab (1990), S. 80]).

Schulz legte nach dieser Kritik eine Nachrichtenwert-Theorie vor, in der Nachrichtenfaktoren nicht mehr als Merkmale von Ereignissen gesehen werden, sondern als „journalistische Hypothesen von Realität“. Die Grundhypothese lautete daher: „Je mehr eine Meldung dem entspricht, was Journalisten für wichtige und mithin berichtenswerte Eigenschaften der Realität halten, desto größer ist ihr Nachrichtenwert“ ([Staab (1990), S. 81]). Die vom Bürger erlebte Realität wird somit zu einer Medien-

[4] Die amerikanische Tradition unterscheidet sich von der europäischen im Wesentlichen durch andere Nachrichtenfaktoren (Unmittelbarkeit, Nähe, Prominenz, Ungewöhnlichkeit, Konflikt und Konsequenz [Staab (1990), S. 49]) und dem Fehlen von Unterteilungen (siehe Kapitel 2.4, Seite 7). Eine detaillierte Darstellung der amerikanischen Tradition der Nachrichtenwert-Theorie findet sich beispielsweise in [Staab (1990), S. 42ff.].
[5] Diese Nachrichtenwerte werden beispielsweise in [Noelle-Neumann, Schulz, Wilke (2000), S. 331] wiedergegeben.

realität, über die die Journalisten aufgrund der journalistischen Nachrichtenauswahl entscheiden (vgl. [Wimmer (1996), S. 43]).

2.4 Die Nachrichtenfaktoren

Schulz überarbeitete und ergänzte 1976 und 1977 den von Galtung und Ruge entwickelten Katalog der Nachrichtenfaktoren und unterschied letztlich zwischen 20 Faktoren, die in sechs Faktorendimensionen zusammengefasst wurden[6] (vgl. [Staab (1990), S. 86ff.]) – siehe nachfolgende Aufzählung (die in Klammern angegebenen Zahlen `[n]` sind zur Zuordnung der Schlüsselwörter zu den einzelnen Faktoren in Kapitel A, Seite 67, angegeben).

1. Status

- `[1]` *Beteiligung von Elite-Nationen:* politische und wirtschaftliche Macht der an einem Ereignis beteiligten Nationen.
- `[2]` *Institutioneller Einfluss:* politische und wirtschaftliche Macht der an einem Ereignis beteiligten Institutionen.
- `[3]` *Beteiligung von Elite-Personen:* Führungs- und Herrschaftsfunktion der Personen, die an einem Ereignis beteiligt sind.

2. Relevanz

- `[4]` *Nähe:* Geografische, politische und kulturelle Nähe des Ereignisortes zum Redaktionssitz bzw. zum Ort der Verbreitung des Mediums.
- `[5]` *Ethnozentrismus:*[7] Bezug eines Ereignisses zur Bevölkerung des Landes (des Gebietes), in dem das jeweilige Medium erscheint.
- `[6]` *Tragweite:* Bedeutsamkeit des Ereignisses in Hinsicht auf die von ihm direkt Betroffenen.
- `[7]` *Betroffenheit:* Konsequenzen des Ereignisses für die Rezipienten des Mediums.

3. Dynamik

- `[8]` *Frequenz:* Kurzfristige Ereignisse haben in einem mit hoher Frequenz erscheinenden Medium höhere Beachtungs- und Publikationschancen.[8]

[6] Schulz sah in der Aufmachung und der Platzierung von Nachrichten Indikatoren für deren Nachrichtenwert.

[7] Besondere Form des Nationalismus, bei der das eigene Volk (die eigene Nation) als Mittelpunkt und zugleich als anderen Völkern überlegen angesehen wird.

[8] Man beachte den Unterschied zwischen *Frequenz des Ereignisses* und *Frequenz des Erscheinens des Mediums.*

- `[9]` *Vorhersehbarkeit:* Je unvorhersehbarer ein Ereignis ist, um so eher berichten die Massenmedien darüber.
- `[10]` *Ungewissheit:* Unsicherheit und Unklarheit der Konsequenzen, des Verlaufs oder des Endes des Ereignisses.
- `[11]` *Überraschung:* Erwartungswidrigkeit des Verlaufs und des Ergebnisses des Ereignisses.

4. Konsonanz[9]

- `[12]` *Kontinuität:* Beachtungsdauer eines Ereignisses in der Medienberichterstattung.
- `[13]` *Thematisierung:* Bezug des Ereignisses zu langfristigem, als kohärent[10] definiertem Geschehen.
- `[14]` *Stereotypie:* Entsprechung der Verlaufsform eines Ereignisses zu etablierten Geschehensmustern.

5. Valenz

- `[15]` *Aggression:* Gewaltsamkeit eines Ereignisses.
- `[16]` *Kontroverse:* Beschreibt die Intensität politischer Meinungsverschiedenheiten und Auseinandersetzungen.
- `[17]` *Erfolg:* Ausmaß positiver wie negativer Veränderungen, die ein Ereignis auf politischem, wirtschaftlichem, kulturellem oder wissenschaftlichem Gebiet bewirkt.
- `[18]` *Werte:* Gefährdung oder Verletzung von Grundwerten.

6. Human Interest

- `[19]` *Personalisierung:* Grad der Beteiligung von Personen an einem Ereignis.
- `[20]` *Emotionalisierung:* Intensität der emotionalen Erfahrungen und Äußerungen der Personen, die in ein Ereignis involviert sind.

Im Rahmen der weiteren Analyse der Nachrichtenfaktoren anhand konkreter Zeitungsartikel zeigte sich, dass bei innenpolitischen Ereignissen fast alle Faktoren als Selektionskriterien fungieren, wenngleich einige dominierend sind (Beteiligung von Elite-Personen, Tragweite, Vorhersehbarkeit, Kontinuität, Kontroverse und Emotionalisierung), während andere eine eher geringe Bedeutung besaßen (Nähe, Aggression, Erfolg und Werte). Bei außenpolitischen Ereignissen spielte zusätzlich der Ethnozentrismus eine Rolle.

[9] lat.: Übereinstimmung

[10] lat.: zusammenhängend

Unterschiede in der Gewichtung der Nachrichtenwerte gab es auch bei verschiedenen Mediengattungen. So etwa hebt die deutsche „*Bild-Zeitung*" Meldungen mit den Werten Emotionalisierung, Aggression und Personalisierung besonders hervor. Eine ähnlich herausgehobene Bedeutung haben emotionalisierende Ereignisse in den Nachrichtensendungen des Fernsehens (vgl. [Staab (1990), S. 88f.]).

2.5 Sprachliche Begriffe zur Repräsentation von Nachrichtenfaktoren

In diesem Kapitel sollen für die im vorangegangenen Kapitel genannten Faktoren Begriffe und Umschreibungen gefunden werden, die die Arbeitsgrundlage für die nachfolgend vorgestellten Algorithmen darstellen.[11]

[11] Eine Übersichtstabelle der hier definierten Wörter findet sich in Anhang A auf Seite 67.

2.5.1 Nachrichtendimension *Status*

Die Nachrichtendimension *Status* steht für die Macht und die Einflussmöglichkeiten der handelnden Akteure (Personen, Gruppierungen und Staaten). Dabei wird nicht zwischen politischem, wirtschaftlichem oder gesellschaftlichem Einfluss unterschieden. Tabelle 2.1 zeigt die Verbegrifflichung der einzelnen Faktoren.

Beteiligung von Elite-Nationen	• Weltpolitisch wichtige Nationen, etwa USA, Russland, Japan, Deutschland • Für Österreich wichtige Nationen, etwa die EU-Staaten und die Nachbarländer; vor allem Deutschland und Italien, aber auch die Schweiz.
Institutioneller Einfluss	• Weltpolitisch und weltwirtschaftlich wichtige Organisationen, etwa die UNO, die OPEC oder die NATO. • Für Österreich wichtige außerstaatliche Organisationen, wie beispielsweise die EU. • Für Österreich wichtige innerstaatliche Organisationen, wie beispielsweise die Krankenkassen, die Interessensvertretungen (Kammern und Gewerkschaften) oder Umwelt- und Konsumentenschutzorganisationen.
Beteiligung von Elite-Personen	• Weltpolitisch wichtige Personen, wie etwa der Präsident der USA oder Russlands, aber auch zB EU-KommissarInnen. • Innerstaatlich wichtige Personen – PolitikerInnen, Vorsitzende großer und einflussreicher Organisationen (siehe oben). • Prominenz: SportlerInnen, SchauspielerInnen, Popstars, FernsehmoderatorInnen . . . , eben im Interesse der Öffentlichkeit stehende Personen.

Tabelle 2.1: Begriffe der Nachrichtendimension *Status*

2.5.2 Nachrichtendimension *Relevanz*

Die Nachrichtendimension *Relevanz* gibt Auskunft über die Wichtigkeit für den Rezipienten der Nachricht und über die Größe der Auswirkungen eines Geschehens auf die direkt Betroffenen. Tabelle 2.2 zeigt die Details.

Nähe	• Geografische Nähe: Bundeshauptstadt, Landeshauptstädte, Bundesländer; Nachbar- und EU-Staaten und deren Hauptstädte. • Politische Nähe: Staaten, deren politisches Gefüge dem österreichischen ähnlich ist; dazu zählen wohl alle demokratischen Republiken. • Kulturelle Nähe: Staaten, die kulturell Österreich sehr ähnlich sind. Dies sind natürlich die Staaten Mittel- und Westeuropas, wohl aber auch die USA (kulturelle Nähe definiert sich beispielsweise auch über die Anzahl der übersetzten Bücher oder Filme).
Ethnozentrismus	• In einem anderen Land stattfindendes Ereignis, das im Verbreitungsgebiet des Mediums nicht, genauso oder besser passieren könnte und so das Verbreitungsgebiet besonders hervorhebt.
Tragweite, Betroffenheit	• Anzahl der direkt durch das beschriebene Ereignis Betroffene, etwa Anzahl der Opfer oder Größe des für die Opfer entstandenen Schadens. • Aber auch Anzahl der von einem Ereignis profitierenden Menschen, etwa bei Steuersenkungen oder erstklassigen Weinjahrgängen.

Tabelle 2.2: Begriffe der Nachrichtendimension *Relevanz*

2.5.3 Nachrichtendimension *Dynamik*

Die Nachrichtendimension *Dynamik* beschreibt einerseits die Dauer von Ereignissen, anderseits auch deren „Überraschungsmoment“, das sich in Vorhersagbarkeit und Ungewissheit manifestiert. In Tabelle 2.3 werden die Begriffe genau definiert.

Frequenz	• Kurzfristigkeit von Ereignissen: Ereignisse von geringer Dauer, die sich beispielsweise mit den Begriffen „schnell“, „kurz“ oder „rasch“ umschreiben lassen.
Vorherseh-barkeit	• Plötzliches, unerwartetes Auftreten eines Ereignisses. Dies kann sowohl Naturgewalten betreffen als auch politische oder wirtschaftliche Ereignisse. Passende Begriffe werden daher Vorfälle aus oben genannten Kategorien sein („Erdbeben“, „Revolution“, „Kurssturz“, ...), die gleichzeitig mit entsprechenden Adjektiven („völlig überraschend“, „unerwartet“, ...) auftreten.
Ungewissheit	• Während sich der Faktor „Vorhersehbarkeit“ auf das Eintreten eines Ereignisses bezieht, bezieht sich die Ungewissheit auf den Verlauf und/oder den Ausgang eines bereits eingetretenen Ereignisses. Die diesen Faktor attributierenden Wörter sind daher annähernd gleich zu den oben bereits erwähnten.
Überraschung	• Unerwarteter Verlauf oder unerwartetes Ergebnis von Ereignissen, charakterisiert durch Wörter wie „überraschend“, „plötzlich“, „unerwartet“ und viele andere. Der Unterschied zum Nachrichtenfaktor „Ungewissheit“ ergibt sich aus dem bereits eingetretenen unerwarteten Verlauf.

Tabelle 2.3: Begriffe der Nachrichtendimension *Dynamik*

2.5.4 Nachrichtendimension *Konsonanz*

Die Nachrichtendimension *Konsonanz* beschreibt Vergleiche neuer Ereignisse zu bereits bestehenden bzw. bekannten Ereignissen. Tabelle 2.4 erläutert die Details.

Kontinuität	• Manche Themen sind sogenannte „Dauerbrenner", deren öffentliche Diskussion sich über Wochen oder gar Monate hinzieht. Entsprechend charakterisieren Wörter bzw. Wortgruppen wie „immer", „immer noch" oder „nach wie vor" die Kontinuität eines Ereignisses.
Thematisierung	• Dieser Faktor beschreibt Ereignisse, die in Bezug zu langfristigem, bereits bekanntem Geschehen stehen. Als passende Wörter wären daher beispielsweise „zugehörig", „neu(e Entwicklung)" oder ähnliche zu sehen.
Stereotypie	• Ein Ereignis verläuft gleich, ähnlich oder vollkommen unterschiedlich zu einem vergleichbaren Ereignis. Dementsprechend bieten sich Wörter wie „vergleichbar" oder „bekannt" an.

Tabelle 2.4: Begriffe der Nachrichtendimension *Konsonanz*

2.5.5 Nachrichtendimension *Valenz*

Die Nachrichtendimension *Valenz* beschreibt den „Grad" des Ereignisses. Je „stärker" ein Ereignis ist, desto eher wird darüber berichtet. In Tabelle 2.5 werden die einzelnen Begriffe definiert.

Aggression	• Das Ereignis selbst ist Indikator für den Grad der Aggression – besonders aggressive Ereignisse sind Krieg, Katastrophe, Erdbeben, Überschwemmung, ... • Zusätzlich umschreiben Adjektive wie „stark", „schwer", „riesig" oder „sehr" den Grad des Ereignisses.
Kontroverse	• Beschreibung des Ereignisses selbst: Diskussion, Meinungsverschiedenheit, Auseinandersetzung, Streit, Wortgefecht, ... • Ein (politischer) Streit innerhalb der selben Fraktion ist anzunehmenderweise interessanter als eine Meinungsverschiedenheit zwischen Regierung und Opposition. • Die oben bereits genannten Adjektive wie „stark", „schwer", „riesig" oder auch „sehr" kommen auch hier wieder zum Tragen.
Erfolg	• Auch hier ist das Ereignis selbst ein Maßstab für seinen Erfolg: Gewinn, Sieg, Erkenntnis, ... • Die oben bereits erwähnten Adjektive, die den Grad des Ereignisses messen: „stark", „sehr", „viel", „erfolgreich", „neu", ...
Werte	• Wiederum beschreiben Begriffe selbst den Vorfall: „Umgehung" von Gesetzen, „Hinterziehung" von Steuern, aber auch „Vergewaltigung", „Verbrechen" oder „Mord". • Das Ereignis zusätzlich beschreibende Adjektive wie „demokratiepolitisch bedenklich", „unüblich", „verwerflich", „unmoralisch", „illegal", ... • Letztlich sind auch wieder die oben bereits erwähnten Adjektive ein Gradmesser für die Schwere des Vorfalls.

Tabelle 2.5: Begriffe der Nachrichtendimension *Valenz*

2.5.6 Nachrichtendimension *Human Interest*

Die Nachrichtendimension *Human Interest* beschreibt, wie stark Personen in ein Ereignis involviert sind oder mit welchen Emotionen Ereignisse behaftet sind; Tabelle 2.6 nennt einige Beispiele.

Personalisierung	• Anzahl der durch das Ereignis direkt betroffenen Menschen; charakterisiert durch das Ereignis selbst: Wahlen, Erdbeben, Steuererhöhungen, Hungersnöte, ... • Zusätzlich umschreiben Adjektive wie beispielsweise „weitreichend“, „groß“, „immens“ oder „schrecklich“ das Ereignis. • Ereignisse können viele Rezipienten eines Mediums betreffen – entsprechende Begriffe wie zB „Demokratie“, „Verfassung“ oder „öffentlich“ sind daher diesem Nachrichtenfaktor zuzuordnen.
Emotionalisierung	• Auch die Emotionalisierung wird durch Begriffe selbst umschrieben: Not, Trauer, Streit, Armut, ... • Einige Begriffe unterliegen einer starken Emotionalisierung; es handelt sich dabei beispielsweise um „Drogen“ oder „Asyl“.

Tabelle 2.6: Begriffe der Nachrichtendimension *Human Interest*

3 Textrepräsentation

Die Suche nach Texten, die bestimmten Kriterien entsprechen, das sogenannte „*Text Mining*“, ist ein Sonderfall des „*Data Minings*“: „*Data mining is the analysis of (often large) observed data sets [...] to summarize the data in novel ways which are both understandable and useful to the database owner*“ ([Lagus (2000), S. 10]).

Um in Texten bestimmte Artikel (hier: Nachrichten) auswählen (suchen) zu können, ist es notwendig, über diese Texte in einer leicht durchsuchbaren und vor allem für den Anwendungszweck adäquaten und repräsentativen Darstellung zu verfügen. Eine einfache Volltextsuche nach bestimmten Suchbegriffen kann kaum jene Artikel zu Tage fördern, die publizierenswert im Sinne der Nachrichtenwert-Theorie erscheinen.

Als sehr fruchtbar erscheint daher die Repräsentation eines Textes mittels des sogenannten „*Vector Space Models*“, das in Kapitel 3.6, Seite 20, diskutiert wird. Dieses bildet die Grundlage für das neuronale Netz, das die Nachrichtenauswahl bewerkstelligen soll und in Kapitel 6, Seite 49, beschrieben wird.

Basis für das Vector Space Model sind sogenannte *Keywords*. Um gute Keywords auswählen zu können, muss zuerst auf die Eigenschaften natürlicher Sprache eingegangen werden; anschließend werden die Eigenschaften von Keywords diskutiert.

3.1 Eigenschaften natürlicher Sprache

Natürliche Sprache hat verschiedene Eigenschaften. Einige davon betreffen das Verstehen und Verstehen-Können von Sprache und sind daher für den Nutzer von Sprache relevant; andere Eigenschaften wiederum sind eher als „Phänomene“ von Sprache zu bezeichnen, die nicht notwendigerweise Einfluss auf deren Verständlichkeit haben.

Um natürliche Sprache wirklich *verstehen* zu können, sind mindestens sechs Voraussetzungen zu erfüllen (nach [Lagus (2000), S. 16f.]; in den Fußnoten jeweils die englische Originalbezeichnung):

1. **Gestalt und Form:**[1] Wissen über die Struktur von Wörtern und deren (gebeugten) Formen.
2. **Satzbau:**[2] Strukturelles Wissen über Wörter und deren Kombinationen, um *syntaktisch* richtige Sätze zu bilden.
3. **Bedeutung:**[3] Wissen um die Bedeutung von Wörtern unabhängig vom jeweiligen Kontext (*Semantik*, siehe Kapitel 3.2, Seite 18).

[1] Morphological Knowlegde
[2] Syntactic Knowledge
[3] Semantic Knowledge

4. **Deutung:**[4] Wissen über den Einfluss des Kontexts auf die Bedeutung der Worte und die Reaktion des Rezipienten (*Pragmatik*).
5. **Satzreihenfolge:**[5] Wissen über die Auswirkungen des aktuellen Satzes auf die Bedeutung des/der folgenden Satzes/Sätze.
6. **„Weltwissen“:**[6] Allgemeines Wissen über das Gebiet („Domäne“) des Textes.

Auch für das Erkennen der Relevanz einer Nachricht gemäß der Nachrichtenwert-Theorie ist vollständiges Verstehen des Inhalts notwendig. Dieser hehre Anspruch kann nur in Ansätzen realisiert werden – doch soll beim Finden der Schlüsselwörter, die die einzelnen Nachrichtenfaktoren charakterisieren, zumindest versucht werden, auf diese Voraussetzungen Rücksicht zu nehmen.

Ebenfalls wichtig für das Auffinden von Keywords scheint das Wissen um einige „Phänomene“ natürlicher Sprache zu sein, die im Folgenden beschrieben werden (nach [Lagus (2000), S. 18ff.]).

- In einer natürlichen Sprache hat ein Wort („Symbol“) eine oder mehrere Bedeutungen – man muss somit zwischen dem Symbol und der Bedeutung trennen.
- Sprache ist aufgrund der Symbolhaftigkeit der einzelnen Wörter diskret (im Sinne von „in einzelne Punkte zerfallend“); kontinuierliche Verläufe sind daher mit Sprache nicht oder nur schwer darstellbar: Man denke etwa an ein Farbspektrum mit letztlich unendlich vielen Farben und der geringen Menge an Wörtern, die uns zur Verfügung steht, all diese Farben zu benennen.
- Unterschiedliche Ausdrücke und Formulierungen lassen im Rezipienten den gleichen Gedanken entstehen. Synonyme[7], Abkürzungen, Akronyme[8], unterschiedliche Bezeichnungen oder auch nur einfache Rechtschreibfehler sind Beispiele für diese Variationen.
- Ein Wort kann unterschiedliche Bedeutungen haben (*Homonym*); diese Mehrdeutigkeit ist bei menschlicher Kommunikation nicht hinderlich, da der Kontext den Kommunikationspartnern bekannt ist. Ein System ohne Wissen kann diese Mehrdeutigkeiten jedoch nicht auflösen.
- Wörter stehen in einem Text in einer bestimmten Reihenfolge; diese Reihenfolge – definiert durch die Syntax – legt auch den Inhalt fest: *„Mann biss Hund“* bedeutet doch etwas wesentlich anderes als *„Hund biss Mann“*. Ähnliches gilt auch für die bereits erwähnte Reihenfolge von Sätzen.

[4] Pragmatic Knowledge
[5] Discourse Knowledge
[6] World Knowledge
[7] **Synonym:** Wort, das inhaltlich mit verschiedenen Wörtern in derselben Sprache übereinstimmt.
[8] **Akronym:** Wort, das aus zusammengerückten Anfangsbuchstaben gebildet ist; auch „Initialwort“ (zB „UNO“).

- Wörter und Texte stehen in Verbindungen zu einander: bei Wörtern sind diese Beziehungen die Synonymie[7] und die Antonymie[9], bei Texten handelt es sich um das Part-Whole-Prinzip[10], und das Subset-Superset-Prinzip[11].
- Sprache verändert sich: Wörter erleben einen Bedeutungswandel, andere geraten in Vergessenheit und gleichzeitig entstehen neue Wörter, in manchen spezialisierten Bereichen entstehen sogar Fachbegriffe oder ganze Fachsprachen (zu denen auch jugendlicher Slang oder Chat-ähnliche Kommunkationsformen zu zählen sind).

Die Vielfalt und Komplexität dieser Phänomene stellt große Anforderungen an das Abbilden der Daten (Texte) in künstlichen Modellen. Selbst wenn man nur auf der Ebene der einzelnen sinngebenden Wörter bleibt, wird man niemals alle repräsentativen Wörter finden können. Letzten Endes scheitert man also am bekannten Problem der Artificial Intelligence: Wie kann Expertenwissen in maschinentauglicher Form dargestellt werden?

3.2 Von der Wortbedeutung – die Semantik

Im Folgenden soll ein sehr kurzer Abriss sprachphilosophischer Grundlagen der Semantik gegeben werden (vgl. [Ruge (1995), S. 19ff.]). Das Erkenntnisinteresse entsteht aus dem Wunsch, für die einzelnen Nachrichtenwerte möglichst aussagekräftige Wörter zu finden – dies kann nur dann erfolgreich geschehen, wenn man sich der unterschiedlichen Dimensionen von Semantik bewusst ist.

- **Modelltheoretische Semantik:** Mit Hilfe der modelltheoratischen Semantik wird versucht, natürlichsprachliche Sätze in logische Formeln überzuführen. Wörter denotieren hier also bestimmte mathematische Konstrukte. Es stellt sich hier jedoch die Frage, ob natürliche Sprache in das enge Korsett einer logischen Sprache gezwungen werden kann.
- **Strukturelle Semantik oder Merkmalssemantik:** Will man einen Begriff erklären, so könnte man alle Objekte aufzählen, die unter diesen Begriff fallen. Gebräuchlicher ist jedoch die Methode, eine abstrakte Beschreibung zu geben, wie es zB in Lexika geschieht. Eine solche Beschreibung kann auch als Liste semantischer Merkmale interpretiert werden.[12]
- **Der WITTGENSTEIN'sche Bedeutungsbegriff:** Diese Theorie betrachtet als ausschlaggebendes Kriterium für die Wortbedeutung den Gebrauch der Wörter. Die Repräsentation wird nicht direkt herangezogen.

[9] **Antonym:** Wort, das einem anderen Wort in Bezug auf die Bedeutung entgegengesetzt ist.
[10] **Part-Whole-Prinzip:** Ein Text ist Teil eines (größeren) Textes.
[11] **Subset-Superset-Prinzip:** Ein Text beleuchtet einen Teilbereich eines größeren „Überbereichs“.
[12] Beispiel „Junggeselle“: menschlich, männlich, nicht verheiratet, erwachsen.

Keine diese Theorien gibt explizit an, wie die Bedeutung von Wörtern zu erfassen sei. Auch liegen beträchtliche Unterschiede in den einzelnen Theorien – doch beschreiben diese unterschiedliche Aspekte von Wortbedeutung: *Zusammenfassung von Objekten*, *semantische Charakteristika* und *Kontextabhängigkeit*. Unterschiedliche Phänomene sind daher mit der einen oder der anderen Theorie beschreibbar.

3.3 Feature Selection

Einzelne, repräsentative Wörter bzw. Wortgruppen werden als *Term* oder *Feature* bezeichnet. Die Auswahl dieser Wörter – die *Feature Selection* – erfolgt entweder manuell durch einen Experten oder durch Maximierung objektiver Kriterien, die die interessierenden Eigenschaften widerspiegeln.

Wichtig ist hierbei, dass relevante Information nicht verworfen wird: Diese kann später nicht mehr wiedergewonnen werden! Andererseits darf auch nicht ein Zuviel an Information existieren: *Rauschen* erhöht nicht nur den Suchaufwand, sondern es können dadurch sogar Daten bei der Suche nicht gefunden werden.

3.4 Wortstämme

Wörter kommen bei weitem nicht nur in der Stammform vor, sondern werden oftmals auch *gebeugt*. Je nach Geschlecht des nachfolgenden Wortes hat beispielsweise das Wort „viel" unterschiedliche Beugungen: viel**e**, viel**er**, viel**en**, viel**es** und viel**em**.

Das Erkennen dieser Beugungen und die Reduzierung auf die Wortstämme – im Englischen auch als *Stemming*[13] bezeichnet – ist ebenfalls Aufgabe der Feature Selection.

3.5 Eigenschaften eines Keywords

Die Eigenschaften eines qualitativ guten Keywords können intuitiv ganz einfach beschrieben werden ([Lagus, Kaski (1999)]): *„A good descriptor of a cluster charcterizes some outstanding property of the cluster in relation to the rest of the collection."* Dies kann leicht auch bei der Selektion von Nachrichten gemäß der Nachrichtenwert-Theorie angewandt werden: Bestimmte Wörter charakterisieren die in Kapitel 2.4, Seite 7, erörterten Nachrichtenfaktoren besser als andere. Anders gesagt sollte ein Keyword daher die beiden folgenden Eigenschaften haben:

1. Das Wort sollte hervorstechend im Vergleich zu anderen Wörtern im gerade betrachteten Nachrichtendokument sein.
2. Das Wort sollte hervorstechend im Vergleich zu anderen Wörtern in der gesamten Dokumentensammlung sein.

[13] Engl.: *stem (of word):* Stamm

Mathematisch betrachtet ergibt sich daraus der in Gleichung 3.1 definierte Zusammenhang:

$$Q(t) = F^{dok}(t) \cdot F^{ges}(t) \tag{3.1}$$

Dabei ist $Q(t)$ die Güte eines Terms t (bzw. eines Wortes oder eines Features), $F(t)$ ist ein Maß für die „Einzigartigkeit" eines Wortes innerhalb eines Dokuments ($F^{dok}(t)$) bzw. innerhalb der Gesamtheit aller Dokumente ($F^{ges}(t)$). Die Güte eines Features ist – wie oben bereits erwähnt – natürlich dann am größten, wenn das Wort hervorstechend sowohl in einem Dokument wie auch in allen Dokumenten ist. Als Maß dafür bietet sich die Häufigkeit des Wortes an.[14]

3.6 Vector Space Model

Yang und Pederson verglichen in [Yang, Pedersen (1997)] unterschiedliche Methoden der Feature Selection. Eine Feature Selection basierend auf der *Document Frequency* brachte ähnlich gute Ergebnisse wie andere Methoden, die wesentlich komplizierter und weniger performant waren (*Information Gain*, *Mutual Information*, *Term Strength* und χ^2-*Test*). Aus diesem Grunde soll hier mit einem ähnlichen Verfahren gearbeitet werden, das im folgenden beschrieben wird.

Dokumente können auf verschiedene Arten repräsentiert werden. Am einfachsten ist wohl die *„Bag of Words"*-Methode ([Lagus (2000), S. 20]), die Dokumente als „Wortcontainer" sieht und die Anzahl der Vorkommnisse der unterschiedlichen Wörter (ohne Rücksicht auf Struktur und Interpunktion) ermittelt. Im folgenden werden anhand des *„Vector Space Model"* die Verbesserungsmöglichkeiten dieses Ansatzes erörtert.

In einem ersten Schritt kann die Anzahl der Wörter reduziert werden. Für diese Diplomarbeit sind nur jene bestimmten Begriffe relevant, die in Kapitel 2.5, Seite 9, festgelegt werden.[15] Diese Begriffe (oder Wörter oder Wortgruppen) werden auch als *„Term"* oder *„Feature"* bezeichnet; der *„Feature Space"* wiederum ist die Menge aller Features (bzw. Terme t_i) als T-dimensionaler Vektor $\mathbf{s}$ betrachtet:

$$\mathbf{s} = (t_1, t_2, \ldots, t_T) \tag{3.2}$$

Ein Dokument $\mathbf{D}_i$ kann als Vektor dargestellt werden (vgl. [Salton (1989), S. 314f.]):

$$\mathbf{D}_i = \mathbf{s}_i = (t_{i1}, t_{i2}, \ldots, t_{iT}) \tag{3.3}$$

Die Gesamtheit M aller Dokumente $\mathbf{D}$ ist daher nach Gleichung 3.4 definiert.

[14] Tatsächlich werden einige Tests mit auf Basis von Häufigkeiten ermittelten Schlüsselwörtern durchgeführt; siehe Kapitel 5.1, Seite 39.
[15] Siehe auch Anhang A auf Seite 67.

$$\mathbf{D} = \begin{pmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \\ \vdots \\ \mathbf{s}_M \end{pmatrix} = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1T} \\ t_{21} & t_{22} & \cdots & t_{2T} \\ \vdots & \vdots & \ddots & \vdots \\ t_{M1} & t_{M2} & \cdots & t_{MT} \end{bmatrix} \tag{3.4}$$

Ein Dokument kann erfrischend einfach „binär“ repräsentiert werden: $t_{ij} = 1$, wenn der Term t_j im Dokument $\mathbf{D}_i$ vorkommt, und 0, wenn der Term nicht vorkommt.

$$t_{ij} = \begin{cases} 0\text{:} & t_j \text{ nicht in } \mathbf{D}_i \\ 1\text{:} & t_j \text{ in } \mathbf{D}_i \end{cases} \tag{3.5}$$

Natürlich kann ein Feature in einem Dokument mehrmals vorkommen: Die Anzahl dieser Vorkommen wird als *„Term Frequency“ tf* bezeichnet. Terme, die in einem Dokument mehrmals vorkommen, erhalten somit auch mehr Gewicht.

$$t_{ij} = tf_{ij} \tag{3.6}$$

Kommt ein Term hingegen in jedem Dokument (annähernd) gleich oft vor, so ist daraus keinerlei Signifikanz für die angestrebte Artikelauswahl abzuleiten. Die *„Document Frequency“ df* gibt an, in wievielen Dokumenten ein Term vorkommt. Gewichtet man die Term Frequency mit der *inversen* Document Frequency *idf*, so lassen sich Aussagen über die Signifikanz eines Terms treffen: Terme, die nur in wenigen Dokumenten vorkommen, sind für die Auswahl wichtiger als Terme, die in viele Dokumente Eingang fanden. Daraus ergibt sich Gleichung 3.7, die oft vereinfachend als $tf \times idf$ dargestellt wird.

$$t_{ij} = tf_{ij} \cdot \left(\frac{1}{df_j}\right) \tag{3.7}$$

Die am weitesten verbreitete Möglichkeit zur Berechnung der Signifikanz eines Terms zeigt Gleichung 3.8. Dabei wird die inverse Document Frequency logarithmiert und zusätzlich die Anzahl M der Dokumente mit einbezogen.[16]

$$t_{ij} = tf_{ij} \cdot \ln\left(\frac{M}{df_j}\right) \tag{3.8}$$

Abschließend kann es noch notwendig sein, die errechneten Vektoren zu normieren.

$$\bar{\mathbf{s}} = \mathbf{s}/t_k \qquad t_k \geq t_l \quad \forall k \neq l \tag{3.9}$$

Eine Implementierung des Vector Space Models gemäß Gleichung 3.8 in der Programmiersprache *perl* wird in Anhang B.1, Seite 81, gezeigt.

Weitere Methoden der Dokumentenrepräsentation, wie zB *Latent Semantic Indexing*, *Random Projection*, *Probalistic Modeling*, *Full Text Indexing* und andere mehr, werden in [Lagus (2000), S. 20ff.] und [Rauber (2000), S. 68ff.] diskutiert.

[16] Weitere Möglichkeiten der Berechnung der Signifikanz eines Terms finden sich beispielsweise in [Rauber (2000), S. 81].

3.7 Exkurs: Ähnlichkeit von Vektoren

Für die Entscheidung, ob ein Artikel veröffentlicht werden soll oder nicht, kann die Kenntnis, ob ähnliche Artikel bereits veröffentlicht wurden, hilfreich sein.[17] *Distanzmaße* oder *Ähnlichkeitsmaße* beschreiben den Grad der Übereinstimmung von Vektoren und damit die Ähnlichkeit von Artikeln, die mit Hilfe des *Vector Space Model* (siehe vorangehendens Kapitel) durch Vektoren repräsentiert werden.

Zwei Distanzmaße sollen hier kurz dargestellt werden.

3.7.1 Kosinusdistanz

Es wird ein n-dimensionaler Vektorraum über den reellen Zahlen $\mathbb{R}$ vorausgesetzt.

Betrag eines n-dimensionalen Vektors a:

$$|a| = \sqrt{a_1^2 + a_2^2 + \cdots + a_n^2} \tag{3.10}$$

Skalarprodukt zweier n-dimensionaler Vektoren a, b:

$$a \cdot b = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n \tag{3.11}$$

Kosinusmaß zweier n-dimensionaler Vektoren a, b:

$$d_{cos}(a, b) = \cos\varphi = \frac{a \cdot b}{|a| \cdot |b|} \tag{3.12}$$

Die Kosinusdistanz $d_{cos}(a, b)$ zwischen zwei Vektoren a, b ist also der Winkel zwischen den beiden Vektoren a, b. Die größte Ähnlichkeit von Vektoren ist naturgemäß bei identischen Vektoren gegeben: $d_{cos}(a, b) = 1$; zeigen die beiden Vektoren exakt in die jeweilige Gegenrichtung, so ist die Ähnlichkeit am geringsten: $d_{cos}(a, b) = -1$; bei orthogonalen Vektoren (also zueinander normalen Vektoren) gilt $d_{cos}(a, b) = 0$.

