Bachelorarbeit

Hochschule Merseburg

FB Ingenieur- und Naturwissenschaften

Lucas Imhof

Geboren am 02. April 1997 in Querfurt

Matrikel-Nr.: 22374

Information Retrieval für eine Web Suchmaschine mithilfe von neuronalen Netzen und klassischen Methoden.

Erstprüfer: Prof. Dr. rer. pol. Uwe Schröter

Zweitprüfer: Prof. Dr. rer. nat. habil. Dr. phil. Michael Schenke

Datum: *11.03.2019*

Inhaltsverzeichnis

1 Einleitung

Die vorliegende Bachelorarbeit befasst sich mit dem Information Retrieval für eine Web Suchmaschine. Die Qualität, der durch das Information Retrieval gesammelten Daten, bestimmt maßgeblich die Effizienz der Suchmaschine, die mithilfe dieser Daten einem Anwender Suchergebnisse liefern soll. Qualität bedeutet, dass ein Anwender ein Ergebnis erhält, welches für seine aktuelle Aufgabe zielführend ist. Zielführend bedeutet in diesem Zusammenhang, dass der Anwender ein Ergebnis erhält, welches aus Daten besteht, die sowohl thematisch wie auch inhaltlich zu seiner Suchanfrage passen. Dabei ist es auch zwingend erforderlich darauf zu achten, die gefunden Ergebnisse abhängig von verschiedenen Faktoren, die in dieser Arbeit auch besprochen werden, korrekt zu priorisieren. Eine besondere Herausforderung stellt dabei die Bildersuche dar, da bei dieser nicht einfach nach Schlagworten gesucht werden kann, sondern der Inhalt des Bildes zunächst vom Information Retrieval-System analysiert und kategorisiert werden muss.

Parallel zu dieser schriftlichen Arbeit ist ein Prototyp für eine Suchmaschine entstanden. Dieser Prototyp besteht aus einem Crawler Netzwerk, welches Informationen von Webseiten, Bildern oder auch Dokumenten extrahiert und diese mithilfe eines Servers in einer Datenbank abspeichert. Zu dem besteht der Prototyp aus einer Suchmaschine, mithilfe dessen man nach Textfragmenten und Bildern suchen kann, die von dem Crawler Netzwerk gefunden wurden.

In den Kapiteln „2.1 Hyperlinks“, „2.2 MIME Types“, „2.3 OCR – Optical Character Recognition“ und „2.4 Geschichte von Suchmaschinen“ werden alle notwendigen Grundlagen, die für das Verständnis dieser Arbeit benötigt werden, erläutert. Dies soll dem besseren Verständnis, der infolge besprochen Themen, helfen. Im Kapitel 3 wird der Prozess des Information Retrievals erläutert und vorgestellt wie dieser in der Praxis von Suchmaschinengiganten eingesetzt wird. Kapitel 4 befasst sich infolge dessen mit neuronalen Netzen, da diese vielseitig für Information Retrieval eingesetzt werden können. Dazu wird zunächst erläutert, was neuronale Netze sind, welche Arten es gibt und wie diese funktionieren. Auch soll kurz vorgestellt werden, wie diese konkret im Information Retrieval angewandt werden können. Das letzte Kapitel befasst sich mit dem parallel zur Arbeit entstanden Prototypen. Bei diesem handelt es sich um ein vollständiges Retrieval-System, welches aus einem Crawler-Netzwerk und einem Suchmaschinen-Frontend besteht. Abschließend wird eine Zusammenfassung gegeben, in der die Zukunftsaussichten und Marktfähigkeit des Prototypen beleuchtet werden. Auch wird erklärt, was eine Suchmaschine theoretisch leisten müsste, um gegen die Marktgiganten ankommen zu können.

2 Grundlagen

2.1 Hyperlinks

Hyperlinks, kurz Links, sind Verknüpfungen im Internet zwischen verschieden Webseiten oder Trigger (Auslöser) für Funktionen auf einer Webseite. Standardmäßig sind diese blau hinterlegt und unterstrichen. Grundsätzlich unterscheidet man drei wesentliche Typen von Links. (HTMLSemi3) (RyteWiki7)

- **Interne Links** (Verweis innerhalb der Seite)
- **Externe Links** (Links auf andere Seiten im Internet)
- **Links mit anderen Funktionen** (z.B.: Starten eines Emailprogrammes)

Das Vorkommen von Links auf Webseiten kann sehr vielschichtig sein. Meist sollten diese als Text daherkommen, in ihrer blauen Schrift und unterstrichen. Jedoch sind auch Links hinter Grafiken möglich und werden sehr häufig verwendet.(HTMLSemi3) (RyteWiki7)

```
<a href="home.html">Zur Hauptseite</a>
```

Das hier gezeigte Quellcode Fragment, stellt zunächst eine einfache Verlinkung innerhalb einer Webseite dar. Dabei erscheint dann auf der Webseite der Text „Zur Hauptseite“ und sobald ein Nutzer auf diesen klickt, wird er zu „home.html“ weitergeleitet. Das „a“ in dem HTML-Tag steht dabei für „ancor“ (engl. für Anker). „href“ steht dabei für **H**yper**ref**erenz und stellt den Verweis dar. (HTMLSemi3) (RyteWiki7)

```
<a href="home.html"><img src="bild.png" alt="Bild"/></a>
```

Sollte man nun anstatt mit einem Text mit einem Bild auf eine andere Seite verlinken wollen kann man einfach Anstatt eines Textes ein Bild innerhalb des „a“-Tags einfügen. (HTMLSemi3)

Neben den drei genannten Typen gibt es noch relative und absolute Verknüpfungen. Relative Verlinkungen werden meist nur bei internen Verlinkungen verwendet sind aber sehr weit verbreitet. Der Vorteil dieser ist meist, dass diese zeitlich schneller umzusetzen sind als absolute Verlinkungen. Relativ heißt hierbei, dass eine Verlinkung abhängig der aktuellen Position erstellt wird. Wo hingegen eine Absolute eine statische Verknüpfung ist, die von jedem Ort aus gleich erreichbar ist. Im Folgenden ist eine Tabelle mit Beispielen für solche Verlinkungen zu sehen. Ausgehend von der URL „*HTTP://example.com/html/“.* (WebRef1)

Relative Verlinkung	Absolute Verlinkung
about.html	http://example.com/html/about.html
tutorial1/	http://example.com/html/tutorial1/
tutorial1/2.html	http://example.com/html/tutorial1/2.html
/	http://example.com/
//www.internet.com/	http://internet.com/
/experts/	http://example.com/experts/
../	http://example.com/
../experts/	http://example.com/experts/
../../../	http://example.com/
./	http://example.com/
./about.html	http://example.com/html/about.html

Tabelle 1: Arten von Links (nach WebRef1)

Dabei handelt es bei allen Verlinkungen, abgesehen von einer Ausnahme, um interne Verlinkungen. Ein Sonderfall ist „//www.internet.com/“, da dies die einzig mögliche Form von Relativen Verlinkungen auf externe Webseiten darstellt. Dabei stellt „//“ eine verkürzte Schreibweise für „http://“ dar. (WebRef1)

2.2 MIME Types

MIME steht für „Multipurpose Internet Mail Extensions“. Sie werden dafür eingesetzt im Internet Dateiformate zur erkennen, um dann die entsprechende Browsererweiterung zu starten die diese Datei lesen und/oder verarbeiten kann. Die MIME-Types werden üblicherweise bei HTTP-Requests im Header im Feld „Content-type“ übertragen. Ein „Content-type“ ist ein Protokollheader, der den Inhalt eines Dokumentes bzw. einer Datei kennzeichnet. Häufig wird in Verbindung mit MIME-Types auch der Begriff „Internet Media Type“ genannt, was nur einen anderen Begriff zu „Multipurpose Internet Mail Extensions“ darstellt. Ursprünglich wurden MIME-Types nur für das SMTP-Protokoll entwickelt und sollten beim Versand von Emails helfen(W3.ORG1). Ein MIME-Type besteht immer aus einem Type und einem Subtype. Der Type ordnet die Datei einer logischen MIME Gruppe zu und der Subtype einem Dateiformat. Ein Beispiel für einen MimeType ist *„image/png“*. Dabei stellt „image“ den Typen dar und „png“ den Subtype. Unter Umständen können MIME-Types weitere Präfixe enthalten. Einer dieser Präfixe kann „x-“ sein. Dieser Präfix bedeutet, dass der aktuelle MIME-Subtype keinem offiziellen Standard der **IANA** (**I**nternet **A**ssigned **N**umbers **A**uthority) zugehörig ist. Ein weiteres Präfix ist „vnd“ und definiert herstellerspezifische Dateiformate. Ein Beispiel hierfür ist ein Adobe Flex Projekt, dessen MIME-Type mit “application/vnd.adobe.fxp“

beschrieben wird. Wenn man nun als Entwickler selbst den MIME-Type seiner Datei bestimmen möchte, gibt es hierfür verschiedene Methoden:

- **Asp-Webanwendungen:** *<% response.ContentType="text/html" %>*,
- **C#:** *response.ContentType = „text/plain";*,
  (mit Verweis auf die Libary System.Web.HTTPResponse),
- **Java:** *response.setContentType(„text/plain");*
  (mit Verweis auf die Libary javax.servlet.ServletReponse),
- **Perl/CGI:** *print „Content-type:text/html";*,
- **PHP:** *<?php header (‚Content-type: text/html');?>*. (FreeForm1)

Die Anzahl der offiziell registrierten MIME-Types ist riesig und wächst stetig an. Zum Zeitpunkt dieser Arbeit lagen 1326 registrierte MIME-Types allein für den Type „application" vor. (IANA1) Zudem ist es jederzeit möglich, neue MIME-Types zu registrieren. Dafür steht ein ausführliches Formular auf der IANA Seite zur Verfügung. (IANA2)

2.3 OCR – Optical Character Recognition

Als OCR bezeichnet man die Erkennung von Buchstaben auf einem digitalen Medium, meist Bilder, durch eine Maschine. Erste Experimente mit OCR gab es bereits in den 1960 Jahren am **MIT** (**M**assachusetts **I**nstitute of **T**echnology) durch Lawrence Roberts, der Versuche zu automatischen Zeichenerkennung unternahm. Die erste Anwendung in der Praxis fand 1965 statt, jedoch noch in Form einer Hardwarelösung. Auch war zu diesem Zeitpunkt die Erkennung auf wenige eigens entwickelte Schriftarten beschränkt. Im Jahre 1976 wurde durch Ray Kurzweil, die erste Schriftartenunabhängige OCR-Erkennung umgesetzt. Durch die immer weiter ansteigende Computerleistung wurde über die letzten Jahre eine immer genauere Analyse möglich. (OCR1) In der Regel werden zunächst alle Bilddatei in Rasterdaten umgewandelt. Diese haben den Vorteil, dass sie Pixelweise analysiert werden können. Veraltete Varianten des OCR bauten darauf auf, ein Bild in verschiede Bereiche zu unterteilen. Dabei wurde das Bild in Bereiche unterteilt, in denen sich Text befindet und in Bereiche wo sich kein Text befindet, um dann im Anschluss mithilfe eines Mustervergleiches zu versuchen Buchstaben zu erkennen. Dieses Verfahren war jedoch sehr mühsam und hat häufig zu Fehlern geführt. Moderne OCR-Verfahren setzen nun auf den Einsatz von künstlichen neuronalen Netzen, die mithilfe eines Backpropagtion-Algorithmuses, der später genauer erläutert wird, den Inhalt eines Bildes zu analysieren. Hierfür wurden die Netze nur speziell auf die Erkennung von Buchstaben und Buchstabenverknüpfungen trainiert. (Wiki6)

2.4 Geschichte von Suchmaschinen

Mit über 90% ist Google heutzutage die meist genutzte Suchmaschine der Welt. (LP1) Diese Dominanz geht soweit, dass selbst googeln, mit Google im Internet suchen, 2004 im Deutschen Duden aufgenommen wurde und im allgemeinen Sprachgebrauch als Synonym für eine Recherche im Internet genutzt wird. (Wik1) Jedoch gab es schon vor Google das Internet mit verschiedenen Suchmaschinen. Einige von diesen, wie zum Beispiel Yahoo!, sind heute noch auf dem Suchmaschinenmarkt vertreten, aber ein Großteil dieser Suchmaschinen sind heute nicht mehr existent. Im Folgenden soll nun kurz an einigen Beispielen erläutert werden, welche Suchmaschinen es in der Vergangenheit gab und welche Suchstrategien diese verfolgten, beziehungsweise verfolgen. Nachdem das Internet in den Jahren 1993/1994 einen wahrlichen Boom am Aufkommen neuer Webseiten erfuhr, kam der Bedarf nach Suchmaschinen auf. Jedoch wurde bereits 1990 an der McGill Universität in Montreal die erste Suchmaschine „Archie“ entwickelt. Diese war jedoch nicht in der Lage, Fließtexte zu analysieren, sondern konnte nur Ordner und Files in FTP-Verzeichnissen aufspüren. Damit war diese für die neuaufkommenden HTTP-Webseiten ungeeignet (NetPlanet1). Infolgedessen entstand ein enormer Bedarf nach neuen Suchmaschinen, die in den folgenden Jahren zwischen 1994 und 1997 rasant aus dem Boden schießen sollten. Eine der ersten Suchmaschinen, die Volltextsuche unterstütze, war AltaVista. AltaVista ist eine Suchmaschine, die aus einem Forschungsprojekts Ende 1995 hervorging. Bevor Google die Marktmacht an sich reiste, war diese eine der bekanntesten Volltext-Suchmaschinen. Für die Suche wurde ein Ranking-Algorithmus verwendet, der hauptsächlich die Meta-Tags eines Webseite und Textfragmente, die nach einer internen Logik gesammelt wurden, berechnete. Nachdem AltaVista von Compaq übernommen wurde, nutzten diese die Marke AltaVista nur noch für Marketingzwecke, um zu demonstrieren wie stark ihre Server sind. Im Jahre 2003 wurde die angeschlagene Suchmaschine von dem Unternehmen aufgekauft, welches ein Subunternehmen der Yahoo! Inc. darstellte (ORF1). Am achten Juli 2013 wurde dann AltaVista durch Yahoo! geschlossen und alle Suchanfragen zur Yahoo-Search weitergeleitet (Heise1). Während viele Suchmaschinen um 1995 bereits mit Bots das Internet nach Webseiten durchsuchten, verfolgte Yahoo! zu diesem Zeitpunkt eine andere Strategie. Sie setzten darauf, alle Webseiten in Kategorien händisch in einen Katalog einzutragen. Dieses System war aber auf Dauer nicht marktfähig, da die Ergebnisse, die die Bots sammelten, weitaus besser waren. Aus diesem Grund kaufte Yahoo! Daten von Konkurrenzunternehmen an und entwickelte daraufhin 2004 eine eigene Suchmaschine. Später ging Yahoo! einen Kooperationsvertrag mit Bing ein und seitdem stehen der Yahoo-Search Ergebnisse von Bing und Yahoo! zur Verfügung (SZ1).

Doch nun stellt sich die Frage, was Google damals anders gemacht hat als die anderen Suchmaschinen, vormals sie erst 1997 an den Markt gegangen sind. Die Gründe für Googles Erfolg sind vielseitig. Aber einer der Hauptgründe waren wohl die guten Suchergebnisse in der Anfangszeit, die ihren Konkurrenten weitaus überlegen waren. Google nutzte damals wie auch noch heute die Popularität einer Webseite, um deren Ranking zu bestimmen. Dies hatte zur Folge, dass Spamseiten mit denen andere Suchmaschinen Anbieter zu kämpfen hatten in ihrer Priorität sehr weit nach hinten gefallen sind. Ein weiterer Grund für den Erfolg von Google war das schlichte Design. Während andere Anbieter ihre Startseite mit Werbung und News überfluteten zeigt Google nur ihre Suchmaske auf der Startseite an (SZ1).

Ist nun Google die unabdingbare Macht am Suchmaschinenmarkt? Wie sich zeigt, wagt sich niemand wirklich an das Thema der Websuche in den letzten Jahren heran, da es zu teuer und viel zu komplex ist. Zuletzt versuchte eine französisches Unternehmen mit Unterstützung der europäischen Union eine „Europäische-Suchmaschine“ zu erstellen die der amerikanischen Übermacht gegenüberstehen kann. Leider fand die Suchmaschine keinen großen gesellschaftlichen Zuspruch, woraufhin die EU im Jahr 2008 die finanziellen Mittel streichte. Das Projekt wurde infolgedessen im Jahr 2013 für endgültig beendet erklärt (Quaero1).

Somit bleibt Google wohl weiterhin unabdingbarer Marktführer.

3 Information Retrieval

3.1 Datenanalyse mittels Information Retrieval

Information Retrieval (IR) ist ein Fachgebiet der Informatik und Computerlinguistik, welches sich mit der computergestützten Suche nach komplexen Inhalten beschäftigt. Komplexe Inhalte hierbei können unter anderem Bilder oder auch ganze Textphrasen sein. Im Wesentlichen geht es beim Information Retrieval darum, bestehende Informationen aufzufinden und keine neuen Strukturen zu entdecken. (Wiki1)

Information Retrieval befasst sich im Kontext dieser Arbeit mit dem Beschaffen von Informationen und deren Analyse. Dabei ist hier das Ziel des Retrieval-Systems, die vorhandenen Informationen so vorbereitet abzuspeichern, damit ein Suchalgorithmus mit diesen Daten schnell Anfragen verarbeiten kann.