3.7.2 Euklidische Distanz

Euklidische Distanz zweier n-dimensionaler Vektoren a, b:

$$d_{euklid}(a, b) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \tag{3.13}$$

Je kleiner die Distanz, desto ähnlicher sind sich die beiden Vektoren. Die Euklidische Distanz ist vereinfacht dargestellt der Abstand zwischen zwei Punkten im Raum; reduziert man Gleichung 3.13 auf zwei Dimensionen und sei a und b die jeweilige

[17] Vgl. Kapitel 6.8, Seite 55

Differenz der Koordinaten zweier Punkte in einer Ebene, so wird daraus der allgemein bekannte *Pythagoräische Lehrsatz*:

$$d_{euklid}(a,b) = \sqrt{a^2 + b^2} \tag{3.14}$$

4 Neuronale Netze

Das menschliche Gehirn vollbringt ganz ohne Programmierungsprobleme, Synchronisations-Deadlocks oder OSI-Protokolle sehr erstaunliche Leistungen – und das, obwohl es mit einer wesentlich kleineren „Taktfrequenz“ arbeitet als unsere modernen, auf dem Konzept von John von Neumann beruhenden Mikroprozessoren. Zwar versuchen moderne Architekturen den „von Neumann-Flaschenhals“[1] zu umgehen, indem mehrere Prozessoren parallel arbeiten, doch sind auch mit diesen Systemen „natürliche“ Aufgabenstellungen wie etwa das kollisionsfreie Umschwirren einer Lichtquelle, das von Insekten problemlos beherrscht wird, nicht genügend genau und genügend schnell lösbar.

Es ist also naheliegend, sich die Natur zum Vorbild zu nehmen und die Funktionen des (menschlichen) Gehirns nachzubilden.[2, 3]

Neuronale Netze haben vielerlei Vorteile, die im folgenden kurz beschrieben werden (vgl. [Kratzer (1991), S. 17f.], [Karagiannis, Telesko (2001), S. 288f.] und [Kohonen (2001), S. 81f.]):

- **Adaptivität („Lernen“):** Das Wissen eines neuronalen Netzes wird nicht explizit programmiert, sondern dem Netz durch die Präsentation von Eingabemustern sowie der mehr oder weniger präzisen Angabe der gewünschten Reaktion „beigebracht“. Man unterscheidet hierbei zwischen *überwachtem* und *unüberwachtem* Lernen.

- **Robustheit, Fehlertoleranz:** Das Wissen eines neuronalen Netzwerks wird in den Gewichten der einzelnen Knoten (siehe Kapitel 4.3.1, Seite 28) gespeichert.[4] Dank dieser „Kollektivverantwortung“ ([Kratzer (1991), S. 17]) kann ein neuronales Netz auch bei Ausfall einzelner Knoten mit *lediglich gering verminderter* Leistung arbeiten.

- **Generalisierungsfähigkeit:** Die von einem neuronalen Netz gelernten assoziativen Beziehungen zwischen Ein- und Ausgabemustern entsprechen nicht einer exakten „wenn-dann“-Beziehung, sondern eher einer statistischen Korrelation. Beim Erkennen eines Musters ist daher keine exakte Übereinstimmung notwendig, das neuronale Netz kann also auch bei ihm unbekannten Daten eine korrekte Lösung finden – man spricht hier von der Fähigkeit der *„Assoziation“* und der *„Generalisierung“*.

[1] Dieser Begriff bezeichnet den Nachteil, dass alle Daten in sequenzieller Abfolge auf dem Datenbus übertragen werden müssen.

[2] Der Gedanke, sich dabei selbst besser zu verstehen, sei eine zusätzliche Motivation zur Beschäftigung mit neuronalen Netzen (vgl. [Brause (1995), S. 14]).

[3] Ausführliche Betrachtungen der biologischen Aspekte finden sich beispielsweise in [Schöneburg, Hansen, Gawelczyk (1990), S. 35ff.], in [Brause (1995), S. 15ff.] oder in [Köhle (1990), S. 35ff.].

[4] Man spricht hier von „verteilter Wissensrepräsentation“.

Weitere Vorteile sind die *Performance* bei Problemstellungen, die keine exakte Lösung benötigen oder eine Lösung nur mit großem Aufwand erreicht werden könnte, und die *modellinhärente Parallelisierungsmöglichkeit*, die die Verwendung paralleler Algorithmen und Hardware erleichtert. (Man spricht bei jeder Form der Informationsverarbeitung in neuronalen Netzwerken daher auch von *parallel distributed processing PDP*.)

Trotz all dieser Vorteile sollten von neuronalen Netzen keine Wunder erwartet werden. Zum einen ist es nach wie vor nicht möglich, menschliches Denken zu simulieren, zum anderen sind neuronale Netze nicht verifizierbar, sondern nur per Test validierbar, wodurch die Anwendungsmöglichkeiten eingeschränkt werden. Weiters ist die Transformation der Ein- und Ausgabedaten in den meisten Fällen mit herkömmlicher Technik zu bewerkstelligen.

Auch sollte nicht vergessen werden, dass das menschliche Gehirn wesentlich komplizierter gebaut ist als die derzeit verwendeten künstlichen neuronalen Netze: Unterschiedlichste Typen von Neuronen und ein Dutzend chemischer Transmitter mit unterschiedlichen Effekten machen den künstlichen Nachbau schwierig bis unmöglich. Teuvo Kohonen spricht es klar aus: „*Quite honestly, one should admit that the brain is a mixture of a vast number of different nonlinear dynamical systems*" ([Kohonen (2001), S. 74]).

4.1 Geschichte[5]

Die ersten Erforschungen der neuronalen Informationsverarbeitung sind kein Verdienst der „klassischen" Informatik bzw. ihrer Vorgänger, sondern stammen aus der Biologie, der Neurophysiologie und der Psychologie[6] Anfang des 20. Jahrhunderts. Bahnbrechend war die Arbeit von Warren McCulloch und Walter Pitts im Jahre 1943, die Neuronen als „Schwellwertschalter" sah (*M-P-Neuronen*).[7]

Diese M-P-Neuronenkomplexe waren in ihrer ursprünglichen Version nicht lernfähig; erst 1949 forumlierte der kanadische Psychologe Donald Hebb eine „Lernregel", die auch in einer Vielzahl heutiger Netze noch (in adaptierter Form) zur Anwendung kommt.[8] 1958 stellte Frank Rosenblatt ein erstes abgeschlossenes Modell eines

[5] Eine ausführliche Darstellung der Entwicklungsgeschichte der neuronalen Netze findet sich beispielsweise in [Schöneburg, Hansen, Gawelczyk (1990), S. 68ff.].

[6] Der US-Psychologe William James nahm schon im Jahre 1890 – noch bevor die Idee des Neurons als informationsverarbeitender Einheit überhaupt geboren war – die Prinzipien der Netzwerkorganisation kortikaler Informationsverarbeitung vorweg. Er beschrieb, dass die Größe der Aktivität eines „Punktes im Kortex" der Summe der Tendenzen aller anderen Punkte entspreche (vgl. [Spitzer (2000), S. 42f.]).

[7] Der Mathematiker Stephen Kleene formulierte ein paar Jahre später für diese Modelle eine Algebra und nannte sie *reguläre Mengen*, die Notation dafür nannte er *reguläre Ausdrücke* und ist heute unter diesem Begriff weit verbreitet (vgl. [Friedl (2003), S. 87]).

[8] Die Hebb'sche Lernregel wird in Kapitel 4.3.3.1, Seite 35, diskutiert.

neuronalen Systems vor: das *Perceptron*, das alle Buchstaben erlernen und wiedererkennen kann. In den darauf folgenden Jahren entwickelten Bernard Widrow und Marcian Hoff ein Netzmodell namens *Adaline* (*Adaptive Linear Neuron*), dessen Lernfähigkeit durch die neu eingeführte „Delta-Regel"[9] verbessert wird. Das wenig später entwickelte Modell *Madaline* (*Multiple Adaline*) ist das erste Netz mit mehreren inneren Schichten („*Hidden Layer*", vgl. Kapitel 4.3.2, Seite 31).

Das 1969 durch Marvin Minsky und Seymour Papert veröffentlichte Werk „*Perceptrons*" beinhaltet eine vernichtende Kritik der damaligen neuronalen Netze: Herkömmliche Systeme waren wesentlich leistungsfähiger, wodurch sich die Forschung und die Finanzierung aus dem Bereich der neuronalen Netze zurückzog. Wie beispielsweise in [Karagiannis, Telesko (2001), S. 221ff.] ausführlich beschrieben, kann man mit dem Perceptron nur Probleme lösen, die *linear trennbar* sind. Die logische Verknüpfung „xor" kann mit einem einfachen Perceptron nicht nachgebildet werden – dazu sind mehrschichtige Netze (*Multi Layer Perceptron*, „MLP") notwendig.

Erst 1982 erhalten die neuronalen Netze wieder Auftrieb: Der Physiker John Hopfield erkennt formale Parallelen zwischen neuronalen Netzen und sogenannten Spingläsern[10]. Teuvo Kohonen liefert 1984 wertvolle Beiträge zur Selbst-Organisation neuronaler Verarbeitungseinheiten und zu topologieerhaltenden Abbildungen ([Kohonen (2001)]). Durch die Generalisierung der Delta-Lernregel („*Backpropagation*"[11]) durch David Rummelhart und Goeffrey Hinton wurde endgültig eine Renaissance der neuronalen Netze eingeläutet, die ungebrochen mit vielen neuen Erweiterungen und Ideen bis in die Gegenwart anhält.

4.2 Klassifikation

Die Unterscheidung von Netzwerktypen ist durch diverse Parameter möglich: Zahl der Schichten, Rückkopplung („feed-back"), Lernmethode und vieles andere mehr. Die in Abbildung 4.1 dargestellte Klassifikation soll einen kurzen Überblick über die in den letzten Jahrzehnten entwickelte große Anzahl neuronaler Netze geben und auch der systematischen Einordnung der in dieser Diplomarbeit verwendeten Netzwerkarten dienen.

Die verschiedenen Netzwerkarten wurden mit zum Teil sehr unterschiedlichen Motivationen und Zielsetzungen entwickelt: Seien es physikalische Grundlagen oder auch biologische oder psychologische Aspekte, die eine Rolle spielten. Eine tiefere Betrachtung würde jedoch den Rahmen dieser Diplomarbeit sprengen; der interessierte

[9] Die Delta-Regel wird in Kapitel 4.3.3.2, Seite 36, diskutiert.

[10] Spingläser sind Metalle, deren Atome im Kristallgitter scheinbar ungeordnet vorliegen. Tatsächlich hängt die jeweilige Anordnung der Atome jedoch davon ab, wie das Metall magnetisch vorbehandelt wurde. Das Metall merkt sich sozusagen seine „magnetische Vergangenheit". Spingläser und das Hopfield-Modell werden beispielsweise in [Kruse, Mangold, Mechler, Penger (1991), S. 163ff.] detailliert dargestellt.

[11] Backpropagation wird in Kapitel 4.3.3.3, Seite 36, diskutiert.

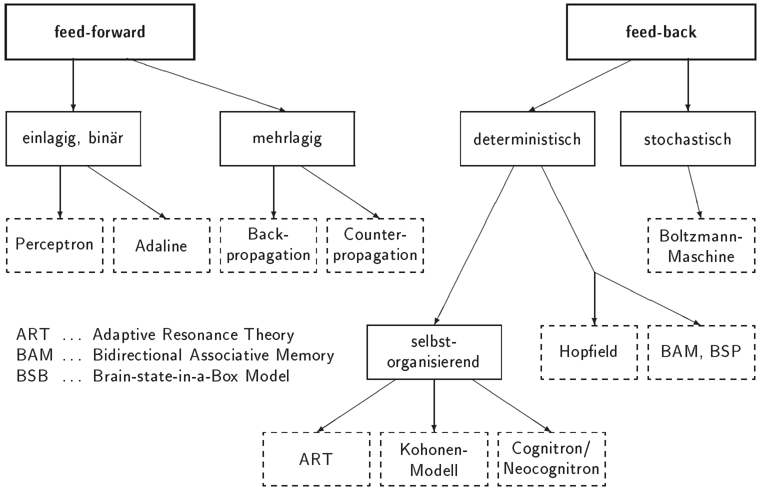


Abbildung 4.1: Klassifikation neuronaler Netze (nach [Schöneburg, Hansen, Gawelczyk (1990), S. 75] und [Karagiannis, Telesko (2001), S. 221]).

Leser sei daher unter anderem auf die Arbeiten von [Karagiannis, Telesko (2001)] oder [Schöneburg, Hansen, Gawelczyk (1990)] verwiesen.

4.3 Charakteristika

Wie in Kapitel 4.1, Seite 25, bereits erwähnt, stammen die ersten Forschungsarbeiten neuronaler Netze aus der Biologie, die damit die Grundlage für künstliche neuronale Netze darstellt: *Soma*[12], *Axon*[13] und *Dendriten*[14] finden sich in künstlichen neuronalen Netzen als Knoten bzw. *Units* (siehe Kapitel 4.3.1, Seite 28) und deren Verbindungen in Form einer *Netztopologie* (siehe Kapitel 4.3.2, Seite 31) wieder. Ausführliche Betrachtungen der biologischen Aspekte finden sich beispielsweise in [Schöneburg, Hansen, Gawelczyk (1990), S. 35ff.] und in [Brause (1995), S. 15ff.].

Grundlegende Elemente eines künstlichen neuronalen Netzes sind:

- Informationsverarbeitung (Knotendynamik); Kapitel 4.3.1
- Netzstruktur, Topologie; Kapitel 4.3.2

[12] Zellkörper
[13] Nervenfaser zur Signalleitung zwischen den Zellen.
[14] Eingänge des Neurons.

- Lernverfahren; Kapitel 4.3.3

Diese Elemente werden in den nachfolgenden Kapiteln beschrieben.

4.3.1 Informationsverarbeitung (Knotendynamik)

Die Verarbeitung von Informationen erfolgt in künstlichen neuronalen Netzen ausschließlich in den sogenannten *Units*[15]. Abbildung 4.2 zeigt den grundlegenden Aufbau einer Unit, der im folgenden beschrieben wird.

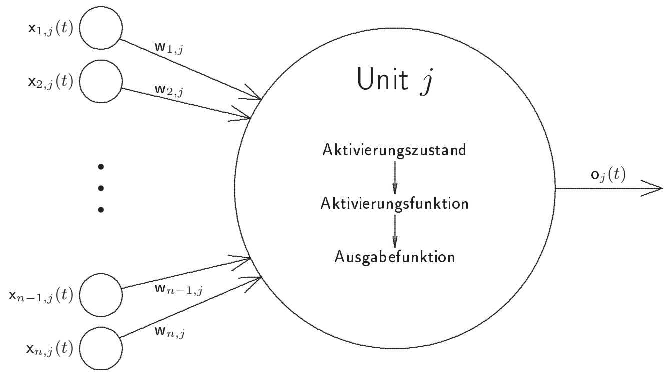


Abbildung 4.2: Schema einer Unit

Das Grundschema einer Unit in allen üblichen neuronalen Netzen stützt sich im Wesentlichen auf das Modell der bereits erwähnten Arbeiten von McCulloch und Pitts aus dem Jahre 1943, die eine Unit als eine Art Summierer mit Schwellwertschalter sahen.

Abbildung 4.2 zeigt links die Inputs $x_{i,j}(t)$ der Unit j, die die Aktivierungen anderer Units darstellen (*Aktivierungszustand*). Diese Inputs werden gewichtet ($w_{i,j}$), anschließend in der Unit addiert und gemäß der *Aktivierungsfunktion* an den Ausgang weiter gegeben, an dem noch eine *Ausgabefunktion* zum Tragen kommt.

[15] Vgl. dazu auch die Kritik Kohonens auf Seite 25, weswegen hier auch immer von „Units" und nicht von „Neuronen" gesprochen wird.

4.3.1.1 Aktivierungszustand

Jede Unit verfügt über eine Menge von n Eingangsaktivitäten x, die gewichtet und summiert werden. Für die Unit j errechnet sich der Nettoinput zum Zeitpunkt t über die sogenannte *Propagierungsfunktion* 4.1.

$$net_j(t) = \sum_{i=1}^{n} w_{i,j} \cdot x_{i,j}(t) \tag{4.1}$$

Dies bedeutet, dass sich der Nettoinput für Unit j aus der Summe der Produkte der Ausgaben der vorgeschalteten Units mit den jeweiligen Verbindungsgewichten ergibt.

4.3.1.2 Aktivierungsfunktion

Der nächste Schritt besteht in der Berechnung der Aktivierung $a_j(t)$ der Unit j mit Hilfe der Aktivierungsfunktion f_{akt}, in die oft neben dem Nettoinput auch der Vorzustand der Unit einbezogen wird. Allgemein kann die Aktivierungsfunktion folgendermaßen definiert werden:

$$a_j(t) = f_{akt}(net_j(t), a_j(t-1)) \tag{4.2}$$

Manchmal ist eine externe Eingabe ex_j ein weiterer Parameter der Aktivierungsfunktion ([Mechler (1995), S. 64]):

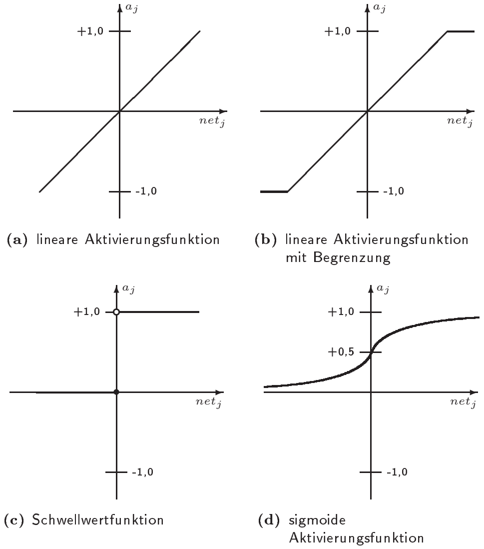
$$a_j(t) = f_{akt}(net_j(t), ex_j, a_j(t-1)) \tag{4.3}$$

Fasst man alle Aktivierungen der Units eines neuronalen Netzes zum Zeitpunkt t zu einem Vektor $\vec{a}(t)$ zusammen, so ist der gesamte Systemzustand zum Zeitpunkt t durch diesen Vektor repräsentiert.

Abbildung 4.3 zeigt einige gebräuchliche Aktivierungsfunktionen.

- **Lineare Aktivierungsfunktion** (Abbildung 4.3(a)): Der Nettoinput wird durch die Identitätsfunktion direkt oder über eine lineare Abbildung in den Aktivierungszustand übergeführt. Dies ist zwar verführerisch einfach, jedoch selten im Einsatz,[16] da sehr kleine bzw. sehr große Aktivierungswerte entstehen können, was sich ungünstig auf das Lernen auswirkt ([Karagiannis, Telesko (2001), S. 216]).
- **Lineare Aktivierungsfunktion mit Begrenzung** (Abbildung 4.3(b)): Um dem Nachteil der gegen $\pm\infty$ gehenden Aktivierungswerte zu begegnen, werden die Werte auf das Intervall $[-1, +1]$ beschränkt.

[16] Die lineare Aktivierungsfunktion wird beispielsweise beim Perceptron-Modell verwendet.



(a) lineare Aktivierungsfunktion **(b)** lineare Aktivierungsfunktion mit Begrenzung

(c) Schwellwertfunktion **(d)** sigmoide Aktivierungsfunktion

Abbildung 4.3: Beispiele von Aktivierungsfunktionen

- **Schwellwertfunktion** (Abbildung 4.3(c)): Diese Aktivierungsfunktion, die auch Treppen- oder Heavyside-Funktion[17] genannt wird, ist sehr populär, da sie eine Möglichkeit darstellt, Units mit diskreten Zuständen zu realisieren: Entsprechend der Klassifikation des Eingangs (ober- oder unterhalb der Konstanten) wird die Aktivierung der Unit bestimmt. Diese Funktionen eignen sich sehr gut für logische Entscheidungen.
- **Sigmoide Aktivierungsfunktion** (Abbildung 4.3(d)): Einige häufig verwendete Netztypen verlangen eine überall differenzierbare Aktivierungsfunktion: Die sigmoide Aktivierungsfunktion hat neben der Stetigkeit auch den Vorteil, durch ihr asymptotisches Verhalten in der Unendlichkeit eine gleichzeitige Be-

[17] Die Heavyside-Funktion $H(x)$ ist folgendermaßen definiert:

$$H(x) = \begin{cases} 0: & x \leq 0 \\ 1: & x > 0 \end{cases} \quad \text{für } x \in \mathbb{R}$$

grenzung des Aktivierungszustandes mit sich zu bringen. Bekannte sigmoide Funktionen sind die Fermi-Funktion[18] und der Tangens Hyperbolicus[19].

4.3.1.3 Ausgabefunktion

Die Weitergabe des durch die Aktivierungsfunktion berechneten Wertes $a_j(t)$ an nachfolgende Units wird durch die Ausgabefunktion bewerkstelligt.

$$o_j(t) = f_{out}(a_j(t)) \tag{4.4}$$

Die meisten Netzmodelle wählen als Ausgabefunktion die Identitätsfunktion, die in Abbildung 4.4 dargestellt ist.

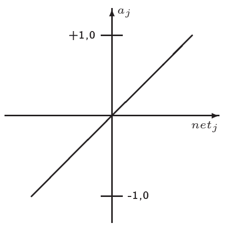


Abbildung 4.4: Identitätsfunktion als Ausgabefunktion

4.3.2 Netztopologie

Ein künstliches neuronales Netzwerk besteht nicht nur aus den oben beschriebenen Units, sondern auch aus den Verbindungen dieser Units. Die Topologie beschreibt, wie diese Units strukturiert (im Sinne von „angeordnet") sind, wie sich Aktivation im Netz ausbreitet und wie die Verbindungsstruktur angepasst werden kann. Ebenfalls von Bedeutung ist Verarbeitungsabfolge, ob also die Units synchron oder asychron die jeweilige Ausgabe berechnen.

Die Struktur eines neuronalen Netzes kann als (gerichteter) Graph oder als Konnektions- oder Adjazenzmatrix dargestellt werden. Diese Matrix enthält in ihren

[18] Die Fermi-Funktion $F(x)$ ist folgendermaßen definiert:

$$F(x) = \frac{1}{1+e^{-x}}$$

[19] Der Tangens Hyperbolicus $\tanh(x)$ ist folgendermaßen definiert:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Elementen die Gewichtungen der Verbindungen, wobei positive Werte eine exzitatorische[20] und negative Werte eine inhibitorische[21] Verbindung bedeuten.

4.3.2.1 Strukturierung

In einer **geschichteten** Netzwerktopologie werden die einzelnen Units mit gleichen Aufgaben zu Funktionsgruppen („*Layer*") zusammengefasst. Dabei existieren jeweils als Schicht *Eingabeneuronen* und *Ausgabeneuronen*. Dazwischen können eine, mehrere oder auch keine *versteckten Schichten („Hidden Layer")* angeordnet sein – siehe Abbildung 4.5.

In einer **ungeschichteten** Netzwerktopologie ist jede Unit mit jeder anderen Unit verbunden (man spricht hier auch von „Vollvernetzung"); eine Unterscheidung nach Funktionalitäten findet nicht statt.

4.3.2.2 Richtung der Aktivationsausbreitung

In **Feedforward-Netzen** erfolgt die Aktivationsausbreitung nur in einer Richtung – vom Input-Layer über allfällig vorhandene Hidden Layer zum Output-Layer. Es kommt jedenfalls zu keinen Rückkopplungen. Abbildung 4.5 zeigt eine schematische Darstellung.

Anders hingegen bei **Feedback-Netzen** (siehe Abbildung 4.6): Hier kann die Aktivationsausbreitung zwischen zwei Units generell in beiden Richtungen erfolgen. Es kommt somit zu Rückkopplungen.[22] Diese Rückkopplungen bringen einen neuen Aspekt: die *Repräsentation von Zeit*. Das Netzwerk bekommt dadurch die Möglichkeit, nicht nur auf Ähnlichkeiten von Mustern, sondern auch auf Reihenfolgen von Mustern differenziert zu reagieren (vgl. [Spitzer (2000), S. 183]).

[20] erregend, (ver-)stärkend

[21] hemmend

[22] Man beachte, dass hierbei nicht jede Unit mit jeder anderen verbunden sein muss: [Jang, Sun, Mizutani (1997), S. 201f.] bezeichnet neuronale Netze als *„recurrent"*, sobald es einen Feedback-Link gibt, der einen Kreis (in graphentheoretischem Sinn) erzeugt.

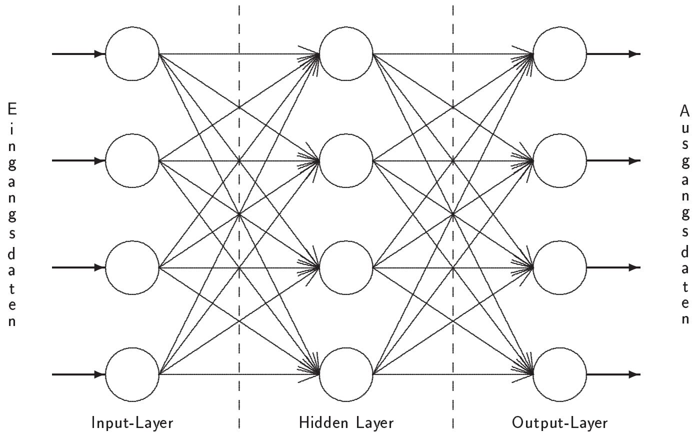


Abbildung 4.5: Struktur eines Feedforward-Netzes

4.3.2.3 Veränderbarkeit der Verbindungsstruktur

In **fixierten** Modellen werden die Verbindungsgewichte nach heuristischen Überlegungen berechnet und fest vorgegeben. Es erfolgt somit kein Lernen im eigentlichen Sinne.

Bei **adaptiven** Modellen hingegen werden die Verbindungsgewichte während eines Lernprozesses (siehe Kapitel 4.3.3, Seite 34) iterativ festgelegt.

4.3.2.4 Verarbeitungsabfolge

Man kann zwei Arten der Verarbeitungsabfolge unterscheiden: Bei der **synchronen** Verarbeitungsabfolge berechnen alle Units zuerst ihren Nettoinput und dann in einem weiteren Schritt den neuen Aktivierungszustand.

Bei der **asynchronen** Verarbeitungsabfolge berechnet eine zufällig ausgewählte Unit sowohl Input wie auch Output. Dieses Ergebnis geht dann in die Berechnungen der nächsten, wieder zufällig gewählten Unit mit ein. Es kommt somit zu Rückkopplungen und unter Umständen zu einem sehr lang dauernden Einschwingprozess.

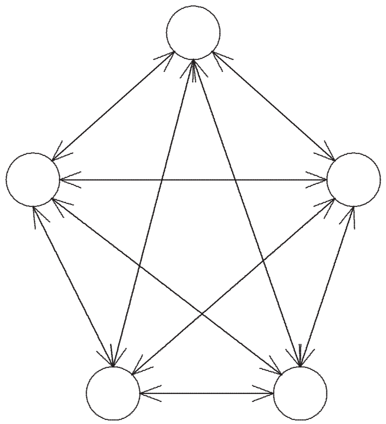


Abbildung 4.6: Struktur eines Feedback-Netzes

4.3.3 Lernverfahren

Wissen[23] wird in einem neuronalen Netz nicht explizit gespeichert, sondern implizit durch die Gewichte zwischen den einzelnen Units repräsentiert. Lernen in einem neuronalen Netz bedeutet daher, in einem vom Arbeitsmodus getrennten Prozess die Gewichte richtig zu trainieren und/oder die Verbindungsstruktur zu modifizieren.

Im Prinzip kann man folgende Lernarten unterscheiden (nach [Zell (1994), S. 94]):

1. Entwicklung neuer Verbindungen
2. Löschen existierender Verbindungen
3. Modifikation der Stärke $w_{i,j}$ der Verbindungen
4. Modifikation des Schwellwerts von Units
5. Modifikation von Propagierungs-, Aktivierungs- und/oder Ausgabefunktion
6. Entwicklung neuer Zellen
7. Löschen von Zellen

[23] Wissen wird in neuronalen Netzen oft auch als „Muster“ bezeichnet.

Die Varianten 1 und 2 sind bei entsprechenden Gewichtungen Spezialfälle von 3. Ähnliches gilt für die Modifikation der Schwellwerte von Units (4), die wie die Modifikation von Gewichten behandelt werden kann. Die Änderung der Propagierungs-, Aktivierungs- und Ausgabefunktionen (5) ist nicht sehr verbreitet und auch biologisch nicht sehr motiviert. Interessant sind wiederum die Punkte 6 und 7, die eine optimale Topologie des Netzes liefern.

Die elementare Unterscheidung der Lernmethoden liegt in der Art der Präsentation der zu erlernenden Muster. Man differenziert dabei zwischen überwachtem und unüberwachtem Lernen.

- **Überwachtes Lernen:** Hier werden dem System alle Eingangsmuster und die zugehörigen Ausgangsmuster („Klassen“) vorgelegt. Die vom Netzwerk produzierten Abweichungen vom gewünschten Ergebnis werden ermittelt, und die Gewichte zwischen den Units werden gemäß einer Lernregel (siehe nachfolgende Kapitel) angepasst, sodass sich die Abweichung zwischen Istwert und Sollwert verringert (siehe Abbildung 4.7). Ziel ist, dass das Netzwerk Muster wiedererkennen kann. Überwachtes Lernen wird auch als assoziatives Lernen oder als *supervised learning* bezeichnet.
- **Unüberwachtes Lernen:** Bei diesem Lernmodus werden dem Netz nur die Eingangsmuster, nicht aber die zugehörigen Klassifizierungen vorgelegt. Ziel ist, die in den Mustern vorhandenen Klassen selbstständig zu finden. Unüberwachtes Lernen wird auch als *unsupervised learning* oder *Clustering* bezeichnet.

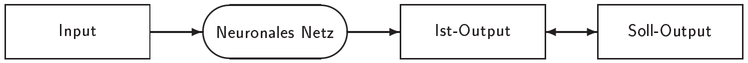


Abbildung 4.7: Grundprinzip des überwachten Lernens

4.3.3.1 HEBB'sche Lernregel

Die von DONALD HEBB bereits im Jahr 1949 – ohne Kenntnis der biologischen Vorgänge, die erst viel später nachgewiesen werden konnten – gefundene Lernregel ist bestechend einfach und besagt:

> *Sind die Units i und j (häufig) gleichzeitig stark aktiviert, so wird die Stärke ihrer Verbindung erhöht.*

Mathematisch lässt sich dieser Zusammenhang durch Gleichung 4.5 ausdrücken, wobei α die sogenannte „*Lernrate*“ darstellt; o_i ist der Output der Vorgängerzelle i und a_j die Aktivierung der Zelle j.

$$\Delta w_{ij} = \alpha \cdot o_i \cdot a_j \qquad 0 \leq \alpha \leq 1 \tag{4.5}$$

Allgemein kann die HEBB'sche Lernregel gemäß Gleichung 4.6 angeschrieben werden, wobei $t_j(t)$ die erwartete Aktivierung („*teaching input*“) repräsentiert. Die Funktionen g und h beschreiben die Gewichtsänderung jeweils proportional zu ihrem ersten Argument ([Mechler (1995), S. 66]).

$$\Delta w_{ij} = \alpha \cdot g(o_i(t), w_{ij}) \cdot h(a_j(t), t_j(t)) \qquad 0 \leq \alpha \leq 1 \tag{4.6}$$

Als Nachteil der HEBB'schen Lernregel ist zu nennen, dass bei anhaltender Aktivierung der beiden Units das zugehörige Verbindungsgewicht ins Unendliche wächst. Als Abhilfe kann eine „Abschwächfunktion“ definiert werden, die das Verbindungsgewicht mit der Zeit wieder reduziert – ähnlich einem „undichten Kondensator“ ([Brause (1995), S. 80]).

Die HEBB'sche Lernregel findet sowohl für überwachtes wie auch für unüberwachtes Lernen Verwendung – je nach dem, ob eine erwartete Aktivierung $t_j(t)$ zur Verfügung steht oder nicht.

4.3.3.2 Delta-Regel

Die Delta-Regel stellt eine Erweiterung der HEBB'schen Lernregel dar. Sie wird auch WIDROW-HOFF-Regel genannt. Die Gewichtsänderung ist proportional zur Differenz δ_j der aktuellen Aktivierung a_j und der erwarteten Aktivierung t_j.

$$\begin{aligned} \Delta w_{ij} &= \alpha \cdot o_i \cdot \delta_j \qquad 0 \leq \alpha \leq 1 \\ \delta_j &= t_j - a_j \end{aligned} \tag{4.7}$$

Die Delta-Regel ist allerdings nur für zweischichtige Netzwerke definiert, da die gewünschte Aktivierung t_j nur für die Output-Units, nicht aber für die Hidden-Units beschrieben ist.

4.3.3.3 Backpropagation

Das Backpropagation-Lernverfahren ist eine Verallgemeinerung der Delta-Regel auf mehrschichtige Netzwerke. Die grundlegende Idee dabei ist, dass die Hidden Units eine „interne Repräsentation“ des gewünschten Outputs durch Rückwärts-Propagation erlernen. Das Lernen erfolgt also in zwei Phasen: Im ersten Schritt werden die Aktivierungen vom Input-Layer über den/die Hidden Layer zum Output-Layer propagiert. In der zweiten Phase wird der am Ausgang errechnete Fehlerwert auf die Hidden Units aufgeteilt, wobei die Units, die näher am Output-Layer liegen, einen stärkeren Anteil haben.

Die Gewichtsänderungen werden durch die folgende Gleichung 4.8 definiert. Die einzelnen Variablen und Funktionen sind in der Tabelle darunter beschrieben.

$$\Delta w_{ij} = \alpha \cdot o_i \cdot \delta_j$$
$$\delta_j = \begin{cases} f'_j(net_j) \cdot (t_j - o_j) & \text{falls } j \text{ eine Output-Unit ist} \\ f'_j(net_j) \cdot \sum_k (\delta_k \cdot w_{jk}) & \text{falls } j \text{ eine Hidden Unit ist} \end{cases} \quad (4.8)$$

w_{ij}	...	Gewicht von Unit i zu Unit j
α	...	Lernrate
o_i, o_j	...	Outputwerte der Units i und j
δ_j	...	Fehler der Unit j (Differenz zwischen Sollwert und dem durch die Unit errechneten Wert)
$f'_j()$	...	Erste Ableitung der Aktivierungsfunktion
t_j	...	Sollwert
net_j	...	Netto-Input der Unit j
k	...	Summationsindex aller direkten Nachfolger der Unit j

Dieses Lernverfahren wird oft auch als „*Vanilla Backpropagation*“ bezeichnet.

4.3.3.4 Backpropagation with Momentum

Eine Verbesserung des oben genannten Lernverfahrens der klassischen Backpropagation stellt die Etablierung eines „Momentum“-Terms dar. Dabei wird die Änderung des Kantengewichts der vorangehenden Berechnung zur Berechnung der neuen Änderung des Kantengewichts herangezogen. Die neue Berechnungsformel wird in Gleichung 4.9 dargestellt.

Der Parameter μ ist eine Konstante, die das Momentum spezifiziert; $\Delta w_{ij}(t)$ ist das Kantengewicht zwischen den Units i und j im Iterationsschritt t; die anderen Variablen und Funktionen sind der Tabelle in Kapitel 4.3.3.3 zu entnehmen.

$$\Delta w_{ij}(t+1) = \alpha \cdot o_i \cdot \delta_j + \mu \cdot \Delta w_{ij}(t)$$
$$\delta_j = \begin{cases} f'_j(net_j) \cdot (t_j - o_j) & \text{falls } j \text{ eine Outputunit ist} \\ f'_j(net_j) \cdot \sum_k (\delta_k \cdot w_{jk}) & \text{falls } j \text{ eine Hidden Unit ist} \end{cases} \quad (4.9)$$

Der Vorteil dieser Erweiterung liegt darin begründet, dass flache Stellen des Lernverlaufs sehr rasch abgearbeitet werden, während für „raue“ Stellen mehr Iterationen durchgeführt werden und somit die Genauigkeit steigt. Daraus ergibt sich eine signifikante Verbesserung der Lerngeschwindigkeit (vgl. [Zell (1995), S. 146]).

4.3.3.5 Backpropagation with Weight Decay

Backpropagation with Weight Decay reduziert die Gewichte der Kanten während des Trainings mit dem Backpropagation-Verfahren. Bei jeder Aktualisierung des Kantengewichts wird das Gewicht des alten Kantengewichts mit dem Faktor d miteinbezogen und vom neuen Gewicht subtrahiert (Gleichung 4.10).

Der Parameter d ist ein konstanter Faktor, der den Einfluss des alten Kantengewichts $w_{ij}(t)$ zwischen den Units i und j auf die Änderung des neuen Kantengewichts $\Delta w_{ij} i(t+1)$ spezifiziert.

$$\Delta w_{ij}(t+1) = \alpha \cdot o_i \cdot \delta_j - d \cdot w_{ij}(t)$$
$$\delta_j = \begin{cases} f'_j(net_j) \cdot (t_j - o_j) & \text{falls } j \text{ eine Outputunit ist} \\ f'_j(net_j) \cdot \sum_k (\delta_k \cdot w_{jk}) & \text{falls } j \text{ eine Hidden Unit ist} \end{cases} \quad (4.10)$$

Durch dieses Lernverfahren konvergieren die Gewichte zu 0; es sei denn, sie werden durch die Backpropagation wieder mit Werten > 0 gesetzt. Der Effekt ist daher ähnlich dem „*Pruning*“ von Netzwerken (siehe [Zell (1995), S. 216ff]), womit versucht wird, unnötige Links und Units zu eliminieren.

4.3.3.6 Backpropagation with chunkwise Update

Als letzte hier vorgestellte Adaption von Backpropagation-Lernverfahren soll die Backpropagation mit *chunkwise Update* diskutiert werden. Während bei der *Vanilla Backpropagation* ein Update der Kantengewichte nach jedem dem Netz präsentierten Muster stattfindet, kann hier ein *Chunk*[24] der Größe N definiert werden. Dies bedeutet, dass die Links zwischen den Units nur nach der Präsentation von N Mustern aktualisiert werden. Dieses Verfahren hat sich vor allem bei großen Trainingsmengen als sinnvoll herausgestellt (vgl. [Zell (1995), S. 146]).

[24] *Chunk*, engl.: großes Stück, Brocken

5 Charakterisierung der Nachrichten-„Daten“

Bevor das Training des neuronalen Netzes in Kapitel 6, Seite 49, erläutert wird, sollen hier die Zusammenhänge zwischen (un-)veröffentlichten Artikeln[1], Schlüsselwörtern[2] und Worthäufigkeiten dargestellt werden. Weiters werden die Artikel mit Hilfe eines sogenannten *Kohonen-Netzes* geclustert, um so ebenfalls Aussagen über die Artikel und ihre „inneren Zusammenhänge“ treffen zu können.

Einen Überblick über die zu erlernenden Daten und Datenmengen gibt Tabelle 5.1. Die Gesamtwörteranzahl beinhaltet alle Wörter der Nachrichten des Betrachtungszeitraums vom 1. Jänner 1999 bis zum 26. Februar 1999.[3] Wörter werden hierbei durch zwischen den Wörtern stehende Satz- und/oder Leerzeichen unterschieden. Bei den unterschiedlichen Wörtern wird kein Wort doppelt gezählt; es werden hiefür alle Wörter verwendet, außer solchen mit weniger als vier Buchstaben, die nicht ausschließlich aus Großbuchstaben bestehen (dabei handelt es sich nämlich um Abkürzungen wie USA, UNO, ...). Die Häufigkeit von ein-, zwei und dreibuchstabigen Wörtern ist im Verhältnis zur Gesamtwortanzahl ohnedies sehr gering und daher vernachlässigbar. Weiters werden für das Deutsche typische Nachsilben wie „e“, „te“ oder ähnliche entfernt.

Artikel gesamt	3.559
Artikel veröffentlicht	670
Artikel unveröffentlicht	2.889
Wörter gesamt	997.094
Unterschiedliche Wörter gesamt	26.697
Schlüsselwörter (manuell)[a]	1.044
Schlüsselwörter (automatisch)[b]	1.075

[a] Manuell ausgewählte Schlüsselwörter, deren *Vorkommen* auf einen hohen Nachrichtenwert schließen lassen

[b] Automatisch ausgewählte Schlüsselwörter, deren *Häufigkeit* auf einen hohen Nachrichtenwert schließen lassen

Tabelle 5.1: Überblick über die zu erlernenden Nachrichten-„Daten“

5.1 Verteilung der Anzahlen der Wörter

Für die genaue Kenntnis des Daten- und Textmaterials scheint es von Interesse zu sein, die Häufigkeit der Vorkommnisse der einzelnen Wörter zu kennen – Abbildung 5.1 zeigt diese Verteilung. (Ein Wort besteht auch hier wiederum aus mindestens drei Buchstaben, außer es handelt sich um Großbuchstaben; siehe oben.)

[1] Die Begriffe „Artikel“ und „Nachricht“ werden synonym verwendet.
[2] Die Begriffe „Schlüsselwort“, „Keyword“ und „Feature“ werden synonym verwendet.
[3] Die beiden letzten Februar-Tage wurden nicht mehr ausgewertet, da die APA-Artikel dieser Tage teilweise erst im März im STANDARD publiziert wurden.

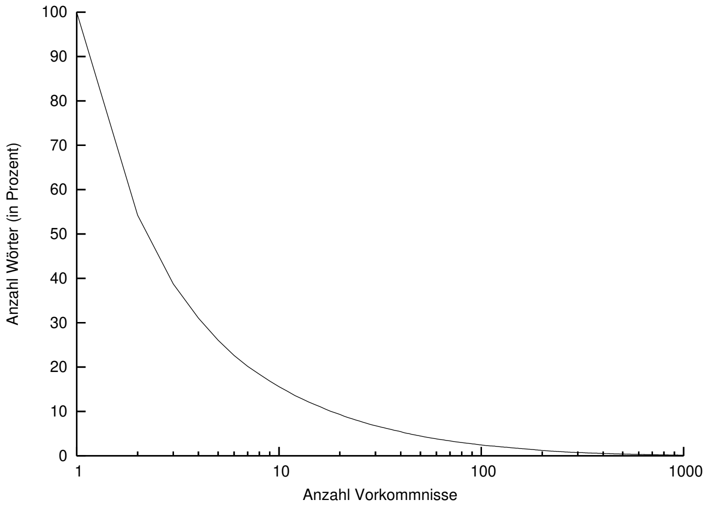


Abbildung 5.1: Wort-Anzahlen

Auf der x-Achse ist die Mindestanzahl der Wörter dargestellt, dh, es wird gezeigt, wieviele Wörter (in Prozent aller Wörter) mindestens n-mal vorkommen. Auf der y-Achse wird die Häufigkeit der Wörter in Prozent dargestellt.[4]

Häufig vorkommende Wörter geben keinerlei Hinweis auf den Nachrichtenwert eines Artikels, da sie ja in annähernd jedem Artikel vorkommen. Analoges gilt für Wörter, die kaum vorkommen: Auch sie stellen keinen Nachrichtenwert dar.[5] Deshalb werden für einen Teil der weiteren Betrachtungen nur jene Features gewählt, die mindestens zwanzig Mal vorkommen. Davon werden wiederum die häufigsten 50 % der Wörter entfernt, sodass letztlich nur jene Schlüsselwörter übrig bleiben, deren Vorkommen auf einen hohen Nachrichtenwert schließen lässt. Es handelt sich dabei um 1.075 Schlüsselwörter.

Für die folgenden Auswertungen werden jedoch lediglich die manuell ausgewählten Schlüsselwörter heran gezogen, da es bei diesen Auswertungen kaum Unterschiede zwischen manuell und automatisch selektierten Schlüsselwörtern gibt.[6]

[4] Wenig überraschend kommen 100 % der Wörter mindestens ein Mal vor.
[5] Vgl. Kapitel 3.5, Seite 19.
[6] Auswertungen mit automatisch selektierten Schlüsselwörtern finden sich in Anhang C.1 auf Seite 106.

5.2 Verteilung der Anzahlen der Keywords

Die folgende Auswertung folgt der Überlegung, dass Artikel mit mehr Schlüsselwörtern eher veröffentlicht werden als solche mit weniger Schlüsselwörtern.

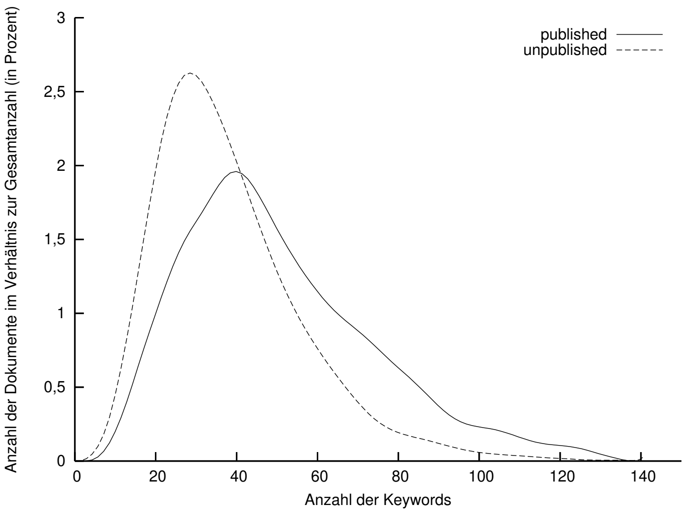


Abbildung 5.2: Verteilung der Anzahlen der Keywords

Abbildung 5.2 zeigt, wieviele Artikel mit einer bestimmten Anzahl von Keywords veröffentlicht („*published*“) bzw. nicht veröffentlicht („*unpublished*“) wurden. Um die Daten vergleichbar zu machen, wurde die jeweilige Anzahl der Artikel mit der jeweiligen Gesamtanzahl (veröffentlicht bzw. nicht veröffentlicht) normiert. Beispielsweise beinhalten knapp zwei Prozent der veröffentlichten 670 Artikel 40 Schlüsselwörter.

Deutlich ist aus Abbildung 5.2 zu erkennen, dass Nachrichten mit geringer Anzahl von Keywords eher unveröffentlicht bleiben, während Nachrichten mit hoher Anzahl von Keywords tendenziell häufiger veröffentlicht werden. Diese Tendenz ist jedoch zu gering, um *allein* die Anzahl der Schlüsselwörter als Auswahlkriterium für die Veröffentlichung von Artikeln heranzuziehen.

5.3 Verteilung der Anzahl der Schlüsselwörter im Verhältnis zur Gesamtwortanzahl

Während in Kapitel 5.2 lediglich die Anzahl der Schlüsselwörter ausschlaggebend war, wird im folgenden auch die Gesamtanzahl der Wörter in einer Nachricht einbezogen.

Die beiden Abbildungen der Wortverteilungen (Abbildung 5.3 und 5.4) zeigen, dass – unabhängig von der Veröffentlichung – in längeren Artikeln auch mehr Keywords vorkommen. Jedoch kann auch hier (wie auch in Kapitel 5.2) kein alleiniges Auswahlkriterium für die Veröffentlichung gewonnen werden, da eine besondere Häufigkeit von Keywords in veröffentlichten Artikeln nicht gegeben ist.

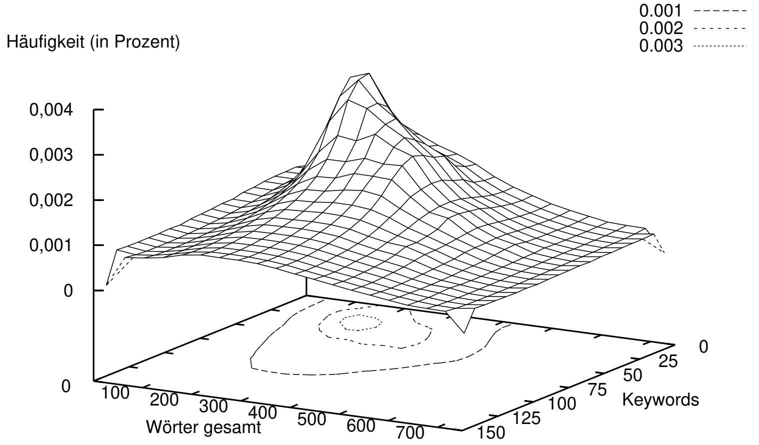


Abbildung 5.3: Wortverteilung unpublizierte Artikel

Weiters lässt sich aus den beiden Abbildungen folgendes erkennen:

- Die Anzahl der Schlüsselwörter steigt linear mit der Gesamtanzahl der Wörter in einem Artikel: Der „Bergrücken“ verläuft sowohl bei den veröffentlichten wie auch bei den unveröffentlichten Artikeln ungefähr diagonal in der durch die Anzahl der Schlüsselwörter und der Gesamtanzahl der Wörter aufgespannten Ebene.
- Ebenfalls unabhängig von der Veröffentlichung kommen Artikel mit rund 300 Wörtern und rund 50 Schlüsselwörtern am häufigsten vor. Je länger ein Artikel ist, desto geringer ist die Wahrscheinlichkeit seiner Veröffentlichung.

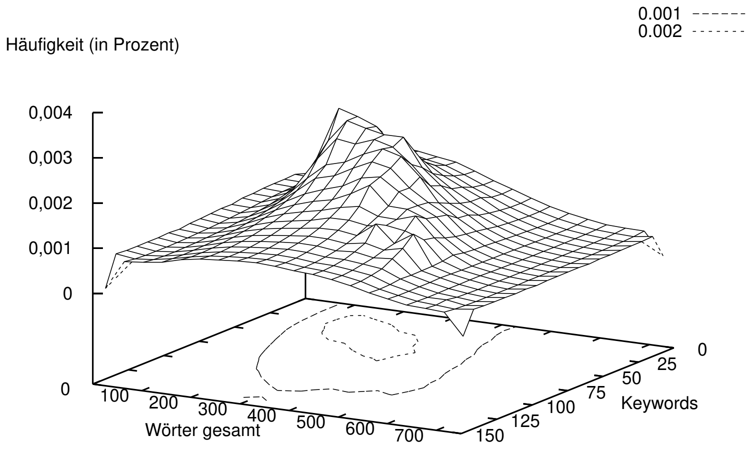


Abbildung 5.4: Wortverteilung publizierte Artikel

- Die Struktur der veröffentlichten Artikel ist „rauer“ als die Struktur der nicht veröffentlichten Artikel. Dies könnte ein Indiz sein, dass die Wichtigkeit (die Veröffentlichungswürdigkeit) eines Artikels nicht so stark mit der Anzahl der (Schlüssel-)Wörter korreliert wie es auf den ersten Blick den Anschein hat.

Insgesamt lässt sich aus dem bis jetzt Gesagten ableiten, dass eine höhere Anzahl von Schlüsselwörtern eher zu einer Veröffentlichung einer Nachricht führt. Doch selbst diese sehr vage Aussage wird relativiert, wenn die Anzahl der Schlüsselwörter relativ zur Gesamtanzahl der Wörter eines Artikels betrachtet wird: Die Wahrscheinlichkeit der Veröffentlichung ist nun nicht mehr an einer bestimmten Stelle höher – es gibt vielmehr mehrere Maxima mit relativ höchster Veröffentlichungswahrscheinlichkeit.

5.4 Ähnlichkeiten von Dokumenten

Abbildung 5.5 zeigt die Ähnlichkeit von Nachrichten über mehrere (Vor-)Tage betrachtet. Als Maß für die Ähnlichkeit dient die *Kosinusdistanz* (siehe Kapitel 3.7.1, Seite 22).

Es wurden die vier folgenden Kombinationen aus veröffentlichten und unveröffentlichten Artikeln untersucht:

1. **Ähnlichkeit der publizierten Artikel:** Es wurden alle veröffentlichten Artikel nur mit anderen veröffentlichten Artikeln für die Berechnung der Ähnlichkeit herangezogen.
2. **Ähnlichkeit der publizierten Artikel mit allen Artikeln:** Alle veröffentlichten Artikel werden mit *allen* anderen Artikeln verglichen.
3. **Ähnlichkeit aller Artikel:** Es werden *alle* Artikel mit allen anderen Artikeln verglichen.
4. **Ähnlichkeit der publizierten Artikel mit den unpublizierten Artikeln:** Die Berechnung der Ähnlichkeit erstreckt sich auf die publizierten Artikel, die mit allen *nicht* publizierten Artikeln verglichen werden.

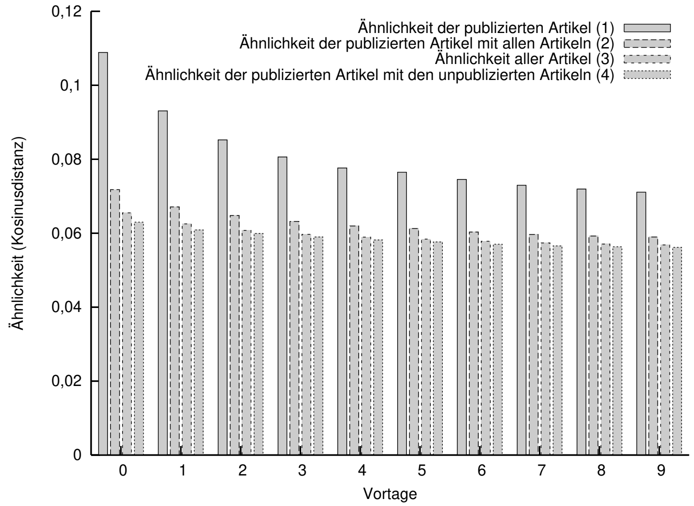


Abbildung 5.5: Ähnlichkeiten von Artikeln über mehrere (Vor-)Tage betrachtet

Wenig überraschend zeigen sich zwei klare Tendenzen: Einerseits ist die Ähnlichkeit der publizierten Artikel untereinander (Fall 1) am höchsten. Dies entspricht auch der Nachrichtendimension „Konsonanz“ (siehe Kapitel 2.4, Seite 7), die unter anderem in der Kontinuität einen Faktor für die Veröffentlichung eines Ereignisses sieht. Zieht man auch nicht publizierte Nachrichten mit in Betracht oder vergleicht man gar die Ähnlichkeiten der publizierten Artikel mit den unpublizierten (Fall 4), so nimmt das Maß für die Ähnlichkeit relativ stark ab.

Auf der anderen Seite zeigt sich auch, dass es einen eindeutigen Trend zur Veröffentlichung von Ereignissen mit kürzerer Dauer gibt: So ist das Ähnlichkeitsmaß für am selben oder am Vortag veröffentlichte Artikel am größten.[7] Dies deckt sich mit der Nachrichtendimension „Dynamik“ und deren Nachrichtenfaktor „Frequenz“, wonach kurzfristige Ereignisse höhere Beachtungs- und Publikationschancen haben.

Für die Artikelsuche mit Hilfe eines neuronalen Netzes bedeutet dies, dass die Ähnlichkeit zu bereits publizierten Nachrichten ebenfalls miteinbezogen werden muss.

5.5 Clustering[8] der Daten mit Hilfe von Karten

Als weitere Methode der Charakterisierung der Nachrichtendaten bietet sich die Möglichkeit des *Clusterings* an. Dabei werden hochdimensionale Daten (im konkreten Anwendungsfall also die von der *APA* publizierten Nachrichten) auf einer Karte gruppiert. Die dadurch entstehenden Gruppen zeichnen sich durch Artikel ähnlicher Inhalte aus.

5.5.1 Grundlagen, Lernverfahren

Das grundlegende Verfahren der *Self Organizing Maps* wurde von Teuvo Kohonen ([Kohonen (2001)]) entwickelt. Es handelt sich dabei um ein neuronales Netz, dessen Output-Units einen zweidimensionalen Merkmalsraum („Karte“) bilden. (Natürlich sind höherdimensionale Merkmalsräume ebenso möglich wie beispielsweise hexagonale Topologien.) Jeder dieser Output-Units ist ein anfangs mit Zufallswerten initialisierter Gewichtsvektor zugeordnet.

Das (unüberwachte) Lernverfahren läuft nach folgendem Schema ab (nach [Dittenbach, Berger, Merkl (2004)]):

1. Zufällige Auswahl eines Inputvektors.
2. Berechnung der Aktivierung der Output-Units; die Aktivierung korreliert mit der Distanz zwischen dem Inputvektor und dem Gewichtsvektor.[9]
3. Auswahl einer „*Winner-Unit*“, also jener Unit mit der geringsten Distanz zwischen Input- und Gewichtsvektor.[9]
4. Anpassung der Gewichtsvektoren der Winner-Unit und der Units in deren Nachbarschaft.

[7] Die sehr hohe Ähnlichkeit zu am selben Tag veröffentlichten Artikeln erklärt sich auch aus der Tatsache, dass zu einem Thema oftmals mehrere Presseaussendungen von unterschiedlichen Interessensgemeinschaften publiziert werden.

[8] *Cluster*, engl.: Gruppe, Haufen

[9] Als Maß für die Distanz wird beispielsweise die Euklidische Distanz verwendet (siehe Kapitel 3.7.2, Seite 22).

5. Wiederholung der Schritte 1–4, bis ein definiertes Kriterium (beispielsweise Anzahl der Iterationen) für das Ende des Lernens erreicht ist.

5.5.2 Visualisierung

Der oben beschriebene Algorithmus ordnet die einzelnen Nachrichten bestimmten Sektoren auf der Karte zu. Doch ohne passender Visualisierung ist nicht erkennbar, wie die einzelnen Nachrichten innerhalb und außerhalb des Sektors zueinander in Verbindung stehen – die *Topologie* der Karte ist nicht bekannt.

Es gibt verschiedene Methoden, die Topologie einer *Self Organizing Map* (im Folgenden als SOM bezeichnet) zu errechnen. Eine davon ist die sogenannte „*Unified Distance Matrix*“ oder kurz „*U-Matrix*“ ([Ultsch, Siemon (1990)]). Dabei wird für jede Output-Unit der Mittelwert der Distanzen zwischen dem eigenen Gewichtsvektor und dem Gewichtsvektor der Nachbarunits berechnet. Das Ergebnis wird jeder Output-Unit der Karte zugeordnet. Mit Hilfe einer Farbskala[10] lässt sich so eine Aussage über die Ähnlichkeit der auf der Karte dargestellten Nachrichten treffen.

Die dabei entstehenden Bergrücken (in Abbildung 5.6 weiß dargestellt) deuten auf Clustergrenzen, die Täler dazwischen (blau) lassen auf Clusterzentren schließen.

5.5.3 Karte der „Nachrichten-Daten“

Die in Abbildung 5.6 dargestellte Karte zeigt die Clusterung der Artikel der *APA*. Rote Kreise bzw. Kreissegmente zeigen den Anteil der nicht veröffentlichten Nachrichten in diesem Sektor, blaue Kreise bzw. Kreissegmente zeigen an, wieviele Nachrichten veröffentlicht wurden.

Die „Landschaft“ setzt sich sehr großzügig aus Bergrücken (weiß) und einigen Tälern (blau) zusammen. Dies lässt auf eine relativ starke inhaltliche Segmentierung der Nachrichten schließen. Die meist gelben Verbindungen weisen auf eine – wenn auch geringe – Ähnlichkeit der Themen hin.

Zur Illustration wurden einige Segmente der Karte mit dem jeweiligen Hauptthema des Segments beschriftet. Es zeigt sich, dass in der Tat einige ähnliche Themen in unmittelbarer Nachbarschaft angeordnet wurden: „ÖVP, Wahlkampf“, „Familienpolitik“, „Karenzgeld“ und „FPÖ, Mietensenkung“ sind soziale Themen bzw. wurden von einer auf Familienpolitik ausgerichteten Partei vertreten. Die für die Repräsentation der Artikel ausgewählten Schlüsselwörer sind also ausreichend in dem Sinne, dass ein Clustering auf dieser Basis möglich ist.

Teilweise wurden auch vollkommen unähnliche Themen in unmittelbarer Nachbarschaft abgebildet („Bundesheer“, „Ewald Stadler“). Dies liegt darin begründet, dass

[10] Natürlich existieren auch andere Möglichkeiten der Visualisierung, etwa eine z-Koordinate für eine dreidimensionale Darstellung.

die Karte für die Anzahl der Features ein wenig zu klein gewählt wurde, wodurch die SOM bei der Berechnung „Kompromisse“ treffen musste. Eine größere Karte hätte allerdings fast nur Bergrücken und einzeln verstreute Täler erzeugt.

Nicht direkt auf der Karte dargestellt ist die Anzahl der Artikel, die durch ein Segment repräsentiert werden. Auswertungen der dieser Karte zugrundeliegenden Daten zeigen jedoch, dass einzelne Nachrichten zu einem Spezialthema kaum veröffentlicht werden (solche Nachrichten finden sich meist in Clustern mit voll ausgefüllten roten Kreisen). Dies ist leicht erklärbar, da natürlich auch die *APA* als Nachrichtenquelle den Gesetzen der Nachrichtenwert-Theorie folgt und für im weitesten Sinne interessante Themen natürlich auch mehr Artikel publiziert.

Weitere Aussagen über die Veröffentlichungswürdigkeit von Nachrichten können anhand dieser Karte jedoch nicht getroffen werden: Die veröffentlichten Artikel gruppieren sich nicht an bestimmten Stellen sondern sind überall ungefähr gleich häufig anzutreffen.

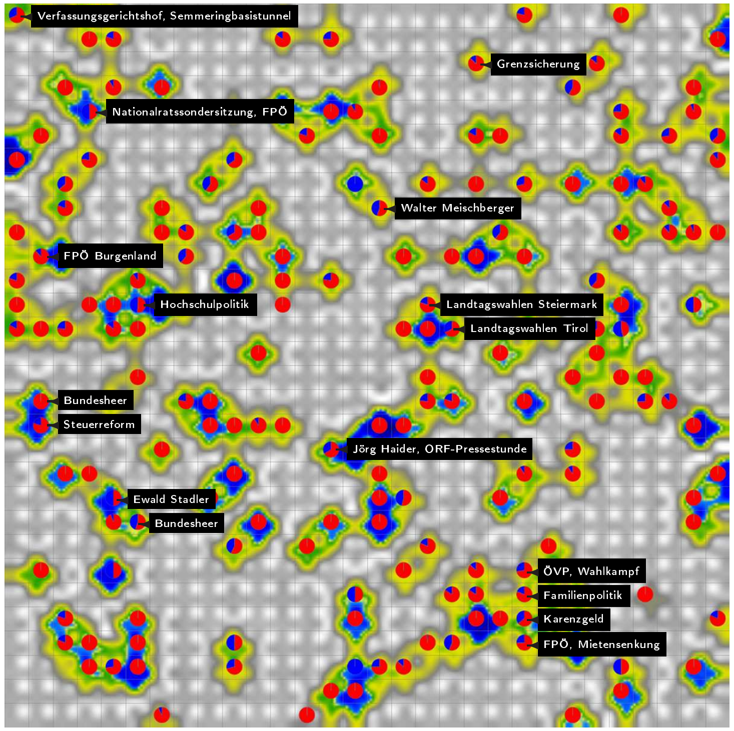


Abbildung 5.6: Clustering der Nachrichten durch eine SOM

6 Artikelsuche mit neuronalen Netzen

Dieses Kapitel beschreibt, wie auf Basis der vorangehenden Abschnitte ein neuronales Netz entwickelt wird, das in der Lage ist, Artikel anhand der Kriterien der Nachrichtenwert-Theorie zu selektieren. Dabei werden Schritt für Schritt die Grundlagen des Trainings und dessen Bewertung erklärt. Zusätzlich werden Optimierungen des Netzwerkes und deren Auswirkungen auf die konkrete Anwendung dargestellt. Den Abschluss bildet die Präsentation der Ergebnisse und deren Analyse.

6.1 Grundprinzip des Trainings

Das Grundkonzept des Trainings des neuronalen Netzes soll *überwachtes Lernen* (Kapitel 4.3.3, Seite 35) sein: Abbildung 6.1 zeigt den schematischen Ablauf.

Dabei werden „Mengen von Nachrichten“ verwendet:

- **Bruttonachrichtenmenge $\mathfrak{B}$:** Damit werden *alle* Nachrichten im Versuchszeitraum bezeichnet. Diese Nachrichten kommen beispielsweise von einer Nachrichtenagentur wie der *APA* und stehen den Journalisten zur Verfügung, um einige auszuwählen und zu publizieren.
- **Nettonachrichtenmenge $\mathfrak{N}$:** Alle von Journalisten im Versuchszeitraum ausgewählten und publizierten Artikel werden als Nettonachrichtenmenge bezeichnet. Die Nettonachrichtenmenge ist also eine Teilmenge der Bruttonachrichtenmenge: $\mathfrak{N} \subset \mathfrak{B}$.

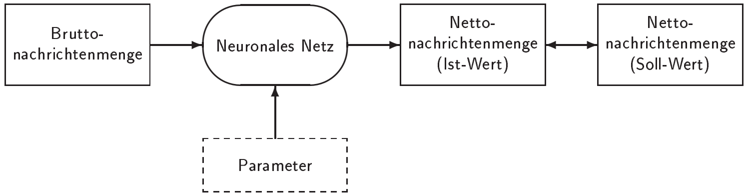


Abbildung 6.1: Grundprinzip des Erlernens der Nachrichtenwerte

Die Zugehörigkeitsfunktion φ gibt an, ob ein Artikel $\mathfrak{a}$ zur Nettonachrichtenmenge $\mathfrak{N}$ gehört oder nicht:

$$\varphi(\mathfrak{a}) = \begin{cases} 0: & \mathfrak{a} \notin \mathfrak{N} \\ 1: & \mathfrak{a} \in \mathfrak{N} \end{cases} \tag{6.1}$$

Beim Training werden dem neuronalen Netz Schritt für Schritt alle Artikel $\mathfrak{a}_i$ der Bruttonachrichtenmenge auf der Inputseite und das Ergebnis der Zugehörigkeits-

funktion $\varphi(\mathfrak{a}_i)$ auf der Outputseite als Sollwert vorgelegt. Daraus folgt auch, dass der Output-Layer des neuronalen Netzes nur genau *eine* Unit besitzt.[1]

Der Test des trainierten Netzes erfolgt auf ähnliche Weise. Allerdings werden dem neuronalen Netz ihm unbekannte Artikel vorgelegt und der daraus resultierende Output mit dem Sollwert (Ergebnis der Zugehörigkeitsfunktion $\varphi(\mathfrak{a}_i)$) verglichen.

6.2 Datenakquisition und Datencodierung

Die Datenakquisition und Datencodierung sind – neben der Wahl der „richtigen“ Topologie des neuronalen Netzes – zwei wesentliche Punkte für das erfolgreiche Lösen der Problemstellung.

Bei der Daten- bzw. Wortakquisition (siehe unten) ist darauf zu achten, dass die gewählten Features „hervorstechend“[2] im Vergleich zu allen anderen Wörtern sowohl im gerade betrachteten Nachrichtendokument wie auch in allen Nachrichten sind, um so den Nachrichtenwert und daraus in weiterer Folge den Veröffentlichungsstatus bestimmen zu können (vgl. Kapitel 3.5, Seite 19, und Kapitel 5.1, Seite 39).

Die Datencodierung erfolgt mit Hilfe des bereits in Kapitel 3.6, Seite 20, vorgestellten *Vector Space Models*. Jedes Dokument (jede Nachricht) wird dabei durch einen Vektor dargestellt, dessen einzelne Komponenten die (normierten) Anzahlen der Keywords des Dokuments darstellen.

6.2.1 Wie geschieht die Wortakquisition?

Im ersten Schritt wurden während der Artikelklassifikation (welche Artikel der *APA* wurden im STANDARD veröffentlicht?) Wörter ausgewählt und den sechs Nachrichtendimensionen bzw. den zwanzig Nachrichtenfaktoren (siehe Kapitel 2.4, Seite 7) zugeordnet.

Nach Abschluss dieser Arbeit wurden in den Texten der *APA* sämtliche bereits gefundenen Schlüsselwörter markiert. Dadurch konnten bereits spezifizierte Schlüsselwörter leicht erkannt und weitere leicht gefunden werden, die nun ebenfalls wieder den sechs Nachrichtendimensionen bzw. den zwanzig Nachrichtenfaktoren zugeordnet wurden (siehe Anhang A, Seite 67).

6.3 Fehlerrate

Im Rahmen der Parametrierung des neuronalen Netzes (siehe folgendes Kapitel 6.4) muss die Güte eines Lernverfahrens zur Beurteilung seiner Qualität gemessen werden.

[1] Im Rahmen der später dargestellten Experimente werden auch neuronale Netze mit drei Output-Units und einer 2-aus-3 Mehrheitsentscheidung verwendet.

[2] Die Wörter sollten weder besonders häufig noch sehr selten vorkommen.

Dies lässt sich am besten über die Fehlerrate bewerkstelligen, die sowohl für die Trainingsdaten wie auch für die Testdaten herangezogen wird.

Die Fehlerrate (SSE, *Sum Squared Error*) ist die Summe der Quadrate der einzelnen Fehler (Gleichung 6.2, [Zell (1995), S. 65]). t_{pj} ist der Sollwert der Unit j bei Anlegen des Musters p; o_{pj} ist der Istwert der Unit j bei Anlegen des Musters p.

$$SSE = \sum_{p \in Muster} \sum_{j \in Output} (t_{pj} - o_{pj})^2 \tag{6.2}$$

Grundsätzlich kann die Fehlerrate auch auf die Anzahl der Muster normiert werden – man erhält dann den *MSE* (*Mean Squared Error*). Dieser in Gleichung 6.3 dargestellte Vergleichsparameter ist jedoch wegen der immer konstanten Anzahl von Trainingsmustern für die weitere Betrachtung kaum von Belang.

$$MSE = \frac{SSE}{n} \tag{6.3}$$

6.4 Parametrierung

Die Parametrierung eines neuronalen Netzes umfasst zum Teil die Definition der Topologie des Netzes (siehe nachfolgendes Kapitel) wie auch die Wahl des Lernverfahrens und dessen Lernraten.