3.1.1 Elementare Suchfunktionen

Unter elementaren Suchfunktionen versteht man mathematische oder auch logische Verfahren, die dazu entwickelt wurden Informationen oder auch Daten fallspezifisch schnell aus einer großen Datenmenge rauszufiltern. (FHKOELN1)

Das ***Boolesche Retrieval*** erlaubt die Suche nach komplexen Suchthemen. So ist es möglich Daten zu analysieren, die aus einer Verbindung von mehreren Sachverhalten bestehen. Hierfür baut das Verfahren auf den drei Logischen Junktoren Negation (NOT), Konjunktion (AND) und Disjunktion (OR) auf. Auch ist dieses Verfahren ein matching-orientiertes Suchverfahren auf der Basis von Zeichenketten und somit ein wichtiger Kernbestandteil aller Information Retrieval-Systeme.

Wie in Abbildung 1 zu sehen ist, liefert die Suche nach einem bestimmten Begriff mithilfe des einfachen Matchings zwei Ergebnismengen zurück. In der einen Menge sind alle Elemente mit diesem Begriff enthalten und in der anderen Menge sind alle Elemente ohne diesen Begriff enthalten. Logisch folgt daraus, dass die Summe aus den beiden Mengen wieder die Anzahl der Elemente in der Datenbank ergeben muss. (FHKOELN1)

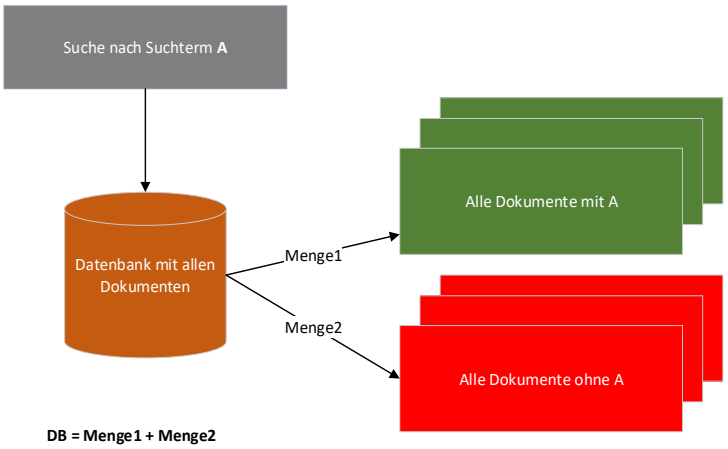

Abbildung 1: Beispiel für einfaches Matching mit einem Suchbegriff (nach FHKOELN1)

Wie in Abbildung 2 zu sehen ist, liefert eine Suche nach zwei Elementen, die mit einem OR verknüpft sind, wieder zwei Mengen zurück. So erhält man eine Menge die entsprechend beider Suchterme und deren Verknüpfung alle Elemente enthalten und eine, in dieser die Elemente nicht enthalten sind. Die Menge1 enthält wiederum zwei Teilmengen und eine Schnittmenge zwischen beiden Suchbegriffen. Elemente aus der Schnittmenge sind höher zu priorisieren als Elemente außerhalb dieser. (FHKOELN1)

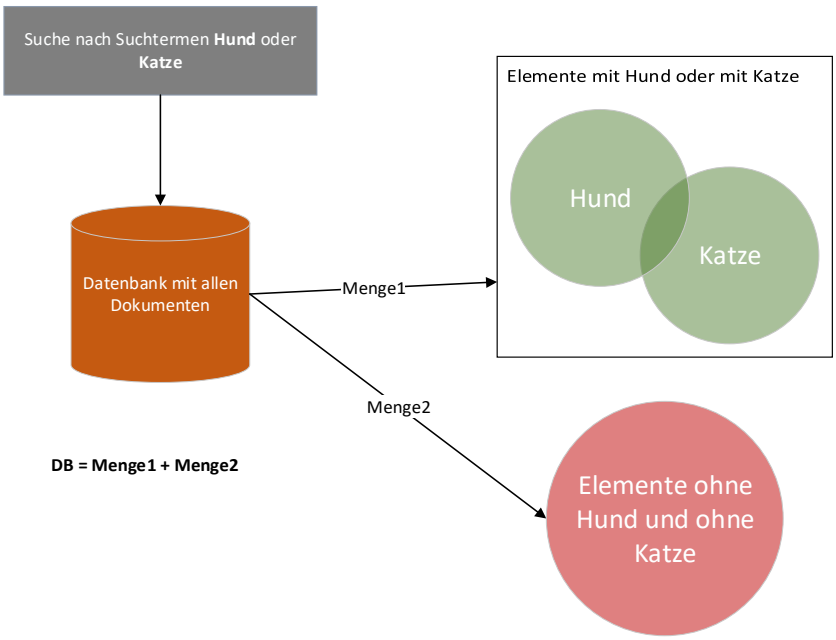

Abbildung 2: Suche nach 2 mit OR verknüpften Elementen (nach FHKOELN1)

Die ***Umgebungssuche*** (engl. ***Proximity Search***) baut darauf auf, dass zwischen mindestens 2 Suchtermen ein maximaler Abstand festgelegt wird. Anschließend wird in der Datenstruktur nach diesen Worten gesucht. Dabei darf der maximale Abstand zwischen den Worten nicht überschritten werden. Auch ist es egal in welcher Reihenfolge die Worte gesucht werden. Somit würde eine Suchanfrage mit den Worten

„**der**“ und „**Apfel**“ auch Ergebnisse finden wie „**der** *rote* **Apfel**“ oder „**Apfel der**“. (FHKOELN1)

Ein wesentlicher Vorteil dieses Verfahrens ist, dass durch das Eingeben mehrere Suchbegriffe auf engem Raum hypothetisch die Wahrscheinlichkeit erhöht wird einen Hinweis auf das gesuchte Thema oder Dokument zu finden. Auch werden durch das enge Suchen feldübergreifende Matchings vermieden. (FHKOELN1)

Mit „engen Raum“ bezeichnet man beim Information Retrieval den nahen Bereich um ein Textfragment. Dabei wird in der Regel ein maximaler Abstand fest definiert, wie weit die Grenzen dieses Abschnittes entfernt sein dürfen.

Der größte Nachteil dieser Methode, ist die Gefahr unter Umständen zu eng zu suchen. Ist die Matching-Distance zu gering gewählt, kann es passieren das man „**der**“ und „**Apfel**“ nicht in Kombination finden kann, wenn in einem Text „**der** *rote leckere und saftige* **Apfel**“ stehen würde. Die Matching-Distance beschreibt einen Zahlenwert, der bestimmt wie weit Wortverknüpfungen voneinander entfernt sein dürfen. (FHKOELN1)

Es existieren prinzipiell drei verschiedene Varianten der Umgebungssuche, die angewandt werden.

Die ***Nachbarschaftssuche*** (engl. Adjaceny Search) schränkt die Umgebungssuche auf unmittelbare Nachbarschaft und eindeutige Richtung ein. Das bedeutet, wenn man nach „**der**“ und „**Apfel**“ sucht findet man auch nur „**der Apfel**“ im Text. Eine weitere Variante ist die feldbezogene Umgebungssuche. Um diese Variante anwenden zu können, muss der Text zunächst in verschiedene Felder unterteilt werden. Idealerweise nutzt man hierfür direkt die Textabsätze falls vorhanden, da man annehmen kann, dass sich ein Absatz in Verbindung mit einem Wort im selben Zusammenhang befinden. Die satzbezogene Umgebungssuche baut darauf auf, dass sich die eingegebenen Suchterme im gleichen Satz befinden müssen. (FHKOELN1)

Die ***Phrasensuche*** ist eine weitere Methode des Information Retrieval. Im Grundlegenden baut diese Methode auf einem Sonderfall des einfachen Matchings auf, indem die Zeichenkette mehrere Wörter und Leerzeichen umfassen kann. Sie ist darauf ausgelegt, mehrere Begriffe als exakte Wortfolge der Eingabe entsprechend zu finden. In vielen Fällen wird für die Phrasensuche eine abgewandelte Form der Nachbarschaftssuche angewandt. Zu einem der größten Vorteile dieser Methode zählt die erhöhte Genauigkeit einzelner Suchanfragen gegenüber der Umgebungssuche. Diese Methode bietet außerdem die Möglichkeit, feststehende Wortwendungen in einem Schlagwortkatalog zu hinterlegen, was dazu führt, dass man trotz der eingeschränkten Suchmöglichkeiten bessere Ergebnisse erhalten kann. Ein großer Nachteil der

Phrasensuche ist, dass man unter Umständen potenzielle Treffer ausschließt, weil eine unbekannte Variante einer Phrase existiert. Dies tritt häufig bei Eigennamen auf, wenn diese in einer leicht abgewandelten Form eingegeben werden. (FHKOELN1)

3.1.2 Suchverfahren in Information Retrieval-Systemen

Bei der ***sequenziellen Suche*** wird eine gesamte Datenbank nach dem gewünschten Ergebnis durchsucht. Dabei untersucht die sequenzielle Suche alle vorhandenen Datensätze nacheinander. In Abbildung 3 ist zu sehen, dass bei einer sequentiellen Suche ein Feld nach dem anderen analysiert wird.

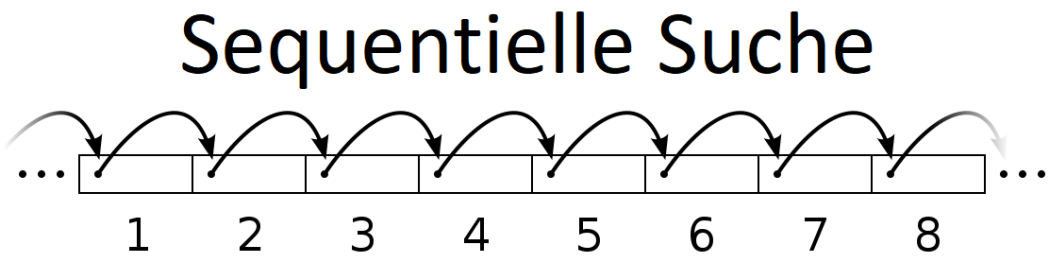

Abbildung 3: Sequenzielle Suche (Wiki7)

Sie beginnt dabei beim ersten Element und hört beim letzten Element auf. Diese Art der Suche hat jedoch die Konsequenz, dass die Performance der Suche mit wachsender Datenbankgröße immer langsamer wird, wenn nur eine unzureichende Indexierung vorhanden ist. Im Gegensatz zu anderen Suchsystemen, kann diese jedoch zu mehr und besseren Ergebnissen führen. Unter Indexierung versteht man, dass bestimmte Schlagwörter nach denen gesucht werden kann abgespeichert werden. (FHKOELN1)

Die ***indexbasierte Suche*** basiert darauf in kleinen Teilmengen eines Datenbestands Informationen zu finden, welche in einem Suchregister alphabetisch, oder in einer anderen Form, sortiert sind. Diese kleineren Datenmengen werden anhand eines übergebenen Index bestimmt. Ab diesem Zeitpunkt sollten die Datenmengen nur noch Informationen enthalten, die dem gesuchten Ergebnis ähnlicher sind als der restliche Datenbestand. Im Allgemeinen besitzt die Indexsuche eine sehr gute Performance, da immer nur kleinere sortierte Datenmengen durchsucht werden müssen. Sollten jedoch teile der Dokumente im Suchregister oder auch im Datenbestand nicht vollständig indexiert sein, besteht die Möglichkeit, dass man entweder nur sehr eingeschränkte Suchmöglichkeiten hat oder auch bei der Suche nur schlechtere Ergebnisse erhält. Aus diesem Grund bedarf es Techniken, die den Indexaufbau und die Indexaktualisierung gut umsetzen. (FHKOELN1)

3.2 Bestimmung des Sucherfolges – Recall und Precision

Jedes Information Retrieval-System kann anhand seiner Effektivität bewertet werden. Dabei spielt neben dem gelieferten Ergebnis auch das nicht gelieferte Ergebnis eine bedeutende Rolle. Denn Ziel ist es, am Beispiel einer Suchmaschine, dem Nutzer möglichst viele genaue Informationen zu liefern und möglichst wenige nicht relevante Informationen. Aber auch sollten möglichst wenige relevante Informationen nicht gefunden werden.

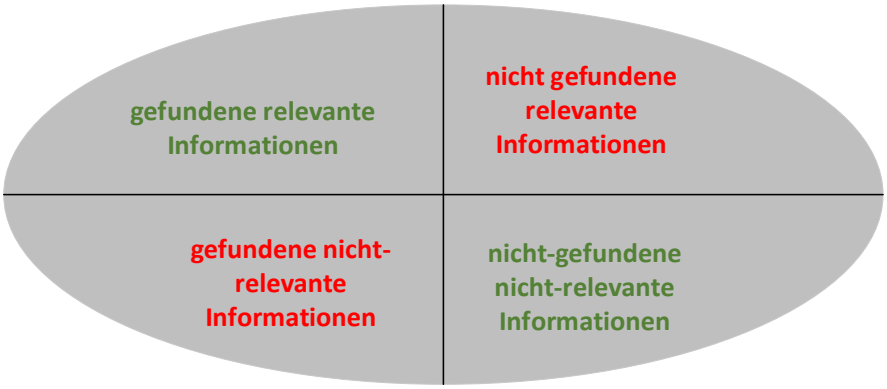

Abbildung 4: Effekt einer Suche auf den gesamten Dokumentenrum

Nach einer Suche kann nun nach den in Abbildung 4 genannten Punkten analysiert werden wie gut das „Maß der Ausbeute“ (Recall) der Suche war. Dieser Recall lässt sich dann wie in Formel 1 zu sehen ist.

$$Recall = \frac{gefundene\ relevante\ Dokumente}{alle\ relevanten\ Dokumente^{1}}$$

Formel 1: Recall

Außerdem kann dann auch das „Maß des Ballastes“ (Precision) einer Suche bestimmt werden, wie in Formel 2 zu sehen ist.

$$Precision = \frac{gefundene\ relevante\ Dokumente}{alle\ relevanten\ Dokumente^{2}}$$

Formel 2: Precision

Wie Ergebnisse von Retrieval-Tests belegen, besteht eine direkte gegenseitige Abhängigkeit zwischen Precision und Recall. So hat eine Maßnahme zur Erhöhung des Recall eine Reduktion der Precision zur Folge. Genauso hat eine Maßnahme zur

[1] gefundene relevante Dokumente + nicht gefundene relevante Dokumente
[2] gefundene relevante Dokumente + gefundene nicht relevante Dokumente

Erhöhung der Precision ein Absinken der Recall zur Folge. Dieses Verhalten lässt sich gut am folgenden Diagramm in Abbildung 5 ablesen.

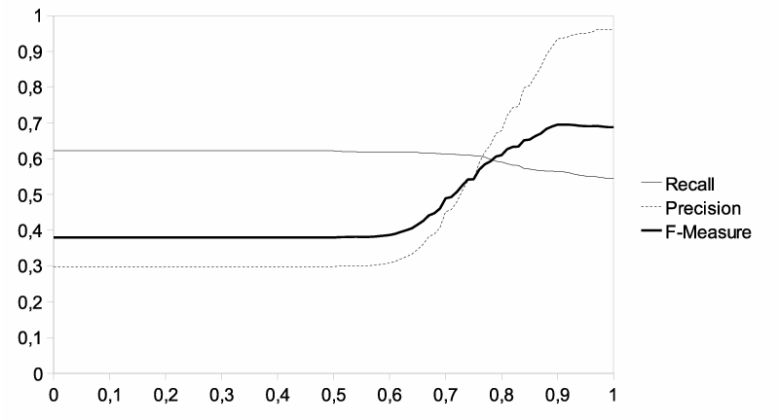

Abbildung 5: Precison und Recall Messung in Abhängigkeit von τ

Das Diagramm stammt aus der Publikation „Skalierbarkeit von Ontology-Matching-Verfahren“ von Heiko Paulheim. Es stellt das Verhältnis zwischen Precison und Recall direkt gegenüber. (FHKOELN1)

3.3 Crawler als Mittel zum Information Retrieval

3.3.1 Crawler allgemein

Crawler, auch Robots oder Spider genannt, sind ein sehr wichtiger Bestandteil von IR-Systemen, die häufig im World Wide Web eingesetzt werden. Crawler die hauptsächlich zur Analyse von Webseiten eingesetzt werden, werden in der Regel als Web-Information Retrieval-Systeme bezeichnet. Sie werden dafür eingesetzt, große Mengen von Daten zu sammeln, sodass diese Informationen Retrieval-Systemen, in Form eines Indexes, in einer zentralen Datenbank zur Verfügung stehen.

3.3.2 Funktionsweise von Crawlern

Im Wesentlichen baut die Funktionsweise eines Crawlers darauf auf, dass dieser eine Webseite aus dem Internet herunterlädt. Anhand des so vorliegenden HTML-Codes können die benötigten Informationen extrahiert und an einer anderen Stelle zwischenspeichert werden.