Als Lernverfahren haben sich aufgrund der gewählten Topologie des neuronalen Netzes (siehe nachfolgendes Kapitel) und nach einer Vorselektion sowohl *Backpropagation with Momentum* (Kapitel 4.3.3.4, Seite 37) als auch *Backpropagation with Weight Decay* (Kapitel 4.3.3.5, Seite 37) als relativ günstig erwiesen.

Die Werte der Parameter Lernrate α und Momentum-Term μ bzw. Faktor d der beiden Lernverfahren wurden zuerst geschätzt und später im Laufe mehrerer Trainingsschritte verfeinert. In den einzelnen Abbildungen in Kapitel 6.10 werden die Parameter immer innerhalb eines bereits sehr günstigen Bereichs variiert.

6.5 Topologie des neuronalen Netzes

Ausgehend von der Aufgabenstellung wurde ein Feedforward-Netz[3] als grundlegende Topologie des Netzes gewählt: Aufgrund bestimmter Eingangsmuster (konkret: in einem Artikel vorkommende Schlüsselwörter) ist ein bestimmtes Ausgangsmuster zu erzeugen (konkret: Veröffentlichung eines Artikels).

Die Anzahl der Input-Units ist deshalb durch die Anzahl der Features vorgegeben. Da die gewünschte Information als Output des neuronalen Netzes lediglich eine Ja-Nein-Entscheidung darstellt, reicht es aus, genau *eine* Output-Unit zu verwenden.

[3] Feedforward-Netze sind in Kapitel 4.3.2.2, Seite 32, erläutert.

Testweise wird gelegentlich mit drei Output-Units und einer 2-aus-3 Mehrheitsentscheidung gearbeitet. Dabei wird als Gesamtergebnis jenes Ergebnis herangezogen, das bei *mindestens* zwei der drei Output-Units identisch ist.

Nun ist nur noch die Anzahl der Hidden Units zu definieren – dies geschieht im folgenden Abschnitt.

6.5.1 Anzahl der Hidden Units

In der Literatur (bspw. [Zimmermann (1995), S. 64]) wird die sinnvolle Anzahl der Trainingsdaten mit der doppelten bis vierfachen Anzahl der Verbindungen in einem neuronalen Netz angegeben. Die Anzahl der Verbindungen errechnet sich nach folgender Gleichung 6.4:

$$Anzahl\ Verbindungen = \sum_{i=1}^{N-1} n_i \cdot n_{i+1} \tag{6.4}$$

Dabei ist N die Anzahl der Schichten, n_i die Anzahl der Units in Schicht i.

Die Anzahl der Trainingsdaten ist aufgrund der im Beobachtungszeitraum durch die Nachrichtenagentur *APA* veröffentlichten Nachrichten fix vorgegeben: Die Trainingsdatenmenge umfasst 3.619 Elemente. Die Anzahl der Units in Schicht 1 (*Input-Layer*) ist die Anzahl der Features und deshalb mit 1.044 ebenfalls vorgegeben.

Aus diesen beiden Kennzahlen folgt nach obiger Richtformel, dass die Anzahl der Units im Hidden Layer ungefähr zwischen 1 und 2 liegen sollte.

Leider erwies sich diese Faustregel (wie Abbildung 6.5 zeigt) als für den konkreten Anwendungsfall unrealistisch. Die Fehlerrate ist bei nur einer Hidden Unit wesentlich größer als bei fünf oder mehr Hidden Units (siehe Kapitel 6.9, Seite 56).

Eine weitere Möglichkeit, die Topologie eines neuronalen Netzes zu gestalten, findet sich in der Anzahl der Hidden Layer. Unter Bedachtnahme auf die oben dargestellte Gleichung 6.4 bleibt hier jedoch wenig Spielraum. In der Tat brachte die Variation der Anzahl der Hidden Layer erwartungsgemäß kaum Verbesserungen (siehe Abbildung C.9, Seite 113, und Abbildung C.13, Seite 117).

6.6 Verhinderung von „*Overfitting*“[4]

Ein neuronales Netz *verallgemeinert* gut, wenn es nach der Lernphase neue, unbekannte Eingabemuster (also solche, die nicht zur Trainingsmenge gehören) korrekt abbildet. Diese Fähigkeit zur Verallgemeinerung geht jedoch verloren, wenn das Netz zu lange trainiert wird: Es gibt nun zwar eine sehr gute Anpassung an einen kleinen

[4] *Overfitting*, engl: Überanpassung

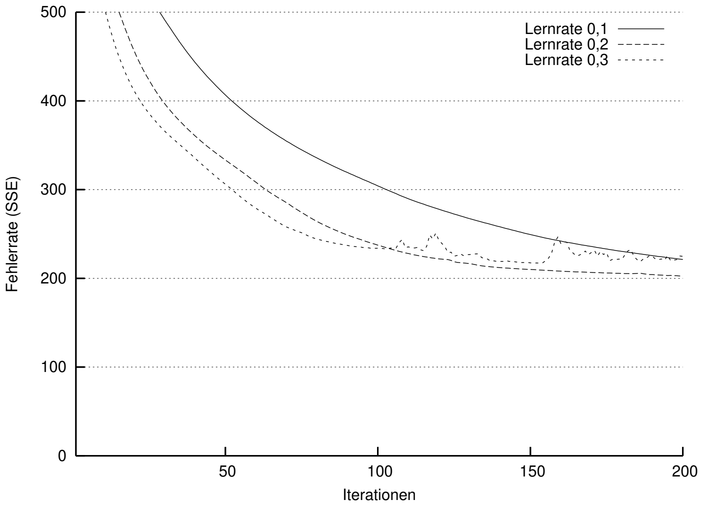


Abbildung 6.2: Backpropagation Momentum mit nur einer Hidden Unit

Weltausschnitt, eine korrekte Reaktion auf neue Daten ist aber nicht mehr gegeben. In dieser Situation – man spricht von *„Overfitting"* – ist der Fehler der Testdatenmenge bei weitem größer als der Fehler der Trainingsdatenmenge.

Abbildung 6.3 zeigt das typische Verhalten des Fehlers der Trainingsdaten und des Fehlers der Testdaten.[5] Es ist deutlich zu erkennen, dass der Fehler der Trainingsdaten asymptotisch abnimmt, während der Fehler der Testdaten ab einer bestimmten Stelle wieder zunimmt. Ab diesem Lernschritt beginnt das Netz „auswendig zu lernen". Durch rechtzeitiges Abbrechen des Trainings kann dieser Effekt vermieden werden.

Neben dem rechtzeitigen Beenden der Lernphase gibt es zwei weitere Methoden, um den Generalisierungsfehler zu reduzieren. Beide beruhen darauf, dass Overfitting ein Resultat der vielen Freiheitsgrade (nämlich der Gewichte) eines neuronalen Netzes ist und die Anzahl dieser Freiheitsgrade möglichst (relativ) klein gehalten werden sollte.

1. Eine Reduktion der Anzahl der Gewichte eines neuronalen Netzes erfolgt durch die Reduktion der Anzahl der Units im Hidden Layer. Dies ist im konkreten

[5] Es handelt sich bei diesem Beispiel nicht um konkrete Berechnungen; vielmehr wurden lediglich Daten für eine anschauliche Präsentation herangezogen.

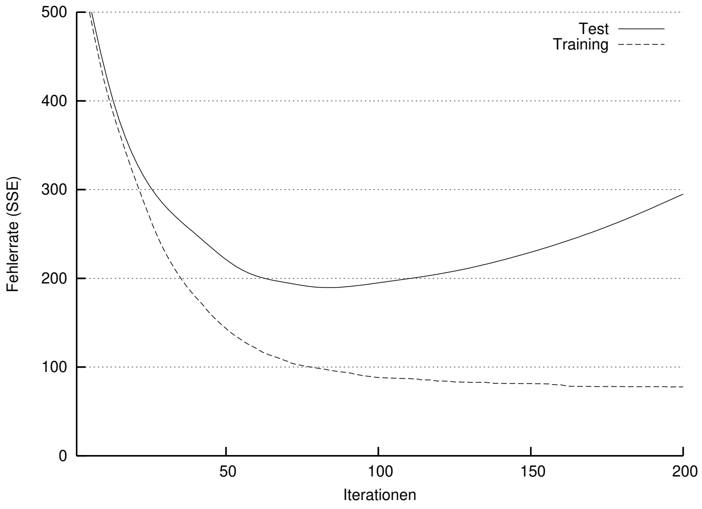


Abbildung 6.3: *Overfitting*

Anwendungsfall nicht mehr möglich, da die durch Testläufe ermittelte Anzahl an Hidden Units ohnehin ein Optimum darstellt.

2. Erhöht man die Anzahl der Lernbeispiele, so reduziert man damit ebenfalls die relative Anzahl der Freiheitsgrade. Da die Anzahl der Trainingsdaten mit den Nachrichten der *APA* aus den Monaten Jänner und Februar 1999 fest vorgegeben ist, sind hier keine Änderungen möglich.

Aufgrund der vorgegebenen Eckdaten (vor allem die Anzahl der zur Verfügung stehenden Nachrichtentexte der *APA*) bleibt als einzige Möglichkeit zur Verhinderung von Overfitting das rechtzeitige Beenden des Trainings.

6.7 Jogging[6] Weights

Gelegentlich steckt das Netzwerk während des Lernens in einem lokalen Minimum. In solchen Fällen ist es ratsam, die Kantengewichte zufällig geringfügig zu ändern. Dabei wird während des Iterationsvorgangs in regelmäßigen Abständen *„zufälliges Rauschen"*[7] auf den Gewichten erzeugt.

[6] *to jog about*, engl.: hin und her gerüttelt werden, durchschütteln
[7] *„random noise"*

Es ist hierbei jedoch wichtig, behutsam vorzugehen: Bereits erlerntes Wissen darf nicht wieder zerstört werden. Deshalb sollte der Grad der Rauscherzeugung mit zunehmender Anzahl der Iterationen (und damit zunehmendem Wissen) reduziert werden. Die folgende Abbildung 6.4 zeigt die zerstörerischen Auswirkungen zu starken Rauschens bei ansonsten identischen Parametern.[8]

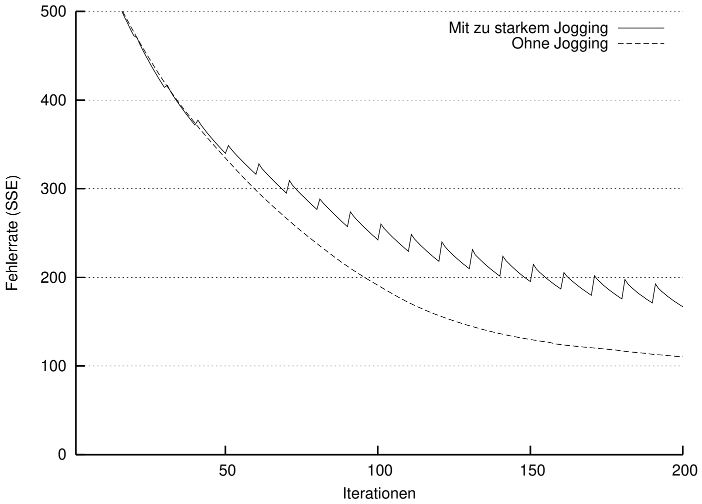


Abbildung 6.4: Fehlerrate mit zu starkem Jogging und ohne Jogging

Im konkreten Anwendungsfall stellt sich heraus, dass der Unterschied zwischen Lernen mit zufälliger Gewichtsveränderung (Abbildung C.4, Seite 108) und ohne (Abbildung 6.5, Seite 58) äußerst gering ausfällt. Es wird daher im Weiteren auf die Verwendung von *Jogging Weights* verzichtet.

6.8 Berücksichtigung der Vortage

Standen in den vorangehenden Abschnitten Verbesserungen des neuronalen Netzes an sich im Mittelpunkt, so widmet sich dieses Kapitel nun der Vergrößerung des Lernerfolges durch Optimierung der Trainingsdaten, indem zusätzliche Input-Units für die Veröffentlichung ähnlicher Artikel an Vortagen geschaffen werden. In Kapitel 5.4,

[8]Backpropagation Momentum ohne Berücksichtung der Vortage, 1 Output-Unit, Lernrate $\alpha = 0.1$.

Seite 43, wurde gezeigt, dass die veröffentlichten Artikel untereinander ähnlicher sind als Artikelmengen, die zum Teil nicht veröffentlicht wurden.

Die Ähnlichkeit anderer Artikel wurde dabei nach folgendem Algorithmus für das Training des neuronalen Netzes aufbereitet:

1. Suche die zehn ähnlichsten Artikel des aktuell betrachteten Artikels an einem bestimmten Vortag.
2. Zähle, wieviele davon veröffentlicht wurden.
3. Normiere die Anzahl der gezählten veröffentlichten Artikel.
4. Wiederhole ab Schritt 1 für andere Vortage.

Die für jeden Vortag ermittelte Anzahl der veröffentlichten Artikel stellt einen Wert für jeweils eine zusätzliche Input-Unit des neuronalen Netzes dar. Wie erwartet ergibt sich unter Einbeziehung der Veröffentlichung ähnlicher Artikel der Vortage ein besserer Lernerfolg – die Abbildungen im folgenden Kapitel und im Anhang ab Seite 106 zeigen die unterschiedlichen Lernerfolge bei Berücksichtigung von bis zu drei Vortagen.[9]

6.9 Training des neuronalen Netzes

Im Folgenden werden die Trainingsverläufe unterschiedlicher Netztopologien unter Berücksichtigung veröffentlichter ähnlicher Artikel der Vortage (siehe Kapitel 6.8, Seite 55) vorgestellt. Das Lernverfahren *Backpropagation Momentum*[10] wird dabei ausführlich mit verschiedenen Parametern variiert:

- Lernrate: $\alpha = 0{,}10$, $0{,}20$ und $0{,}30$
- Anzahl der Vortage: 0, 1, 2 und 3
- Anzahl der Hidden Units: 5, 10, 15 und 20 bzw. 6, 12, 18 und 24
- Anzahl der Output-Units: 1 oder 3

Die Lernerfolge sind im Großen und Ganzen sehr ähnlich, wenngleich sich der Trend abzeichnet, dass eine größere Anzahl von Hidden Units bei Fortschreiten des Trainings eine geringere Fehlerrate ergibt. Eine größere Anzahl von Hidden Units verstößt

[9]Abbildung 5.5 zeigte, dass die Ähnlichkeit der Dokumente bei mehr als drei Tagen relativ unverändert bleibt. Ein Informationsgewinn ist daher nur bei nicht sehr weit in der Vergangenheit liegenden Tagen zu erwarten.

[10]Weiters werden Berechnungen mit dem Verfahren *Backpropagation Weight Decay* und mit den zwanzig durch die Nachrichtenwert-Theorie definierten Kategorien durchgeführt. Ein Trainingsdurchgang mit vier Layern wird ebenfalls vorgestellt. Alle diese Verfahren und Ansätze bringen jedoch lediglich gleiche oder gar schlechtere Ergebnisse, weswegen hier nicht näher darauf eingegangen werden soll. Die grafische Darstellung dieser Lernverläufe und ausgewählte Berechnungen mit 1.075 Schlüsselwörtern finden sich im Anhang ab Seite 106.

jedoch gegen die in Kapitel 6.5.1, Seite 52, postulierte Faustformel, wonach die Anzahl der Trainingsdaten ungefähr zwei- bis viermal so hoch sein sollte wie die Anzahl der Verbindungen in einem neuronalen Netz.

6.9.1 Backpropagation Momentum mit einer Output-Unit

Die folgende Abbildung 6.5 zeigt den Lernverlauf eines neuronalen Netzes mit 1.044 Schlüsselwörtern (Input-Units) und einer Output-Unit, das mit dem Algorithmus *Backpropagation Momentum* (siehe Kapitel 4.3.3.4, Seite 37) trainiert wird. Bei den Berechnungen wurde die Anzahl der Hidden Units und die Lernrate unterschieden. Der für *Backpropagation Momentum* spezifische *Momentum Term* wurde dabei im Rahmen von hier nicht näher dargestellten Versuchen optimiert.

Allen Berechnungen ist gemeinsam, dass die Fehlerrate asymptotisch absinkt (mit einer größeren Lernrate rascher), sich aber immer bei einer Fehlerrate (SSE) von etwa 100 einpendelt.

Weitere Lernverläufe, die die Veröffentlichung von Artikeln an ein, zwei oder drei Vortagen berücksichtigt, sind im Anhang ab Seite 109 dargestellt.

6.9.2 Backpropagation Momentum mit drei Output-Units

Abbildung C.8, Seite 112, zeigt den Lernverlauf eines neuronalen Netzes mit 1.044 Schlüsselwörtern (Inputunits) und drei Output-Units, das mit dem Algorithmus *Backpropagation Momentum* (siehe Kapitel 4.3.3.4, Seite 37) trainiert wird. Als Sollwert der drei Output-Units wurde während des Lernens jeweils angegeben, ob die zugehörige (an den Input-Units vorliegende) Nachricht veröffentlicht wurde oder nicht. Das endgültige Ergebnis wurde durch eine 2-aus-3 Mehrheitsentscheidung errechnet.

Auch hier zeigt sich wieder, dass die Lernverläufe sehr ähnlich sind: Je größer die Lernrate ist, desto schneller ist die Untergrenze der Fehlerrate (SSE) bei einem Wert von etwa 100 erreicht.

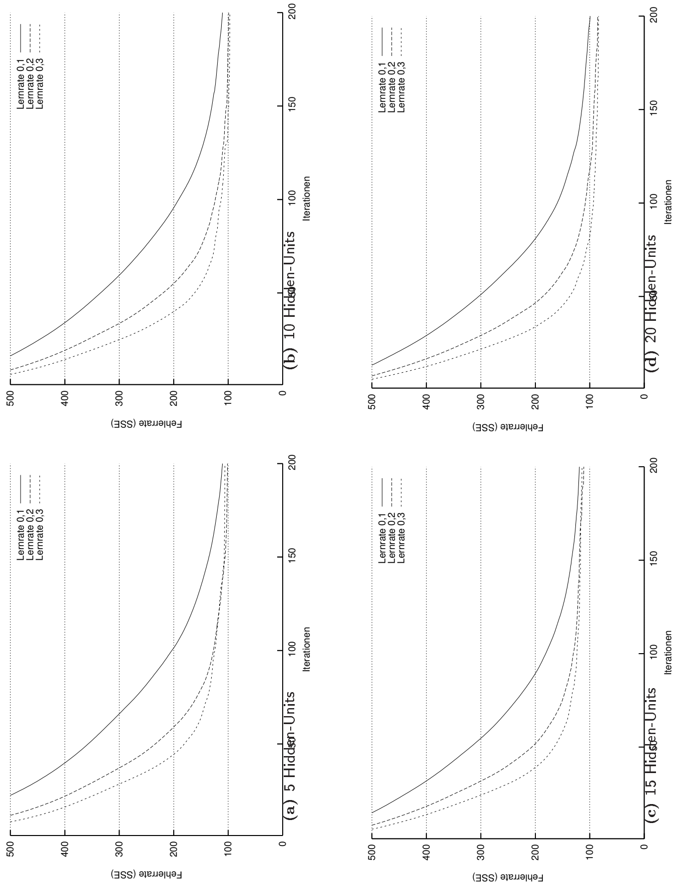


Abbildung 6.5: Backpropagation Momentum ohne Berücksichtigung der Vortage, 1 Output-Unit, 1.044 Schlüsselwörter

6.9.3 Backpropagation Momentum mit vier Layern

Abbildung C.9, Seite 113, zeigt den Lernverlauf eines neuronalen Netzes mit 1.044 Schlüsselwörtern (Input-Units), 10, 15 oder 20 Units im ersten Hidden Layer, 5 Units im zweiten Hidden Layer und einer Output-Unit, das mit dem Algorithmus *Backpropagation Momentum* (siehe Kapitel 4.3.3.4, Seite 37) trainiert wird.

Die Units zwischen den beiden Hidden Layern sind dabei nicht vollverbunden, sondern sektorisiert: Das bedeutet, dass der erste Sektor im ersten Hidden Layer *nur* mit der ersten Unit im zweiten Hidden Layer verbunden ist usw.

6.9.4 Backpropagation Weight Decay

Abbildung C.10, Seite 114, zeigt den Lernverlauf eines neuronalen Netzes mit 1.044 Schlüsselwörtern (Input-Units) und einer Output-Unit, das mit dem Algorithmus *Backpropagation Weight Decay* (siehe Kapitel 4.3.3.5, Seite 37) trainiert wird.

Die Fehlerrate (SSE) ist im Vergleich zum Training mit dem Algorithmus *Backpropagation Momentum* sehr hoch; zu beachten ist jedoch, dass sich die Fehlerrate der drei Lernraten nicht *einem* Grenzwert annähert, sondern jede Lernrate sozusagen ihren „eigenen" Grenzwert hat.

6.9.5 Training mit zwanzig Kategorien

Abbildung C.11, Seite 115, zeigt den Lernverlauf eines neuronalen Netzes mit 1.044 Schlüsselwörtern, die allerdings in die zwanzig Kategorien („Nachrichtenfaktoren") der Nachrichtenwert-Theorie (siehe Kapitel 2.4, Seite 7) zusammengefasst sind. Leider zeigt auch diese Methode ein sehr schlechtes Ergebnis (es musste sogar die Skalierung geändert werden, um den Lernverlauf darstellen zu können).

Die Ursache für dieses unbefriedigende Ergebnis liegt wohl darin begründet, dass annähernd *jeder* Nachrichtenfaktor in jedem Dokument vorkommt, weswegen daraus kaum die für die Beurteilung der Veröffentlichungswürdigkeit notwendige Information gewonnen werden kann: Die Definition der Eigenschaften guter Keywords (Kapitel 3.5, Seite 19) geht davon aus, dass Schlüsselwörter sowohl innerhalb eines Dokuments wie auch innerhalb aller Dokumente einzigartig sind. Genau dies trifft auf die Nachrichtenfaktoren eben nicht zu.

6.10 Test des trainierten neuronalen Netzes

Wie bereits in Kapitel 6.1, Seite 49, beschrieben, erfolgt der Test eines trainierten neuronalen Netzes mit Hilfe von dem Netz unbekannten Artikeln, die zu klassifizieren

sind. Das Ergebnis wird mit der „Realität" (Ergebnis der Zugehörigkeitsfunktion $\varphi(\mathfrak{a}_i)$) verglichen.

6.10.1 Qualität der Klassifikation – *Precision* und *Recall*

Precision und *Recall* sind zwei allgemein gebräuchliche Maßzahlen, um die Güte der automatisierten Klassifikation von Daten zu bestimmen. Grundlage dieser Maßzahlen sind die in Tabelle 6.1 dargestellten direkt messbaren Werte (vgl [Sebastiani (2002), S. 40f.]).

		Der Standard	
		veröffentlicht	nicht veröffentlicht
Trainiertes Netz	veröffentlicht	TP	FP
	nicht veröffentlicht	FN	TN

TP ... *true positives* FP ... *false positives*
FN ... *false negatives* TN ... *true negatives*

Tabelle 6.1: Klassifizierung der Treffer des trainierten Netzes

Die *Precision* π (oder Relevanzquote oder Genauigkeit) gibt den Prozentsatz der korrekt als publiziert klassifizierten Daten an. Die Berechnung erfolgt durch den Quotienten aus der Anzahl der korrekt als publiziert erkannten Nachrichten und der Anzahl aller als publiziert erkannten Nachrichten.

$$\pi = \frac{TP}{TP + FP} \tag{6.5}$$

Recall ρ (oder Nachweisquote oder Vollständigkeit) wiederum ist der Prozentsatz jener publizierten Nachrichten, die korrekt als publiziert erkannt wurden und sich aus dem Quotienten der Anzahl der korrekt als publiziert erkannten Nachrichten und der Anzahl aller publizierten Nachrichten errechnet.

$$\rho = \frac{TP}{TP + FN} \tag{6.6}$$

6.10.2 Auswahl geeigneter trainierter neuronaler Netze

Es wurden jene Topologien und Parameter ausgewählt, die sich während des Trainings (siehe vorangehendes Kapitel 6.9) als günstig im Sinne einer geringen Fehlerrate erwiesen.

- Backpropagation Momentum mit 15 Hidden Units und einer Output-Unit unter Berücksichtigung von zwei Vortagen; Lernrate $\alpha = 0.3$, Momentum Term $\mu = 0.1$ (Abbildung 6.6).

- Backpropagation Momentum mit 24 Hidden Units und drei Output-Units unter Berücksichtigung von drei Vortagen; Lernrate $\alpha = 0.3$, Momentum Term $\mu = 0.1$ (Abbildung C.14).
- Backpropagation Momentum mit vier Layern (15 Hidden Units im zweiten und 5 Hidden Units im dritten Layer) ohne Berücksichtung von Vortagen; Lernrate $\alpha = 0.3$, Momentum Term $\mu = 0.1$ (Abbildung C.15).

Testweise wurden auch trainierte Netze mit schlechterer Fehlerrate für die Artikelselektion herangezogen. Diese brachten aber ähnliche Ergebnisse wie die drei oben ausgewählten Topologien, sodass hier nicht näher darauf eingegangen wird.

6.10.3 Ergebnis der Testläufe

Die folgende Abbildung 6.6 zeigt die während des Trainings (1) bzw. die während des Tests (2) errechnete Fehlerrate (SSE) bei jedem Iterationsschritt. Weiters sind die Werte *Precision* π (3) und *Recall* ρ (4) dargestellt (in Prozent).[11]

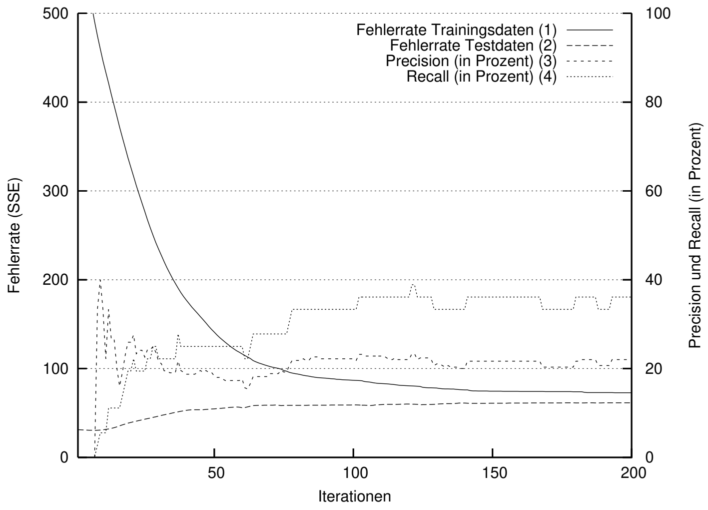


Abbildung 6.6: Test des trainierten neuronalen Netzes

[11] Die Abbildungen der beiden Testläufe mit 24 Hidden Units und mit vier Layern sind in Anhang C.3, Seite 118, dargestellt.

Das Ergebnis[12] aller drei trainierten Netze ist äußerst unbefriedigend und letztlich sogar widersprüchlich: Die Genauigkeit (*Precision*) eines trainierten Netzes ist dann am höchsten, wenn das Netz noch fast gar nicht trainiert wurde (also bei den ersten Iterationen). Eine Precision von rund 20 % bedeutet, dass das trainierte neuronale Netz bei den durch das Netz als publiziert klassifizierten Artikeln viermal mehr Nachrichten falsch denn richtig klassifizierte!

Ähnliches gilt bei der Maßzahl *Recall*: Von den durch den STANDARD publizierten Artikeln wurden nicht einmal 40 % korrekt klassifiziert.[13]

6.11 Analyse der Testergebnisse

Wie sich im vorangehenden Kapitel zeigte, kann das trainierte neuronale Netz kaum die gestellten Anforderungen erfüllen. Die Klassifizierung eines Artikels nach seiner Veröffentlichungswürdigkeit scheint überspitzt formuliert mehr dem Zufall denn zielorientierten Handelns zu folgen. Die vermutlichen Gründe für dieses Scheitern sollen im Anschluss kurz erörtert werden.

6.11.1 Zu geringe Anzahl der Trainingsdatensätze

Gemäß den Ausführungen zur Anzahl der Hidden Units in Kapitel 6.5.1, Seite 52, sollte die Anzahl der Trainingsdaten rund der zwei- bis vierfachen Anzahl der Verbindungen in einem neuronalen Netz entsprechen.

Bei einer Anzahl von beispielsweise zehn Hidden Units und der fest vorgegebenen Anzahl von 1.044 Keywords (also Input-Units) ergeben sich mehr als 10.000 Verbindungen im neuronalen Netz, wodurch mehr als 20.000 oder gar 40.000 Trainingsdatensätze notwendig wären. Leider stehen nur die 6.112 Artikel des ersten Quartals 1999 zur Verfügung, wovon ein Drittel für den Test des Netzes mit ihm unbekannten Daten vorgesehen ist. Geht man von den 6.000 Artikeln des ersten Quartals 1999 aus, müsste man über mehr als 1,5 Jahre hindurch alle Artikel manuell klassifizieren, um sie so für das Training aufzubereiten! Allerdings führt dies aufgrund der größeren Anzahl von handelnden Personen und behandelten Themen wiederum zu einer Steigerung der Schlüsselwörter, wodurch noch eine wesentlich größere Trainingsdatenmenge notwendig wäre.

Eine Reduktion der Keywords, damit der Input-Units und damit wiederum der Anzahl der notwendigen Trainingsdatensätze erscheint wenig zielführend, da der in den Artikeln verwendete Wortschatz sehr groß ist – man denke nur an die Anzahl der handelnden Personen und deren Ämter bzw. Organisationen.

[12] Die Sprünge bei Precision und Recall resultieren aus der relativ geringen Anzahl der Artikel.

[13] Die Ergebnisse der beiden anderen trainierten Netze unterscheiden sich nur marginal von diesem schlechten Ergebnis.

Für zukünftige Arbeiten bestünde die Möglichkeit der Reduktion des Vektorraums durch die *„Principal Component Analysis“ (PCA)*. Die PCA ist eine lineare Abbildung, die einen hochdimensionalen Raum unter Beibehaltung größtmöglicher Varianz auf einen Raum von wesentlich geringerer Dimensionalität reduziert. Dadurch könnte die Anzahl der Input-Units ohne nennenswerten Informationsverlust verringert werden.

6.11.2 Grundsätzliche Lösbarkeit des Problems?

Die beiden folgenden Tabellen 6.2 und 6.3 von Schlüsselwörtern entstammen den beiden in Anhang D, Seite 119, vorgestellten *APA*-Artikeln. Beide Artikel sind ungefähr gleich lang; der eine wurde im STANDARD veröffentlicht, der andere nicht.

berichtet	Berufung	bevorstehenden	Bundesgeschäftsführer
Bundeskanzler	Fehler	FPÖ	FPÖ-Stimmen
Frauen	frauenfeindliche	Frauenstimmen	Generalsekretär
Generalsekretariat	Haider	harte	Hilfe
Klima	Nachrichtenmagazin	Nationalrat	Nationalratswahlen
neue	nicht	Obmann	Profil
Prozent	Rudas	sicher	Spitze
SPÖ	Themen	vielfältigen	Wahl
Wahlkampf	wahlkämpfen	weniger	Westenthaler
Wien	Zielgruppe	Zitatensammlung	

Tabelle 6.2: Schlüsselwörter eines unpublizierten Artikels

arbeiten	bereits	Forschung	Graz
heimischen	Hochschulen	Hochschulsektion	Inkrafttreten
Innsbruck	insgesamt	Jahres	Jahresbeginn
Kraft	Leiters	mehr	neue
nicht	Organisationsrecht	Peter Skalicky	Reform
Rektor	Salzburg	sondern	startete
trat	TU Wien	Unis	Universität
Universitätsorganisationsgesetz	UOG	Vizerektor	Wien
Wissenschaftsministerium	zugleich		

Tabelle 6.3: Schlüsselwörter eines publizierten Artikels

Es zeigt sich, dass allein das Vorhandensein bestimmter Wörter nicht genügt, um als Mensch die Entscheidung zu treffen, ob ein Artikel veröffentlicht wird oder nicht. Ein erfahrener Journalist verfügt über „Weltwissen“, um diese Entscheidung treffen zu können. Tatsächlich bringt das neuronale Netz bessere Ergebnisse, wenn der Veröffentlichungsstatus der Vortage – sozusagen als kleiner Teil des Weltwissens – miteinbezogen wird.

6.11.3 Kritik an der Nachrichtenwert-Theorie

Vor allem Kapitel 6.9.5, Seite 59, hat deutlich gezeigt, dass die bloße Anwesenheit der zwanzig Nachrichtenfaktoren in den Artikeln nicht ausreicht, diese Artikel als veröffentlichenswert zu klassifizieren. Dies lässt den Schluss zu, dass die Nachrichtenwert-Theorie zwar notwendige, nicht aber hinreichende Kriterien für die Veröffentlichungswürdigkeit von Nachrichten beschreibt.

In der Tat definiert die Nachrichtenwert-Theorie zwar die einzelnen Nachrichtenfaktoren, trifft aber keinerlei Aussage über deren Gewichtung – genau dies ist die Domäne des erfahrenen Journalisten, der die für die Rezipienten des Mediums interessanten Nachrichten auswählt. Zwar müsste ein neuronales Netz einen Teil dieser journalistischen Erfahrung während des Trainings erlernen können, doch wird ein nicht unerklecklicher Teil dieses „Weltwissens“ den in dieser Diplomarbeit betrachteten neuronalen Netzen und Netztopologien verschlossen bleiben.

7 Zusammenfassung

Ziel dieser Diplomarbeit ist, auf Basis der aus der Publizistik und Kommunikationswissenschaften stammenden Nachrichtenwert-Theorie Werkzeuge zu schaffen, die Journalisten eine Vorselektion veröffentlichenswerter Artikel einer Nachrichtenagentur (wie beispielsweise der *APA*) ermöglichen.

Der erste Teil dieser Arbeit befasst sich daher mit den theoretischen Grundlagen sowohl auf kommunikationswissenschaftlicher wie auch auf informatischer Ebene. In Kapitel 2 wird die Nachrichtenwert-Theorie ausführlich dargestellt: Neben verschiedenen Konzepten für die Theorie der Nachrichtenauswahl wird über die historische Entwicklung der Nachrichtenwert-Theorie der aktuelle Stand der Forschung herausgearbeitet und die einzelnen Nachrichtenfaktoren, zusammengefasst in sechs Nachrichtendimensionen, dargestellt.

Im darauffolgenden Kapitel 3 werden zuerst Eigenschaften natürlicher Sprache diskutiert, um die Problematik der automatisierten Texterkennung und -klassifikation zu beleuchten. Anschließend werden die Eigenschaften guter Schlüsselwörter und deren Auswahl beschrieben. Den Abschluss bildet das *Vector Space Model*, mit dessen Hilfe die Nachrichten repräsentiert werden und das die Grundlage für die Artikelselektion bildet.

Im Anschluss daran versucht Kapitel 2.5 erste Begriffe für die einzelnen Nachrichtendimensionen und Nachrichtenfaktoren zu bestimmen und diese allgemein zu umschreiben.

Kapitel 4 stellt schließlich die informatischen Grundlagen dieser Arbeit – neuronale Netze – vor. Nach einer kurzen Definition der Eigenschaften neuronaler Netze und der Darstellung der geschichtlichen Entwicklung wird eine Einteilung der verschiedenen Netztypen und -topologien getroffen. Daran anschließend wird die Informationsverarbeitung innerhalb des Netzes vorgestellt, um schließlich die verschiedenen Lernverfahren zu diskutieren.

Den Abschluss des ersten Teils bildet Kapitel 5, das sich mit der Charakterisierung der Nachrichten-„Daten“ auseinander setzt. Es zeigt verschiedene Statistiken über Wortverteilungen ebenso wie die Ähnlichkeiten (nicht) veröffentlichter Artikel am selben Tag bzw. an ein bis drei Vortagen. Zusätzlich werden die Artikel inhaltlich mit Hilfe einer *Self Organizing Map* klassifiziert, um zu zeigen, dass die gewählten Schlüsselwörter die Nachrichten ausreichend beschreiben und dadurch die Artikel unterschieden bzw. in Gruppen eingeteilt werden können („*Clustering*“).

Der zweite Teil dieser Arbeit widmet sich der konkreten Realisierung der Nachrichtenauswahl auf Basis der in den vorangegangenen Kapiteln diskutierten Grundlagen und bildet somit die „Brücke“ zwischen den Kommunikationswissenschaften und der Informatik. Kapitel 6 beschreibt schrittweise den Aufbau eines neuronalen Netzes und dessen Parametrierung, um ein für die Aufgabenstellung optimales, trainiertes

neuronales Netz zu erhalten. Leider stellt sich beim Test des trainierten Netzes heraus, dass ebendiese Aufgabenstellung nicht erfüllt werden kann – am Ende dieses Kapitels wird daher versucht, Gründe für dieses Scheitern zu finden und zu analysieren.

Damit hat sich im Rahmen der Experimente dieser Arbeit gezeigt, dass es mit den gewählten Mitteln und Methoden nicht möglich ist, mit Hilfe eines neuronalen Netzes jenes Weltwissen zu erwerben, über das Journalisten verfügen. Auch konnte der Inhalt von Artikeln nicht derart gewonnen werden, dass Aussagen über die Veröffentlichungswürdigkeit möglich sind.

So soll diese Arbeit Anregung sein, andere Methoden zu finden, mit deren Hilfe Artikel klassifiziert werden können, um damit Journalisten in ihrer Arbeit der Auswahl veröffentlichenswerter Artikel zu unterstützen. So könnte ein erster Verbesserungsansatz in der in Kapitel 6.11.1, Seite 62, kurz angerissenen *Principal Component Analysis* liegen. Eine weitere Methode der Textklassifizierung, die die Selektion von Artikeln auf Basis der Veröffentlichungswürdigkeit gewährleisten könnte und für große Merkmalsräume (meint: eine hohe Anzahl von Features) geeignet ist, stellen beispielsweise *Support Vector Machines*[1] dar.

[1] `http://svmlight.joachims.org/`

A Keywords

In den nachstehenden Tabellen werden alle Wörter angeführt, die der Auswahl der Artikel dienen.[1] Eine genaue Beschreibung findet sich in Kapitel 2.5 ab Seite 9.

Die Einteilung der einzelnen Keywords in die durch die Nachrichtenwert-Theorie definierten Kategorien ist nicht immer eindeutig möglich, weswegen sich hier durchaus einige Unschärfen ergeben: Beispielsweise passt das Feature „teurer“ sowohl in die Nachrichtendimension *Dynamik*, als auch in die Dimensionen *Konsonanz* und *Human Interest*. Die Bewertung erfolgt jedoch für alle Wörter gleichrangig, sodass die genaue Einteilung letztlich nicht von Belang ist.

A.1 Syntaktische Darstellungen

Die nachfolgenden Schlüsselwörter sind reguläre Ausdrücke (Regex), die direkt in den Nachrichtentexten der *APA* gesucht werden. Die hier verwendeten Metazeichen werden in der nachfolgenden Tabelle A.1 erläutert; eine ausführliche Beschreibung regulärer Ausdrücke findet sich beispielsweise in [Friedl (2003)].

Metazeichen	**Bedeutung**
( )	Gruppierung
?	0 oder 1 Vorkommen
*	Beliebige Anzahl von Vorkommen
\w	Wortbestandteil ([a-zA-Z0-9_])
[ ]	Zeichenklasse (ein Zeichen aus der Klasse)

Tabelle A.1: Metazeichen regulärer Ausdrücke

Die bei den einzelnen Schlüsselwörtern in Klammern angegebenen Zahlen stellen die 20 Nachrichtenfaktoren dar, zu denen die Schlüsselwörter zugeordnet wurden. Die Liste der Faktoren ist in Kapitel 2.4, Seite 7, zu finden.

[1] Da die Nachrichtentexte der *APA* aus dem Jahre 1999 stammen, wurden die Wörter nicht in die neue Rechtschreibung transkripiert.

A.2 Status

[aäÄ]rzt (3)	(Abdullah)?Öcalan (3)	Abgeordnete (2, 3, 19)
AK (2, 19)	AKW (6, 7)	(Albert)?Angerer (3)
(Alexander)?van der Bellen (3, 17)	(Alfred)?Dallinger (3)	Ampelkoalition (2, 7)
Amt (2)	(Andreas)?Khol (3, 17)	(Andreas)?Rudas (3)
\w*anwalt (3)	Anwälte (3)	Aschermittwoch (2, 16, 20)
ATA (2)	Ausland (1, 4, 5)	\w*ausschuß (2)
Bank Austria (2)	(Barbara)?Helige (3, 17)	(Barbara)?Prammer (3, 17)
Behörde (2)	\w*beirat (3)	Berlin (1, 4, 5)
(Bernhard)?Görg (3)	Bezirk (2, 4)	\w*bischof (3)
Bischöfe (3)	Bregenz (1, 4)	Brüssel (1, 4, 5)
Bund (2)	Bundesgeschäftsführer (2)	Bundeskanzler (3)
Bundesland (1, 2, 6)	Bundespräsident (3)	Bundesr[aä]t (2, 3)
Burgenl[aä]nd (1, 2, 6)	Bürgermeister (3, 4)	Caritas (2)
Caspar Einem (3)	\w*chef (3)	(Christof)?Zernatto (3, 17)
(Christoph)?Schönborn (3)	Creditanstalt (2)	CV-Verbindung (2)
(Cyriak)?Schwaighofer (3)	\w*delegation (2)	Deutschland (1, 4, 5)
Die Grünen (2)	Diözese (2, 4)	\w*direktion (2)
Direktor (3)	(Ed)?Fagan (3)	Einkommen (6)
Eisenstadt (1, 4)	(Elisabeth)?Gehrer (3, 17)	(Erwin)?Pröll (3)
Erzbischof (3)	(Erz)?diözese (2)	Europa (1, 4, 5)
europäisch (1, 4, 5)	Europaparlament (2)	(Ewald)?Stadler (3)
Experte (3)	Fachmann (3)	favori (3, 17)
FP (2)	FPÖ (2)	Fraktion (2)
(Franz)?Fischler (3, 17)	(Franz)?Schausberger (3, 17)	(Franz)?Vranitzky (3, 17)
(Franz)?Fiedler (3)	(Fritz)?Karmasin (3)	(Fritz)?Neugebauer (3)
(Fritz)?Verzetnitsch (3)	\w*führ (3)	Funktion (3)
Gemeinde (2)	General (3)	(Gerhard)?Hirschmann (3)

(Gerhard)?Kratky (3)
Gewerkschaft (2)
Graz (1, 4)
(Hannes)?Farnleitner (3, 17)
(Heinz)?Fischer (3, 17)
(Helmut)?Schüller (3)
(Hermann)?Groer (3)
Innsbruck (1, 4)
Italien (1, 4, 5)
(Jörg)?Haider (3)
\w*kammer (2, 19)
Kardin[aä]l (3)
(Karl)?Schlögl (3, 17)
Kärnt (1, 3, 6)
Klausur (2)
Klubobfrau (3)
Kommission (2)
\w*kongreß (2)
(Kurt)?Krenn (3)
Landesgeschäftsführer (3)
Landeshauptstadt (1, 4)
\w*leit (3)
Liberalen Forum (2)
Linz (1, 4)
Macht (1, 2, 3, 17)
(Maria)?Rauch-Kallat (3, 17)
(Martin)?Strutz (3)
(Michael)?Häupl (3)
national (1, 4, 5)

\w*gericht (2, 18)
Gipfel (2)
Gremium (2)
(Hannes)?Swoboda (3)
(Helene)?Partik-Pable (3)
(Helmut)?Zilk (3)
Hochschule (2)
Institut (2)
Jahnturnhalle (2, 16, 20)
(Josef)?Pühringer (3, 17)
\w*kandid (3)
(Karl)?Habsburg (3)
(Karl)?Stix (3)
Kirche (2)
Klerus (2)
Klubtagung (2)
kompeten (3, 17, 18)
Krankenkass (2)
Land (1, 4)
Landeshauptleute (2, 3)
\w*landtag (2)
Liberale (2)
Liberales Forum (2)
(Lore)?Hostasch (3)
(Madeleine)?Petrovic (3)
(Maria)?Schaffenrath (3)
medizin (3, 18)
\w*minist (3, 17)
Nationalrat (2)

\w*geschäftsführer (3)
GÖD (2)
Grüne (2)
(Heide)?Schmidt (3)
(Helmut)?Kukacka (3)
(Herbert)?Sausgruber (3, 17)
(Ingrid)?Korosec (3)
international (1, 5)
(Johannes)?Voggenhuber (3)
Justiz (2, 18)
Kanzler (3)
(Karl)?Öllinger (3)
(Karli Heinz)?Grasser (3)
Klagenfurt (1, 4)
Klubklausur (2)
Koalition (2)
\w*konferenz (2)
kurd (1, 5)
Länder (1, 4)
Landeshauptmann (3)
Landtagspräsident (3)
Liberale Forum (2)
LIF (2)
(Ludwig)?Adamovich (3)
Mandat (2, 3)
(Martin)?Bartenstein (3, 17)
(Michael)?Ausserwinkler (3)
Ministerrat (2)
Nationalräte (2, 3)

Nationalratspräsident (3)
Neujahrstreffen (2)
(Nikolaus)?Michalek (3, 17)
\w*obmann (3)
Opposition (2)
Österreich (1, 4, 5, 7, 18)
(Otto)?Habsburg (3)
Paris (1, 4, 5)
Parteivorsitz (2, 3)
(Peter)?Pelinka (3)
(Peter)?Skalicky (3)
PKK (2)
popul (3, 18)
Präsidiale (2)
Profi (3)
Rechnungshof (2)
Regierungsklausur (2)
renommier (2, 3, 17, 18)
Richter (3, 18)
(Rudolf)?Parnigoni (3)
Senior (18, 19)
Sondersitzung (2, 16)
Spitzenkandidat (3, 16)
St. Pölten (1, 4)
(Susanne)?Riess-Passer (3, 17)
Temelin (1, 18, 20)
Tirol (1, 2, 4, 6)
(Ulrike)?Lunacek (3)
Universität (2)

NATO (2)
New York (1, 4, 5)
ÖAAB (2)
OECD (2)
ORF (2)
österreichisch (1, 2, 4, 5, 7, 19)
ÖVP (2)
Parlament (2)
(Peter)?Ambrozy (3)
(Peter)?Pilz (3)
(Peter)?Ulram (3)
\w*polit (2, 3)
Porträt (3, 18)
Präsidium (2)
prominen (3)
\w*referent (3)
region (2, 4)
Republik (2, 18)
Ried (2, 16, 20)
Salzburg (1, 2, 4, 6)
Session (2)
Sozialdemokraten (2)
SPÖ (2)
Status (3, 17, 18)
Symposium (2)
(Theresia)?Haidlmayr (3)
Treffen (2)
Uni (2)
Unternehmen (2)

Neujahrsansprache (3, 18)
Niederösterreich (1, 2, 6)
Oberösterreich (1, 2, 6)
ÖGB (2)
\w*organis (2)
(Othmar)?Karas (3)
P[aä]pst (3)
Partei (2)
(Peter)?Kostelka (3)
(Peter)?Rosenstingl (3)
(Peter)?Westenthaler (3)
\w*polizei (2, 18)
Präsident (3)
Presse (2)
\w*rat (2, 3)
Regierung (2)
Rektor (3)
Ressort (2, 3)
(Rudolf)?Edlinger (3, 17)
Schweiz (1, 4, 5)
\w*sitzung (2)
SP (2)
\w*sprecher (3)
Steiermark (1, 2, 4, 6)
\w*system (2, 18)
(Thomas)?Klestil (3, 18, 17)
tschech (1, 4, 5)
universit (2)
(Ursula)?Stenzel (3, 17)

USA (1, 4, 5)	Vatikan (1, 4, 5, 18)	Verband (2)
(Viktor)?Klima (3, 17)	Vize (3)	Vizekanzler (3, 17)
(Volker)?Kier (3)	Volksanw[aä]lt (2, 3)	Volkspartei (2)
Vorarlberg (1, 2, 4, 6)	Vorbild (3, 18)	Vorsitz (3)
Vorstand (2, 3)	VP (2)	(Walter)?Meischberger (3)
(Waltraud)?Klasnic (3, 17)	(Wendelin)?Weingartner (3, 17)	(Werner)?Amon (3)
(Werner)?Fasslabend (3, 17)	Wien (1, 2, 4, 6)	(Wilhelm)?Molterer (3, 17)
(Wolfgang)?Gattringer (3)	(Wolfgang)?Petritsch (3)	(Wolfgang)?Schüssel (3, 17)
World Vision (2, 18)	Zentrum (17, 18)	

Tabelle A.2: Keywords der Nachrichtendimension *Status*

A.3 Relevanz

Abgabe (6, 7)	Abwehr (15, 16)	Aggression (15, 16, 18)
alle (6, 7, 17, 19)	Ansprache (6, 16, 19)	attack (15, 16, 18)
auffällig (6, 9, 17)	auswirk (6, 7, 19)	best (6, 15, 16, 17, 18)
Bevölkerung (6, 7, 18, 19)	Bürger (6, 7, 18, 19)	bürokrat (6, 7, 19)
deutlich (17, 18)	Einrichtung (2, 4)	Ergebnis (6, 16, 17, 18)
Finanzierung (6, 16, 17, 18)	Freiheit (6, 7, 17, 18, 19)	Gebühr (6)
global (1, 4, 5, 6, 7)	haupt (6, 18)	hauptsächlich (6, 18)
Heimat (4, 5, 6, 7, 18)	heimisch (4, 5, 6, 7, 18)	\w*höchst (6, 10, 11, 17, 19, 20)
(ins)?gesamt (6, 17)	intensiv (6, 16, 17, 20)	jedwede (6)
Konsequen (6, 15, 16)	\w*kontroll (6, 18)	lediglich (6, 17, 18, 19)
Manöver (4, 6, 15, 16, 20)	mindest (6, 17, 18, 19)	Nachteil (6, 16, 18, 20)
\w*niedrig (6, 11, 14)	\w*pflicht (6, 7, 19, 20)	primär (6, 16, 18)
radikal (6, 15, 16, 18, 20)	Recht (6, 7, 16, 18)	Relevan (6, 14, 16)
sämtlich (6, 17, 18)	SBT (4, 5, 6, 10, 12)	schlimm (6, 10, 11, 18, 20)
Schwerpunkt (6, 8, 12)	Semester (6, 10, 12, 13, 14, 19)	\w*steuer (6, 7, 13, 15, 16, 19, 20)

\w*studie (3, 6, 11, 16)	Tiefpunkt (6, 8, 11, 18, 20)	Transit (4, 6, 7, 12, 13, 16)
überwiegend (6, 11, 14)	umfass (6, 19)	(un)?bedingt (6, 15, 16)
(un)?entbehr (6, 15, 16, 17, 18)	untergeordnet (6, 14)	\w*veranstaltung (6, 8, 20)
verwalt (6, 7, 12, 13, 14, 15, 18, 19, 20)	vorrang (6, 16)	Vorteil (6, 14, 17)
Wehrpflicht (6, 7, 12, 13, 18, 19, 20)	wichtig (6, 17, 18)	zahlen (6, 8, 9, 18, 20)
zentral (4, 6)	zig (6)	

Tabelle A.3: Keywords der Nachrichtendimension *Relevanz*

A.4 Dynamik

abseits (9, 11)	(ab)?sinken (9, 10, 11, 14)	ankünd (8, 13, 16)
Anlaß (8, 9, 13)	aufhorchen (9, 10, 11, 16, 20)	äußerst (6, 10, 11, 14, 17, 19)
Aussicht (9, 10, 11)	ausweit (8, 9, 10, 11)	bald (9)
Beginn (8, 9, 10, 11)	bevorstehen (8, 9, 10, 11)	blitzartig (8, 9, 11)
Detail (11, 13)	direkt (8, 14)	Durchbruch (8, 11, 17)
dynam (9, 11, 13, 14)	Einbruch (8, 9)	Einführung (8, 9, 11, 12, 13)
eklatant (9, 10, 11, 15, 16, 18, 20)	Ende (8, 9, 12)	endgültig (8, 9, 12)
endlich (9, 10, 11, 17)	enttäusch (9, 10, 11, 17, 20)	Ereignis (8, 9)
\w*erhöh (8, 9, 17, 18)	Eröffnung (3, 6, 8, 13, 17)	erreich (9, 11, 17)
erst (9, 11, 12, 13, 14)	Existenz (9)	extrem (6, 9, 10, 11, 17, 20)
fehlen (9, 10, 11, 17, 18)	flexib (9, 13, 14)	frisch (8, 9)
früh (8, 9, 10, 11)	gestiegen (9, 10, 11, 17)	gigant (9, 10, 11, 17)
\w*gr[uü]nd (8, 9, 17, 18, 19)	\w*harr (9, 11)	Idee (8, 9, 17)
Innovation (8, 9, 17, 18)	innovativ (8, 9, 17, 18)	(jeden)?fall (8, 9, 10, 11, 14)
j[uü]ng (6, 7, 9, 11, 12, 13, 14)	kaum (6, 9, 10, 14)	knapp (10, 11, 14)
Konkurrenz (14, 15, 16)	Kontinuität (12, 13, 14)	\w*kündig (8, 13, 16)
kurz (8, 11, 14)	lang (8, 11, 14)	längst (9, 10, 11, 13, 14)
Legislaturperiode (2, 3, 4, 6, 7, 8, 10–14)	letzt (8, 9, 11, 14)	Lösung (8, 11, 13, 14, 17)

maximal (6, 11, 17, 20)	mehr (6, 9, 10, 14)	minimal (6, 8, 11, 17, 20)
minus (9, 11, 17)	Modell (10, 11, 12, 14)	nachfolge (8, 9, 11)
Nachricht (8, 9, 11)	oben (9, 11, 17)	\w*offensiv (8, 9, 11, 15, 16)
prognos (9, 10)	rasch (8, 10, 11, 14)	\w*real (18)
reduzier (9, 11)	Regel (10, 11, 12, 14)	Schaffung (8, 11, 17)
schärfer (9, 11, 14, 15, 16, 20)	Schluß (8)	Schlußlicht (9, 11, 17, 18)
schnell (10, 11, 14, 17)	schon (8, 9, 11, 13, 14)	Schritt (8, 9, 11)
\w*schub (8, 9, 11, 12)	schwach (6, 9, 10, 14)	\w*senkung (8, 9)
setz (9, 11, 12)	\w*sicher (10)	Signal (8, 12, 14, 15, 16)
sofort (8, 17)	Stabilität (9, 12, 17, 18)	start (8, 9, 11, 12, 17)
steigen (10, 11, 12, 13, 14)	Steigerung (8, 10, 11, 12, 13, 14)	stop (8, 9, 11, 12)
Tagesmeldung (1–20)	Tendenz (10, 11)	teurer (11, 12, 14, 18, 20)
(un)?gewiss (10, 14)	untätig (10, 11, 15, 16, 18)	unten (9, 11)
unverrückbar (12, 14, 18)	(un)?vorhersehbar (9, 10, 11)	[Üü]berrasch (9, 10, 11)
verhalten (11, 15, 16)	verhärten (8, 11, 15, 16)	verschärfen (8, 11, 15, 16)
vorzeitig (8, 9, 11, 14)	wachsen (11, 12, 14)	wahrschein (10)
Wechsel (8, 9, 11)	weg (8, 9, 11)	weit (9, 11, 14)
Weiterentwicklung (8, 9, 11, 12, 13, 14)	wenig (11, 14)	wieder (12, 13, 14)
\w*wunde (9, 11, 15, 16, 18, 20)	(zu)? tief (9, 10, 11, 13, 14)	zuk[uü]nft (9, 10, 12, 14)
zurück (9, 11, 13, 14)	zus[aä]tz (9, 11, 13, 14)	Zuwachs (8, 9, 11, 13, 14)

Tabelle A.4: Keywords der Nachrichtendimension *Dynamik*

A.5 Konsonanz

Änderung (10, 11, 14)	bereits (12, 14)	bisher (12, 13, 14)
Entwicklung (10, 11, 14)	erneut (9, 10, 11, 13)	fortsetz (12, 13)
immer (12, 13)	Jahr (12)	künftig (10, 11, 13, 14)
laufend (12, 13)	nochmal (9, 10, 11, 12)	Situation (8, 12, 14)

stabil (12, 13)	ständig (12, 13)	täglich (8, 12)
Them (13, 16)	Trendwende (12, 14)	(un)?gewöhn (13, 14)
verringer (10, 11, 13, 14)	(wieder)?gestiegen (10, 11, 13, 14)	

Tabelle A.5: Keywords der Nachrichtendimension *Konsonanz*

A.6 Valenz

Abhilfe (17)	Absage (17)	abschaff (6, 16, 17, 18)
Abschied (6, 18, 20)	Abschluß (17)	absetz (15, 16, 17)
\w*absich (16, 17)	absolut (17, 18)	absurd (11, 15, 16)
\w*aktion (16, 17)	Alarm (9, 10, 11, 16)	Alternative (11, 16, 17)
Angriff (15, 16)	Anliegen (16, 17)	anreg (16)
anrüchig (20, 15, 16, 18)	Anspruch (16, 17)	Anstreng (16, 19)
Ansturm (16, 17, 19)	anti (15, 16)	\w*antrag (8, 15, 16)
Appell (15, 16, 18, 20)	\w*ärger (15, 16, 18, 20)	Argument (15, 16)
Armut (6, 7, 18, 20)	Aufarbeitung (13, 16, 17)	Aufgabe (16)
aufklär (11, 16, 17)	aufregend (15, 16, 17, 18, 20)	Aufregung (15, 16, 17, 18, 20)
\w*aufstockung (17, 18)	Aufwand (10, 11, 16, 18)	Augenmerk (16, 17)
Ausbau (16, 17, 18)	ausdr[uü]ck (15, 16, 18, 20)	Auseinandersetzung (15, 16, 20)
außerordentlich (8, 9, 10, 11, 15, 16, 18, 20)	\w*ausf[aä]ll (16, 18)	aus(ge)?zeich (18)
Ausmaß (6, 15, 16, 20)	Aussage (16)	ausschließ (16)
Aussendung (8, 9, 16)	Auszeich (18)	bedauer (15, 16, 20)
bedenk (16)	Bedingung (15, 16)	bedrohen (15, 16, 18, 20)
bedroht (15, 18)	Befugnis (3, 17)	begrüß (16)
Behauptung (16, 18)	Beitrag (18, 20)	Bekenntnis (18, 20)
\w*belastung (16, 18, 19, 20)	Bericht (16, 10, 11)	beschließen (2, 3, 16, 17, 18)
Beschl[ou]ß (2, 3, 16, 17, 18)	Beschwerde (15, 16, 19)	besonders (6, 16, 17, 18, 19)
\w*besser (6, 11, 17, 18)	bestmöglich (6, 16, 17, 18, 20)	beton (16, 20)

Beweis (16, 17, 18)
br[ae]nn (6, 15, 16, 20)
Chance (17, 18)
(da)?gegen (16)
dementier (15, 16)
Differenz (16)
Dorn (16, 20)
droh (15, 16, 20)
durchsetz (16, 17)
effizient (17, 18)
Einsatz (17)
Endphase (10, 11, 17)
entscheid (2, 16, 17, 18)
entzwei (15, 16)
Erfolg (17, 18)
Ers[ae]tz (10, 11, 16, 17)
exzellent (17)
Familie (6, 7, 17, 18, 19, 20)
feier (17)
Forschung (2, 18)
ganz (17, 18)
geeignet (17, 18)
gehörig (17, 18, 20)
gemeinsam (16, 18, 19, 20)
gering (16, 18)
Gewinn (17, 18, 20)
\w*günstig (17, 18)
Hälfte (17, 18)
Hindernis (15, 17, 18)

Blamage (16, 17, 18, 20)
brisan (6, 15, 16, 18, 20)
Chaos (15, 16, 18)
Debakel (17, 20)
Demokrat (5, 6, 7, 18, 19, 20)
\w*disku (16)
dräng (16)
Drohung (15, 16, 20)
echt (18)
(ein)?dring (6, 16, 20)
einzig (17, 18)
enorm (6, 10, 11, 16, 17, 18, 20)
Entschluß (16, 17)
erbärmlich (15, 16, 20)
erfolgreich (17, 18)
erschreckend (16, 18, 20)
Fach (2, 3, 17, 18)
\w*faschismus (6, 15, 16, 18, 20)
\w*forder (16)
Frage (16)
Garantie (18)
\w*gegner (15, 16)
gel[au]ng (18)
genau (18)
(ge)?senk (17)
glücklich (17, 18, 20)
gut (17, 18)
hart (15, 16, 18, 20)
hitzig (15, 16, 20)

Blockade (15, 16, 20)
Causa (16)
Chuzpe (15, 16, 18, 20)
Debatte (15, 16)
Dialog (16)
Disput (15, 16, 20)
drastisch (6, 10, 11, 15, 16, 20)
Druck (15, 16, 20)
\w*effekt (17)
\w*einflu (2, 3, 17, 18)
empör (15, 16, 20)
Entschärfung (16, 17, 18, 20)
entw[aeu]rf (16)
erbost (15, 16, 20)
ernstlich (6, 16, 19)
Experiment (18)
falsch (15, 16, 18)
Fehler (16, 17, 18)
Forderung (16)
fundament (16, 18)
Geburtstag (6, 18, 20)
geheim (18)
\w*geld (6, 7, 18)
gerecht (17, 18)
Gespräch (16)
Grundstein (17, 18)
\w*haft (15, 16, 17)
heftig (15, 16, 20)
hoch (17, 18, 20)

h[oö]he (17)
in Gefahr (16, 18, 20)
invest (17, 18)
Jugend (6, 18, 19)
kein (10, 11, 16, 18)
komplett (16, 17)
\w*konkret (10, 18)
konter (15, 16)
korrekt (16, 18)
Kreuzfeuer (15, 16, 20)
kritisier (15, 16)
liberal (18)
Maßnahme (17)
menschenverachtend (6, 15, 16, 18, 19, 20)
Mittelpunkt (16, 18)
Mrd (18)
Nachdruck (15, 16, 20)
Negativrekord (17, 18, 20)
nicht (16, 18)
notwendig (15, 16, 18)
optimal (10, 11, 17, 18)
passiv (10, 11, 16)
Platz (16, 17, 18)
Potential (17, 18)
\w*problem (15, 16)
Projekt (16, 17)
Prozent (17)
Querelen (15, 16)
Rekordwert (17)

ideal (17, 18)
Initiative (9, 10, 11, 16)
irrsinnig (15, 16, 20)
\w*k[aä]mpf (15, 16, 20)
[kc]ontra (15, 16)
Konflikt (15, 16)
Konsens (16, 17)
Kontroverse (15, 16)
korrupt (7, 15, 16, 18)
Kritik (15, 16)
leicht (11, 17)
lüg (15, 16, 18, 20)
Match (15, 16)
Mill (18)
möglich (10, 11, 16)
müss (16, 18)
nachhaltig (17, 18)
nein (16, 18)
Niederlage (15, 16, 17)
ober (17, 18)
Ordnung (18)
peinlich (15, 16, 20)
Position (16, 17, 18)
Praxis (18)
profit (17, 18, 20)
Propaganda (15, 16)
\w*prozeß (15, 16, 18)
rechtswidrig (15, 16, 18)
\w*reserven (18)

illegal (15, 16, 18, 20)
initiier (9, 10, 11, 16)
irrt[uü]m (10, 11, 16)
Kampagne (15, 16)
\w*kl[aä]r (16)
Konfrontation (15, 16)
konstruktiv (16)
Konzept (16, 17)
\w*kr[aä]ft (15, 16)
kritisch (15, 16)
\w*leistung (17, 18)
maßlos (15, 16, 20)
Meinung (16)
mißacht (15, 16, 18)
Motto (16, 17, 18)
m[uü]ß (16, 18)
negativ (16, 18)
neu (9, 10, 11, 16, 18)
nominier (16, 17)
objektiv (16, 18)
Packelei (15, 16, 18, 20)
Plädoyer (16, 18)
positiv (17, 18)
Prinzip (18)
\w*programm (17, 19)
Protest (15, 16, 20)
pur (17)
Rekord (17)
resign (17, 20)

richtig (17, 18)	riesig (17)	Rücktritt (15, 16, 20)
Rückzug (15, 16, 20)	Ruf (18)	Sanktion (15, 16, 20)
schaff (17)	\w*schaffen (17)	scharf (15, 16)
scheiter (15, 16, 17, 20)	Schlagwort (15, 16)	schlecht (15, 16, 18)
schlicht (16, 17)	Schlüsse (16, 17)	schön (18)
\w*schr[aä]nk (16, 18)	schroff (15, 16, 20)	Schußfeld (15, 16)
Schutz (16, 18)	schwer (17)	schwierig (17)
sehr (15, 16)	sehr überrascht (9, 10, 11, 16, 20)	senken (6, 16, 17)
Sieg (15, 16, 17)	Sinn (16, 18)	sinnvoll (16, 18)
skurril (9, 10, 11, 16, 18)	sonder (9, 18)	\w*st[aä]rk (17, 18)
stark (16, 17, 20)	Strafe (15, 16, 17, 18)	\w*streb (16, 18)
Streit (15, 16)	Streitthema (15, 16)	streng (16)
strikt (16, 20)	\w*stritt (15, 16, 20)	Summe (16, 17)
Tabu (18, 20)	Tisch (15, 16)	total (17, 18)
Triumph (16, 17, 20)	turbulent (10, 11, 15, 16)	überflüssig (15, 16)
überhaupt nicht (15, 16)	umfang (17)	umsetz (17)
umstritten (15, 16)	(un)?abhängig (18, 20)	(un)?absicht (15, 16, 18)
unangefochten (17, 18)	(un)?einig (15, 16)	(un)?fähig (15, 16, 20)
(un)?genügend (15, 16)	(un)?gerecht (18)	(un)?moralisch (18, 20)
(un)?Mut (10, 11, 16, 17)	(un)?qualifiziert (17)	(Un)?schuld (18)
Unsinn (15, 16, 18)	Unterstützung (16)	Untersuchung (15, 16, 18, 20)
(un)?üblich (15, 16, 18)	(un)?verantwortlich (15, 16, 18)	(un)?verzicht (16, 18)
(un)?wahr (15, 16, 18)	(un)?wirksam (17)	(un)?zufrieden (16, 18, 20)
\w*urteil (16, 17, 18)	[üÜ]bersch[uü] (16)	[Üü]berzeug (16, 17, 18)
vehement (16, 20)	\w*verantwort (15, 16, 18)	Verbesserung (17)
Verbrech (15, 18, 20)	verdammt (15, 16, 18)	verfahren (16, 20)
Verhandlung (16)	verhinder (15, 16)	verlang (15, 16)
verlieren (15, 16, 17)	verloren (15, 16, 17)	Verlust (15, 16, 17)
\w*vermögen (17, 18)	Vernunft (16, 18)	vernünftig (16, 18)

versagen (15, 16, 17)	verspr[ae]ch (16, 18)	verteidig (15, 16)
\w*vertr[aä]g (17, 18)	vertraulich (10, 11, 18, 20)	vertret (16)
verurteil (15, 16, 20)	verwerflich (15, 16, 18, 20)	Veto (15, 16)
viel (6, 11, 16, 17)	voll (6, 11, 16, 17)	völlig (6, 11, 16, 17)
\w*voraussetzung (16)	\w*vorhaben (17)	Vorreiterrolle (11, 16, 17)
Vorschl[aä]g (9, 11, 16)	Vorstellung (16)	Vorstoß (9, 11, 16)
vorw[aeiuü]rf (15, 16, 20)	vorwerfen (15, 16, 20)	wähle (18)
warn (16, 18)	weiche (16)	weisungsfrei (18, 20)
wert (16, 18)	wesentlich (16, 18)	widerruf (11, 16)
widerst[aä]nd (15, 16, 18)	wirk (17)	wirklich (17, 18)
\w*wissen (17, 18)	w[uü]nsch (16)	Zerstrittenheit (15, 16)
Ziel (16, 17, 18)	zurücktreten (8, 9, 10, 11, 15, 16, 20)	zurücktritt (8, 9, 10, 11, 15, 16, 20)
Zustimmung (16, 17)	Zwielicht (15, 16, 18)	

Tabelle A.6: Keywords der Nachrichtendimension *Valenz*

A.7 Human Interest

Abfangjäger (4, 15, 16, 19, 20)	Abfertigung (18, 19, 20)	abschieben (6, 15, 16, 19, 20)
\w*abstimm (19)	Abtreibung (6, 18, 20)	Affäre (15, 16, 18, 20)
Allgemeinheit (6, 7, 18, 19, 20)	Alpen (4, 6, 7, 18, 19, 20)	alt (6, 18, 20)
Angst (20)	\w*anschlag (15, 16, 18, 20)	Arbeit (6, 7, 18, 19, 20)
Arbeitslosigkeit (6, 19, 20)	Arbeitsplatzinitiative (6, 18, 19, 20)	Asyl (6, 15, 16, 18, 19, 20)
Ausbildung (18, 19, 20)	Ausländ (5, 6, 19, 20)	Auslieferung (18, 20)
Austritt (6, 15, 16, 18, 20)	Baccalaureat (17, 20)	Bakkalaureat (17, 20)
Basis (6, 19)	Bedürfnis (18, 20)	bedürftig (18, 20)
Belästigung (18, 20)	Beruf (6, 7, 17, 18, 19, 20)	Beschäftigung (6, 7, 17, 18, 19, 20)
Betr[aä]g (18, 20)	betreu (18, 19, 20)	betroffen (19, 20)
\w*bildung (6, 7, 17, 18, 19, 20)	blank (15, 16, 20)	brauch (4, 18, 19, 20)