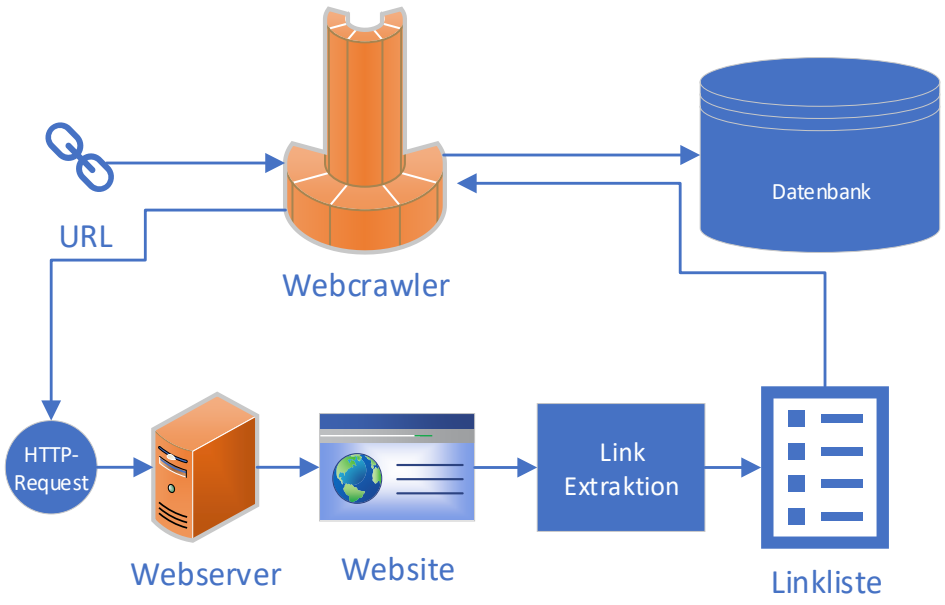

Abbildung 6: allgemeine Funktionsweise eines Crawlers (nach IRWiki1)

Jeder Crawler holt sich zunächst alle auf der aktuellen Webseite hinterlegten Links, diese wiederum werden nach Abschluss der aktuellen Analyse untersucht. Um bei der Rekursion doppelte Webseitenaufrufe zu vermeiden, besitzen die meisten Crawler eine Datenbank, in der die getätigten Aufrufe mit einem Zeitstempel hinterlegt werden. Dies wird gemacht, um nach einem bestimmten Zeitintervall alle Informationen aktualisieren zu können, um so veraltete Datenbestände zu vermeiden. Häufig werden mehrere Rechnersysteme zu einem Crawlernetzwerk zusammen geschalten. Diese nutzen dann für die Datenhaltung einen zentralen Server, der sich dann unter anderem auch um die Datenhaltung kümmert. (IRWiki1)

3.3.3 Probleme und Schwierigkeiten

Die Qualität der Ergebnisse eines Crawlers hängt von vielen verschieden Faktoren ab. Jedoch ist bei der Entwicklung eines Crawlers nicht nur darauf zu achten, dass dieser sehr schnell viele Seiten durchsucht, sondern auch rechtliche Rahmenbedingungen und Standards eingehalten werden. So kann es mitunter vorkommen, dass ein Webseitenbetreiber nicht möchte, dass der Inhalt auf seiner Webseite von einem Crawler indexiert und auf einer anderen Plattform zur Suche zur Verfügung steht. Um dies zu kennzeichnen muss dieser eine sogenannte „robots.txt“ auf seinem Webserver hinterlegen. (siehe 3.6.4) Sollte eine solche Konfiguration vorliegen, sollte der Crawler in der Lage sein, die Anweisungen in dieser zu verstehen und befolgen zu können. Auch kann es Vorkommen das urheberrechtlich geschützte Texte indexiert werden und sobald dessen Inhalte in den Suchergebnissen angezeigt werden, kann unter Umständen eine Urheberrechtsverletzung vorliegen. Sollte nun in einer „Robots.txt“ gekennzeichnet worden sein, dass eine solche Datei nicht indexiert werden darf und dies vom Entwickler des Crawlers nicht berücksichtigt wurde, kann dies auch rechtliche Folgen haben. Unter anderem auch aus diesem Grund halten sich alle seriösen Crawler an die definierten Regeln in der „Robots.txt“. Jedoch teilt Google selbst auf ihrer offiziellen Supportseite mit, dass der einzige Schutz gegen einen Datendiebstahl durch, ein Passwortschutz der jeweiligen Webseite darstellen kann. (IRWiki1) (Google5)

Damit ein Crawler möglichst viele verschiedene Webseiten schon zum Start finden kann, ist es ratsam, Linkverzeichnisse als Startpunkt für den Crawler zu definieren. Diese enthalten meist sehr artverschieden Content und können so schnell eine große Breite an Inhalten wiedergeben. Außerdem sollte sich ein Crawler stets nach dem PageRank-Verfahren (siehe Kapitel PageRank) orientieren, da es unmöglich ist das gesamte Internet herunterzuladen und zu analysieren wie es für einen Crawler wie dem von Google erforderlich wäre. Eine große Gefahr für Crawler sind Webseiten die kontinuierlich neuen Links generieren. Hier muss ein Crawler in der Lage sein, dies zu erkennen um nicht in dieser hängen zu bleiben. (IRWiki1)

3.4 Robots.Txt

Die „robots.txt“, für ein Beispiel siehe Abbildung 7, ist eine Datei, die auf einem Webserver neben der „index.html“ von einem Webseitenbetreiber hinterlegt werden kann. Sie dient im wesentlich dazu, alle Crawler davon abzuhalten von dem Webserver Daten herunterzuladen. Dabei hat ein Webseitenbesitzer die Möglichkeit, in dieser Datei festzulegen, wie die Crawler auf verschiedenen Content reagieren sollen. Als Entwickler eines Crawlers hat man selbstverständlich die Pflicht diese Standards einzuhalten, unter Umständen kann das nichteinhalten Copyrightverletzungen zur Folge haben. Wie in

Abbildung 7 zu sehen ist, ist einem Crawler mit dieser Konfiguration nicht gestattet, den ‚/fotos/' Pfad oder auch den ‚/temp/' Pfad zu durchsuchen. Auch ist es diesem untersagt die Seite ‚fotoalbum.html' zu analysieren. (IRWiki1)

```
# robots.txt zu http://www.example.org/

User-agent: UniversalRobot/1.0
User-agent: mein-Robot
Disallow: /quellen/dtd/

User-agent: *
Disallow: /fotos/
Disallow: /temp/
Disallow: /fotoalbum.html
```

Abbildung 7: Beispiel für eine robots.txt

Auch Google informiert auf seiner offiziellen Support Seite über die Nutzung der „robots.txt". Sie selbst bezeichnen diese nur „als Tool zur Verwaltung von Webseiten Traffic durch Crawler". Jedoch ist dies aus Sicht des Autors nicht die primäre Aufgabe einer solchen Datei, wie schon im vorrangegangenen Absatz erläutert wurde. Google hat richtig erkannt, dass die „robots.txt" nur Richtlinien enthält wie sich ein seriöser Crawler auf der Webseite verhalten soll, warnt jedoch davor, dass sich andere Crawler nicht zwingend an diese Richtlinien halten müssen. Es empfiehlt sich zudem, die Daten durch andere Blockierungsmethoden wie einen Passwortschutz zu sichern. (Google5)

3.5 Meta Tags und Meta Keywords

Meta Tags sind im <Head> Bereich einer Webseite hinterlegte Informationen oder auch Hinweise, die für einem normalen Nutzer unsichtbar sind. Sie dienen in der Regel dazu Maschinen, somit auch Crawlern, bestimmte Informationen über die Webseite zu liefern. Jedoch stammt diese Technologie aus den 90er Jahren, eine Zeit in der Computersysteme und Software noch nicht so ausgreift waren wie heutige Systeme. Somit war früher die Bedeutung dieser Meta Tags enorm höher, da so viele Entscheidungen nicht mehr von dem zugreifenden System getroffen werden mussten, sondern direkt von der Webseite übermittelt wurden. (NLSEO1) Es gibt eine Vielzahl von Einsatzmöglichkeiten von Meta Tags, im Folgenden sollen jedoch nur die, die für die Crawler relevant sind, betrachtet werden.

```
<meta name="name" content="inhalt" />
```

Abbildung 8: Beipsiel eines Metatags

Der Aufbau eines jeden Meta Tags ist recht einfach gehalten. Jedes Meta Tag enthält einen Namen und eine Beschreibung. Der Inhalt dieser Meta Tags ist Variable. So

können sie sich in Art und Form auf jeder Webseite leicht unterscheiden. Jedoch ist als Webseitenbetreiber darauf zu achten, einen formalen Standard einzuhalten, den Retrieval-Systeme auch verstehen können, wenn dessen Inhalt für solche bestimmt ist. Dies ist aber immer von dem Zweck des jeweiligen Meta Tags abhängig, da nicht jeder Meta Tag zwingend für eine Retrieval-System vorgesehen sein muss. (NLSEO1)

Meta Keywords sind eine spezielle Form von Meta Tags. Heutzutage werden diese kaum noch verwendet da Google diese meist vollständig nicht berücksichtigt und diese bei anderen Suchmaschinen wie Yahoo! oder Bing nur eine sehr untergeordnete Rolle spielen. Somit ist der Aufwand/Nutzen Faktor zum einpflegen dieser für die meisten Softwarelösungen zu gering, weshalb auf diese dann verzichtet wird. Softwarelösungen wie WordPress unterstützen jedoch trotzdem noch diese Funktion. Prinzipiell ist die Anzahl der Meta Keywords unbeschränkt, jedoch ist es aufgrund der Webseitenoptimierung zu empfehlen, nur 3 bis 5 solcher Keywords zu nutzen, da sonst sehr schnell der Webseiten Quelltext überladen wird. (NLSEO1)

Meta Robots sind eine Alternative oder auch eine Erweiterung zur zuvor genannten „Robots.txt“. Ähnlich wie diese bieten Meta Robots auch die Möglichkeit Crawler zu beeinflussen. So kann in diesen konfiguriert werden ob die aktuelle Seite in den Suchergebnissen angezeigt werden darf oder ob Links den PageRank übertragen dürfen, um nur zwei Beispiele zu nennen. Jedoch ist diese Umsetzung nicht so gängig wie die mit der „Robots.txt“ und wird nicht so strikt umgesetzt. Die drei großen, Google, Bing und Yahoo!, halten sich jedoch an diese Meta Robots. (NLSEO1)

3.6 Googlebot

Abbildung 9: Aktuelles Google-Logo

Auch Alphabet Inc. hat für seine Suchmaschine Google ein eigenes Crawlernetzwerk, mit dem es Informationen sammelt. Im Folgenden ist mit Google sowohl die Suchmaschine wie auch der Mutterkonzern (Alphabet Inc.) gemeint, da sich beide nur schwer voneinander trennen lassen aufgrund der stark verschränkten Firmenstruktur. Google selbst nennt ihr Crawlernetzwerk „Googlebot“. Wie sie auf ihrer offiziellen Webseite mitteilen nutzen sie ihr Netzwerk, um neue und aktualisierte Webseiten aufzuspüren und um diese dem Google-Index hinzuzufügen. Der Google Index ist eine gigantische Datenbank indem alle für Google wichtigen Webseiteninformationen gespeichert sind. (Google1)

Eine Webseite gilt für Google als indexiert, sobald sie auf ihren Inhalt und Bedeutung hin analysiert wurde und im Google-Index gespeichert wurde. Nachdem eine Webseite

erfolgreich indexiert wurde, kann diese auch mithilfe der Google Suchmaschine gefunden werden. Wenn nun eine Webseite aufgrund einer „Robots.txt“ oder eines Meta-Eintrages nicht an auf ihren Inhalt hin analysiert werden konnte gilt diese trotzdem als indexiert. (Google3)

Der Googlebot greift dabei bei der Analyse von Webseiten „alle paar Sekunden“ auf jeder ihre bekannten Webseiten zu. Dieser wurde von Google auf die Verteilung auf mehrere Computer konzipiert. So ist es Google möglich, ihr Bot-Netzwerk mit Wachstum des Internetzes zu flexibel zu skalieren. Dabei ist Google bestrebt, die entstehende Bandbreite während der Analyse zu reduzieren indem sie diese auf Server auslegen, die sich in der Umgebung der Webseite befinden. (Google1)

Das Zeitintervall indem eine Webseite vom Googlebot aufgerufen wird, wird als Crawling-Frequenz bezeichnet. Google selbst ist in der Lage eine optimale Crawling-Geschwindigkeit für eine Webseite zu bestimmen. Nach eigenen Angaben machen sie dies mit einem hoch entwickelten Algorithmus. Trotzdem bietet Google die Möglichkeit an, das Zeitintervall für das Crawling einer Webseite selbst anzupassen, falls man Besitzer der Domain ist und Probleme mit dem daraus resultierenden Netzwerk-Traffic oder hoher Serverauslastung bekommt. (Google4)

Google verwendet für verschiedene Einsatzgebiete auch verschiedene Crawler. Der eben genannte Googlebot dient nur zur Analyse von Webseiten. Diese verschiedenen Crawler nutzen immer eigene „User-Agents“, die Google auf seiner Support-Webseite definiert hat. Jeder dieser verschiedenen Crawler identifiziert sich mit einem sogenannten UserAgentString. Auch verwendet jeder User-Agent einen eignen Token in der „robots.txt“ um zu überprüfen, ob dieser auf die Webseite zugreifen darf. (Google2) Weitere Crawler, die Google einsetzt, sind unter anderem APIs-Google, ein Tool für Entwickler (Google6), oder auch Werbecrawler wie AdSense, welches dazu dient, relevante Anzeigen auf einer Webseite schalten zu können. (Google7)

Ein User-Agent ist in der Regel ein Schnittstellenprogramm zwischen einem Client und einem Internetdienst. Zur Identifikation bei diesem sendet das Clientprogramm einen sogenannten *UserAgentString* im Header des HTTP-Requests. Solche Schnittstellenprogramme können zum Beispiel Webbrowser oder E-Mail-Programme sein. (Wiki2) Wie im vorangegangenen Absatz erwähnt, nutzen Information Retrieval-Systeme auch diese User-Agents zur Identifikation. In diesem Fall sind die Crawler die Clients, die auf einen Webserver zugreifen.

3.7 Slurp

Mit über hundert Millionen Nutzern ist Yahoo! eine beachtliche Konkurrenz für den aktuellen Monopolisten Google. Yahoo!`s Webcrawler heißt Slurp und ist eine Weiterentwicklung der ehemaligen Inktomi Suchmaschine Slurp und gehört seit 2002 zu ihrem Portfolio. (SEOChat1) (RyteWiki1)

Im Gegensatz zu anderen Information Retrieval-Systemen, speichert Slurp den gesamten gefunden Text in einer Datenbank ab und aktualisiert diesen regelmäßig. Dadurch ist es Yahoo! möglich den Content von dynamischen Webseiten schneller und leichter zu analysieren, da die wechselnden Inhalte stets aktualisiert in der Datenbank vorliegen. Slurp hält sich dabei an alle Richtlinien die in der „robots.txt“ angegeben sind und und beachtet alle Meta Tags. (SEOChat1) (RyteWiki1)

3.8 Bingbot

Bingbot ist der Webcrawler von Microsoft und sucht im Internet Informationen für die Suchmaschinen Bing und Yahoo!. Dieser Umstand kommt zustande, da Microsoft, Yahoo! aufgekauft hat und nicht beide Suchmaschinen miteinander fusionieren wollte. Die Gründe hierfür sind nicht ganz klar, jedoch wird vermutet das Microsoft sein eigenes Produkt (Bing) unbedingt weiterführen und Yahoo! aufgrund seiner doch großen Nutzerbasis nicht auflösen wollte. Die Bing-Suchmaschine ist im Vergleich zu Google und Yahoo! eine noch relativ junge Suchmaschine, da diese erst im Jahre 2009 entstanden ist, als Microsoft ihren Dienst „Live Search“ eingestellt hat. Ähnlich wie der Googlebot verfolgt der Bingbot Hyperlinks auf Webseiten, um deren Inhalt auszulesen. (RyteWiki2)

Wie auch Google verwendet Microsoft neben dem Bingbot noch weitere Crawler für verschiedene Aufgabenbereiche. So nutzen sie unter anderem „AdIdxBot“ zur Bestimmung der Platzierung von Werbeanzeigen auf Webseiten oder „BingPreview“ um Vorschau-Snapshots von Webseiten in der Suche anzuzeigen. Dabei halten sich nach offiziellen Angaben von Microsoft alle diese Crawler an die Richtlinien der „robots.txt“. (Bing1)

3.9 Suchmaschinen Optimierung

3.9.1 Duplicate Content

Als Duplicate Content versteht man Inhalte auf verschieden Webseiten, oder auch verschieden Sub Domains innerhalb eines Webservers, die identisch sind oder sich stark ähneln. Dies kann zur Folge haben, dass ein Information Retrieval-System, diese Webseite schlechter rankt, da aus Sicht des Crawlers die Webseite nur automatisch Content generiert oder diesen dupliziert damit die Webseiten mehr Verlinkungen und

Content aufweisen. Daher ist es Ratsam solchen Content mittels der „robots.txt“ für Suchmaschinen auszuschließen. (RyteWiki3) (Google8)

Gründe für Duplicate Content können sein:

- Inhalte werden auf andere Webseiten dupliziert, dies kann zur Folge haben, dass die Webseite des Urhebers schlechter gerankt wird,
- Inhalte auf einer Webseite sind über mehrere Subdomains erreichbar (Beispielsweise mit und ohne „www“),
- Gleiche Inhalte erscheinen in verschiedenen Kategorien. Dies tritt unter anderem auf, wenn neu geposteter Inhalt im News Bereich erneut erstellt wird,
- Verschiedene Attribute verlinken in Online Shops häufig zu dem identischen Produkt,
- Webseiten geben Inhalte für verschiedene Länder in der gleichen Sprache aus (z.B.: Deutsch in Deutschland, Österreich und der Schweiz).