\w*budget (6, 7, 15, 16, 19, 20)	Bundesheer (6, 19, 20)	Bundeshymne (4, 6, 18, 19, 20)
Defizit (6, 7, 15, 16, 19, 20)	Diversion (18, 20)	Drogen (18, 20)
Ehrlichkeit (18, 20)	\w*einfach (15, 16, 18, 20)	Einsparung (6, 19, 20)
emanz (18, 19, 20)	Emotion (20)	Entlastung (6, 17, 19)
ernst (15, 16, 20)	EU-Erweiterung (1, 2, 4, 6, 7, 8, 12, 13–20)	evangelisch (18, 19)
finanz (6, 18, 19, 20)	Flüchtling (6, 15, 16, 18, 19, 20)	\w*föderal (4, 6, 7, 18, 19, 20)
Förderung (17, 18, 19)	Frau (6, 7, 18, 19, 20)	freiwillig (17, 18, 20)
\w*freu (17, 20)	Friede (18, 19, 20)	Front (18, 19, 20)
Geburt (18, 20)	\w*gef[aä]hr (16, 18, 20)	Gehalt (6, 7, 17, 18, 19, 20)
\w*geiz (18, 20)	Gerechtigkeit (6, 18, 20)	\w*gesetz (6, 7, 18, 19, 20)
Gesundheit (6, 7, 18, 19, 20)	gewalt (15, 16, 18, 19, 20)	\w*gleich (18, 20)
Gleichbehandlung (18, 19, 20)	\w*grenz (18, 20)	gr[oö]ß (6, 9, 10, 11, 18, 19, 20)
harmoni (6, 18, 19, 20)	Heer (6, 7, 15, 16, 18, 19, 20)	Heeresgesetz (6, 7, 18, 19, 20)
Hilf (18, 20)	Historikerkommission (2, 3, 18, 20)	Holocaust (6, 7, 18, 19, 20)
Industrie (6, 7, 17, 18, 19)	Interesse (18, 20)	jüdisch (6, 18, 19, 20)
Karenz (6, 18, 20)	Karenzgeld (6, 18, 20)	Karenzgeld für alle (6, 16, 18, 20)
\w*katastroph (6, 9, 11, 19, 20)	katholisch (6, 18, 19, 20)	Kind (6, 18, 19, 20)
klein (6, 9, 10, 11, 18, 19, 20)	Konkordat (18, 19, 20)	\w*kosten (18, 20)
krank (19, 20)	Krankenkassenbeiträge (6, 7, 19, 20)	Krankenversicherung (6, 7, 19, 20)
Krieg (6, 15, 16, 18, 19, 20)	Krise (16, 18, 19, 20)	Kultur (6, 7, 18, 19)
(Kündigungs)?schutz (6, 7, 19, 20)	Landtagswahl (6, 7, 10, 12, 14, 19, 20)	\w*last (6, 18, 19)
Lawine (6, 19, 20)	Leben (6, 7, 19, 20)	Legalitätsprinzip (18)
Lohn (6, 7, 18, 19, 20)	M[aä]ngel (15, 16, 20)	Mandatsstand (17, 18)
Markt (6, 18, 19)	Mensch (6, 7, 18, 19, 20)	Miete (6, 7, 18, 19, 20)
Mifegyne (6, 18, 19, 20)	Militär (6, 19, 20)	Militärbefugnisgesetz (6, 19, 20)
mißbrauch (6, 15, 16, 18, 20)	Mißst[aä]nd (6, 15, 16, 18, 20)	\w*mittel (18, 19)
modern (18)	moral (18)	Nationalratswahl (6, 7, 10, 12, 14, 19, 20)
Neid (18, 20)	Neuwahl (6, 7, 9, 10, 11, 15, 16, 18, 20)	Not (19, 20)
Oberster Gerichtshof (2, 6, 17, 18, 20)	OGH (2, 6, 17, 18, 20)	ökologisch (18, 19, 20)

\w*[öÖ]ffent (6, 7, 18)	Opfer (6, 18, 19, 20)	panik (6, 20)
panisch (6, 20)	Paragraph (6, 18)	Patient (6, 19, 20)
Pension (6, 7, 18, 19, 20)	pfleg (20)	Pflichtmitgliedschaft (2, 6, 19, 20)
Politikerpension (3, 6, 15, 16, 20)	Preis (6, 18, 19, 20)	Pressestunde (2, 20)
privat (6, 18, 19, 20)	Privilegien (18, 20)	Privilegienabbau (18, 20)
Rechtsunsicherheit (6, 7, 18, 20)	\w*reform (6, 18, 20)	\w*reich (18, 20)
Revolution (15, 16, 18, 19, 20)	sch[aä]d (15, 16, 20)	Schilling (6, 7, 18, 19, 20)
Schubhaft (18, 20)	Schul (6, 18, 19, 20)	Schüler (6, 18, 19, 20)
Schwarzarbeit (18, 20)	Selbstbehalt (6, 7, 18, 19, 20)	Semesterticket (19, 20)
Semmering (4, 18, 20)	\w*seriös (18, 20)	Sex (6, 7, 19, 20)
Sicherheit (6, 18, 19, 20)	Skandal (18, 20)	Soldat (6, 19, 20)
sorg (18, 20)	sozial (6, 18, 19, 20)	Sozialpartnerschaft (6, 18, 19)
\w*spar (18, 20)	Staat (1, 4, 5, 6, 7, 18, 19, 20)	Steuerreform (6, 7, 18, 19, 20)
stolz (20)	streich (18, 20)	Student (6, 19)
Suchtgift (6, 18, 20)	Superwahljahr (6, 7, 10, 11, 12, 14–20)	\w*t[aä]t (18, 20)
Täter (18, 20)	terror (6, 15, 18, 20)	teuer (18, 20)
Tod (6, 20)	\w*tragödie (15, 18, 20)	\w*tunnel (18, 20)
überleb (6, 17, 20)	Umfrage (19)	Unbehagen (15, 16, 20)
Unfall (6, 20)	\w*unglück (6, 20)	(ver)?billig (18, 20)
Verbot (18, 19, 20)	Verd[aä]cht (18, 20)	Verfassung (6, 7, 18, 19)
Verfassungsgerichtshof (2, 6, 15, 16, 18, 20)	Verständnis (18)	Vertrau (18, 20)
Verwaltungsgerichtshof (2, 6, 15, 16, 18, 20)	VfGH (2, 6, 15, 16, 18, 20)	Volk (4, 5, 6, 7, 18, 19, 20)
Volksabstimmung (6, 7, 10, 11, 14–16, 19, 20)	Volksbefragung (6, 7, 10, 11, 14–16, 19, 20)	\w*volksbegehren (6, 7, 10, 11, 14–16, 19, 20)
Vorschrift (6, 19, 20)	VwGH (2, 6, 15, 16, 18, 20)	Wahl (6, 7, 10, 11, 12, 14, 15, 16, 17, 19, 20)
Wahlkampf (6, 7, 10, 11, 12, 14–20)	Wahlkampfauftakt (6, 8, 12, 14–16, 19, 20)	Wahltermin (6, 8, 10, 15, 16, 19, 20)
Weisung (6, 20)	weitverbreiet (6, 9, 10, 11, 18, 19, 20)	Welt (1, 4, 5, 6, 19)
Wirtschaft (6, 7, 18, 19)	wohn (6, 20)	\w*zahl (18, 19, 20)
Zuschuß (18, 19, 20)	zuversichtlich (20)	Zwangsmitgliedschaft (6, 15, 16, 19, 20)

Tabelle A.7: Keywords der Nachrichtendimension *Human Interest*

B Listings

In diesem Kapitel werden die Listings der in dieser Diplomarbeit verwendeten Algorithmen und Programme dargestellt. Hauptaugenmerk wird dabei auf die Implementierung des *Vector Space Models* gelegt, das gleich im Folgenden vorgestellt wird.

B.1 Vector Space Model

Das in Kapitel 3.6, Seite 20, ausführlich dargestellte Vector Space Model dient im Grunde der Zählung und Normierung der Anzahl der Vorkommnisse der für die Selektion eines Artikels relevanten Wörter und Wortgruppen. Nachfolgend werden einige Beispielinputdaten, die mit SNNS weiter verwendbaren Outputdaten und der Sourcecode selbst gezeigt.

Die Implementierung gliedert sich in vier grundlegende Teile, deren einzelne Funktion leicht im Sourcecode erkennbar ist, wohin der interessierte Leser für eine ausführliche Dokumentation verwiesen sei.

1. Zählung der einzelnen Terme (`Calculate_TF()`)
2. Ermittlung der Häufigkeit der Vorkommnisse der Terme in den Dokumenten (*document frequency*, `Calculate_DF()`)
3. Berechnung der inversen Termfrequenz ($tf \times idf$, `Calculate_DFxIDF()`)
4. Normierung des Gesamtvektors (`Calculate_TFxIDF_standardized()`)

Die in diesem Beispiel verwendeten Schlüsselwörter sind im Array `@arFeatures` gespeichert und werden durch die Funktion `ReadKeywords()` aus eigens für LaTeX formatierten Dateien gelesen, um im gesamten Projekt auf die *selben* Daten zugreifen zu können.

B.1.1 Inputdaten

Die Schlüsselwörter wurden hervorgehoben.

• `msg0001.txt`

Heute fällt ***sehr viel*** Regen.
Ein Neger mit Gazelle zagt im Regen ***nie***.
Fällt ***ganz viel*** Regen, führt dies zu ***sehr*** zagenden Gazellen.

• `msg0002.txt`

Diesmal wird es ein ***besonders*** schöner Abend, der voller Freude
von ***vielen***, ***vielen*** Menschen genossen werden sollte.
Besonders die am Abend mit ***viel*** Freude gesehene Fernsehserie
sollte unser Glück ***nie*** trüben.

• `msg0003.txt`

Auch dies ist ein ***sehr*** wichtiger Text. Er wird ***sehr***, ***sehr oft***
gelesen werden. Jedenfalls hoffe ich dies ***ganz besonders***; denn nur
so sind ***besondere*** Werte möglich, die beim Finden der Keywords ***sehr***
stark mithelfen werden.

• `msg0004.txt`

Judy Marshall wird keineswegs von den Geräuschen geweckt, mit denen
French Landing erwacht. Im Gegenteil, sie liegt bereits schon seit
drei Uhr mit starrem Blick wach, sucht den Schatten nach etwas ihr
Unbekanntem ab, flüchtet vor Träumen, die zu grässlich sind, um sich
an sie zu erinnern. Stephen King, Das schwarze Haus

• `msg0005.txt`

Besondere Momente im Leben ***besonders*** hervorzuhebender, jedoch leider nur
selten vorkommender und doch ***oft*** gesehener ***sehr*** großer Menschen finden
nur ***sehr*** selten statt. Hier kommt das Wort "***besondere***" vor.
Und hier stehen ***viele***, ***viele*** Wörter.

B.1.2 Veröffentlichungsstatus

In dieser Datei wird für jeden einzelnen Nachrichtentext gespeichert, ob die betreffende Nachricht veröffentlicht oder nicht veröffentlicht wurde. Dies stellt den Sollwert für das Training bzw. den Test des neuronalen Netzes dar. Die Daten werden von SNNS allerdings nicht direkt aus dieser Datei verwendet; vielmehr werden diese Daten mit Hilfe des später vorgestellten Programms `vsm.pl` für SNNS aufbereitet.

```
##########################################################################
# status.txt                                                             #
# Liste aller Meldungen mit einem Flag,                                  #
# ob diese Meldung veröffentlicht wurde.                                 #
# Thomas Pfeiffer, Oktober 2002 - Juni 2004                              #
##########################################################################

msg0001.txt 1 #
msg0002.txt 1 #
msg0003.txt 1 #
msg0004.txt 0 #
msg0005.txt 1 #
```

B.1.3 Trainingsdaten

In der von `vsm.pl` erzeugten Datei `TFxIDF.pat` werden die Trainingsdaten für das neuronale Netz gespeichert. Es handelt sich dabei konkret um die normierten Inputwerte für jede Input-Unit und den Sollwert (Veröffentlichungsstatus, siehe oben) für die Output-Unit(s).

```
################################################################################
# Anzahl der bearbeiteten Dokumente (Patterns): 5
# Anzahl der Features (Inputunits): 13
# (c) Thomas Pfeiffer, 9325691, Oktober 2002 - Mai 2004
################################################################################

SNNS pattern definition file V3.2
generated at Wed Jun  9 23:09:39 2004
No. of patterns : 5
No. of input units : 13
No. of output units : 1

################################################################################
# Features in alfabetischer Reihenfolge:
# "besondere", "besonderer", "besonders", "ganz viel",
# "ganz viele", "ganz vielen", "nie", "oft",
# "sehr", "viel", "viele", "vielen",
# "vieler",
################################################################################

# Inputpattern (Datei msg0001.txt) #1:
0.00000 0.00000 0.00000 1.00000 0.00000 0.00000 0.56932 0.00000
0.63479 0.63479 0.00000 0.00000 0.00000
# Outputpattern #1:
1

# Inputpattern (Datei msg0002.txt) #2:
0.00000 0.00000 0.31739 0.00000 0.00000 0.00000 0.28466 0.00000
0.00000 0.47609 0.56932 1.00000 0.00000
# Outputpattern #2:
1

# Inputpattern (Datei msg0003.txt) #3:
0.44844 0.00000 0.25000 0.00000 0.00000 0.00000 0.00000 0.44844
1.00000 0.00000 0.00000 0.00000 0.00000
# Outputpattern #3:
1

# Inputpattern (Datei msg0004.txt) #4:
0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000 0.00000
0.00000 0.00000 0.00000 0.00000 0.00000
# Outputpattern #4:
0

# Inputpattern (Datei msg0005.txt) #5:
1.00000 0.00000 0.27875 0.00000 0.00000 0.00000 0.00000 0.50000
0.55749 0.55749 1.00000 0.00000 0.00000
# Outputpattern #5:
1
```