Neben dem Duplicate Content spricht man auch manchmal von „Near Duplicate Content“. Dabei handelt es sich um Inhalte, die auf vielen verschiedenen Webseiten sehr ähnlich sind. Solche Inhalte können unter anderem vielfach kopierte Textbausteine sein, wie sie als Teaser auf Newsseiten häufig verwendet werden. Auch wenn der restliche Content unterschiedlich zu dem Content auf anderen Webseiten ist, kann es vorkommen das Suchmaschinen diesen als Duplicate Content einstufen. (RyteWiki3)

Jede Suchmaschine analysiert Duplicate Content anders und ist dabei leider wenig transparent. Google hatte erstmals 2004, im Zuge ihres Brandy Updates, Webseiten auf Duplicate Content analysiert. Dieser dort verwendete Algorithmus hatte aber zur Folge, dass viele Webseiten nicht mehr gefunden wurden. Dieses Problem wurde jedoch erst im Sommer 2005, im Bourbon Update behoben. Mit diesem Update entfernte Google viele Regeln zur Erkennung von Duplicate Content. (RyteWiki3) (Google8)

Dabei sind die Konsequenzen von Duplicate Content auf eine Webseite offensichtlich. Google selbst versichert zwar, dass nur Seiten heruntergestuft oder aus der Suche blockiert werden, die dies für „böse Absichten“ nutzen, aber im Zweifel kann man sich nicht sicher sein ob Google die eigne Webseite nicht doch aus diesem Grund herabstuft. Daher ist es zwingend erforderlich, selbst Duplicate Content auf seiner eignen Webseite zu verhindern. (RyteWiki3) (Google8)

3.9.2 OnPage-Optimierung

OnPage-Optimierung, auch OnSite-Optimierung genannt, bezeichnet einen wichtigen Teil der Suchmaschinenoptimierung. Dieser besteht aus technischen, inhaltlichen und strukturellen Anpassungen einer Webseite, die dazu führen sollen, das

Suchmaschinenranking dieser nachhaltig zu verbessern. Das Hauptziel einer OnPage-Optimierung soll sein, letztendlich eine technisch perfekte und schnell ladende Webseite zu haben. Darüber hinaus soll auch drauf geachtet werden, dass kein Duplicate Content auf der Webseite vorkommt und Inhalte schnell mit einer Suchmaschine gefunden werden können. (RyteWiki4)

Zu den technischen Faktoren zählt unter anderem die Ladegeschwindigkeit einer Webseite, da Google diese direkt als Multiplikator für das Ranking nutzt. Ein weiterer technischer Aspekt ist die Optimierung für die Mobile Ansicht, da Google auch die „Mobile Friendliness“ direkt in das Ranking einer Seite einfließen lässt. (RyteWiki4)

Außerdem zählen zur OnPage-Optimierung die Lesbarkeit einer Webseite für Crawler zu erhöhen und diesen das Verarbeiten dieser zu erleichtern. Unter anderem zählt hierzu die schon zuvor genannte Methode der „robots.txt“. Darüber hinaus bietet Google die Möglichkeit an, eine Webseite in der Google Search Console zu registrieren, was positiv dazu beitragen kann. Die Google Search Console ist ein Tool für Webseitenbetreiber, um Webseiten und deren Sitemaps zu verwalten. (RyteWiki4)

Mithilfe von internen Links auf einer Webseite kann die Priorität einzelner Unterseiten, in Bezug auf ein Schlüsselwort, erhöht werden. Hierfür müssen diese nur innerhalb der Webseite „hart“ verlinkt werden. Dies ist möglich, da für interne Links keine Spamfilter existieren. (RyteWiki4)

Es ist zwingend erforderlich alle HTML-technischen Hervorhebungen korrekt zu verwenden, da eine Vielzahl der Suchmaschinen heutzutage in der Lage sind, Content innerhalb einer Webseite anhand der HTML-Tags unterschiedlich zu priorisieren. So können Suchmaschinen fettgedruckte oder auch kursivgedruckte Texte erkennen und aus diesen wichtige Keywords auslesen, die letztendlich das Ranking einer Seite beeinflussen können. (RyteWiki4)

```
<h1>Große Überschrift</h1>

<h2>Kleinere Überschirft</h2>

<I>kursive Schrift</I>

<h2><I>kursive Überschrift</I></h2>

<strong>1. fett hervorgehobener Text</strong>

<b>2. fett hervorgehobener Text</b>

<b><FONT SIZE="6">Falsche H1 Überschrift</FONT></b>
```

Große Überschrift

Kleinere Überschirft

kursive Schrift

kursive Schrift

1. fett hervorgehobener Text

2. fett hervorgehobener Text

Falsche H1 Überschrift

Abbildung 10: HTML Beispiel

Wie in der Abbildung 10 zu sehen ist gibt es verschieden Möglichkeiten, um mit HTML Texte zu gestalten. Die ***„<h1>“*** bis ***„<h6>“*** Tags stellen dabei in HTML Überschriften dar. Dabei unterscheiden sich diese nicht nur optisch voneinander, sondern auch von ihrer Priorität bzw. Wichtigkeit. Der ***„<h1>“*** stellt dabei die wichtigste Überschrift im HTML-Dokument dar und ***„<h6>“*** die mit der niedrigsten Priorität. So priorisiert ein gutes Retrieval-System den Inhalt einer Seite nach diesen Tags. Es sollte bei der Erstellung von Content darauf geachtet werden, dass nur eine ***„<h1>“*** Überschrift vorkommt. Sollten mehrere großgeschriebene Texte innerhalb einer Seite nötig sein, so ist es auch möglich diese mit anderen HTML-Tags nachzubilden, wie dies in der Abbildung 10 dargestellt wurde. (W3Schools1) Neben den Überschriften sind fett hervorgehobene Textpassagen auch wichtige Anhaltspunkte für Retrieval-Systeme. Um Textpassagen *„fett“* zu schreiben gibt es zwei verschiedene Möglichkeiten ohne CSS zu verwenden. Eine Variante ist die Verwendung des HTML-Tags ***„<b>“***, was so viel bedeutet wie ***bold*** (engl. für fett gedruckt) oder die Verwendung von ***„“*** (engl. für überzeugend, kräftig oder stark). Optisch besteht zwischen beiden in den meisten Browsern kein Unterschied. Jedoch sind die beiden in ihrer Bedeutung sehr unterschiedlich. Der Tag ***„“*** sollte immer dann Verwendung finden, wenn etwas inhaltlich und logisch wichtig ist wo hingegen der ***„<b>“***-Tag nur dann eingesetzt werden sollte, wenn auf einen Textabschnitt die Aufmerksamkeit gelenkt werden soll und der Inhalt dessen nicht zwingend wichtig ist. Gute Retrieval-Systeme erkennen somit auch den Inhalt aus ***„“***-Tags als wichtiger an als den aus ***„<b>“***-Tags. (HTMLSemi1)

Ähnlich ist dies auch bei kursiv geschrieben Texten. Hier gibt es einmal den Tag ***„<i>“***, was für *italic* (engl. für italienisch) steht und den Tag ***„“***, was für *emphasis* (engl. für Betonung, Gewichtung) steht. Die Bezeichnung *italic* beruht darauf, dass der italienische Schriftsetzer Aldus Manutiusm, als erster schräge Buchstaben entwickelte, um mehr Inhalt auf eine Buchseite bringen zu können und die Entwickler von HTML diesem ein Andenken setzen wollten. Bis zu HTML5 waren beide ähnlich einzuordnen wie die Tags ***„<b>“*** und ***„“***, wobei ***„<i>“*** das logische Äquivalent zu ***„<b>“*** war und ***„“*** zu ***„“***. Doch dies hat sich seit HTML5 stark geändert, was viele HTML-Designer verunsichert. Demnach sollen ab jetzt ***„<i>“***-Tags, genau dann eingesetzt werden, wenn dieser ohne besondere Bedeutung oder Betonung ist. Wohingegen vor HTML5 dieser für besondere Betonungen eingesetzt wurden. Der ***„“***-Tag ist dagegen jetzt umso bedeutungsloser geworden. Er soll nur noch dann eingesetzt werden, wenn etwas beim mündlichen Aussprechen anders betont wird. Ein gutes Beispiel hierfür ist: „Rufe einen Arzt *sofort*!“. In wie weit deren neue Bedeutung das Verhalten von Information Retrieval-Systemen beeinflusst, kann nur schwer gesagt werden, aktuelle sorgt es in HTML-Expertenkreisen für Unklarheit. (HTMLSemi2)

Für die Bildersuche ist es unter anderem auch wichtig, dass der Dateiname eines Bildes, mit dem es auf der Webseite eingebunden wurde, aus einem wichtigen Keyword besteht, damit dieses später auch mithilfe dessen gefunden werden kann. Ein Keyword ist in diesem Sinne ein Schlüsselwort, welches den Inhalt auf einem Bild beschreiben kann. Auch ist die Verwendung von Alt-Attributen nicht zu unterschätzen. Ursprünglich dienten diese nur als alternativ Texte falls ein Teil des Contents nicht geladen werden konnte. Doch seitdem Tools für Sehbehinderte diese Tags nutzen, um ihren Anwendern den Inhalt des Elementes auszugeben, ist die Wichtigkeit für diese gestiegen und Suchmaschinen berücksichtigen diese. (RyteWiki4)

3.9.3 OffPage-Optimierung

Neben der OnPage-Optimierung ist die OffPage-Optimierung ein wichtiger Bestandteil für die Suchmaschinenoptimierung. Im Gegensatz zur OnPage-Optimierung kann die OffPage-Optimierung nur im geringsten Teil von einem Webseitenbetreiber beeinflusst werden. Bei der OffPage-Optimierung handelt es sich um die Verbesserung der Reputation, also dem Ansehen, der Webseite. Dies kann wenig bis gar nicht auf der Webseite selbst geschehen. Viel mehr hängt dies vorrangig von den Backlinks ab. Backlinks sind Links auf anderen Webseiten, die wiederum auf die aktuelle Webseite verlinken. Dabei kann man prinzipiell sagen, dass eine hohe Anzahl von Backlinks das Ranking einer Seite massiv verbessern kann. Google selbst empfiehlt, sich persönlich nicht bei der OffPage-Optimierung einzumischen, da ein Nutzer, dem der Content auf der Seite gefällt, diesen auch auf anderen Webseiten oder auch Social-Media teilen wird. Dabei rät Google davon ab seinen Link auf künstliche Weise im Internet zu verbreiten, da dies der Algorithmus erkennen kann und in der Lage ist, einen solchen Link abzustrafen. Eine Abstrafung kann sich beispielsweise darin äußern, dass der PageRank der Seite reduziert wird. Dagegen wirken sich positiv auf den Link, auf einer externen Seite aus, dass zwischen diesem und dem Content auf der Seite ein thematischer Zusammenhang besteht. Auch ist es wichtig, dass der Link nicht im Header einer Seite stehen sollte, da dies häufig ein Platz für Werbelinks ist und er so als „Spam“ eingestuft werden könnte. Arten einen Link „*auf künstliche Weise*“ zu verbreiten sind zum Beispiel, dass ein Webseitenbetreiber sich bei Linkindexen einkauft damit diese, seinen Link auf ihrer Webseite posten oder auch das wahllose Verbreiten von Links in verschieden Foren. (RyteWiki5) (SEL1)

3.9.4 Ranking

Als Ranking einer Webseite bezeichnet man die Position, an welcher diese bei einer Suche nach einem bestimmten Themenbegriff von der Suchmaschine eingeordnet wird.

Google hat in seinem Webmasterbereich Punkte definiert, die das Ranking einer Webseite positiv und negativ beeinflussen. (Google9) (RyteWiki6)

Positive Rankingfaktoren können sein:

- klare Seitenstruktur (Content im Content Bereich, Headinhalte im Head),
- Jede Unterseite sollte über einen statischen Link aufrufbar sein,
- Hinterlegung einer Sitemap,
- „vernünftige“ Menge an Links,
- Verwendung von relevanten Suchbegriffen, die ein Nutzer in die Suchmaschine eingeben kann,
- Hinterlegung eines ALT-Attributes bei Bildern und Textinhalten,
- korrekter Einsatz von HTML-Syntax,
- Vermeidung fehlerhafter Links,
- zielgerichtet Einsatz von Medien (Bilder und Videos). (Google9)

Neben den genannten positiven Faktoren erwähnt Google auch negative Faktoren. Grundsätzlich wirken sich die Punkte negativ auf das Ranking aus, die es Google erschweren die Webseite zu indexieren oder das Ranking einer Webseite manipulieren sollen. (Google9) (RyteWiki6)

Negative Rankingfaktoren können sein:

- Einsatz von Skriptsprachen die die Lesbarkeit und Erreichbarkeit reduzieren,
- Einsatz von Sitzungs-Id´s oder Parametern für die Aufzeichnung des Wegs, den der Crawler nimmt,
- DoubleClick- oder AdSense-Links,
- lange Ladezeiten. (RyteWiki6)

4 Künstliches neuronales Netz

4.1 Definition

In den Neurowissenschaften wird eine beliebige Anzahl miteinander verbundener Neuronen als neuronales Netz bezeichnet. Diese bilden in Form eines Nervensystems einen Zusammenhang, der einer bestimmten Funktion dienen soll. In der Informatik und anderen Informationswissenschaften wird nun versucht, eine solche Struktur künstlich nachzubilden und zu simulieren. Diese nachgebildeten Strukturen nennt man dann **künstliche neuronale Netze**. (Wiki3)

An künstlichen neuronalen Netzen wird schon seit ca. 1943 geforscht. Vorreiter in dieser Wissenschaft waren Warren McCulloch und Walter Pitts, die zunächst Formale Modelle eines Neurons entwickelten. Die Forschung in diesem Bereich hatte aber erst ab ca. 1986 markant zugenommen. Bis heute sind neuronale Netze, die selbst Dinge lernen ein großer Faszinationspunkt für Menschen, vor allem da diese nur auf Matrizenberechnung aufbauen. (NN1)

Der Anwendungsbereich von künstlichen neuronalen Netzen lässt sich mittlerweile in zwei große Teilbereiche unterteilen. So gibt es die, die dahingehend motiviert sind, menschliches Verhalten und Erleben, also die Funktionsweise des Gehirns, besser verstehen zu können. Dann gibt es auch noch die, die dazu dienen, konkrete Anwendungsprobleme zu lösen, die aus den verschiedensten Bereichen stammen können. Zum Beispiel der Statistik, Wirtschaftswissenschaften oder auch der Technik. (NN1)

4.2 Units

Als Units werden die Neuronen innerhalb eines neuronalen Netzes bezeichnet. Manchmal werden diese auch als Einheiten oder Knoten bezeichnet. Ihr Zweck ist es, Informationen von anderen Neuronen aufzunehmen und an andere Units in veränderter Form abzugeben. (NN2)

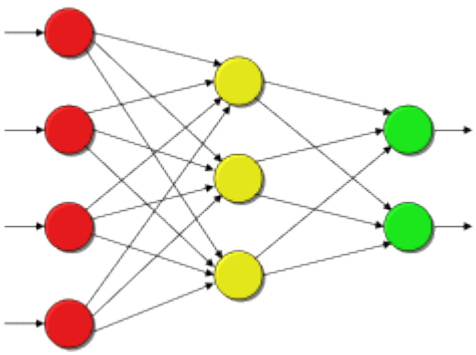

Abbildung 11: Darstellung eines neuronalen Netzes (nach NN2)

Im Wesentlichen unterscheidet man 3 verschiedene Arten von Neuronen, die auch in Abbildung 11 zu sehen sind. Die in rot gefärbten Units sind **Input-Units**. Diese sind in der Lage, verschiedene Reize und Muster aus der Außenwelt aufzunehmen und an die hier gelben **Hidden-Units** weiterzuleiten. Deren Aufgabe besteht wiederum darin, die Inputs von der linken Seite aufzunehmen und an die rechte Seite nach ihrer Verarbeitung weiterzugeben. Sie stellen damit eine interne Repräsentation der Außenwelt dar. Auf der ganzen rechten Seite befinden sich in hier in Grün die **Output-Units**. Ihre Aufgabe ist es, Signale an die Außenwelt abzugeben. Units die sich auf einer Ebene (von oben nach unten Betrachtet) befinden, nennt man im Zusammenschluss Layer. Jedes neuronale Netz in der Informatik besitzt einen Input- und einen Output-Layer. Die Anzahl der Hidden-Layer unterscheidet sich bei jedem Netztypen. Auch sind Netze ohne Hidden-Layer möglich. (NN2)

Eine besondere Form der Units sind die sogenannten **Bias-Units**. Diese besitzen immer unabhängig von deren Input einen Aktivitätslevel von Eins. Ein Gewicht zwischen Bias-Unit und einer anderen Unit kann sowohl einen positiven wie auch negativen Wert annehmen. Sollte von anderen Units kein starker Input auf eine andere Unit erfolgen kann eine Bias-Unit dazu beitragen, dass eine Einheit bei positivem Gewicht trotzdem aktiv bleibt. Mithilfe eines negativen Gewichtes kann eine Unit im negativen Zustand festgesetzt werden. Dieser Effekt kann dann nützen, wenn man eine gewisse Schwelle benötigt (bei einem negativen Bias), die andere Input-Units erst überschreiten müssen. Dieser Schwellenwert ist auch während des Lernprozesses jederzeit veränderbar, da anders als bei der Schwelle einer Aktivierungsfunktion, hier das Gewicht zwischen Bias- und Empfängerunit den Schwellenwert bestimmt. Sollte erwünscht sein, dass ein Neuron besonders oft „feuert“, dann muss man dem Bias einen positiven Wert zuweisen.

4.3 Verbindungen zwischen Units

Units aus verschieden Layern sind immer mit **Kanten** verbunden. Mithilfe eines Gewichtes an dieser Kante wird die Stärke der Verbindung zwischen zwei Units festgelegt. Dabei ist zu unterscheiden, ob es sich um ein positives Gewicht, ein negatives Gewicht oder einem Gewicht von Null handelt. Ein positives Gewicht verkörpert einen erregenden Einfluss auf das andere Neuron, wo hingegen ein negatives Gewicht von hemmender Natur ist. Sollte ein Gewicht Null sein bedeutet dies, dass es keinen Einfluss zwischen beiden Units gibt. Das gesamte Wissen eines Neuronalen Netzes ist in seinen Gewichten gespeichert. (NN3)

4.4 Netzinputs

Als Input bezeichnet man den Eingang eines mathematischen Zahlenwerts in eine Unit. Dieser ist meist abhängig von dem Output eines anderen Neurons und dem Gewicht zwischen beiden. Häufig werden diese einfach multiplikativ miteinander verbunden. Somit ergibt sich das mit erhöhtem Output des sendenden Neurons und einem höheren Gewicht zwischen beiden, der Einfluss von dem sendenden Neuron auf das empfangende Neuron erhöht wird. Auch kann man daraus schlussfolgern, dass sobald einer der beiden Terme Null wird, auch das Ergebnis Null werden muss. Der gesamte Input einer Unit wird als Netzinput bezeichnet. Dieser wird mithilfe Propagierungs-funktion bestimmt. (NN4)

$$netzinput_i = \sum_j a_j * w_{ij}$$

Formel 3: Propagierungsfunktion

Die am häufigsten verwendete ist die Linearkombination, bei dem sich der Netzinput aus der Summe aller einzelnen Inputs zusammensetzt, die das Neuron von den anderen Neuronen erhält. (NN4)

4.5 Aktivitätsfunktion

Der Zusammenhang zwischen Netzinput und dem Aktivitätslevel eines Neurons werden durch die Aktivitätsfunktion dargestellt. Visualisiert wird diese in einem Zwei-dimensionalen Diagramm, wobei an der X-Achse der Netzinput und an der Y-Achse der Aktivitätslevel eingetragen werden. Dieser Aktivitätslevel wird mithilfe einer Ausgabefunktion transformiert, welchen dann dieses Neuron an ein anderes weitersendet. Sehr oft wird als Ausgabefunktion, die Identitätsfunktion verwendet. Bei dieser ist der Output gleich dem Aktivitätslevel. (NN5)

Prinzipiell unterscheidet man zwischen vier verschiedenen Aktivitätsfunktionen.

1. Bei der **linearen Aktivitätsfunktion** ist der Zusammenhang zwischen Netzinput und Aktivitätslevel linear.

2. Diesem kann ein Schwellenwert hinzugefügt werden, der zunächst überschritten werden muss, bevor ein Anstieg möglich ist. Diese nennt man dann **lineare Aktivitätsfunktion mit Schwelle**.

3. Eine weitere ist die **binäre Schwellenfunktion**. Bei dieser gibt es, wie es der Name schon vermuten lässt, in der Regel nur die zwei Zustände Eins und Null. Zu beachten ist allerdings noch, dass in Sonderfällen hier aber auch minus Eins möglich ist!

4. Die **Sigmoide Aktivitätsfunktion** ist die von den meisten Modellen verwendete Methode, da diese kognitive Prozesse simulieren kann. Dabei ähnelt diese Funktion stark den Log- und Tangens-Funktionen. Ist der Betrag des Netzinputs groß und negativ, dann ist dessen Aktivitätslevel nahe Null (log-Funktion) oder -1(Tan-Funktion), um danach langsam anzusteigen, dann immer steiler zu werden bis diese in einer nahezu linearen Funktion ausläuft. Sobald die Werte des Netzinputs einen hohen Wert erreichen beginnt die Sigmoide-Funktion sich asymptotisch der Eins anzunähern. Somit tun sich zwei wesentliche Vorteile der Sigmoide-Funktion auf. Durch die Begrenzung des Aktivitätslevels nach oben und nach unten kann man sagen, dass in Bezug auf die linearen Aktivitätsfunktionen hier eine höhere biologische Plausibilität vorliegt, da biologische Neuronen auch nur einen begrenzen Wertebereich annehmen können. Auch wird die Fehleranfälligkeit des Netzes dadurch reduziert, da so die Werte nicht zu stark über die Grenzen hinweg laufen können wie es häufig bei rekurrenten Verbindungen passieren kann. Im Gegensatz zu den binären Schwellenfunktion, ist es bei den Sigmoide-Funktionen an jeder Stelle möglich klar einen Wert zu differenzieren. (WIKI4) (NN5)

4.6 Trainings und Testphase

Bevor ein Neuronales-Netz Anfragen verarbeiten kann muss dieses zunächst erst einmal lernen, damit es korrekt arbeiten kann. Dieser Lernprozess wird dabei in die **Trainingsphase** und die **Testphase** aufgeteilt. (NN6)

In der **Trainingsphase** versucht das neuronale Netz zu lernen, wie es Entscheidungen richtig treffen kann. Dazu werden die Gewichte zwischen den einzelnen Neuronen angepasst. Mithilfe einer gewählten Lernregeln nimmt dann das Neuronale-Netz Veränderungen auf eine bestimmte Art und Weise vor. Mehr zu Lernregeln im Abschnitt 4.8 Lernregeln. Eine Vielzahl der Lernregeln lassen sich in ***supervised learning*** und ***unsupervised leraning*** unterteilen. Beim „supervised learning“ (engl. für überwachtes bzw. beaufsichtigtes lernen) werden die Gewichte nach einem zuvor bekannten Output optimiert. Dieser Output wird als „teaching vector“ bezeichnet. Dagegen wird beim „unsupervised learning“ kein Output vorgegeben. Alle Gewichtsänderungen erfolgen bei diesem Verfahren in Abhängigkeit der Ähnlichkeit der Gewichte mit den Inputreizen. (NN6)

Die **Testphase** schließt direkt nach der Trainingsphase an. Bei dieser finden keine Gewichtsänderungen mehr statt. Stattdessen wird auf Grundlage der bereits modifizierten Gewichte untersucht, ob das Netzt korrekt gelernt hat. Hierfür wird ein oder mehrere Inputreize in das Netz eingespeist und validiert, ob das erzielte Ergebnis mit dem erwartetem übereinstimmt. Prinzipiell gibt es zwei Arten von Reizen, die hier

unterschieden werden können. **Ausgangsreize** sind Reize, mithilfe dessen geprüft werden kann, ob ein neuronales Netz das eingegebene Trainingsmaterial korrekt erfasst hat. Mithilfe von **neuen Reizen** kann festgestellt werden, ob ein Netz auch über die Trainingsdaten hinweg in der Lage ist seine Aufgabe korrekt zu lösen. (NN6)

4.7 Matrizendarstellung

Für eine mathematische Berechnung eines neuronalen Netzes kann dieses auch in der Matrizenschreibweise niedergeschrieben werden. So können die mathematischen Berechnungen relativ einfach zusammengefasst und vorgenommen werden. Da das Lernen in neuronalen Netzen mithilfe der Gewichte stattfindet und in diesen das gesamte Wissen des Netzes enthalten ist, werden diese in einer solchen Matrix dargestellt. Die Darstellung der Gewichte eines neuronalen Netzes in einer Matrix nennt man **Gewichtsmatrix**. Sollte ein neuronales Netz keinen Hidden-Layer enthalten, reicht diese für die Darstellung des neuronalen Netzes aus. Durch die Hinzunahme von weiteren Hidden-Layers wird immer eine weitere Gewichtsmatrix benötigt. So sind bei zwei Hidden-Layern drei Gewichtsmatrixen erforderlich. Siehe Abbildung 12.(NN6)

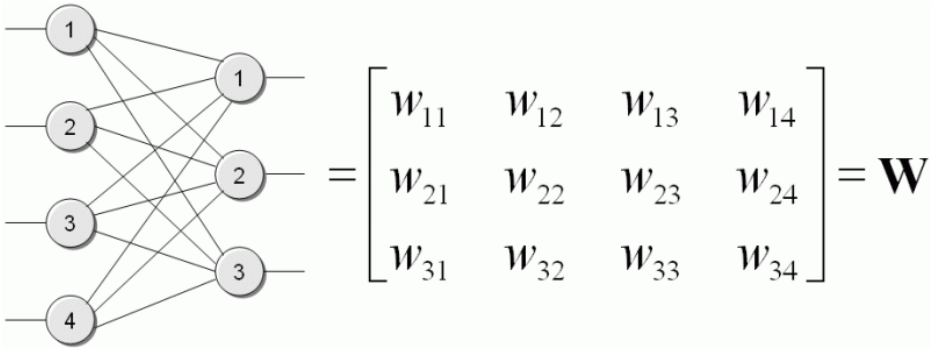

Abbildung 12: Gewichtsmatrix eines neuronalen Netzes (NN7)

Wie in Abbildung 12 zu sehen ist können die Gewichte auch in einer Matrix niedergeschrieben werden. Wie mit diesen gearbeitet wird, wird in dem nächsten Abschnitt lernregeln erläutert.

4.8 Lernregeln

Für die Modifikation der Gewichte während des Trainingsprozesses werden Lernregeln benötigt, die bestimmen, wie deren Veränderungen vorgenommen werden. Eine Lernregel ist dabei ein Algorithmus der Auskunft darüber gibt, welche Gewichte eines Netzes wie stark erhöht oder reduziert werden sollen. Im Folgenden werden die wichtigsten Lernregeln besprochen. (NN7)

Die **Hebb-Regel** ist mitunter eine der einfachsten Lernregeln und besitzt dabei eine große biologische Plausibilität. Formuliert wurde diese 1949 von Donald Olding Hebb, einem Psychologen, aus Kanada. (NN8)

In seinem Buch „The Organization of Behaivor“ schrieb er:

„Wenn ein Axon der Zelle A ... Zelle B erregt und wiederholt und dauerhaft zur Erzeugung von Aktionspotentialen in Zelle B beiträgt, so resultiert dies in Wachstumsprozessen oder metabolischen Veränderungen in einer oder in beiden Zellen, die bewirken, dass die Effizienz von Zelle A in Bezug auf die Erzeugung eines Aktionspotentials in B größer wird.“ (Hebb1)

In Bezug auf neuronale Netze lässt sich nun die Hebb-Regel folgendermaßen formulieren:

„Das Gewicht zwischen zwei Einheiten wird dann verändert, wenn beide Units gleichzeitig aktiv sind.“ (NN8)

Somit lässt sich die Größe der Gewichtsveränderung aufgrund der folgenden Werte bestimmen:

- Aktivitätslevel (bzw. Output) der sendenden Unit a_j,
- Aktivitätslevel (bzw. Output) der empfangenden Unit a_i,
- vorher festgelegten, positiven Lernparameter ϵ.

Diese drei Parameter lassen sich nun produktiv zusammenfassen, damit erhält man dann die Formel zur Hebbregel. (NN8)

$$\Delta w_{ij} = \epsilon * a_i * a_j$$

Formel 4: Hebbregel

Eine weitere Lernregel ist die **Delta-Regel**. Diese baut auf einem Vergleich zwischen dem gewünschten und dem tatsächlich beobachteten Output einer Output Unit i auf. Delta ergibt sich dann aus der Subtraktion aus dem gewünschten Ergebnis und dem beobachteten Ergebnis. (NN9)

$$\delta = a_i(\text{erwartet}) - a_j(\text{beobachtet})$$

Formel 5: Deltaberchenung Hebbregel

So können nun drei verschieden mögliche Ergebnisse betrachtet werden. Sollte die beobachtetet **Aktivität zu niedrig** sein, muss diese gesteigert werden. Sofern nun der Input der sendenden Unit positiv ist, kann dies erreicht werden indem die Gewichte zwischen beiden erhöht werden. Anderenfalls müssen diese reduziert werden, falls die beobachtete **Aktivität zu groß** sein sollte. Dies kann dadurch erreicht werden indem die Verbindungen in den der Input positiv ist geschwächt wird und die mit einem negativen Input verstärkt werden. Ist die beobachtete **Aktivität äquivalent** mit dem erwarteten Resultat, ist keine Veränderung notwendig. (NN9)

All diese Komponenten sind in der Formel der Delta-Regel enthalten (NN9):

$$\Delta w_{ij} = \epsilon * \delta_i * a_j$$

Formel 6: Delta-Regel

Außerdem kann diese Formel sicherstellen, dass die Größe der Gewichtsveränderung proportional zur Größe des Fehlers ist. Durch die Multiplikation mit a_j werden diejenigen Gewichte zu den sendenden Units stärker verändert, die einen größeren Einfluss auf den Fehler-Term ausüben. ϵ ist der Lernparameter. Mithilfe dessen wird vor der Trainingsphase festgelegt, wie groß die Gewichtsveränderung pro Lerndurchgang ausfallen soll. (NN9)

Leider unterstützen die beiden eben genannten Verfahren keine Hidden-Layer, welche bei modernen neuronalen Netzen in der Regel unverzichtbar sind, da eine Vielzahl von Problemen ohne diese nicht gelöst werden können. Ein Beispiel hierfür ist das XOR-Problem, welches ohne Hidden-Units nur schwer gelöst werden kann. (NN10)

Backpropagation ist eine weitere Lernregel und unterstützt die eben genannten Hidden-Layer. Im Wesentlichen stellt das Backpropagation Verfahren eine Rechen-vorschrift dar, mithilfe dessen die Gewichte der Hidden-Units modifiziert werden können. Das Verfahren wurde bereits in den 70er Jahren entwickelt, geriet dann jedoch bis in die späteren 80er in Vergessenheit, als Rumelhart, Hinton und Williams eine Verallgemeinerung der Delta-Regel vorstellten. (Na1) Viele neuronale Netze die konkrete Anwendungsprobleme lösen sollen, greifen typischerweise auf das Backpropagation Verfahren zurück. Jedoch liefert dieses Verfahren in Bezug auf die biologische Plausibilität sehr fragwürdige Ergebnisse zurück, da diese nicht mit der Erklärung der Funktionsweise des menschlichen Gehirns zusammen zu führen sind. Dazu mehr im Kapitel über die Probleme von neuronalen Netzen. (NN10)

Ein großes Problem bei Netzen mit Hidden-Units ist, dass man keinen direkten Fehler für Neuronen in der Hidden-Schicht bestimmen kann. Dieses Problem beruht darauf, dass die Outputs der Hidden-Units unbekannt sind und hieraus einer Ermittlung des Fehlerterms nur schwer oder gar nicht möglich ist. Damit eine Änderung der Gewichte in der Trainingsphase möglich ist, wird diese in drei Schritte unterteilt. Im **Forward-Pass** werden, wie schon bekannt ist, den Inputneuronen Reize präsentiert und aus diesen dann der Output des neuronalen Netzes berechnet. Anschließen erfolgt die **Fehlerbestimmung**. Bei diesem Schritt werden die im Forward-Pass erzeugten Werte mit den gewünschten Outputs validiert. Sollten die errechneten Werte einen gewissen Schwellenwert nicht überschreiten, wird der Trainingsprozess an dieser Stelle abgebrochen. Sonst wird ein dritter Schritt eingeleitet, der **Backward-Pass**. Dieses

Verfahren stellt den innovativen Kern des Backpropagation dar. Hier breiten sich die in entgegengesetzter Richtung die Fehlertermen in Richtung der Input-Schicht aus. Mithilfe dieser Fehlerterme werden nun nach und nach zwischen den Units die Gewichte des Netzes modifiziert, sodass die Fehlerterme kleiner werden. Diese 3 Schritte werden solange wiederholt bis das neuronale Netz einen zielführenden Output innerhalb der Schranken erzeugt. (NN10)

Die Gewichtsanpassung ist hierbei ein wichtiger Teilbereich. Wenn jede Kombination von Gewichten innerhalb eines Gesamtfehlerterms bestimmt werden sollte, um den kleinsten Gesamtfehlerterm und somit die optimale Lösung zu finden, würde dies nicht in absehbarer Zeit berechenbar sein. Hierfür wäre die Berechnung einer Hyperebene erforderlich. Stattdessenn verwendet man hierfür das **Gradientenabstiegs-verfahren**, um die Gewichte zu modifizieren. Da es bei diesem nicht erforderlich ist die Gesamtgewichte zu kennen. Das Gradientenabstiegsverfahren startet zunächst mit zufällig gewählten Gewichten. Für diese wird dann der Gradient bestimmt und so die Gewichte angepasst. Der Gradient ist dabei eine Funktion des Skalar Feldes, welche Änderungsrate und Richtung der größten Änderung in Form eines Vektorfeldes angibt. Für die so neu erhaltene Gewichtskombination wird erneut der Gradient bestimmt und die Gewichte dessen angepasst. Dieser Prozess wird so oft wiederholt, bis ein lokales Minimum, ein globales Minimum oder eine maximale Anzahl von Wiederholungen erreicht worden ist. Leider birgt dieses Verfahren einige Probleme in sich. Diese treten auf, da immer nur die lokale Umgebung (Gradient) zu einem Punkt bekannt ist. So kann es vorkommen, insbesondere bei großen Netzdimensionen, dass man nur ein **Lokales Minima** erhält. Durch die Erhöhung der Netz Dimensionen resultiert eine Vergrößerung des Fehlerterms der Hyperebene, was dazu führt, dass diese stärker zerklüftet und die Anzahl der lokalen Minima erhöht wird. Das Gegenteil hierzu sind **Flache Plateaus**. Bei diesen existieren innerhalb der Hyperebene kaum „Berge und Täler“, sondern nur relativ flache „Plateaus“. Dies hat zur Folge, dass der Gradient sehr klein wird und so das nächste „Tal“ nicht mehr erreicht werden kann. Auch kann es passieren, dass **ein gutes Minima verlassen** wird. Dies tritt dann auf, wenn ein „tiefes Tal“ mit einer relativ geringen Ausdehnung in der Hyperebene liegt und so nur ein lokales Minimum identifiziert wird. Im Falle einer **direkten Oszillation** wird weder ein lokales Minimum noch ein globales Minimum erkannt. Dies passiert immer dann, wenn der Gradient von einem „Abhang“ eines Tales zum gegenüberliegenden „Abhang“ springt und von dort wieder zu selben Stelle zurück. So sind die Beträge der Gradienten gleich. So gelingt es dem Verfahren nicht, in die „Tiefe der Hyperebene hervorzustoßen“. Sollte das Verfahren nicht direkt zurückspringen, sondern hierfür mehrere Schritte benötigen, um zum Ausgangspunkt zurückzukehren, spricht man von einer **indirekten Oszillation**. (NN10)