B.1.4 Implementierung in *perl*

Das Programm `vsm.pl` erzeugt das Pattern-File (die Trainingsdaten) `TFxIDF.pat`, das direkt von SNNS für das Training des neuronalen Netzes verwendet werden kann. Weiters wird die Datei `nwt.net` erzeugt, die die Netztopologie und die (noch untrainierten) Kantengewichte enthält. Optional können verschiedene Statistiken (beispielsweise die Wortverteilungen, siehe Kapitel 5, Seite 39) und andere Dateien (Daten für *Self Organising Maps*, siehe Kapitel 5.5, Seite 45) generiert werden.


```
#!/usr/bin/perl
################################################################################
# vsm.pl                                                                       #
# Liest die als Einzelfiles vorliegenden Nachrichten, zählt die Vorkommnisse  #
# der Features je Dokument und gibt diese Daten als SNNS-Pattern aus.          #
# Als zusätzliches Feature können bei Bedarf die Keywords in Kopien der        #
# Nachrichten-Textfiles hervorgehoben werden.                                  #
# Thomas Pfeiffer, 9325691                                                     #
my($cszCopyrightDate) = "Oktober 2002 - Juni 2005";                            #
################################################################################

use DB_File;
use Data::Dumper;
use File::Basename;
use Getopt::Long;
use strict;
use warnings;

$| = 1;                                          # Autoflush für stdout et al. #

(defined $ENV{DADir}) or die "Do not know working directory!\n";

our ($bCreateDistances,
     $bCreateSOMFiles,
     $bCountKeywords,
     $bDeleteUselessWhitespaces,
     $bDemo,
     $bHelp,
     $bHighlightKeywords,
     $bNetworkOnly,
     $nOutputUnits,
     $bPrintKeywords,
     $bSelectKeywords,
     $bTestNetwork,
     $bUseNoWhitespaceFiles,
     $bVerbose) = (0,0,0,0,0,0,0,0,0,0,0,0);
$nOutputUnits = 1;

GetOptions("createdistances"           => \$bCreateDistances,
           "createsomfiles"            => \$bCreateSOMFiles,
           "deleteuselesswhitespaces"  => \$bDeleteUselessWhitespaces,
           "demo"                      => \$bDemo,
           "help"                      => \$bHelp,
           "highlight"                 => \$bHighlightKeywords,
           "network"                   => \$bNetworkOnly,
           "outputunits=s"             => \$nOutputUnits,
           "printkeywords"             => \$bPrintKeywords,
           "selectkeywords"            => \$bSelectKeywords,
           "testnetwork"               => \$bTestNetwork,
           "usenowhitespacefiles"      => \$bUseNoWhitespaceFiles,
           "verbose"                   => \$bVerbose) or Usage(1);

my  ($szHostname) = `hostname`; chomp $szHostname;
our ($cszDataDir)        = our ($cszStatusDir)      = our ($cszKeyDir) =
our ($cszPatternfileDir) = our ($cszNetworkfileDir) =
our ($cszGraphDir)       = our ($cszDistanceDir)    =
our ($cszSOMDir)         = $ENV{DADir};
```

```perl
$cszDataDir          .= ($bDemo) ? "demo/" :
                        ($bTestNetwork) ? "data/apa/test/" : "data/apa/train/";
$cszDistanceDir       = $cszDataDir . "distances/";
$cszStatusDir        .= ($bDemo) ? "demo/" :
                        ($bTestNetwork) ? "data/apa/test/" : "data/apa/train/";
$cszKeyDir           .= ($bDemo) ? "demo/" : "data/";
$cszPatternfileDir   .= ($bDemo) ? "demo/" : "trained/$szHostname/";
$cszNetworkfileDir   .= ($bDemo) ? "demo/" : "trained/$szHostname/";
$cszSOMDir           .= ($bDemo) ? "demo/" : "trained/$szHostname/";
$cszGraphDir         .= "graphs/";

our ($cszMsgFile)           = ($bDemo) ? "msg*.txt" :
                              ($bTestNetwork) ? "1999030[1-5]_APA*.txt " :
                                                "*APA*.txt";
our ($cszStatusFile)        = "status.txt";
our ($cszKeyFile)           = ($bDemo) ? "keywords_?.tex" : "keywords_?.tex";
#our ($cszKeyFile)          = ($bDemo) ? "keywords_?.tex" : "keywords20_05.tex";

our ($cszPatternfile)         = ($bTestNetwork)?"TFxIDF_test.pat":"TFxIDF.pat";
our ($cszNetworkfile)         = "nwt.net";
our ($cszSOMInputvectorfile)  = "input.som";
our ($cszSOMTemplatefile)     = "template.som";
our ($cszHighlightExt)        = "hl";
our ($cszNoWhiteSpaceExt)     = "nws";
our ($cszSelectExt)           = "sel";
our ($cszDistanceCosineExt)   = "dco";
our ($cszDistanceEuklidExt)   = "deu";
our ($cszSelectcountFile)     = "selectcounts1.csv";
our ($cszWordspreadFile)      = "selectcounts2.csv";

Usage(0) if ($bHelp);

&DeleteUselessWhitespaces(), exit(0) if ($bDeleteUselessWhitespaces);

our (@arStatistics) = ();
our ($dnAnzPublished) = 0;
our ($dnAnzUnpublished) = 1;

our ($cnMaxPatternsPerLine) = 8;

our (%asarNWTFaktoren);
our (@arFeaturesSorted) = ReadKeywords();
our ($cnFeatures) = $#arFeaturesSorted + 1;
#our ($cnFeatures) = scalar keys %asarNWTFaktoren;

our ($cnAnzHiddenunits1) = 15;
#our ($cnAnzHiddenunits2) = 5;
die "Wrong number of hidden units; died"
    if ($cnAnzHiddenunits1 % $nOutputUnits);

&PrintKeywords(), exit(0) if ($bPrintKeywords);      # Option --printkeywords #

our (%asarStatus)     = ReadStatus();
our (@arFiles)        = undef;
our (@arnVS)          = undef;
our (@arnDF)          = undef;
our ($cnDokumente)    = scalar keys %asarStatus;
our ($cnAllFeatures) = $cnDokumente*$cnFeatures;

my ($szTieFile_arnVS) = "/tmp/arnVS.array.pid$$";
tie (@arnVS, "DB_File", $szTieFile_arnVS, O_RDWR|O_CREAT, 0600, $DB_RECNO)
    or die "Cannot tie $szTieFile_arnVS, died";
```



```perl
our (@arTage); # Alle Tage sortiert, an denen Nachrichten publiziert wurden. #
our (%asarTage);              # Alle Tage mit Liste der Nachrichten des Tages. #
our (@arMsgFiles) = sort keys %asarStatus;          # Alle Nachrichten sortiert. #

my (%seen);
@arTage = sort grep { !$seen{$_}++ } map { &GetDayOfMsg($_) } keys %asarStatus;
foreach (@arMsgFiles) { push @{ $asarTage{&GetDayOfMsg($_)} }, $_; }

our ($nPrevDays) = 2;              # Anzahl der betrachteten vergangenen Tage. #
our ($nAnzDistances) = 10;     # Anzahl der betrachteten ähnlichsten Artikel. #

&Create_SNNS_Network(), exit(0) if ($bNetworkOnly);          # Option --network #
&HighlightKeywords(), exit(0) if ($bHighlightKeywords);# Option --highligh...#
&SelectKeywords(), exit(0) if ($bSelectKeywords);  # Option --selectkeywords #

&InitVectorSpace();
&Calculate_TF();
&Calculate_DF();
&Calculate_TFxIDF();
&Calculate_TFxIDF_standardized();
&CreateSOMFiles() if ($bCreateSOMFiles);
&Calculate_Kosinusdistanz() if ($bCreateDistances);
&Create_SNNS_Patternfile();
&Create_SNNS_Network() if (!$bTestNetwork);

untie (@arnVS);

exit 0;                                                        # Hauptprogramm #
################################################################################

################################################################################
# ReadKeywords                                                                 #
# Liest die Keyword-Dateien ein und bildet eine alfabetische Liste der         #
# Schlüsselwörter (einschließlich der Vorkommen in den NWT-Kategorien.         #
# Zu zerlegende Zeilen:                                                        #
# KW1 (1, 2) & KW2 (2, 3) & KW3 (1,3) \tabularnewline                          #
################################################################################
sub ReadKeywords
 {
   my($szFile);
   my(@arszKeywords) = ();
   my(%asarszKeywords) = ();

   &Message("Reading keywords ...\n");

   FILE:
   foreach $szFile (<$cszKeyDir$cszKeyFile>)
     {
       next FILE unless -T $szFile;

       if (!open(KEY, $szFile))
         {
           print "Cannot open $szFile; continuing ...\n";
           next FILE;
        }

       while (<KEY>)
         {
           next if /^%/;                       # Kommentarzeilen überspringen. #
           next if /^\s*$/;                        # Leerzeilen überspringen. #
           next if /^\\/;             # Zeilen mit LaTeX-Befehlen ignorieren. #
```

```
          s/\\tabularnewline//;
          s/ignore//g;
          while (/\s$/) { chop; }
                  # Aufbau eines Arrays, dessen Elemente wiederum ein Array #
                  # darstellen. Dieses zweite Array hat nur zwei Einträge:  #
                  # - das Keywort selbst und die Datei, in der es vorkommt. #
                  # - das Keywort selbst und ein Array mit den Faktoren.    #
          foreach (split /\s*&\s*/) # Schlüsselwörter einer Zeile trennen. #
            {
              # Schlüsselwort (in $1) und NWT-Kategorien (in $2) aufteilen. #
              /(.*)\s*\((\d+(, *\d+)*)\).*/;

              if (defined $2)
                {
                  my(@arKat) = split(/, */, $2);# NWT-Kategorien auftrennen.#
                  foreach (@arKat) { $asarNWTFaktoren{$_}++ }
                  my($x) = $1; while ($x =~ /\s$/) { chop $x; }
                  push @arszKeywords, [ $x, [ @arKat ] ];
                }
              else
                {
                  print "Warnung: Keine Kategorieren definiert: $_\n";
                  push @arszKeywords, [ $_ ];
                }
            }
        }

      close(KEY);
    }

  foreach (@arszKeywords)        # Features unabhängig von Groß-/Kleinschrei- #
    {                            # bung und Sonderzeichen sortieren.          #
      my(@K) = @{ $_ };

      next if ($K[0] =~ /^$/);                # Leere Keywords überspringen. #
      $K[0] =~ s/\(/\(?:/g;        # Gruppierungen ohne Capturing "(?: ...)" #
      $K[0] =~ s/\\\\/\\/g;        # \\ --> \; zB: \\w --> \w (wegen LaTeX)  #
      $K[0] =~ s/\\w\*//g;
      $K[0] =~ s/\$\\backslash\$/\\/ig;

      my($szKW) = $K[0];
      $szKW =~ s/\\w\*//g;
      $szKW =~ tr/äöüÄÖÜß\(\)\?\|\[\]:/aouaous/d;
                     # Bei gleichen Sortierwörtern ist ein Hash notwendig;  #
                     # Bsp.: "zusatz" und "zusätz" wird beides zu "zusatz". #
      push @{ $asarszKeywords{lc $szKW} }, [ @K ];
    }

  @arszKeywords = ();
  foreach my $szKW (sort keys %asarszKeywords)
    {
      foreach ( @{ $asarszKeywords{$szKW} } )
        {
          push @arszKeywords, $_;                    # $_ ist [ $K ] von oben. #
        }
    }

  return @arszKeywords;
}                                                            # ReadKeywords #
##############################################################################

##############################################################################
```



```
# PrintKeywords                                                              #
# Gibt alle Schlüsselwörter alfabetisch sortiert einschließlich der Anzahl   #
# aus. Hilfreich beim Debuggen und zur Überprüfung, ob die manuell oder      #
# automatisiert gefundenen Schlüsselwörter verwendet werden.                 #
##############################################################################
sub PrintKeywords
  {
    foreach (@arFeaturesSorted) { print "@{$_}[0]\n"; }
    print "="x60, "\n";
    print "Anzahl: ", $#arFeaturesSorted+1, " Wörter \n";
}                                                           # PrintKeywords #
##############################################################################

##############################################################################
# ReadStatus                                                                 #
# Liest aus der Statusdatei die einzelnen Nachrichten und den zugehörigen    #
# Veröffentlichungsstatus; die Daten werden als Hash zurück gegeben.         #
##############################################################################
sub ReadStatus
  {
    my(%HashStatus);
    my($szKey, $szValue);

    &Message("Reading status file ...\n");

    open (STATUS,"<$cszStatusDir$cszStatusFile") or
         die "Cannot open $cszStatusDir$cszStatusFile!\n";

    foreach (<STATUS>)
      {
        next if /^#/;                           # Kommentarzeilen überspringen. #
        next if /^\s*$/;                            # Leerzeilen überspringen. #
        s/\s*#.*$//;                  # Kommentare innerhalb der Liste entfernen. #
        ($szKey, $szValue) = split;

                  # Falls nur (un)veröffentlichte Artikel bearbeitet werden. #
        # next if ($szValue != 0);
        $HashStatus{$szKey} = $szValue;

        $arStatistics[($szValue) ? $dnAnzPublished : $dnAnzUnpublished]++;
      }

    close (STATUS);
    return %HashStatus;
}                                                              # ReadStatus #
##############################################################################

##############################################################################
# Match                                                                      #
# Versieht das übergebene Feature mit regulären Ausdrücken:                  #
# - Zeilenanfang oder ein Nichtbuchstabe am Anfang                           #
# - Eine beliebige Menge von Nicht-Whitespace-Zeichen am Ende                #
# Dadurch ist sichergestellt, dass nur die Wortanfänge übereinstimmen müssen #
# und Beugungen keinerlei Rolle spielen. Das übergebene Feature kann bereits #
# reguläre Ausdrücke beinhalten.                                             #
##############################################################################
sub Match
 {
   my($szFeature) = @_;

   return qr/(?:^|\W)$szFeature\S*/i;
}                                                                   # Match #
```

```
###########################################################################

###########################################################################
# HighlightKeywords                                                       #
# Konvertiert in allen APA-Nachrichten die Keyswords in Großbuchstaben.   #
# Dadurch kann manuell herausgefunden werden, ob ein Artikel aufgrund der #
# Keywords veröffentlichenswert erscheint.                                #
# Die Dateien mit den hervorgehobenen Keywords werden nach folgendem Schema #
# benannt: JJJJMMTT_APAxxx.txt.y.hl;                                      #
# y steht für 0=nicht veröffentlicht, 1,2=veröffentlicht                  #
###########################################################################
sub HighlightKeywords
 {
   my($szFileIn, $szFileOut);
   my($szF);
   my($szExtension) = ($bUseNoWhitespaceFiles) ? ".".$cszNoWhiteSpaceExt : "";
   # my($i) = 0;

   &Message("Highlighting keywords ...\n");

   undef $/;              # Dateien werden mit <MSG> als ganzes eingelesen. #

   FILE:
   foreach $szFileIn (<$cszDataDir$cszMsgFile>)
     {
       next FILE unless -T $szFileIn.$szExtension;        # Nur Textdateien. #
       # next FILE unless ($asarStatus{basename($szFileIn)});

       if (!open(MSG, $szFileIn.$szExtension))
         {
           print "Cannot open $szFileIn; continuing ...\n";
           next FILE;
         }

       # if ($i++ > 10) { return; }

       $szFileOut = join("", $cszDataDir, basename($szFileIn),
                    $szExtension, ".",
                    $asarStatus{basename($szFileIn)}, ".", $cszHighlightExt);

       &Message(join("", "Reading: ", basename($szFileIn), $szExtension,
                     " --> ", basename($szFileOut), "\n"));

       if (!open(OUT,">$szFileOut"))
         {
           print "Cannot open $szFileOut; continuing ...\n";
           close MSG;                                # Wir sind ordentlich. #
           next FILE;
         }

       print OUT "Veröffentlicht: ".$asarStatus{basename($szFileIn)}."\n\n";

       $_ = <MSG>;
       foreach $szF (@arFeaturesSorted)
         {
           my ($szM) = &Match(@{$szF}[0]);
           s/($szM)/\U$1/g;
         }
       print OUT $_;
       close(MSG);
       close(OUT);
   }
```



```
}                                                       # HighlightKeywords #
###########################################################################

###########################################################################
# SelectKeywords                                                          #
# Selektiert alle Keywords und schreibt sie in Dateien nach dem Muster    #
# nnnn_19990101_APAxxx.txt.[012].sel                                      #
# nnnn steht für die Anzahl der Keywords in dieser Datei                  #
# Zusätzlich werden allerlei Statistik-Dateien in &WriteSelectcountFiles() #
# geschrieben.                                                            #
###########################################################################
sub SelectKeywords
 {
   my($i) = 0;
   my($szFileIn, $szFileOut);
   my($szF);
   my(@arPublished)  = ();
   my(@arWordspread) = ();
   my($nMaxKeywords) = 0;
   my($szExtension) = ($bUseNoWhitespaceFiles) ? ".$cszNoWhiteSpaceExt" : "";

   &Message("Selecting keywords ...\n");

   undef $/;              # Dateien werden mit <MSG> als ganzes eingelesen. #

   FILE:
   foreach $szFileIn (<$cszDataDir$cszMsgFile>)
     {
       my (@arKeywords) = ();

       next FILE unless -T $szFileIn.$szExtension;
       next FILE unless defined $asarStatus{basename($szFileIn)};

       if (!open(MSG, $szFileIn.$szExtension))
         {
           print "Cannot open $szFileIn; continuing ...\n";
           next FILE;
         }

       # if ($i++ >= 2) { last; }

       &Message(join("", "Reading: ",
                     basename($szFileIn.$szExtension), " --> "));

       $_ = <MSG>;
       foreach $szF (@arFeaturesSorted)
         {
           my($szM) = &Match(@{$szF}[0]);
           push @arKeywords, $1 while (/($szM)/g);
         }

       $szFileOut = join("", $cszDataDir, sprintf("%04d_",$#arKeywords+1),
                         basename($szFileIn), ".",
                         $asarStatus{basename($szFileIn)}, ".",
                         $cszSelectExt);

       &Message(join("", basename($szFileOut), "\n"));

       if (!open(OUT,">$szFileOut"))
         {
           print "Cannot open $szFileOut; continuing ...\n";
           close MSG;                                # Wir sind ordentlich. #
```

```
          next FILE;
        }
      print OUT @arKeywords;
      close(MSG);
      close(OUT);

      $arPublished[$#arKeywords+1]->
                  [($asarStatus{basename($szFileIn)}) ? 1 : 0]++;
      my($n) = scalar (@_ = split);
      $arWordspread[$n]->[$#arKeywords+1]->
                   [($asarStatus{basename($szFileIn)}) ? 1 : 0]++;
      $nMaxKeywords = $#arKeywords+1 if ($#arKeywords+1 > $nMaxKeywords);
    }

  &WriteSelectcountFiles(\@arPublished,\@arWordspread,$nMaxKeywords);
}                                                       # SelectKeywords #
############################################################################

############################################################################
# WriteSelectcountFiles                                                    #
############################################################################
sub WriteSelectcountFiles
 {
  my($refarPublished, $refarWordspread, $nMaxKeywords) = @_;
  my($i, $j);

  &Message("Writing $cszGraphDir$cszWordspreadFile ...\n");
  if (!open(OUT,">$cszGraphDir$cszWordspreadFile"))
    {
      print "Cannot open $cszGraphDir$cszWordspreadFile: $!\n";
      exit 1;
    }

  for ($i=0; $i<=$#$refarWordspread; $i++)
    {
      if (defined $refarWordspread->[$i])
        {
          for ($j=0; $j<=$nMaxKeywords; $j++)
            {
              print OUT "$i\t$j";
              foreach my $p (0..1)
                {
                  if (defined $refarWordspread->[$i]->[$j]->[$p])
                    {
                       print OUT "\t", $refarWordspread->[$i]->[$j]->[$p] /
                             $arStatistics[($p) ? $dnAnzPublished :
                                                  $dnAnzUnpublished];
                    }
                  else { print OUT "\t0"; }
                }
              print OUT "\n";
            }
        }
      else
        {
          for ($j=0; $j<=$nMaxKeywords; $j++) { print OUT "$i\t$j\t0\t0\n"; }
        }
    }
  close(OUT);

  &Message("Writing $cszGraphDir$cszSelectcountFile ...\n");
  if (!open(OUT,">$cszGraphDir$cszSelectcountFile"))
```



```
    {
      print "Cannot open $cszGraphDir$cszSelectcountFile: $!\n";
      exit 1;
    }

  for ($i=0; $i<=$#$refarPublished; $i++)
    {
      print OUT "$i";
      foreach $j (0..1)
        {
          if (defined $refarPublished->[$i]->[$j])
            {
              print OUT "\t", $refarPublished->[$i]->[$j] /
                     $arStatistics[($j) ? $dnAnzPublished:$dnAnzUnpublished];
            }
          else { print OUT "\t0"; }
        }
      print OUT "\n";
    }
  close(OUT);
}                                                 # WriteSelectcountFiles #
############################################################################

############################################################################
# DeleteUselessWhitespaces                                                 #
# Löscht unnötige Leerzeichen und Leerzeilen und entfernt Satzzeichen aus  #
# den Nachrichtendateien. Vorteil: Raschere Bearbeitung bei der Feature-   #
# Suche.                                                                   #
############################################################################
sub DeleteUselessWhitespaces
 {
  my($szFileIn, $szFileOut);
  my($szInput, $szOutput);

  &Message("Deleting useless whitespaces ...\n");

  undef $/;               # Dateien werden mit <MSG> als ganzes eingelesen. #

  FILE:

  print "$cszDataDir\n";
  print "$cszMsgFile\n";

  foreach $szFileIn (<$cszDataDir$cszMsgFile>)
    {
      next FILE unless -T $szFileIn;

      if (!open(MSG,$szFileIn))
        {
          print "Cannot open $szFileIn; continuing ...\n";
          next FILE;
        }

      $szFileOut = join("", $cszDataDir, basename($szFileIn), ".",
                        $cszNoWhiteSpaceExt);

      &Message(join("", "Reading: ", basename($szFileIn), " --> ",
                    basename($szFileOut), "\n"));

      if (!open(OUT,">$szFileOut"))
        {
          print "Cannot open $szFileOut; continuing ...\n";
```

```
            close MSG;                              # Wir sind ordentlich. #
            next FILE;
          }

        $_ = <MSG>;
        s/\n+/ /gm;
        tr/\-\.:,;\(\)\?\!\/\"\=\*/ /;               # Satzzeichen entfernen. #
        s/ +/ /gm;
        print OUT $_;
        close(MSG);
        close(OUT);
    }
}                                               # DeleteUselessWhitespaces #
############################################################################

############################################################################
# InitVectorSpace                                                          #
# Initialisiert den gesamten Vektorraum: $cnDokumente*$cnFeatures          #
############################################################################
sub InitVectorSpace
 {
   my($i);

   &Message("Init vector space (all features: $cnAllFeatures) ...\n");
   for ($i=0; $i<=$cnAllFeatures; $i++) { $arnVS[$i] = 0; }
}                                                        # InitVectorSpace #
############################################################################

############################################################################
# Calculate_TF                                                             #
# Berechnet die Term Frequency: Es wird die Anzahl der Vorkommen eines     #
# Features in einem Dokument für jedes Dokument ermittelt.                 #
############################################################################
sub Calculate_TF
 {
   my($szFile,$nAktFeature,$szF);
   my($nAktDokument) = 0;
   my($szExtension) = ($bUseNoWhitespaceFiles) ? ".".$cszNoWhiteSpaceExt : "";
   my($i) = 0;

   &Message("Calculating term frequency ...\n");

   undef $/;              # Dateien werden mit <MSG> als ganzes eingelesen. #

   FILE:
   foreach $szFile (sort <$cszDataDir$cszMsgFile>)
     {
        next FILE unless -T $szFile.$szExtension;
        if (!open(MSG, $szFile.$szExtension))
          {
            print "Cannot open $szFile; continuing ...\n";
            next FILE;
          }

        &Message("Reading: " . basename($szFile) . $szExtension . "\n");
        push @arFiles, basename($szFile);

        $_ = <MSG>;
        $nAktFeature = 0;
        foreach $szF (@arFeaturesSorted)
          {
            my($szM) = &Match(@{$szF}[0]);
```



```
            while (/$szM/g)
              {
                $arnVS[$nAktDokument*$cnFeatures+$nAktFeature]++;
                # Arbeit mit Kategorien/Dimensionen:
                # Jede Dimension eines Features macht ++
                # foreach my $szD (@{@{$szF}[1]})
                #   { $arnVS[$nAktDokument*$cnFeatures+$szD]++; }
             }
            $nAktFeature++;
         }
        close(MSG);
        $nAktDokument++;
    }

   # print "_TF: ", join(" ",@arnVS), "\n\n";
}                                                           # Calculate_TF #
############################################################################

############################################################################
# Calculate_DF                                                             #
# Berechnet die Document Frequency: Es wird ermittelt, wie wievielen Doku- #
# menten ein Feature vorkommt.                                             #
############################################################################
sub Calculate_DF
 {
   my($i);

   &Message("Calculating document frequency ...\n");

   for ($i=0; $i<$cnFeatures; $i++) { $arnDF[$i] = 0; }
   for ($i=0; $i<$cnAllFeatures; $i++)
     {
       $arnDF[$i % $cnFeatures]++ if ($arnVS[$i] != 0);
    }

   # print "_DF: ", join(" ",@arnDF), "\n\n";
}                                                           # Calculate_DF #
############################################################################

############################################################################
# Calculate_TFxIDF                                                         #
# Berechnet die mit der inversen Document Frequency gewichtete Term Fre-   #
# quency. Terme, die nur in wenigen Dokumenten, dort dafür aber umso häu-  #
# figer vorkommen, erhalten mehr Gewicht.                                  #
############################################################################
sub Calculate_TFxIDF
 {
   &Message("Calculating TFxIDF ...\n");

   for (my $i=0; $i<$cnAllFeatures; $i++)
     {
       if ($arnDF[$i % $cnFeatures])
         {
           $arnVS[$i] *= log ($cnDokumente/$arnDF[$i % $cnFeatures]);
         }
    }

   # print "_TFxDF: ", join(" ",@arnVS), "\n\n";
}                                                       # Calculate_TFxIDF #
############################################################################

############################################################################
```

```
# Calculate_TFx_IDF_standardized                                            #
# Normiert jeden Dokumentvektor, indem alle Komponenten des Vektors durch   #
# die betragsmäßig größte Komponente dividiert werden.                      #
##############################################################################
sub Calculate_TFxIDF_standardized
 {
   my($nMax) = 0;

   &Message("Calculating TFxIDF standardized ...\n");

   # Normierung jedes einzelnen Dokument-Vektors. #
   my ($nStart) = 0;
   while ($nStart < $cnAllFeatures-2)
     {
       foreach ($nStart..$nStart+$cnFeatures-1)
         {
           $nMax = $arnVS[$_] if ($nMax < $arnVS[$_]);
         }
       if ($nMax != 0)
         {
           $arnVS[$_] /= $nMax foreach ($nStart..$nStart+$cnFeatures-1);
           $nMax = 0;
         }
       $nStart += $cnFeatures;
   }
}                                          # Calculate_TFxIDF_standardized #
##############################################################################

##############################################################################
# Calculate_Kosinusdistanz_AB                                               #
# Berechnet die Kosinusdistanz cos phi zweier Vektoren refA, refB.          #
# cos phi = (a * b) / |a|*|b|                                               #
# +1: maximale Ähnlichkeit                                                  #
# -1: minimale Ähnlichkeit                                                  #
#  0: orthogonale Vektoren                                                  #
##############################################################################
sub Calculate_Kosinusdistanz_AB
 {
   my($refA, $refB) = @_;
   my($i);

   my $Betrag = sub {
                      my($refar) = @_;
                      my($sum) = 0;

                      for (my $i=0; $i<=$#$refar; $i++)
                        {
                          $sum += $refar->[$i]**2;
                        }
                      return sqrt($sum);
                    };

   my $Skalarprodukt = sub {
                             my($refA, $refB) = @_;
                             my($sum) = 0;

                             for (my $i=0; $i<=$#$refA; $i++)
                               {
                                 $sum += $refA->[$i] * $refB->[$i];
                               }
                             return $sum;
                           };
```



```
   return &$Skalarprodukt($refA,$refB) / (&$Betrag($refA) * &$Betrag($refB));
}                                             # Calculate_Kosinusdistanz_AB #
##############################################################################

sub GetDayOfMsg
 {
   $_[0] =~ /(\d{8})_APA\d{4}\.txt/;
   return $1;
}                                                             # GetDayOfMsg #
##############################################################################

sub GetIndexOfDay
 {
   my($Tag,$refarTage) = @_;
   for (my $i=0; $i<=$#$refarTage; $i++)
     {
       return $i if ($refarTage->[$i] == $Tag)
     }
   return 0;
}                                                           # GetIndexOfDay #
##############################################################################

sub GetIndexOfMsg
 {
   my($Msg,$refarMsgFiles) = @_;
   for (my $i=0; $i<=$#$refarMsgFiles; $i++)
     {
       return $i if ($refarMsgFiles->[$i] eq $Msg);
     }
   return 0;
}                                                           # GetIndexOfMsg #
##############################################################################

##############################################################################
# Calculate_Kosinusdistanz                                                  #
# Berechnet für alle Dokumente die Kosinusdistanzen zu allen Dokumenten der #
# letzten 0..9 Tage und speichert diese Werte in jeweils eigenen Files ab.  #
##############################################################################
sub Calculate_Kosinusdistanz
 {
   for (my $i=0; $i<$cnDokumente; $i++)
     {
       my(@a, @b);
       my(%arDistances) = ();

       &Message("Calculating distances for document $i ...\n");

       @a = @arnVS[$i*$cnFeatures..($i+1)*$cnFeatures-1];
       $arDistances{"number"}    = $i;
       $arDistances{"document"}  = $arMsgFiles[$i];
       $arDistances{"day"}       = &GetDayOfMsg($arDistances{"document"});
       $arDistances{"distances"} = ( );
       my($nIndexOfDay)          = &GetIndexOfDay($arDistances{"day"},\@arTage);

       for (my $nPre=0; $nPre<=9; $nPre++)       # Alle Vortage abklappern. #
         {
           my $nDay;
           if (($nDay = $nIndexOfDay-$nPre) >= 0) # Gab es Vortage? (1. 1.!) #
             {                                # Jede Msg des Vortags abklappern. #
               foreach (@{ $asarTage{$arTage[$nDay]} })
                 {
```

```
                    my $nDok = &GetIndexOfMsg($_,\@arMsgFiles);
                    @b = @arnVS[$nDok*$cnFeatures..($nDok+1)*$cnFeatures-1];
                    my $erg = &Calculate_Kosinusdistanz_AB(\@a,\@b);
                    push @{ $arDistances{"distances"}{$nPre}{$erg} }, $nDok;
                }
            }
        }

        my($szF) = join("", $cszDistanceDir,              # Extension entfernen. #
                        (split(/\.txt/, $arDistances{"document"}))[0],
                        ".", $cszDistanceCosineExt);
        open DIST, ">$szF" or die "Cannot open $szF, died";
        print DIST Data::Dumper->Dump([\%arDistances], ['*arDistances']);
        close DIST;
    }
}                                                    # Calculate_Kosinusdistanz #
################################################################################

################################################################################
# PrevDays                                                                     #
# Berechnet die normierte Anzahl der publizierten ähnlichen Artikel eines      #
# bestimmten Artikels der Vortage.                                             #
################################################################################
sub PrevDays
 {
   my($szFile) = $cszDistanceDir . $arMsgFiles[$_[0]];
   our(%arDistances);
   my($nDay);
   my($nDist);
   my(@arVortage); for (my $i=0; $i<$nPrevDays; $i++) { $arVortage[$i]=0; }

   $szFile =~ s/txt$/$cszDistanceCosineExt/;
   unless (my $return = do $szFile) {
        die "couldn't parse $szFile: $@" if $@;
        die "couldn't do $szFile: $!"    unless defined $return;
        die "couldn't run $szFile"       unless $return;
     }

   # 1: Für alle Vortage werden die ...                                        #
   # 2: ... $nAnzDistances nächsten (ähnlichsten) Artikel betrachtet.          #
   # 3: Manchmal gibt es weniger als $nAnzDistances Artikel (zb 2. 1. 1999).   #
   # 4: Jede Distanz kann mehrere Artikel beinhalten.                          #
   # 5: Wurde ein Artikel am Tag $nDay veröffentlicht, so wird der Ver-        #
   #    öffentlichungszähler in @arVortage erhöht.                             #
   # 6: Normierung der Veröffentlichungszählers auf die Anzahl der $nAnz-      #
   #    distances nächsten Artikel.                                            #
   for ($nDay=1; $nDay<=$nPrevDays; $nDay++)                               # 1 #
     {
       foreach $nDist ((sort { $b <=> $a } keys                            # 2 #
                  %{ $arDistances{"distances"}{$nDay} })[0..$nAnzDistances-1])
         {
           next if (!defined $nDist);                                      # 3 #
           foreach (@{ $arDistances{"distances"}{$nDay}{$nDist} })         # 4 #
             {                                                             # 5 #
               if ($asarStatus{$arMsgFiles[$_]}) { $arVortage[$nDay-1]++ }
             }
         }
     }

   @arVortage = map { $_ / ($nAnzDistances+1) } @arVortage;               # 6 #
   return @arVortage;
}                                                                 # PrevDays #
```



```
################################################################################

################################################################################
# CreateSOMFiles                                                               #
# Erzeugt die beiden für ghsom notwendigen Files                               #
# $cszSOMDir$cszSOMInputvectorfile und $cszSOMDir$cszSOMTemplatefile.          #
# ghsom ist ein Programm zur Erzeugung von Self Organizing Maps.               #
################################################################################
sub CreateSOMFiles
 {
   my $CreateInputvectorFile = sub
     {
       Message("Creating SOM inputvector file ...\n");
       open(IVF, ">$cszSOMDir$cszSOMInputvectorfile") or
          die "Cannot create $cszSOMDir$cszSOMInputvectorfile, died";

       print IVF "\$TYPE vec\n";
       print IVF "\$XDIM $cnDokumente\n";
       print IVF "\$YDIM 1\n";
       print IVF "\$VEC_DIM $cnFeatures\n";

       my $i;
       for ($i=0; $i<$cnAllFeatures; $i++)
         {
           if (($i != 0) && (($i % $cnFeatures) == 0))
             {
               print IVF "$arFiles[$i/$cnFeatures]\n";
             }
           printf IVF "%2.5f ", $arnVS[$i];
         }
       print IVF "$arFiles[$i/$cnFeatures]\n";

       close IVF;
   };

   my $CreateTemplatevectorFile = sub
     {
       my ($i) = 0;

       Message("Creating SOM templatefile ...\n");
       open(TVF, ">$cszSOMDir$cszSOMTemplatefile") or
          die "Cannot create $cszSOMDir$cszSOMTemplatefile, died";

       print TVF "\$TYPE template\n";
       print TVF "\$XDIM 2\n";
       print TVF "\$YDIM $cnDokumente\n";
       print TVF "\$VEC_DIM $cnFeatures\n";

       foreach (@arFeaturesSorted)
         {
           my($szKW) = @{ $_ }[0];
           $szKW =~ s/\\w\*//g;
           $szKW =~ tr/äöüÄÖÜß\(\)\?\|\[\]:/aouaous/d;
           $szKW =~ s/ +$//;
           $szKW =~ tr/ /\_/d;
           print TVF $i++, " ", $szKW, "\n";
         }

       close TVF;
   };

   &$CreateInputvectorFile();
```

```
   &$CreateTemplatevectorFile();
}                                                        # CreateSOMFiles #
##############################################################################

##############################################################################
# Create_SNNS_Patternfile                                                    #
# Erzeugt ein sehr, sehr großes Patternfile, in dem für alle Dokumente die   #
# TFxIDF-Werte jedes Features (jeder Inputunit) und das gewünschte Output-   #
# pattern stehen. Dieses File wird von SNNS zum Lernen verwendet.            #
##############################################################################
sub Create_SNNS_Patternfile
 {
   my $GetIgnoredDocsCount = sub
     {
       my($nCount) = 0;

       for (my $i=0; $i<$nPrevDays; $i++)
         {
           my(@t) = @{ $asarTage{$arTage[$i]} };
           $nCount += $#t+1;
         }
       return $nCount;
     };

   my($i,$nPatternID,$nPatternsPerLine);

   open(PATTERNFILE, ">$cszPatternfileDir$cszPatternfile") or
      die "Cannot create patternfile $cszPatternfileDir$cszPatternfile, died";

   &Message("Creating pattern file ...\n");
   &SNNSPattern_Header($cnDokumente,&$GetIgnoredDocsCount(),
                       $cnFeatures+$nPrevDays);
   &SNNSPattern_Preambel($cnDokumente-&$GetIgnoredDocsCount(),
                         $cnFeatures+$nPrevDays);
   &SNNSPattern_Features();

   for ($i=0; $i<$cnAllFeatures; $i++)
     {
       unless ($i % $cnFeatures)
         {
           $nPatternID = $i/$cnFeatures + 1;

           my($szFilename) = $arFiles[$nPatternID];
           my($nAktDay) = &GetIndexOfDay(&GetDayOfMsg($szFilename),\@arTage);

                  # Es sollen nur für jene Tage Patterns erzeugt werden, wo #
                  # es $nPrevDays Vortage gibt. Die Einbeziehung der Vortage #
                  # ist nur sinnvoll, wenn auch Vortage vorhanden sind.      #
           if ($nAktDay-$nPrevDays < 0)
             {
               $i += $cnFeatures-1;              # next führt $i++ am Ende ... #
               next;                             # ... jeder Schleife aus.     #
             }

           print PATTERNFILE "# Inputpattern (Datei ",
                $szFilename, ") #", $nPatternID, ":\n";
           foreach (&PrevDays($nPatternID-1))
             {
               printf PATTERNFILE "%2.5f ", $_;
             }
           print PATTERNFILE "\n";
           $nPatternsPerLine = 0;
```



```
         }

       printf PATTERNFILE "%2.5f ", $arnVS[$i];

       $nPatternsPerLine++;
       unless ($nPatternsPerLine % $cnMaxPatternsPerLine)
         {
           print PATTERNFILE "\n";
           $nPatternsPerLine = 0;
         }

       unless (($i+1) % $cnFeatures)
         {
           print PATTERNFILE "\n# Outputpattern #", $nPatternID, ":\n";
           # print PATTERNFILE $asarStatus{$arFiles[$nPatternID]}, "\n\n";
           print PATTERNFILE
                 &Create_Outputpattern($asarStatus{$arFiles[$nPatternID]}),
                 "\n\n";
         }
     }
   close(PATTERNFILE);
}                                                 # Create_SNNS_Patternfile #
##############################################################################

##############################################################################
# Create_Outputpattern                                                       #
# Erzeugt das Outputpattern gemäß $nOutputUnits.                             #
##############################################################################
sub Create_Outputpattern
 {
   my($bPublished) = @_;

   if    ($nOutputUnits == 1) { $bPublished }
   elsif ($nOutputUnits == 2) { ($bPublished) ? "1 0" : "0 1" }
   elsif ($nOutputUnits == 3) { ($bPublished) ? "1 1 1" : "0 0 0" }
   elsif ($nOutputUnits == 5) { ($bPublished) ? "1 1 1 1 1" : "0 0 0 0 0" }
   else  { "error" }
}                                                    # Create_Outputpattern #
##############################################################################

sub SNNSPattern_Header
 {
   my($nDokumente,$nIgnored,$nInputunits) = @_;

   print PATTERNFILE "#" x 78, "\n";
   print PATTERNFILE "# Anzahl der bearbeiteten Dokumente (Patterns): ",
                     "$nDokumente,\n";
   print PATTERNFILE "# davon mangels Vortagen ignoriert: $nIgnored\n";
   print PATTERNFILE "# Anzahl der Inputunits: $nInputunits\n";
   print PATTERNFILE "# (c) Thomas Pfeiffer, 9325691, $cszCopyrightDate\n";
   print PATTERNFILE "#" x 78, "\n\n";
}                                                      # SNNSPattern_Header #
##############################################################################

sub SNNSPattern_Preambel
 {
   my($nPatterns,$nUnits) = @_;

   print PATTERNFILE "SNNS pattern definition file V3.2\n";
   print PATTERNFILE "generated at ", scalar localtime;
   print PATTERNFILE "\n";
   print PATTERNFILE "No. of patterns : $nPatterns\n";
```

```
   print PATTERNFILE "No. of input units : $nUnits\n";
   print PATTERNFILE "No. of output units : $nOutputUnits\n";
   print PATTERNFILE "\n\n";
}                                                      # SNNSPattern_Preambel #
##############################################################################

sub SNNSPattern_Features
 {
   my($i) = 0;

   print PATTERNFILE "#" x 78, "\n";
   print PATTERNFILE "# Features in alfabetischer Reihenfolge:\n# ";
   foreach (@arFeaturesSorted)
     {
       print PATTERNFILE "\"@{$_}[0]\", ";
       $i++;
       $i=0, print PATTERNFILE "\n# " unless ($i % ($cnMaxPatternsPerLine/2));
    }
   print PATTERNFILE "\n", "#" x 78, "\n\n";
}                                                      # SNNSPattern_Features #
##############################################################################

sub Create_SNNS_Network
 {
   my($nInputUnits)      = $cnFeatures + $nPrevDays;
   my($szConnections)    = "";
   my($nSektoren)        = $nOutputUnits;
   # my($nSektoren)       = $cnAnzHiddenunits2;

   my($nSektorgroesse) = $cnAnzHiddenunits1 / $nSektoren;
   my($i);

   &Message("Creating network ...\n");
   &Message("Features:       $cnFeatures\n");
   &Message("Previous days: $nPrevDays\n");
   &Message("Inputunits:     $nInputUnits\n");
   &Message("Hiddenunits1:  $cnAnzHiddenunits1\n");
   # &Message("Hiddenunits2:  $cnAnzHiddenunits2\n");
   &Message("Outputunits:    $nOutputUnits\n");

   # Die Connections von Layer 2 nach Layer 3 sind nicht mehr voll-
   # verbunden (also jede Unit mit jeder Unit), sondern nur mehr
   # zu gleichen Anteilen zu den Outputunits. Beispielsweise bei
   # 15 Hiddenunits und 3 Outputunits werden die Hiddenunits zu
   # jeweils 5 gruppiert und jede dieser Gruppen nur mit jeweils
   # einer Outputunit vollverbunden. Bei Existenz eines vierten
   # Layers sind wieder alle möglichen Verbindungen zwischen den
   # Units des dritten und vierten Layers vorhanden.

   for ($i=1; $i<=$nSektoren; $i++)
     {
       $szConnections  .= join(" ", " -l 2 1",
                                    ($i-1)*$nSektorgroesse+1, "1",
                                    $nSektorgroesse,
                                    "+ 3 1", $i, "1 1 +");
    }

   my($szCmd) = join("",
                     "ff_bignet",
                     " -p 1 $nInputUnits",
                     " -p 1 $cnAnzHiddenunits1",
```



```
                     # Bei vierlagigem Netzwerk
                     # " -p 1 $cnAnzHiddenunits2",

                     " -p 1 $nOutputUnits",
                     " -l 1 1 1 1 $nInputUnits + 2 1 1 1 $cnAnzHiddenunits1 +",
                     $szConnections,

                     # Bei vierlagigem Netzwerk
                     # " -l 2 1 1 1 $cnAnzHiddenunits1 + 3 1 1 1 3 +",
                     # " -l 3 1 1 1 2 + 4 1 1 1 1 +",
                     # " -l 3 1 3 1 2 + 4 1 2 1 1 +",
                     # " -l 3 1 5 1 2 + 4 1 3 1 1 +",
                     # " -l 3 1 1 1 $cnAnzHiddenunits2 + 4 1 1 1 1 +",

                     " $cszNetworkfileDir$cszNetworkfile");

   &Message("\n$szCmd\n\n");
   system(split / +/, $szCmd) == 0 or die "Cannot create network file!\n";
}                                                       # Create_SNNS_Network #
##############################################################################

##############################################################################
# Message                                                                    #
# Ausgabe von Debug-Infos; gesteuert über $bVerbose.                         #
##############################################################################
sub Message
 {
   print $_[0] if ($bVerbose);
}                                                                   # Message #
##############################################################################

##############################################################################
# Usage                                                                      #
# Erläutert die Nutzung dieses Programms durch Kommandozeilenparameter.      #
##############################################################################
sub Usage
 {
   print "Usage: ", basename($0),
         " [--createdistances] [--createsomfiles]",
         " [--deleteuselesswhitespaces] [--demo] [--help] [--highlight]",
         " [--network] [--outputunits=<n>] [--printkeywords]",
         " [--selectkeywords] [--testnetwork] [--usenowhitespacefiles]",
         " [--verbose]\n\n";

   print <<EOF;
--createdistances
  Erzeugt die Distanzfiles aller Nachrichten für die letzten 0..9 Tage
  im Verzeichnis $cszDistanceDir.
  Die Dateien werden mit der Endung ".$cszDistanceCosineExt" versehen.

--createsomfiles
  Erzeugt die Dateien für die Weiterverarbeitung mit ghsom, einem
  Programm zur Erzeugung von Self Organizing Maps.
  $cszSOMDir$cszSOMInputvectorfile
  $cszSOMDir$cszSOMTemplatefile

--deleteuselesswhitespaces
  Entfernt alle unnötigen Leerzeichen und Zeilenumbrüche. Dadurch können
  auch mehrwortige Keywords gefunden werden, die durch einen Zeilenumbruch
  separiert werden.

  Die Ausgabedateien werden mit der Endung ".nws" (NoWhiteSpace) versehen.
```

```
  Siehe auch Option --usenowhitespacefiles.

--demo
  Demo-Modus für die Beispieldateien in Anhang B.

--help
  Gibt diese Hilfe aus.

--highlight
  Markiert die Keywords in den Dateien und versieht sie mit der Endung
  [012].hl - 0, 1 und 2 gibt an, wie oft eine APA-Nachricht in einem
  Artikel des Standards verwendet wurde.

--network
  Erzeugt nur das Netzwerk (es werden keine Pattern berechnet).

--outputunits
  Anzahl der Outputunits; muss zwischen 1 und 5 liegen.
  Standardwert: $nOutputUnits

--printkeywords
  Gibt eine alfabetische Liste der Keywords aus.

--selectkeywords
  Erzeugt Dateien nnnn_19990101_APAxxxxx.txt.[012].sel; in diesen
  Dateien stehen nur die Keywords; nnnn gibt die Anzahl der Keywords
  in der Datei an.
  Zusätzlich wird die Datei selectcounts.csv erzeugt, in der die Anzahl
  der Dokumente je gefundener Keywords steht.

--testnetwork
  Erzeugt die Daten für das Testen eines trainierten Netzes.

--usenowhitespacefiles
  Verwendet beim Trainieren des Netzes die mit --deleteuselesswhitespaces
  erzeugten Dateien (*.nws).

--verbose
  Ausgabe von allerlei Meldungen.

EOF

    exit $_[0];
}                                                                 # Usage #
###########################################################################

## Ende vsm.pl ###
```

B.2 Training des Netzes mit SNNS

Die Dateien `TFxIDF.pat` und `nwt.net`, beide erzeugt von `vsm.pl` (siehe oben), sind die Grundlage für das Trainieren des neuronalen Netzes. Dieses Training erfolgt mit

Hilfe des Batchfiles `train.bat`, das von `batchman` – ein Teilprogramm von SNNS – verwendet wird.[1]

```
###########################################################################
# train.bat                                                               #
# SNNS-Batchfile, um das Netz mit den Echtdaten zu trainieren.            #
# Thomas Pfeiffer, 9325691                                                #
# September 2002 - Mai 2005                                               #
###########################################################################

loadNet("nwt.net")
loadPattern("TFxIDF.pat")
setLearnFunc("BackpropMomentum", 0.10, 0.10)
setUpdateFunc("Topological_Order")
setInitFunc("Randomize_Weights", 1.0, -1,0)
initNet()

while CYCLES < 200 and SIGNAL == 0 do
  execute("date",day,month,day,hour)
  print(month, " ", day, " ", hour, ": ", "Cycles: ", CYCLES, " SSE: ", SSE)
  trainNet()

  saveNet("nwt_"+CYCLES+".trained.net")
endwhile

execute("date",day,month,day,hour)
print(month, " ", day, " ", hour, ": ", "Cycles: ", CYCLES, " SSE: ", SSE)

saveResult("nwt.res", 1, PAT, TRUE, TRUE, "create")
saveNet("nwt.trained.net")
```

B.2.1 Trainingsergebnis

Der Output eines Trainingszyklus zeigt die Fehlerrate (SSE) bei jeder Iteration. Diese Fehlerrate ist auch die Grundlage für die in den vorangegangenen Kapiteln dargestellten Trainingsverläufe.[2]

```
#  Net nwt.net loaded
#  Patternset TFxIDF.pat loaded; 1 patternset(s) in memory
#  Learning function is now BackpropMomentum
#  Parameters are: 0.3 0.1
#  Update function is now Topological_Order
#  Init function is now Randomize_Weights
#  Parameters are: 1 -1 0
#  Net initialized
May 13 00:04:41: Cycles: 0 SSE: 3.40282e+38
May 13 00:04:56: Cycles: 1 SSE: 582.223
May 13 00:05:13: Cycles: 2 SSE: 580.052
May 13 00:05:29: Cycles: 3 SSE: 580.014
May 13 00:05:46: Cycles: 4 SSE: 580.026
May 13 00:06:02: Cycles: 5 SSE: 580.038
```

[1] Der Batchbetrieb bietet auch den Vorteil, das Training unbeaufsichtigt über Nacht mit unterschiedlichen Parametern durchführen zu können.

[2] Die grafische Darstellung des hier abgebildeten Trainings findet sich in Abbildung C.9(c) auf Seite 113.

```
May 13 00:06:17: Cycles: 6 SSE: 580.011
May 13 00:06:33: Cycles: 7 SSE: 579.903
May 13 00:06:49: Cycles: 8 SSE: 579.649
May 13 00:07:04: Cycles: 9 SSE: 579.147
May 13 00:07:20: Cycles: 10 SSE: 578.187
May 13 00:07:36: Cycles: 11 SSE: 576.339
May 13 00:07:51: Cycles: 12 SSE: 572.7
May 13 00:08:07: Cycles: 13 SSE: 565.774
May 13 00:08:23: Cycles: 14 SSE: 554.587
May 13 00:08:38: Cycles: 15 SSE: 540.719
May 13 00:08:54: Cycles: 16 SSE: 526.131
May 13 00:09:10: Cycles: 17 SSE: 511.483
May 13 00:09:26: Cycles: 18 SSE: 497.51
May 13 00:09:41: Cycles: 19 SSE: 484.511
May 13 00:09:57: Cycles: 20 SSE: 472.272
May 13 00:10:14: Cycles: 21 SSE: 460.571
May 13 00:10:29: Cycles: 22 SSE: 449.298
May 13 00:10:46: Cycles: 23 SSE: 438.359
May 13 00:11:02: Cycles: 24 SSE: 427.76
May 13 00:11:18: Cycles: 25 SSE: 417.654
May 13 00:11:33: Cycles: 26 SSE: 408.037
May 13 00:11:49: Cycles: 27 SSE: 398.921
May 13 00:12:05: Cycles: 28 SSE: 389.857
May 13 00:12:20: Cycles: 29 SSE: 380.521
May 13 00:12:36: Cycles: 30 SSE: 371.067
May 13 00:12:52: Cycles: 31 SSE: 361.939
[...]
May 13 00:51:40: Cycles: 178 SSE: 57.2901
May 13 00:51:55: Cycles: 179 SSE: 57.8331
May 13 00:52:11: Cycles: 180 SSE: 57.8992
May 13 00:52:27: Cycles: 181 SSE: 64.4202
May 13 00:52:42: Cycles: 182 SSE: 60.3361
May 13 00:52:58: Cycles: 183 SSE: 57.4533
May 13 00:53:14: Cycles: 184 SSE: 56.2959
May 13 00:53:30: Cycles: 185 SSE: 56.0292
May 13 00:53:45: Cycles: 186 SSE: 55.9661
May 13 00:54:01: Cycles: 187 SSE: 56.7141
May 13 00:54:17: Cycles: 188 SSE: 55.9155
May 13 00:54:32: Cycles: 189 SSE: 55.8324
May 13 00:54:48: Cycles: 190 SSE: 55.6696
May 13 00:55:05: Cycles: 191 SSE: 56.043
May 13 00:55:21: Cycles: 192 SSE: 55.6695
May 13 00:55:36: Cycles: 193 SSE: 55.6916
May 13 00:55:53: Cycles: 194 SSE: 55.7131
May 13 00:56:09: Cycles: 195 SSE: 55.6648
May 13 00:56:24: Cycles: 196 SSE: 55.7095
May 13 00:56:40: Cycles: 197 SSE: 55.5742
May 13 00:56:56: Cycles: 198 SSE: 55.6243
May 13 00:57:12: Cycles: 199 SSE: 55.6424
May 13 00:57:27: Cycles: 200 SSE: 55.5465
# Result file nwt.res written
# Network file nwt.trained.net written
```


C Weitere Auswertungen

C.1 Wortverteilungen

Die drei folgenden Abbildungen entsprechen den Auswertungen aus den Kapiteln 5.2 und 5.3 auf den Seiten 41ff. Der Unterschied besteht darin, dass hier nicht die manuell gewählten 1.044 Schlüsselwörter, sondern die automatisch selektierten 1.075 Schlüsselwörter zur Berechnung herangezogen wurden.

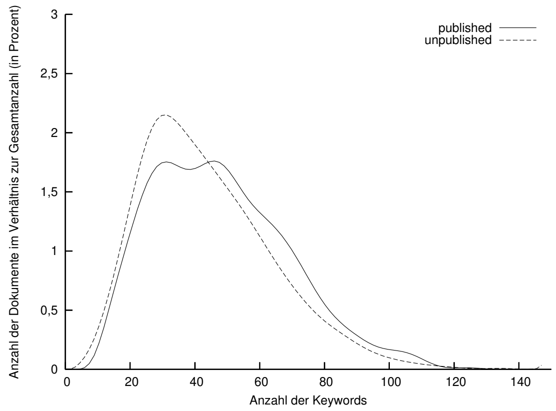


Abbildung C.1: Verteilung der Anzahlen der Keywords

C.2 Ausgewählte Trainingsverläufe

Die folgenden Abbildungen C.4 bis C.13 zeigen ausgewählte Trainingsverläufe der in Kapitel 6.9, Seite 56, erörterten Netztopologien und Lernverfahren.