Es gibt verschiedene Ansätze, um die Probleme des Gradientverfahrens zu lösen. Eine Variante ist es zunächst die Anfangsgewichte zu modifizieren. Dazu ist es wichtig einen guten Startpunkt für das Verfahren zu wählen, da dieser einen zentralen Einfluss auf die Werte der Gewichte im Verlauf nimmt. Zudem ist die Art der Initialisierung der Werte bedeutend. Sind die Werte schlecht gewählt, können diese entweder sehr stark um Null fluktuieren oder in sehr großen Zahlenräumen. Zu empfehlen ist deren Erzeugung mithilfe eines normalverteilten Zufallsalgorithmus. Eine zweite Variante ist die Veränderung der Lernrate. Diese sollte aber erst dann versucht werden, wenn die Neu-Initialisierung der Gewichte nicht erfolgreich war. Es ist hierbei grundsätzlich zu betrachten, dass es keine optimale Lernrate für alle Arten Neuronen existiert. Durch **Erhöhung der Lernrate** bewirkt man, dass die Sprünge in der Hyperebne größer werden. Dies bringt die Vorteile mit sich, dass flache Plateaus schneller durchlaufen bzw. überwunden werden und vom Startpunkt weit entfernte Minima schneller erreicht werden. Jedoch resultiert hier raus auch, dass gute Minima häufig übersprungen werden und somit auch die Gefahr der Oszillation steigt. Wenn man hingegen die **Lernrate reduziert** führt dies dazu das kleinere Schritte vorgenommen werden. So werden hierdurch gute Minima nicht mehr so leicht übersprungen und die Gefahr der Oszillation sinkt. Auch werden so komplexe Daten und eine große Datendichte besser bewältigt. Allerding erhöht sich hierdurch die Trainingszeit zum Erreichen eines Minimums. Diese kann unter Umständen hierdurch inakzeptabel groß werden. Auch werden flache Plateaus langsamer durchlaufen oder garnichtmehr überwunden. Manchmal können lokale Minima von diesem Verfahren nicht mehr verlassen werden. (NN10)

Wie die Universität Münster empfiehlt, ist es ratsam die Lernrate im Laufe des Testprozesses anzupassen. Sie schlagen vor, mit einer Lernrate von 0.7 zu beginnen und bei unbefriedigendem Lernerfolgs schrittweise um 0.1 zu reduzieren. (NN10)

Im Gegensatz zum Backpropagation-Verfahren kann das **competitive Learning** auch ohne Vorgabe eines korrekten Output-Reizes seine Gewichte anpassen. Hierbei handelt es sich um ein unsupervised Verfahren, welches in der Lage ist anhand von Ähnlichkeiten der Input-Reize eine Kategorisierung vorzunehmen. (NN11)

Das competive Learning arbeitet nach dem Prinzip „The Winner takes it all“, welches sich in den folgenden 3 Phasen wiederspiegelt. (NN11)

1. **Erregung** (excitation): Hier werden wie gewohnt alle Output-Units des Netzinputs bestimmt
2. **Wettbewerb** (competition): Nun werden die Netzinputs sämtlicher Output-Units miteinander verglichen. Die Unit mit dem höchsten Netzinput ist der Gewinner.

3. **Adjustierung** der Gewichte (weight adjustment): In diesem Schritt werden alle Verbindungen die zur „Gewinner-Unit“ führen angepasst. Alle anderen Gewichte bleiben unverändert. Die Gewichte zum Gewinner werden so modifiziert, dass sie dem Input ähnlicher gemacht werden. (a_j-w_{ij}) Dies lässt sich wie Folgt auch als Formel darstellen. Dabei ist ϵ wieder der zuvor festgelegte Lernparameter.

$$\Delta w_{ij} = \epsilon^* (a_j - w_{ij})$$

Formel 7: Bestimmung des Lernparamters

(NN11)

Hieraus lässt sich ersehen, dass bei diesem Verfahren immer nur die Einheit mit dem höchsten Gewicht über das Voranschreiten des Lernerfolges entscheidet.

	Hebb-Regel	**Delta-Regel**	**Backprpogation**	**Competitive Learning**
Kernkonzept	Gleichzeitige Aktivierung	Vergleich zwischen gewünscht und beobachten Verhalten; Gradientenverfahren	Backward-pass; Gradientenverfahren	„The winner takes it all.“
Art der Lernregel	Supervised, unsupervised, reinforcement learning möglich	Supervised learning	Supervised learning	Unsupervised learning
biologische Plausibilität	Teilweise	Nicht vorhanden	Nicht vorhanden	Teilweise
Netztypen die diese Lernregel nutzen	Pattern Associator, Auto Associator	Pattern Associator; Auto Associator	Simple Recurrent Networks, Jordan Netze	Kompetitive Netze; konzeptuell auch in Kohonennetzen
Vorteile	Einfacheit, biologische Plausibilität	Einfachheit, relativ leicht zu implementieren	Auch bei Netzen mit Hidden-Units einsetzbar; größere Mächtigkeit im Ver-gleich zur Delta-Regel	Unsupervised learning; biologische Plausibilität
Nachteile	Überlaufen der Gewichte ist möglich und geringe Mächtigkeit des Systems	Nicht bei Netzen mit Hidden-Units einsetz-bar; fragwürdige bio-logische Plausibilität; geringe Mächtigkeit des Systems	Fragwürdige biologische Plausibilität; lokale Minima	Einzelne Output-Unit kann alle Inputmuster „an sich reißen“ → keine Kategorisierung mehr

Tabelle 2: Übersicht Lernregeln neuronales Netz (NN18)

Wie in der vorangegangen Tabelle zu sehen ist, besitzt jede Lernregeln ihre Vor- und Nachteile. Jedoch ist schwierig zu differenzieren, welche dieser Regeln im Allgemeinen

besser ist, da jedes Netz nur mit bestimmten Netztypen anwendbar ist und nur bei einigen eine biologische Plausibilität gegeben ist.

4.9 Netztypen

Man ist in der Lage, neuronale Netze in verschiedene Netztypen zu unterteilen. Dabei klassifiziert man diese nach verschiedenen Punkten, die im Folgenden erläutert werden. Es ist möglich, dass verschiedene Netztypen die gleiche Lernregel verwenden oder sich auch mit mehreren verschiedenen Lernregeln realisieren lassen. Weitere Aspekte, die dazu dienen einen Netztypen zu klassifizieren, kann das (nicht) vorhanden sein, von Hidden-Units sein. Ein weiterer wichtiger Punkt ist die Trainingsphase, hier ist zu unterscheiden, ob in dieser mit supervised oder unsupervised Learning gearbeitet wurde. Zudem sind die Verkopplungsstrukturen der Neuronen untereinander zu betrachten und welchem Anwendungszweck das Netz dienen soll. (NN14)

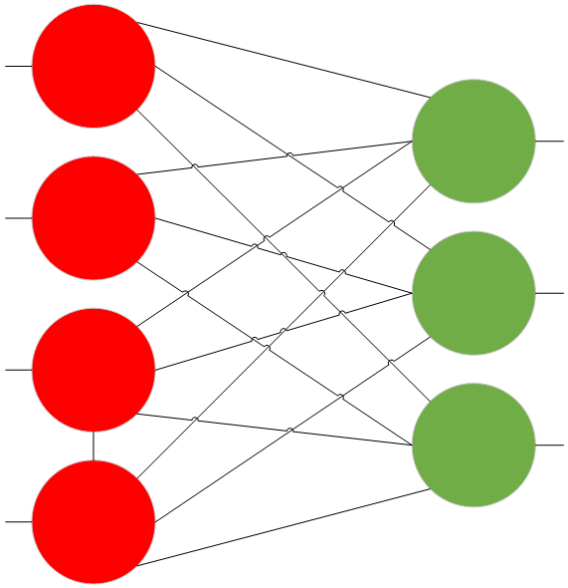

Abbildung 13: Schematische Darstellung des Patter Associator (nach NN13)

Der **Pattern Associator** ist ein Netztyp zu dem ein neuronales Netz zugeordnet werden kann. Dieser Netztyp ist in der Lage, Muster zu erkennen, welche es zuvor gelernt hat. Dazu lernt das Netz abhängige Verknüpfungen (Assoziationen) zwischen verschiedenen Reizpaaren zu erkennen. Bei diesem Verfahren existieren keine Hidden Units, somit besteht dieses Netz nur aus einer Input- und einer Outputschicht. Lernen kann der Patter Associator in der Trainingsphase entweder mit der Hebb-Regel oder aber auch mit der Delta-Regel. Dieses Verfahren verfügt auch über eine Reihe von guten Eigenschaften. Somit ist dieses Verfahren in der Lage eine Generalisierung von Eigenschaften durch Reize zu erkennen. Der Patter Associator kann so nicht nur eine Katze als Katze erkennen, sondern auch verschiedene Katzenrassen, falls dies gewünscht ist. Jedoch

kann es vorkommen, dass eine Übergeneralisierung stattfindet. So kann es passieren, dass es einen Delphin zu der Kategorie der Fische zuordnet. Auch hat dieses Verfahren den Vorteil, dass ihm das Absterben einzelner Neuronen innerhalb des Netzes nicht großartig Schaden können und es trotzdem korrekte Ergebnisse liefern kann. Sollten bereits an den Input-Daten Schäden vorliegen, ist das Netz in der Lage über diese hinweg Muster zu erkennen. So ist es beispielsweise mögliche ein Gesicht, trotz verdeckter Nase, zu erkennen. Ähnlich wie der Mensch auch bildet dieses Netz zentrale Tendenzen beziehungsweise Kategorien anhand der gelernten Inputmuster aus. (NN13)

Ein weiterer Netztyp sind die **rekurrenten Netze**. Mithilfe von diesen Netzen sollen häufig zeitlich codierte Informationen, anhand der Inputreize, entdeckt werden. Dies wird mithilfe von Rückkopplungen zwischen Neuronen innerhalb einer Schicht oder zu vorangegangenen Schichten erreicht. Dabei lassen diese sich wiederum in Untergruppen aufteilen. (NN12)

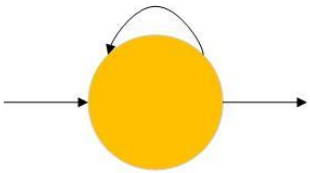

Abbildung 14: Direkte Rückkopplung

Bei der **direkten Rückkopplung** existieren Verbindungen zwischen dem Output und dem Input derselben Unit. Dies hat zur Folge, dass der Aktivitätslevel der Einheit zum Input der gleichen Einheit wird. (NN12)

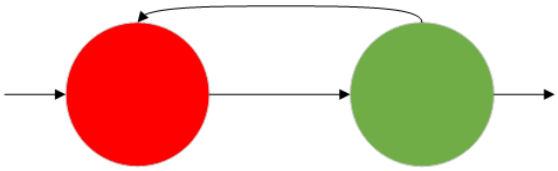

Abbildung 15: Indirektes Feedback

Bei der **indirekten Rückkopplung** hingegen werden die Output Reize an eine vorangegangene Unit zurückgegeben. (NN12)

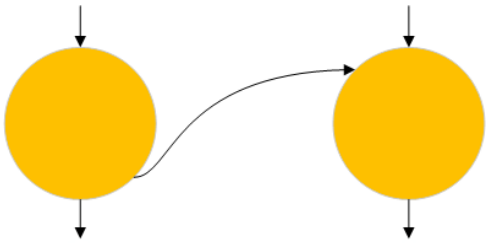

Abbildung 16: Seitliche Rückkopplung

Eine weitere Art der Vernetzung ist die **seitliche Rückkopplung**. Bei dieser existieren Verbindungen zwischen Units einer einzelnen Schicht. Dieses Verhalten von Neuronen existiert auch in der Natur. Zu beobachten ist dies in den Horizontalzellen im menschlichen Auge. (NN12)

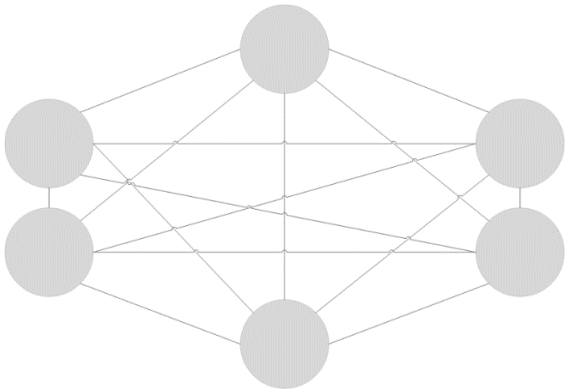

Abbildung 17: Vollständige Verbindung

Auch ist eine **vollständige Verbindung** zwischen allen Neuronen im Netz möglich, dabei sind Rückkopplungen nicht ausgeschlossen. (NN12)

Eine Form der Rekurrenten Netze sind die **Simple Recurrent Networks** (SRNs). Diese besitzen Kontext-Einheiten die sich in der Theorie auf gleicher Ebene wie die Inputschichten befinden und von der Hidden-Schicht Informationen erhalten. Draus ergibt sich, dass dann diese Informationen wieder um einen Schritt verzögert an den Hidden-Units ankommen. Innerhalb des Netzes existieren also genauso viele Kontext-Units wie auch Hidden-Units. Dabei erhält jede Kontexteinheit immer nur Informationen von einer Hidden-Unit. Deswegen wird in den Kontext-Units auch der Input und nicht der Netzinput einem Aktivitätslevel zugeordnet. Außerdem werden alle Gewichte zwischen den Hidden-Units und den Kontext-Einheiten permanent auf plus eins fixiert, dadurch erhalten die Kontext-Units in jedem Durchlauf eine exakte Kopie des Outputs der Hidden-Schicht, mit welcher sie verbunden sind. Somit geben die Kontext-Unit dann einen Schritt später ihren Output wieder zurück an die Hidden Schicht. Dabei ist jede Kontext-Unit mit jeder Unit aus der Hidden-Schicht verbunden. Die Gewichte zwischen diesen Verbindungen können auch während der Lernphase des Netzes modifiziert werden. Dadurch das die Kontext-Unit immer den Aktivitätslevel vom vorangegangenen Durchlauf speichern kann, kann man davon sprechen, dass alle Kontexteinheiten Teilinformationen aus sämtlichen vorangegangen Zeitpunkten bzw. Durchläufen besitzen. Man spricht dann von indirekten Teilinformationen über allen vorrangegangenen Zeitpunkten. Als dynamisches Gedächtnis kann man dann die Kontext-Einheiten mit ihren Gewichten betrachten. Ein identischer Input im Netz wird durch die Existenz der Kontext-Units, dann kontextabhängig modifiziert. (NN12)

Rekurrente Netze finden meist dann Einsatz, wenn zeitabhängige Prozesse analysiert werden sollen. So können diese eingesetzt werden, wenn Prognosen über die Zukunft getroffen werden sollen. Beispielsweise können diese den Input einer nächsten Sequenz mit einer Wahrscheinlichkeit voraussagen. Auch im menschlichen Gehirn spielt das Treffen von Vorhersagen eine essenzielle Rolle beim Lernen. Ebenso sind diese Netze in der Lage, menschliche Verhaltenssequenzen zu simulieren. So sind diese unter anderem auch in der Lage, die Steuerung der Motorik zu übernehmen oder auch verständliche Sprache zu reproduzieren. (NN12)

Ein weiterer Netztyp sind die **kompetitive Netze**. Die einfachste Form dieses Netzes arbeitet nur mit einer Input Schicht und einer Output Schicht. Jedoch ist es möglich Hidden-Units zu verwenden. Kompetitive Netze sind auch in der Lage ohne Vorgabe eines externen Output Reizes zu lernen. Diesen Prozess nennt man dann „*unsupervised learning*“.

Die **Trainigsphase** wird hier in 3 Schritte unterteilt:

1. **Erregung**
2. **Wettbewerb**
3. **Adjustierung der Gewichte**

Bei „kompetitive Netzen“ ist es zwingend erforderlich, eine Begrenzung der Gewichtsvektoren vorzunehmen da es bei diesen Netzen vorkommen kann, dass die Gewichte zu einer einzelnen oder einigen wenigen Output-Units zu groß werden. Hierdurch ist es möglich, dass solche Units unabhängig des Inputmusters gewinnen. Wenn dies passiert, liefert dieses Verfahren keine gute Kategorienbildung mehr zurück. Um dies zu verhindern, hat man die Möglichkeit den Absolutbetrag einzelner Gewichtsvektoren einer Schicht auf einen konstanten Wert festzulegen. Ein Gewichtsvektor sind hierbei alle Gewichte aus einer bestimmten Output -Unit. (NN15)

Eingesetzt werden kompetive Netze unter anderem zur Erzeugung von Output Mustern, die weniger korreliert sind als der Input. Dies ist möglich, weil diese Netze in der Lage sind Redundanzen aus dem Input herauszufiltern. Auch ist es möglich dieses Verfahren einem anderen Netz vorzuschalten. So kann durch die Verringerung der Korrelation der Pattern Associator bessere Ergebnisse liefern. Ähnlich wie beim Patter Associator kann es bei der Musterklassifikation bei beispielsweise der Mustererkennung helfen. (NN15)

Eine weitere Netzart sind die **Kohonennetze**. Sie stellen eine Erweiterung kompetivier Netze dar und arbeiten unsupervised. Kohonennetze sind in der Lage, auf selbstorganisierende Weise zu lernen, indem sie Karten (In diesem Zusammenhang auch häufig Maps genannt) von einem Inputraum erstellen. Der Name dieser Netzart ist

auf den finnischen Ingenieur Teuvo Kohonen zurückzuführen, der auf der Basis der Arbeiten von Stephen Grossberg und Christoph von der Malsburg im Jahre 1982 eine bekannte Form der Kohonennetze konzipierte. (NN16)

Die Kohonennetze besitzen im Vergleich zu anderen Netzarten eine hohe biologische Plausibilität, da angenommen wird das Menschen logische Probleme ohne externen Lehrer lösen. So wurde bereits im Jahre 1962 von David Hubel und Thorsten Wiesel nachgewiesen, dass das sogenannte selbstorganisierende Lernen beim Sehsinn in Okzipitallappen stattfindet. (NN16)

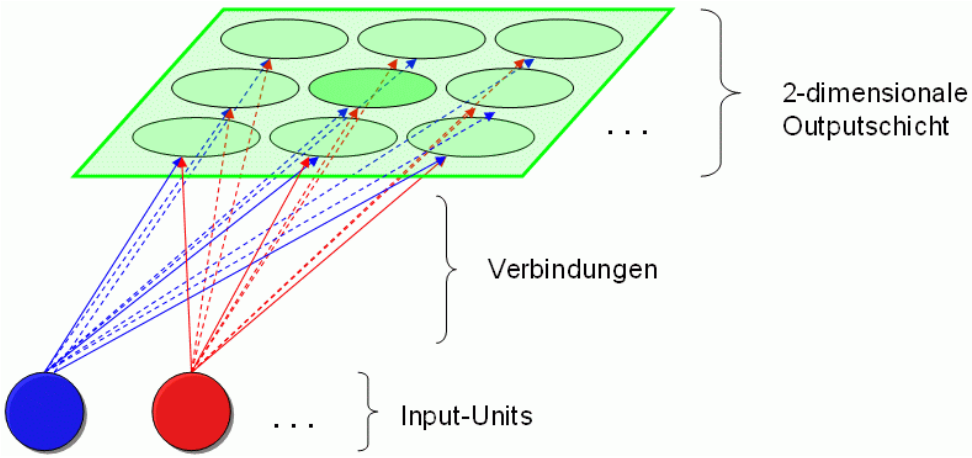

Abbildung 18: Aufbau eines Kohonennetze (nach NN16)

In der Regel bestehen Kohonennetze nur aus einer Input-Schicht und einer Output-Schicht. Eine Umsetzung mit Hidden-Units ist theoretisch möglich wird aber nur selten umgesetzt. Eine große Besonderheit stellt bei diesem Verfahren die Output-Schicht dar, da diese hier häufig sich in einem zweidimensionalen Raum aufgebaut ist (siehe Abbildung). Dabei spielt der Abstand zwischen den einzelnen Output-Neuronen eine entscheidende Rolle. Von jedem Input-Neuron existiert eine Verbindung zu jedem Output-Neuron. (NN16)

Die **Trainingsphase** bei den Kohonennetze kann in fünf Schritte unterteilt werden.

1. **Startwerte festlegen:** Zu Beginn des Prozesses müssen alle Gewichte zufällig ausgewählt werden. Anschließend muss eine Lernkonstante festgelegt werden, sowie die Maximale Anzahl von Durchläufen. Zudem muss bei diesem Verfahren ein Radius für die 2D Output Funktion bestimmt werden und eine Nachbarschaftsfunktion hierzu.
2. **Auswahl des Inputvektors:** Ein Inputvektor kann entweder ausgewählt werden oder wird mithilfe mathematischer Verfahren zufällig erzeugt.
3. **Aktivitätsberechnung und Auswahl:** An dieser Stelle wird die Aktivität der Output Neuronen berechnet. Anschließend wird die Unit ausgewählt, die dem Inputvektor

am ähnlichsten ist. Diese Auswahl erfolgt anhand der maximalen Erregung der Units. Diese Erregung entspricht dem Gewichtsvektor und dessen Distanz zum Inputmuster. Desto kleiner die Distanz umso höher ist die Erregung.

4. **Gewichtsmodifikation:** Alle Gewichte, die zur Gewinner-Unit gehören, müssen so verändert werden, sodass sie dem Input-Vektor leicht ähnlicher werden. Ebenso in abgeschwächter Form die Gewichte aus den Nachbarschaftsunits. Außerdem wird mit jedem Durchlauf der Lernparameter reduziert und, falls nötig, auch der Radius, der die Nachbarschaft eingrenzt. Anschließend geht alles wieder bei Schritt eins los.
5. **Abbruch:** Der Abbruch erfolgt sobald die maximale Anzahl der Durchläufe erreicht wurde. (NN16)

Einzelne Gewichte werden durch die ständige Gewichtsänderung im Trainingsprozess bestimmten Inputvektoren immer ähnlicher. Durch die ständige Reduzierung der Lernkonstante fällt am Beginn die Korrektur der Gewichte größer aus. Dadurch ergibt sich allmählich während des Prozesses ein stabiler Zustand. Zusätzlich besteht die Möglichkeit, den Radius der Nachbarschaft zu reduzieren, um den Einfluss von den Nachbarn der Gewinner-Unit kontinuierlich zu reduzieren. (NN16)

Damit dieses Netz einen stabilen Zustand erreichen kann, sind einige Aspekte sehr bedeutend. Nur mithilfe eines **Zählers** ist es möglich die Anzahl der Durchläufe zu zählen, also die Anzahl der Inputvektoren und damit wie viele Gewichtsmodifikationen vorgenommen werden müssen. Zudem ist der **Radius zur Nachbarschaft** sehr wichtig, da dieser Faktor bei der Gewichtsanpassung berücksichtigt wird. Der **Lernparameter** bestimmt wie stark die Gewichte zwischen den einzelnen Units verändert werden. Die Anzahl der Output-Neuronen ist abhängig von der **Matrixgröße**. Je mehr Output-Neuronen ein Kohonennetze hat, umso genauer kann ein Clustering des Inputraumes vorgenommen werden. Besonders wichtig ist auch die **Form der Nachbarschaftsfunktion**, da diese im Wesentlichen bestimmt in welcher Art und Weise benachbarte Neuronen von der Gewichtsveränderung betroffen sind. So kann die Stärke der Veränderung mit zunehmender Distanz vom Gewinner linear oder gar exponentiell abnehmen. Auch ist die Art und Weise wie Lernparameter, Radiusgröße und Nachbarschaftsfunktion im Lerndurchgang verändert werden ein wichtiger Prozess, der das Ergebnis hauptsächlich beeinflusst. Auch ist die Wahl der **Dimensionen**-Anzahl des Netzes einer großen Bedeutung, da auch N-Dimensionale Netze möglich sind und somit auch ein immer größer werdender Rechenaufwand nötig. (NN16)

Die Anwendungsbereiche dieses Netzes sind vielseitig. So kann es unter anderem Approximationen von einer Funktion finden, bei denen analytisch keine Lösungen

existieren. Sie können Probleme der inversen Kinematik lösen, dass heißt sie sind in der Lage, Roboterarme in einem zweidimensionalen Raum zu steuern. In dessen Rahmen sind sie auch in der Lage, den kürzesten Weg zwischen zwei Punkten zu finden, auch wenn sich auf dem direkten weg Hindernisse befinden, solange sich die Hindernisse nicht bewegen. Mithilfe dieses Netztes ist man auch in der Lage eine n-dimensionale Figur in eine zwei oder mehr dimensionale Figur zu transformieren, ohne dabei kritische Informationen zu verlieren. Dadurch ist es möglich höherdimensionale Figuren für Menschen leichter verständlich zu machen indem man diese visualisiert. Auch kann mit diesem Verfahren das Rundreiseproblem auf eine schnelle und elegante Weise gelöst werden. Zudem zeigt dieses Verfahren seine Stärken in der Mustererkennung. So findet es seinen Einsatz in der Sprach-, Unterschiften- oder auch der Gesichtserkennung. (NN16)

	Pattern Associator	*Rekurrente Netze*	*Kompetitive Netze*	*Kohonennetze*
Kernkonzept	Assoziationen zwischen verschiedenen Reizpaaren bilden	Rückkopplungen zu derselben oder einer vorherigen Schicht	1. Erregung 2. Wettbewerb 3. Gewichts-modifikation	Wie Kompetitive Netze, nur mit mehrdimensionaler Output-Schicht
Lernregel	Hebb-Regel; Delta-Regel	Backpropagation	Competitive Learning	Konzeptuell: Competitive Learning
Rückkopp-lungen	nicht vorhanden	vorhanden	nicht vorhanden	nicht vorhanden
Hidden-Units	nicht vorhanden	sind möglich	sind möglich	in der Regel nicht
Art der Lernregel	Supervised learning	Supervised learning	Unsupervised learning	Unsupervised learning
Vorteile	Einfachheit	Entdeckung zeitlich codierter Informationen	Biologische Plausibilität	Biologische Plausibilität
Nachteile	Keine Hidden-Units und biologisch eher unplausibel	Überlaufen der Aktivität ist möglich	Erstärken einzelner Output-Units verhindert sinnvolle Kategorisierung	Wahl zahlreicher Parameter entscheidend für adäquate Clusterung

Tabelle 3: Netzarten NN19

In der vorangegangen Tabelle sind die Unterschiede der einzelnen Netzen klar zu erkennen. Jedes Netz verfügt über seine eignen Vor- und Nachteile und verfolgt unterschiedliche Kernkonzepte, die mit und ohne Hidden-Units umgesetzt werden können. Eine Verallgemeinerung zu Treffen, welches Netzt „besser“ ist, ist nicht möglich.

4.10 Probleme der neuronalen Netze

Auch neuronale Netze sind nicht perfekt und so gibt es um sie auch diverse Probleme. So werden häufig neuronale Netze zur Erklärung des menschlichen Gehirns herangezogen. Jedoch ist es in Wirklichkeit so, dass neuronale Netze theoretisch jede menschliche Verhaltensweise simulieren können, aufgrund der großen Anzahl frei variierbaren Parameter. Hierdurch ist das Verhalten dieser nicht falsifizierbar (nicht wiederlegbar). Sie sind durch die Variation verschiedener Parameter immer durch Falsifikation geschützt. Dieses Verhalten nennt man in diesem Zusammenhang Immunisierungsstrategie. Karl R. Popper beobachtet dieses Verhalten als einer der ersten. (NN17) Dieses Problem betrifft aber nur neuronale Netze, deren Ziel es ist, die Verhaltensweise des menschlichen Gehirns zu simulieren. Neuronale Netze die konkrete Anwendungsprobleme lösen sollen sind hingegen davon nicht betroffen. Ein weiterer Kritikpunkt, an neuronalen Netzen, ist die fragwürdige biologische Plausibilität. So wiedersprechen eine Vielzahl von neuronalen Netzen den biologischen Grundannahmen und sind somit als Modell zur Erklärung des menschlichen Verhaltes nur mäßig gut geeignet. Ein Beispiel hierfür ist die Rückwärtsausbreitung beim Backpropagation. Das wohl größte Problem aller neuronalen Netze ist deren gigantischer Rechenaufwand. Meist ist es so, dass ein Problem mit herkömmlichen Rechenverfahren schnell gelöst werden kann, dies jedoch mit neuronalen Netzen viel komplexer ist. Auch ist zu beachten, dass neuronale Netze nur selten hundertprozentige Lösungen ausgeben, sondern nur Näherungswerte. Klassische Methoden hingegen liefern stets vollkommen korrekte Ergebnisse. (NN17)

4.11 Neuronale Netze im Information Retrieval

Zum aktuellen Zeitpunkt werden nur wenige neuronale Netze in kommerziellen Information Retrieval-Systemen eingesetzt. Jedoch sind die sogenannten **Spreading Activation Netzwerke** im Information Retrieval soweit entwickelt, dass diese in realistischen Umgebungen aktiv getestet werden. Das Spreading Activation Modell ist ein sehr einfach aufgebautes Modell, welches meist nur aus zwei Schichten besteht. Weitere Modelle, die im Information Retrieval eingesetzt werden, sind unter anderem die Kohonennetze für das Clustering oder aber auch Assoziativspeicher für die fehlertolerante Suche. (UNIH1)

Spreading Activation ist ein schon ein seit Mitte der 80er Jahre entwickeltes neuronales Netz für den Einsatz im Information Retrieval. Entwickelt wurde es von Belew und Kwock (UNIH1). Dieses Modell wurde bis in die 90er Jahre immer weiterentwickelt und 1996 auf der TREC von einem kleinen Team vorgestellt. Die **TREC** (**T**ext **Re**trieval **C**onference) ist eine Wissenschaftskonferenz, die sich ausschließlich mit dem Thema

des Information Retrievals beschäftigt. (WIKI5) Nach der Teilnahme an der TREC gelang dem kleinen Team erstmals der Schritt weg von experimentellen Systemen hin zu echten Massendaten. Hierzu wurden Massendaten aus Zeitungstexten extrahiert. Das Spreading Activation Netzwerk ist ein bidirektionales, symmetrisches Netzwerk mit Aufteilung in Schichten. Typischerweise existieren zwei Schichten die untereinander Aktivierungen austauschen. Eine der beiden Ebenen ist dabei üblicherweise für Dokumente zuständig und die andere für Terme. Die Gewichte der Verbindungen werden mithilfe einer Dokument-Term-Matrix aus der Indexierung initialisiert. Sozusagen wird das Gewicht zwischen einem Dokument-Neuron und einem Term-Neuron von einem Indexierungs-Algorithmus und einem Gewichts-Algorithmus bestimmt. Es ist zu beobachten, dass eine Vielzahl der Verbindungen ein Gewicht von Null erhalten, da viele Terme in Dokumenten gar nicht vorkommen. Wie üblich wird von einem Benutzer nun eine Anfrage an das Retrieval-System gestellt. Das System aktiviert dann die gewählten Terme und die Aktivierung breitet sich im Netz aus. Zunächst werden alle Dokument-Neuronen aktiviert, mit denen die Anfragen-Terme indexiert sind. Im nächsten Schritt senden alle aktivierten Dokument-Neuronen einen Aktivierungsimpuls an die Terme, mit denen sie in Verbindung stehen. Bereits an diesem Punkt ist schon häufig zu beobachten, dass jetzt Terme aktiviert sind die gar nicht in der eigentlichen Suchanfrage vorkamen, somit tritt eine Term-Expansion als inhärente Eigenschaft auf. Die Aktivierungsausbreitung endet, sobald ein bestimmter Aktivierungswert oder eine bestimmte Anzahl von Schritten erreicht ist und die nun am stärksten aktivierten Dokumente werden dem Benutzer als Ergebnis präsentiert. (UNIH1)

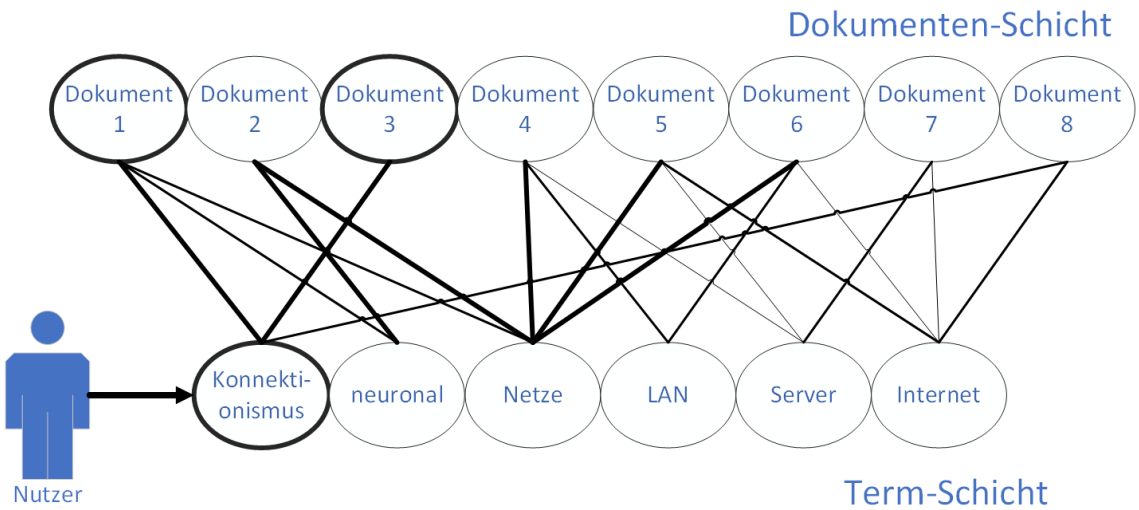

Abbildung 19: Beispiel einer Suchanfrage (nach UNIH1)