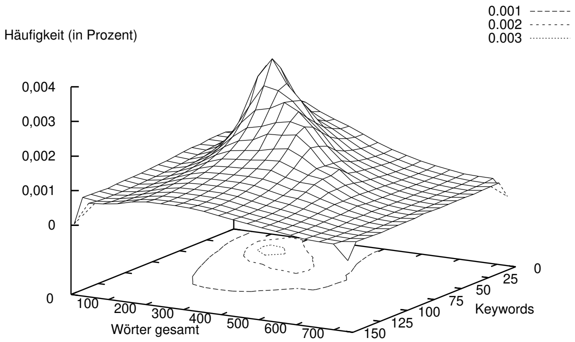


Abbildung C.2: Wortverteilung unpublizierte Artikel

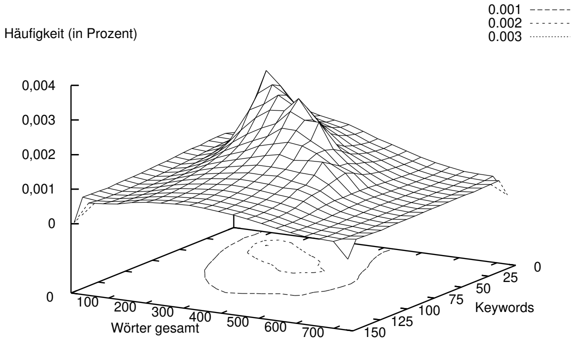


Abbildung C.3: Wortverteilung publizierte Artikel

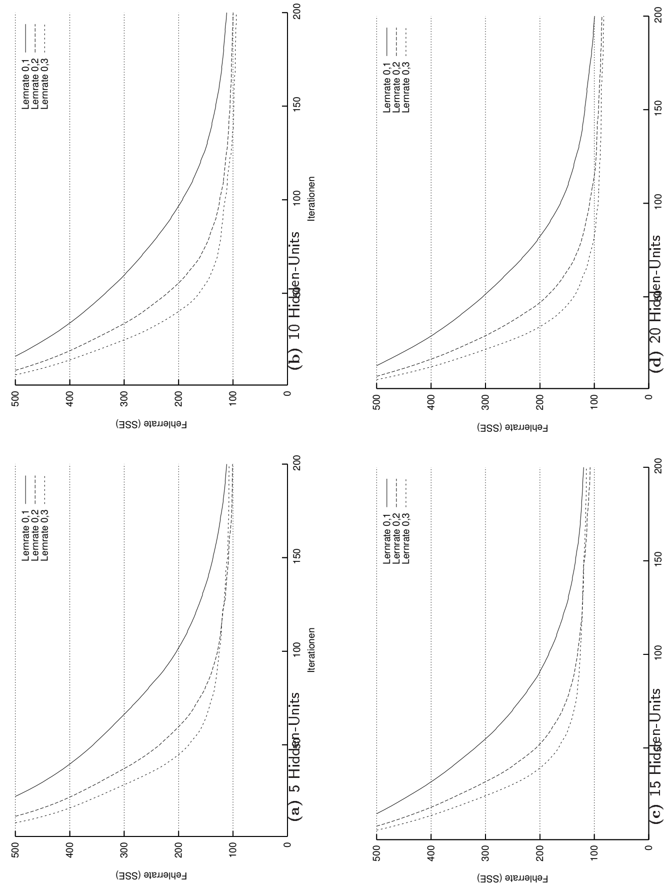
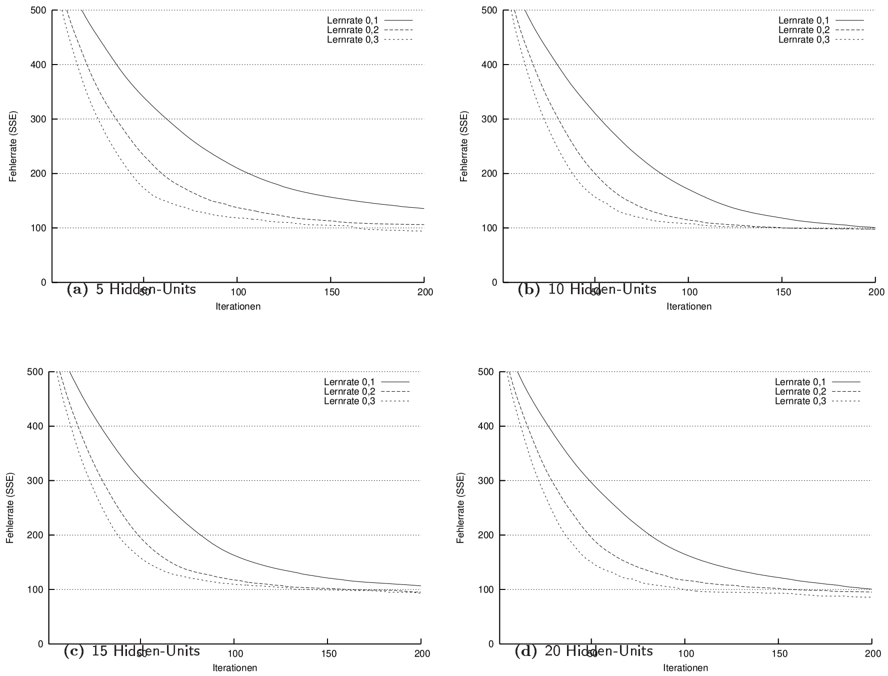
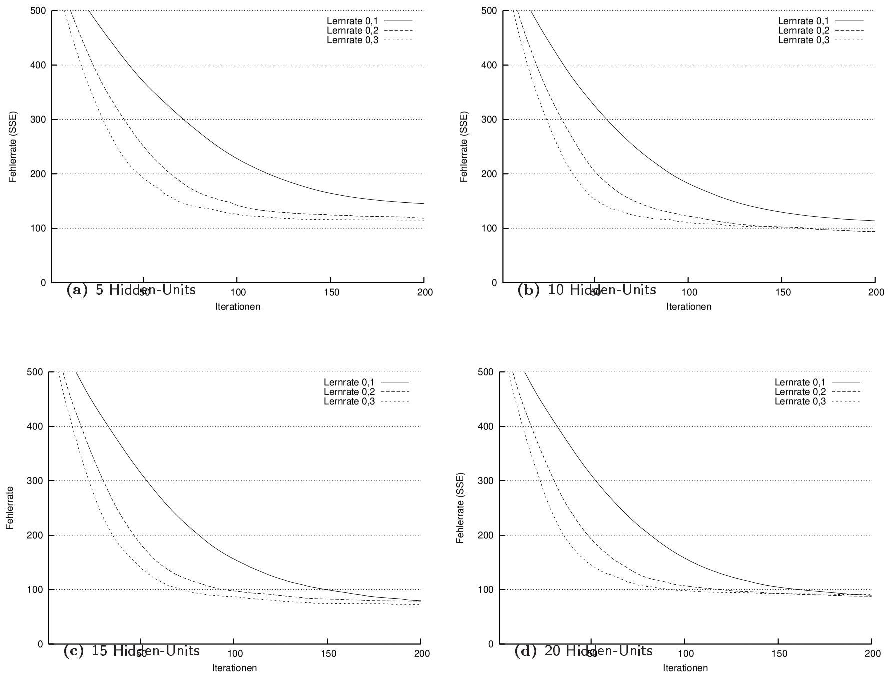


Abbildung C.4: Backpropagation Momentum ohne Berücksichtigung der Vortage, 1 Output-Unit, 1.044 Schlüsselwörter, jogging weights



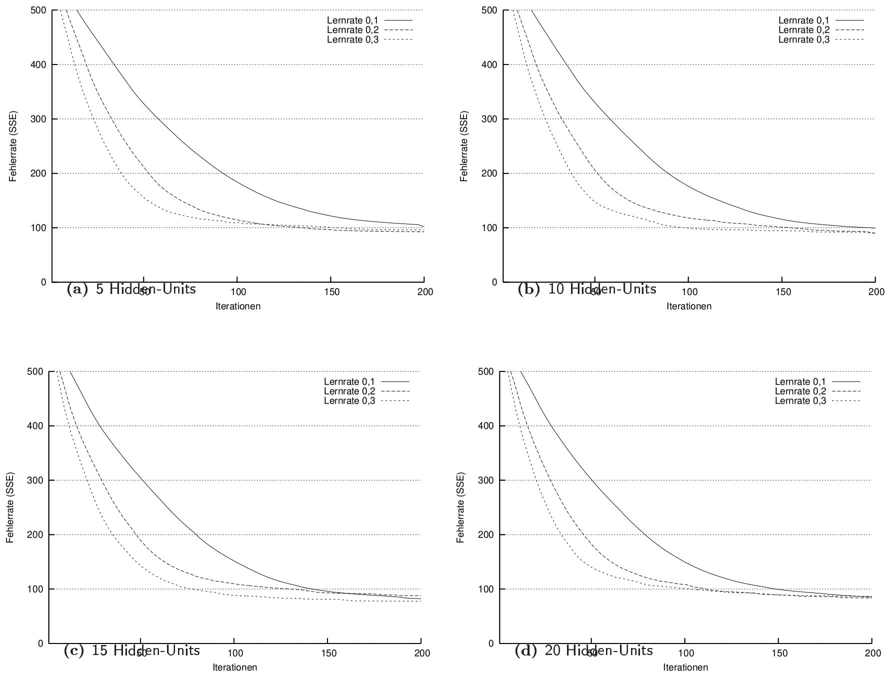
(a) 5 Hidden-Units (b) 10 Hidden-Units (c) 15 Hidden-Units (d) 20 Hidden-Units

Abbildung C.5: Backpropagation Momentum mit Berücksichtigung eines Vortages, 1 Output-Unit, 1.044 Schlüsselwörter



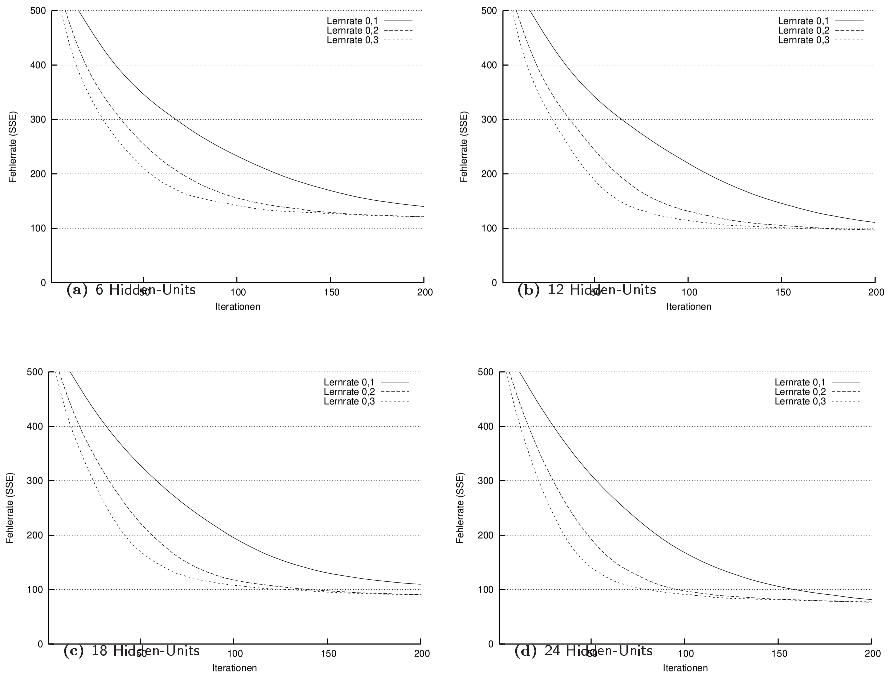
(a) 5 Hidden-Units (b) 10 Hidden-Units (c) 15 Hidden-Units (d) 20 Hidden-Units

Abbildung C.6: Backpropagation Momentum mit Berücksichtigung zweier Vortage, 1 Output-Unit, 1.044 Schlüsselwörter



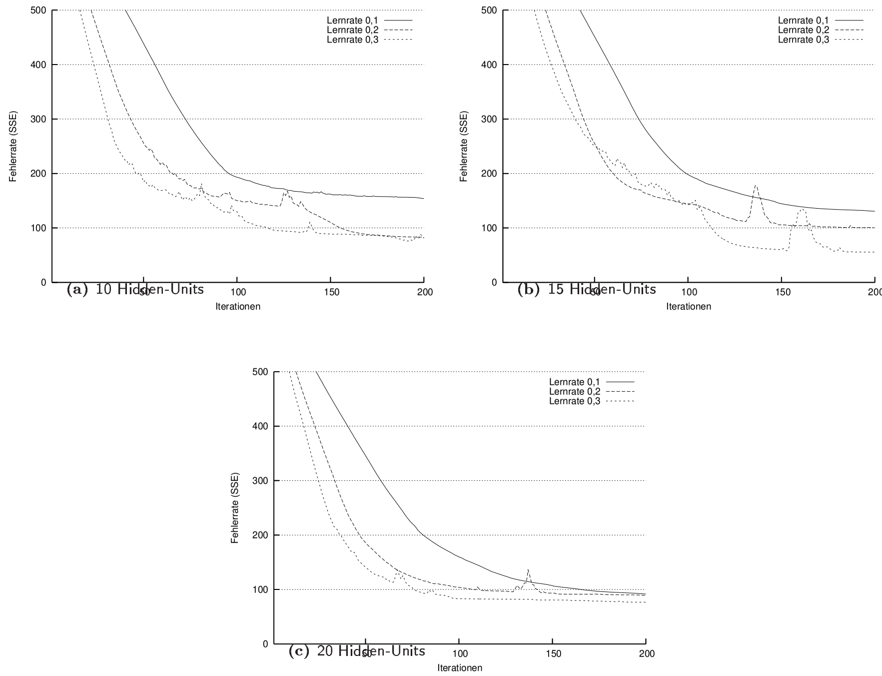
(a) 5 Hidden-Units (b) 10 Hidden-Units (c) 15 Hidden-Units (d) 20 Hidden-Units

Abbildung C.7: Backpropagation Momentum mit Berücksichtigung dreier Vortage, 1 Output-Unit, 1.044 Schlüsselwörter



(a) 6 Hidden-Units (b) 12 Hidden-Units (c) 18 Hidden-Units (d) 24 Hidden-Units

Abbildung C.8: Backpropagation Momentum ohne Berücksichtigung der Vortage, 3 Output-Units, 1.044 Schlüsselwörter

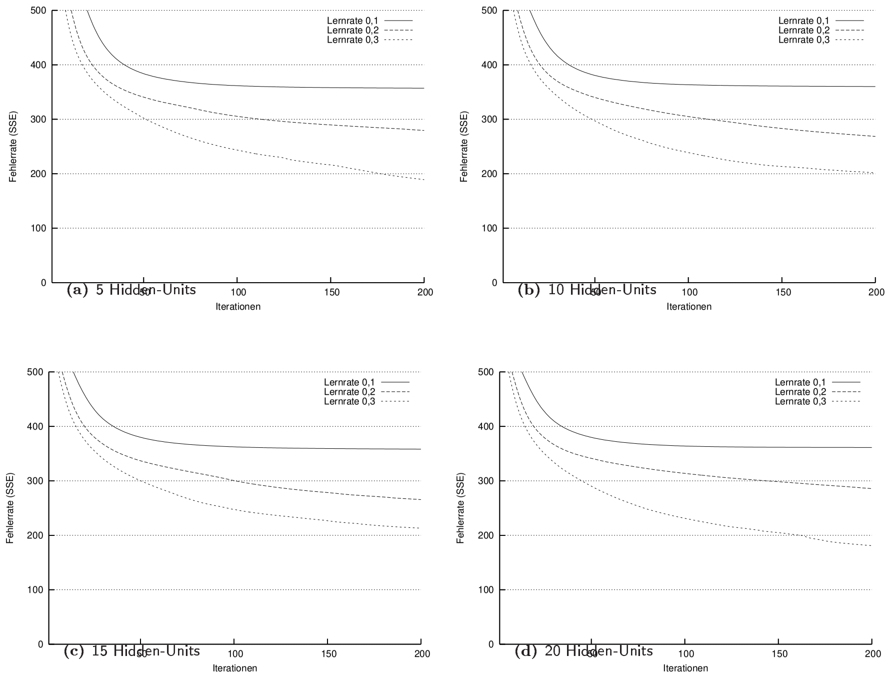


(a) 10 Hidden-Units

(b) 15 Hidden-Units

(c) 20 Hidden-Units

Abbildung C.9: Backpropagation Momentum ohne Vortage, 4 Layer, 1 Output-Unit, 1.044 Schlüsselwörter



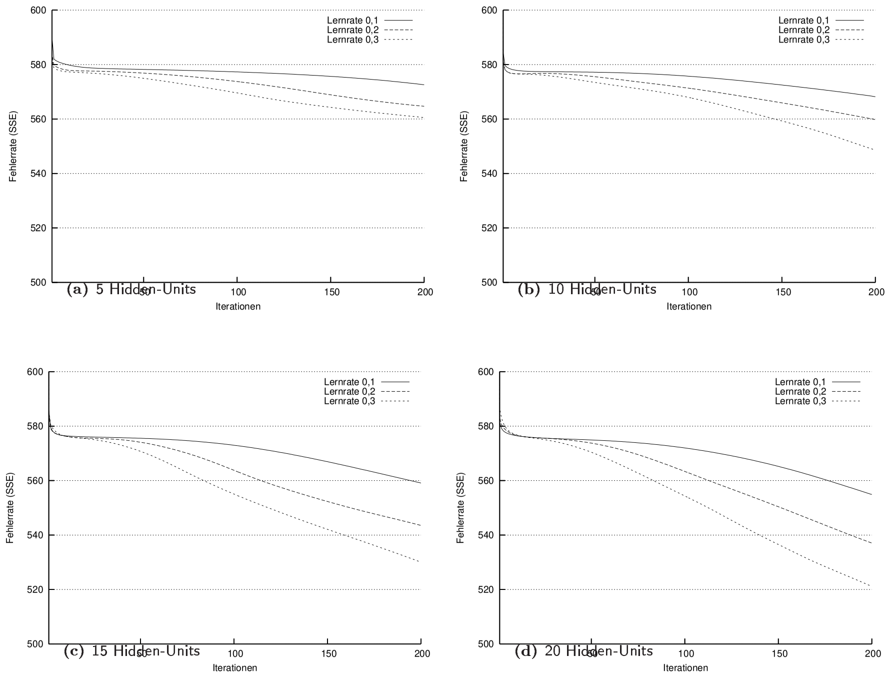
(a) 5 Hidden-Units

(b) 10 Hidden-Units

(c) 15 Hidden-Units

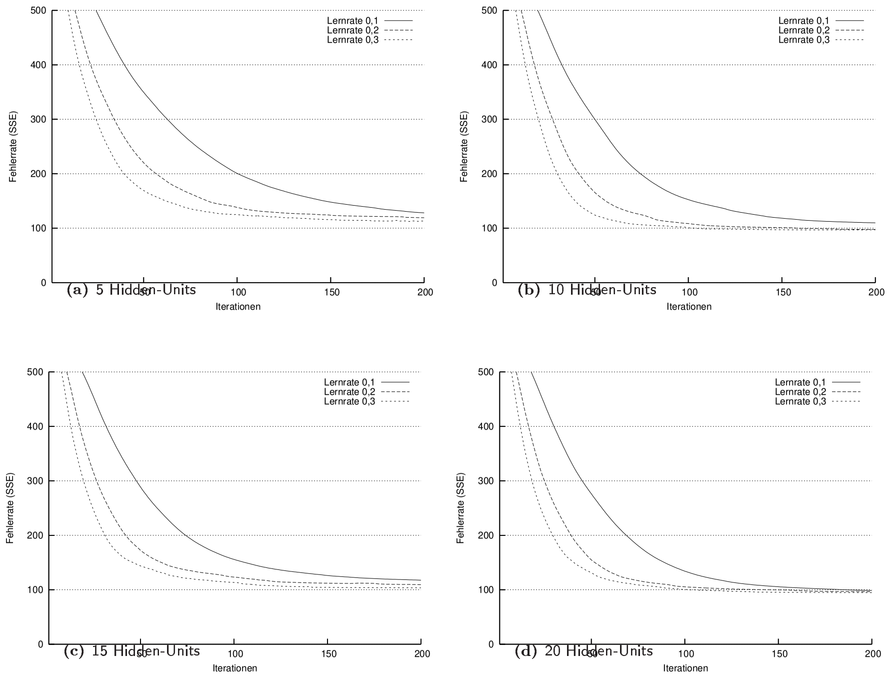
(d) 20 Hidden-Units

Abbildung C.10: Backpropagation Weight Decay ohne Berücksichtigung der Vortage, 1 Output-Unit, 1.044 Schlüsselwörter



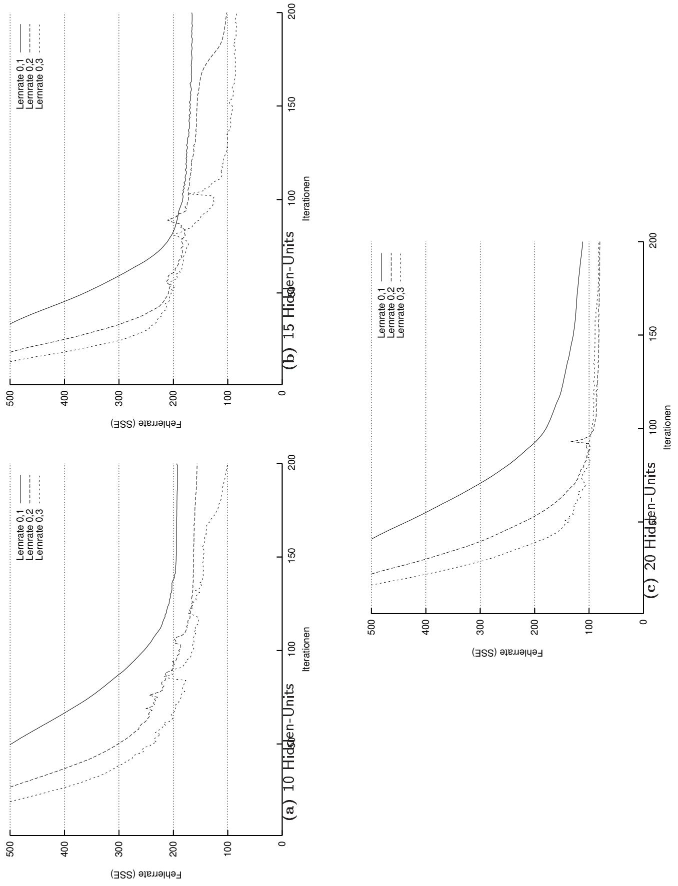
(a) 5 Hidden-Units (b) 10 Hidden-Units (c) 15 Hidden-Units (d) 20 Hidden-Units

Abbildung C.11: Backpropagation Momentum mit zwanzig Kategorien, 1 Output-Unit, 1.044 Schlüsselwörter



(a) 5 Hidden-Units (b) 10 Hidden-Units (c) 15 Hidden-Units (d) 20 Hidden-Units

Abbildung C.12: Backpropagation Momentum ohne Berücksichtigung der Vortage, 1 Output-Unit, 1.075 Schlüsselwörter



(a) 10 Hidden-Units

(b) 15 Hidden-Units

(c) 20 Hidden-Units

Abbildung C.13: Backpropagation Momentum ohne Berücksichtigung der Vortage, 4 Layer, 1 Output-Unit, 1.075 Schlüsselwörtern

C.3 Test des trainierten neuronalen Netzes

Die beiden folgenden Abbildungen basieren auf den Beschreibungen aus dem Kapitel 6.10.2, Seite 60.

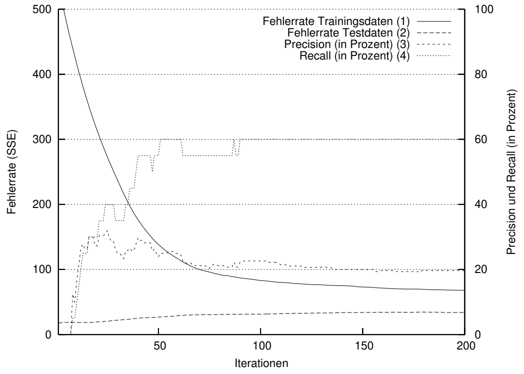


Abbildung C.14: Test des trainierten neuronalen Netzes mit drei Output-Units unter Berücksichtigung von drei Vortagen

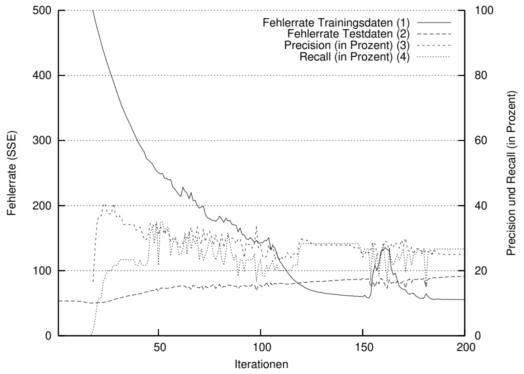


Abbildung C.15: Test des trainierten neuronalen Netzes mit vier Layern

D Beispiele für *APA*-Artikel

Dieses Kapitel enthält zwei zufällig ausgewählte, etwa gleich lange Artikel der *APA* vom Jänner 1999. Der erste Artikel wurde nicht im STANDARD veröffentlicht, der zweite schon. Weiters wurden die Schlüsselwörter[1] mit Hilfe von Großbuchstaben kenntlich gemacht.

D.1 Beispiel eines unveröffentlichten Artikels

```
APA0037 5 II 0123                                   10.Jän 99

WAHL/NATIONALRAT/FPÖ/FRAUEN

FRAUEN als NEUE FPÖ-ZIELGRUPPE

Utl.: HAIDER WENIGER "HARTE" THEMEN im WAHLKAMPF als 1995 =

WIEN (APA) - Die FPÖ will sich bei den BEVORSTEHENDEN WAHLKäMPFEN
gezielt um FRAUENSTIMMEN bemühen. Dies BERICHTET das
NACHRICHTENMAGAZIN "PROFIL" in seiner am Montag erscheinenden Ausgabe
unter BERUFUNG auf FPÖ-OBMANN JöRG HAIDER und GENERALSEKRETäR Peter
WESTENTHALER. Bei den NATIONALRATSWAHLEN von 1995 waren nur 38
PROZENT der FPÖ-STIMMEN von FRAUEN gekommen, dies will die FPÖ-SPITZE
nun ändern. ****

"Den FEHLER von 1995, nur 'HARTE' THEMEN im WAHLKAMPF zu
präsentieren, machen wir SICHER NICHT", umreißt HAIDER die NEUE
Linie. Man wolle "FRAUEN in ihrer VIELFäLTIGEN Zwangslage
ansprechen". Das FPÖ-GENERALSEKRETARIAT erarbeite derzeit eine
ZITATENSAMMLUNG, mit deren HILFE BUNDESKANZLER Vikrot KLIMA und
SPÖ-BUNDESGESCHäFTSFüHRER ANDREAS RUDAS "FRAUENFEINDLICHE Äußerungen"
nachgewiesen werden sollen.

(SCHLUß) mk/me

APA0037   1999-01-10/09:52

100952 Jän 99
```

[1] Es wurde die Menge der 1.044 manuell selektierten Schlüsselwörter verwendet.

D.2 Beispiel eines veröffentlichten Artikels

```
APA0203 5 II 0143 XI                                   11.Jän 99

UNIVERSITäT/REFORM

Zwei Drittel der UNIS im NEUEN ORGANISATIONSRECHT

Utl.: TU WIEN STARTETE mit JAHRESBEGINN ins UOG '93 =

   WIEN (APA) - An der Technischen UNIVERSITäT (TU) WIEN TRAT mit
JAHRESBEGINN das NEUE UNIVERSITäTSORGANISATIONSGESETZ (UOG) '93 in
KRAFT. Damit ARBEITEN BEREITS zwei Drittel der INSGESAMT zwölf
HEIMISCHEN UNIVERSITäTEN nach dem NEUEN ORGANISATIONSRECHT, das den
HOCHSCHULEN MEHR Autonomie einräumt. ****

   Nach Angaben des LEITERS der HOCHSCHULSEKTION im
WISSENSCHAFTSMINISTERIUM, Sigurd Höllinger, sollen noch im Lauf
dieses JAHRES die UNIVERSITäTEN GRAZ, INNSBRUCK und SALZBURG auf das
UOG '93 umstellen. Die UNIVERSITäT WIEN folgt voraussichtlich Anfang
2000.

   An der TU WIEN stehen mit INKRAFTTRTEEN des UOG '93 dem REKTOR
PETER SKALICKY drei VIZEREKTOREN zur Seite: Als VIZEREKTOR für Lehre
ist Hans Kaiser TäTIG, Franz G. Rammerstorfer agiert als VIZEREKTOR
für FORSCHUNG. Gerhard Schimak ist NICHT nur als VIZEREKTOR für
Ressourcen zuständig, SONDERN ZUGLEICH Stellvertreter des REKTORS.

 (SCHLUß) cm/wea/je

APA0203   1999-01-11/12:02

111202 Jän 99
```

E Literaturverzeichnis

[Brause (1995)] BRAUSE, RÜDIGER: *Neuronale Netze. Eine Einführung in die Neuroinformatik*, 2. Auflage, Teubner, Stuttgart, 1995

[Burkart, Hömberg (1995)] BURKART, ROLAND und HÖMBERG, WALTER (Hg.): *Kommunikationstheorien*, Reihe Studienbücher zur Publizistik- und Kommunikationswissenschaft, Band 8, Verlag Wilhelm Braumüller, Wien, 1995

[Dittenbach, Berger, Merkl (2004)] DITTENBACH, MICHAEL; BERGER, HELMUT und MERKL, DIETER: *Improving Domain Ontologies by Mining Semantics from Text* in *Proceedings of the 1st Asia-Pacific Conference on Conceptual Modelling (APCCM 2004)*, Dunedin, New Zealand, 2004

[Erbring (1989)] ERBRING, LUTZ: *Nachrichten zwischen Professionalität und Manipulation. Journalistische Berufsnormen und politische Kultur*, in: [Gottschlich, Langenbucher (1999), S. 155ff.]

[Friedl (2003)] FRIEDL, JEFFREY: *Reguläre Ausdrücke*, 2. Auflage, O'Reilly, Köln, 2003

[Gottschlich, Langenbucher (1999)] GOTTSCHLICH, MAXIMILIAN und LANGENBUCHER, WOLFGANG (Hg.): *Publizistik- und Kommunikationswissenschaft*, Reihe Studienbücher zur Publizistik- und Kommunikationswissenschaft, Band 1, Verlag Wilhelm Braumüller, Wien, 1999

[Holzer (1973)] HOLZER, HORST: *Massenkommunikation als Kapitalverwertungsprozess und die Rolle des Publikums*, in: [Burkart, Hömberg (1995), S. 69ff.]

[Jang, Sun, Mizutani (1997)] JANG, JYH-SHING ROGER; SUN, CHUEN-TSAI und MIZUTANI, EIJI: *Neuro-Fuzzy and Soft Computing. A Computational Approach to Learning and Soft Computing*, Prentice Hall, Upper Saddle River, New Jersey, 1997

[Karagiannis, Telesko (2001)] KARAGIANNIS, DIMITRIS und TELESKO, RAINER: *Wissensmanagement. Konzepte der künstlichen Intelligenz und des Softcomputing*, Oldenbourg Verlag, München, Wien, 2001

[Köhle (1990)] KÖHLE, MONIKA: *Neurale Netze*, Springer-Verlag, Wien, New York, 1990

[Kohonen (2001)] KOHONEN, TEUVO: *Self-Organizing Maps*, 3. Auflage, Springer, Berlin, Heidelberg, 2001

[Kratzer (1991)] KRATZER, KLAUS PETER: *Neuronale Netze. Grundlagen und Anwendungen*, 2. Auflage, Carl Hanser Verlag, München, Wien, 1993

[Kruse, Mangold, Mechler, Penger (1991)] KRUSE, HILGER; MANGOLD, ROLAND; MECHLER, BERNHARD und PENGER, OLIVER: *Programmierung Neuronaler Netze. Eine Turbo Pascal Toolbox*, Addison-Wesley, Bonn, München, 1991

[Lagus, Kaski (1999)] LAGUS, KRISTA und KASKI, SAMUEL: *Keyword Selection Method for Characterizing Text Document Maps*, Helsinki University of Technology, 1999 (`http://lib.tkk.fi/Diss/2000/isbn9512252600/article5.pdf`, 3. Juni 2005)

[Lagus (2000)] LAGUS, KRISTA: *Text Mining with the WEBSOM*, Dissertation at the Department of Computer Science and Engineering at Helsinki University of Technology, Finland, 2000 (`http://lib.tkk.fi/Diss/2000/isbn9512252600/`, 3. Juni 2005)

[Mechler (1995)] MECHLER, BERND: *Intelligente Informationssysteme*, Addison-Wesley, Bonn, 1995

[Noelle-Neumann, Schulz, Wilke (2000)] NOELLE-NEUMANN, ELISABETH; SCHULZ, WINFRIED und WILKE, JÜRGEN (Hg.): *Publizistik. Massenkommunikation*, Verlag Fischer Taschenbuch, 6. Auflage, Frankfurt am Main, 2000

[Rauber (2000)] RAUBER, ANDREAS: *Digital Libraries or The Art of Storing, Drawing, and Exploring Document Collections*, Dissertation an der Fakultät für Technische Naturwissenschaften und Informatik der Technischen Universität Wien, 2000

[Ruge (1995)] RUGE, GERDA: *Wortbedeutung und Termassoziation. Methoden zur automatischen Klassifikation*, Reihe Sprache und Computer, Band 14, Georg Olms Verlag, Hildesheim, 1995

[Salton (1989)] SALTON, GERARD: *Automatic Text Processing. The Transformation, Analysis, and Retrieval of Information by Computer*, Addison-Wesley, Massachusetts, 1989

[Sebastiani (2002)] SEBASTIANI, FABRIZIO: *Machine Learning in Automated Text Categorization*, ACM Computing Surveys, Vol. 34, 2002

[Schöneburg, Hansen, Gawelczyk (1990)] SCHÖNEBURG, EBERHARD; HANSEN, NIKOLAUS und GAWELCZYK, ANDREAS: *Neuronale Netzwerke. Einführung, Überblick und Anwendungsmöglichkeiten*, Markt&Technik Verlag, München, 1990

[Spitzer (2000)] SPITZER, MANFRED: *Geist im Netz. Modelle für Lernen, Denken und Handeln*, Spektrum Akademischer Verlag, Heidelberg, Berlin, 2000

[Staab (1990)] STAAB, JOACHIM FRIEDRICH: *Nachrichtenwert-Theorie: Formale Struktur und empirischer Gehalt*, Reihe Alber-Broschur Kommunikation, Band 17, Verlag Karl Alber, München, 1990

[Ultsch, Siemon (1990)] ULTSCH, ALFRED und SIEMON, H. PETER: *Kohonen's Self Organizing Feature Maps for Exploratory Data Analysis*, in Proceedings of International Neural Networks, 305–308, Kluwer Academic Press, Paris, 1990

[Wimmer (1996)] WIMMER, ALEXANDER: *Die EU-Berichterstattung in Standard und Presse anhand der Reflexionshypothese der Agenda-Setting Forschung*, Diplomarbeit an der Grund- und Integrativwissenschaftlichen Fakultät der Universität Wien, 1996

[Yang, Pedersen (1997)] YANG, YIMING und PEDERSEN, JAN: *A Comparative Study on Feature Selection in Text Categorization*, `http://nyc.lti.cs.cmu.edu/yiming/Publications/icml97.ps`, 3. Juni 2005

[Zell (1994)] ZELL, ANDREAS: *Simulation neuronaler Netze*, Addison-Wesley, Bonn, 1994

[Zell (1995)] ZELL, ANDREAS: *SNNS – Stuttgart Neural Network Simulator. User Manual, Version 4.2*, Report No. 6/95, `http://www-ra.informatik.uni-tuebingen.de/SNNS/`, 3. Juni 2005

[Zimmermann (1995)] ZIMMERMANN, HANS-JÜRGEN (Hg.): *Datenanalyse. Anwendung von DataEngine mit Fuzzy Technologien und Neuronalen Netzen*, VDI-Verlag, Düsseldorf, 1995

F Index