In der Abbildung 19 ist Beispiel für ein solches Netzwerk, während des Retrieval Prozesses zu sehen. Im Beispiel hat der Nutzer nach TCP/IP gesucht. Während des

Prozesses sind nun mehrere Dokumente die in Verbindung mit TCP/IP stehen aktiviert worden. Auch wurden Begriffe, die mehrfach innerhalb der Dokumente mit TCP/IP erwähnt wurden, wie beispielsweise *network*, *server* und *client* aktiviert. Bemerkenswert ist auch noch, dass auch ein Dokumenten-Neuron als Anfrage dienen kann und die Funktionalität äquivalent bleibt. Auch ist es möglich, dass ein Benutzer die Ergebnisse beeinflussen kann. Hierzu wird die Erweiterung „relevance feedback“ in den Prozess mit einbezogen. Bei diesem Verfahren evaluiert ein Benutzer nach einer gewissen Anzahl von Aktivierungsschritten das Ergebnis. Nun ist er in der Lage die Ergebnisdokumente aus dem Ergebnis manuell zu priorisieren. Dabei kann er diese entweder Auf- oder auch Abstufen. Im weiteren Verlauf wird so das Ergebnis, das beim Nutzer ankommt beeinflusst. (UNIH1)

Im Laufe der Zeit wurde dieses Verfahren durch weitere Schichten ergänzt. (UNIH1) So führte Belew eine Schicht ein, die Autoren repräsentieren sollte und direkt mit den Dokumenten verbunden ist. Auch sind in seinem abgewandelten Modell nun Verbindungen zwischen einzelnen Schichten möglich, dass bedeutet, dass es direkte Beziehungen zwischen einzelnen Termen geben kann. Kwok präsentierte ein Modell, dass eine Anfragenschicht beinhaltete, welche nun vor die Term Schicht gestellt worden ist. (UNIH1) Dabei ist festzustellen, dass das Spreading Activation Modell sehr dem Vektorraum-Modell ähnelt. An dieser Stelle sei noch anzumerken, dass dieses Modell nicht die Lernfähigkeit neuronaler Netze ausschöpft. (UNIH1)

Das **COSIMIR-Modell** (**Co**gnitive **SIM**ilarity Learning in **Information Retrieval**) versucht die Schwächen der Spreading-Activation Modelle zu überwinden. Dazu nutzt es Benutzerurteile zum Lernen innerhalb eines Backpropagation-Netzwerkes. Wie dieser Netz-Typ funktioniert, wurde im letzten Kapitel bereits ausreichend besprochen. Der Zentrale Prozess des Backpropagation wird im COSIMIR-Modell für das Information Retrieval umgesetzt. Dazu nutzt dieses den Abgleich zwischen Anfrage und Dokument-Repräsentation innerhalb eines einfach Backpropagation-Netzwerkes. Hierdurch ist es in der Lage, formal mehr Klassen von Funktionen zu implementieren als ein Spreading Activation Netzwerk, da es subsymbolische Repräsentationsmechanismen nutzt. Beim COSIMIR-Modell werden an die Eingangsschicht ein Query und ein Dokument gelegt. Nun wird über eine oder mehrere Hidden-Schichten die Aktivierung propagiert. Während der Trainingsphase muss somit die Relevanz zwischen verschieden Kombinationen aus Dokumenten und Anfragen erlernt werden. Um dies umzusetzen ist es zunächst erforderlich Relevanzurteile von Benutzern zu sammeln. So ist dann das COSIMIR-Modell in der Lage eine kognitive Ähnlichkeitsfunktion zu implementieren. (UNIH1)

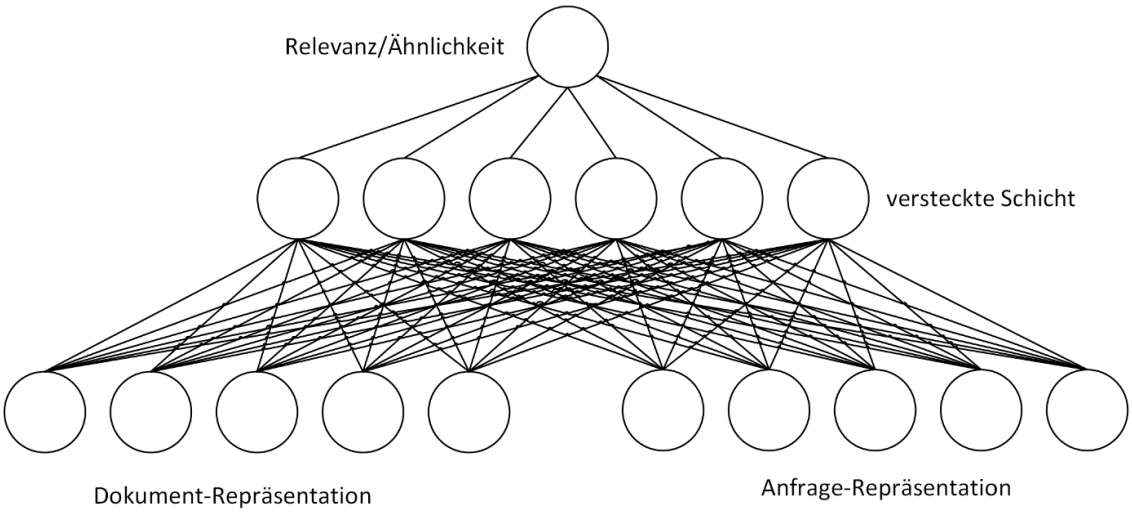

Abbildung 20: COSIMIR-Modell (nach UNIH1)

Damit das COSIMIR-Modell (siehe Abbildung 20) korrekt arbeiten kann, werden sehr viele Trainingsdaten benötigt. Grund hierfür ist, dass jeder Term zwei Input-Neuronen benötigt und dadurch das Netz entsprechend groß wird. Somit kann es nur dann hinreichend gut trainiert werden, wenn jeder Term mindestens einmal in einem Trainingsbeispiel vorkommt. Es ist zwingend erforderlich, auch Trainingsbeispiele mit einer geringen Relevanz und Nicht-Relevanz einzubauen, da sonst das Netz unter Umständen nur lernt, wie es mit einer hohen Relevanz umzugehen hat. Durch diesen Prozess steigt die Anzahl der potenziellen Trainingsdaten enorm an, denn so können alle als nicht Relevanz gekennzeichneten Dokumente als Negativbeispiele genutzt werden. (UNIH1)

Das COSIMIR-Modell besitzt einige wesentliche Eigenschaften, die es von anderen IR-Systemen positiv abgrenzt. Viele Retrieval-Systeme sind nicht in der Lage, die Eigenschaften einer unlineare Ähnlichkeitsfunktionen zu erfüllen, wie sie beim Menschen auftreten. Aufgrund von verschiedenen menschlichen Ähnlichkeitsurteilen sind diese nicht immer symmetrisch oder transitiv. Das COSIMIR-Modell ist eines der wenigen Modelle, die dieses unlineare Verhalten von Ähnlichkeitsempfinden wiederspiegeln können. In einer Vielzahl von Information Retrieval-Systemen wird die Entscheidung einer Ähnlichkeitsfunktion auf heuristische Verfahren aufgebaut. Auch diese entfallen bei COSIMIR vollständig. Einige Information Retrieval-Systeme erwarten die paarweise unabhängig zwischen den Eingabetermen. Jedoch wird diese Eigenschaft in vielen Fällen der Daten nicht erfüllt. COSIMIR kann auf dieses Verhalten verzichten und ist in der Lage, sich selbst die Zusammenhänge und weitere Abhängigkeiten innerhalb des Netzes zu modellieren. Zudem ist innerhalb des COSIMIR-Models mehr Wissen enthalten als beispielsweise im Spreading-Activation-Netzes, da hier noch die Benutzerurteile zu den sowieso schon verwendeten Daten hinzukommen. Zwar könnte

theoretisch auch das Spreading-Activation-Netze, um solche Daten ergänzt werden, jedoch wäre es dann nicht in der Lage zwischen Ausgangsdaten und der Ähnlichkeitsfunktion zu trennen, wie es das COSIMIR-Modell kann. Auch benötigt das COSIMIR-Modell keine Gleichförmigkeit zwischen dem Dokumenten- und Anfragevektor. Beide können in unterschiedlichen Formen vorliegen. Bei großen Testdaten ist es sogar möglich selbst bei Vorliegen in unterschiedlichen Sprachen möglich Entscheidungen zu treffen. Somit ist das COSIMIR-Modell auch für Retrieval in heterogenen Umgebungen geeignet. (UNIH1)

Die Effizienz des COSIMIR-Models wurde bislang noch nicht in der Praxis nachgewiesen, sondern nur in experimentellen Retrieval-Tests. Die größte Schwäche dieses Models besteht nach wie vor in der benötigten gigantischen Menge an Trainingsdaten. Die Implementierung dieses Models ist relativ trivial, da hierzu auf Standard Schnittstellen von anderen neuronalen Netzen zugegriffen werden kann. Nur die Vor- und Nachbereitung der Daten ist sehr aufwendig. (UNIH1)

5 Prototyp

5.1 Grundlegendes

Im Rahmen dieser Arbeit ist ein Prototyp eines Information Retrieval-Systems entstanden. Dieser enthält alle wesentlichen Merkmale eines Information Retrieval-Systems. Dabei besteht dieses Retrieval-System aus drei voneinander getrennten Modulen.

1. Crawler
2. Serverseitige Datenbankschnittstelle für die Crawler
3. Web-Frontend für die Suchmaske

Als Betriebssystem wird bei allen Modulen auf ein UNIX System gesetzt. Bei den Crawlern wird ein *Ubuntu 16.04* verwendet, da dieses mit allen verwendeten Programmschnittstellen und benötigten Bibliotheken kompatibel ist. Auf dem Server, der sowohl Host für die Datenbank wie auch für den Webserver ist, wird ein *Ubuntu 18.04* verwendet.

Im Rahmen einer Bachelorarbeit ist es nicht möglich eine komplexe Suchmaschine, wie beispielsweise Google, zu erstellen. Der hier dargelegte Prototyp basiert auf dem Schema der Volltextsuche ähnlich wie es Yahoo! macht. Außerdem werden bei der Suche, falls vorhanden, die auf der Webseite hinterlegten META-Keywords beziehungsweise HTML-Tags, wie *„<title>“*, mit einbezogen.

Die Crawler erhalten nach Verbindung zum Server automatisch einen Link von dem Server, den sie analysieren sollen. Bei diesem wird zunächst der Mimetype bestimmt, um die weitergehende Vorgehensweise zu bestimmen. Handelt es sich um eine HTML-Seite bzw. ein Textfile, werden alle HTML-Tags entfernt und der Inhalt aus dem Body dem Server übermittelt. Außerdem werden, falls vorhanden, der Titel der Seite und Meta-Keywords übertragen. Zudem analysiert er alle HTML-Files auf Vorhandensein weiterer Links. Alle gefunden Links werden auch zum Server übermittelt.

Bei PDF-Files oder auch Office Dokumenten wird mithilfe der jeweiligen Schnittstelle der Inhalt des Dokumentes extrahiert und in der Datenbank abgespeichert.

Erhält der Crawler einen Link zu einem Bild, analysiert dieser das Bild mithilfe von einem neuronalen Netz. Außerdem sucht er mithilfe eines OCR-Tools nach Buchstaben bzw. Texten auf dem Bild. Die Ergebnisse der OCR-Analyse und des neuronalen Netzes werden zum Server übermittelt.

Die Aufgabe des Servers ist es die Anfragen der Clients, hier die Crawler, zu verarbeiten. Wenn ein Crawler nun die Anfrage nach einem neuen Link an den Server stellt

durchsucht der Server die Datenbank nach dem ältesten Link. Sollten noch keine Links vorhanden sein, steht dem Server eine Datenbank mit Host´s zur Verfügung, die er sonst als Basis nutzen kann. Links aus der Host Tabelle sind auch höher priorisiert als die in der Link Datenbank. Die Analyse der Links basiert auf dem **FiFo**-Prinzip (**F**irst **i**n **F**irst **o**ut). Dies bedeutet, dass der älteste Link in der Datenbank wird immer als nächstes den Crawlern zur Analyse zur Verfügung gestellt wird. Dies wird erreicht, indem zum Zeitpunkt des Hinzufügens eines neuen Datensatzes in die Link-Datenbank, der aktuelle Zeitpunkt, in Unix Zeit durch zwei dividiert, als TimeStamp übermittelt wird. Nachdem ein Link von einem Client analysiert wurde und die gesammelten Informationen zum Server gesendet wurden, wird der Inhalt des Datensatzes entsprechend angepasst. Dabei wird der TimeStamp in der Datenbank auf die aktuelle Zeit gesetzt. Sollte ein Link nicht erreichbar sein, übermitteln dies die Clients dem Server auch entsprechend und löschen den Link aus der Datenbank. Außerdem überprüft der Server beim Hinzufügen eines Links ob dieser bereits in der Datenbank vorkommt und bricht gegebenen falls den Vorgang ab. Alle Links, die hinzugefügt wurden, werden Server sowie Clientseitig in einem kleinen Rahmen gecacht, sodass die Anzahl der Datenbank-Querys mit doppelten Links reduziert wird. Dies ist aber aufgrund des begrenzten Arbeitsspeichers nur bedingt möglich.

Eine weitere Aufgabe des Servers ist die Verwaltung eines Webservers, der das Frontend der Suchmaschine darstellt. Im Rahmen dieses Prototyps wird als Basis ein Apache Tomcat Webserver eingesetzt. Dieser ermöglicht es, neben JavaScript, auch mit Java zu programmieren.

5.2 Netzwerkaufbau

Der gesamte Versuchsaufbau befindet sich innerhalb des Intranetzes der Hochschule Merseburg. Da die Hochschule eine öffentliche Einrichtung ist, sind diverse Sicherheitsvorkehrungen innerhalb des Intranetzes und für die Kommunikation mit dem Internet getroffen wurden, die im Folgenden erläutert werden (siehe dazu auch Abbildung 21:). Das in Abbildung 21, als **Netzwerk A** bezeichnete Netzwerk, ist das öffentliche Netzwerk innerhalb der Hochschule. Über dieses Netzwerk läuft regulär die Kommunikation zahlreicher Poolrechner und privater Laptops, die über eine LAN-Dose angeschlossen sind. Dieses Netzwerk ist so konfiguriert das es nur nach außerhalb kommunizieren kann und nicht von anderen Diensten im Internet erreicht werden kann. Die Kommunikation zwischen Rechner innerhalb des Netzes ist uneingeschränkt möglich. Der Server des Projektes befindet sich in **Netzwerk B** der DMZ der Hochschule. Als **DMZ** (*Demilitarized Zone*) bezeichnet man einen Netzwerkbereich innerhalb eines Unternehmens oder auch Organisation, welcher besonders geschützt ist von Zugriffen aus dem Internet und unter Umständen auch aus dem Intranet. In der Regel ist kein Server innerhalb der DMZ von außerhalb erreichbar, sodass keine sensiblen Unternehmensdaten an die Öffentlichkeit gelangen können (SearchSec1). Die DMZ der Hochschule Merseburg, funktioniert analog dazu. Der hier verwendete Server kann im Internet nur zuvor freigehaltene Netzwerkadresse erreichen. In diesem Fall ist das nur der UNIX-Update Server. Auch die Erreichbarkeit des Servers aus dem Internet ist stark begrenzt. So sieht es die Sicherheitspolitik der Hochschule Merseburg vor, das jeder Port, der aus dem Internet erreichbar sein soll, bei dem leitenden Netzwerkadministrator beantragt werden muss. Analog gilt dies auch für die Erreichbarkeit eines Ports aus dem Intranet. Geregelt werden diese Netzwerkzugriffe mithilfe von verschiedenen Firewalls. Innerhalb der Hochschule Merseburg gibt es noch weitere Subnetze, auf die aber nicht weiter eingegangen werden soll. Jedes Subnetz verfügt über seinen eignen IP-Bereich. Dies ist möglich, weil die Hochschule über ein eigenes Class-B Netz verfügt und somit 16.384 Subnetze erzeugen kann. Jedes dieser Subnetze ist wiederum in der Lage 65.534 Hosts zu verwalten die Wiederum 254 separate Sub-Hosts betreiben können. Somit ist die Hochschule theoretisch in der Lage, 272.722.100.224 Netzwerkgeräte zu betreiben.

Für den hier dargelegten Anwendungsfall ist es notwendig, dass der Server für die Administration über einen SSH-Port (*22022*) und für die Weboberfläche über einen separaten Port (*8080*) aus dem Internet erreichbar ist. Außerdem ist der Server für die Crawler-Schnittstelle über Port 2156 innerhalb des Intranetz erreichbar. All diese Netzwerkkonfigurationen mussten für den entsprechenden Anwendungsfall beim Netzwerkadministrator beantragt und genehmigt werden.

Für die Konfiguration der Crawler musste keine gesonderte Konfiguration vorgenommen werden, da diese standardmäßig nach dem Anschluss an das „Laptop-Netzwerk“ über alle nötigen Berechtigungen verfügten.

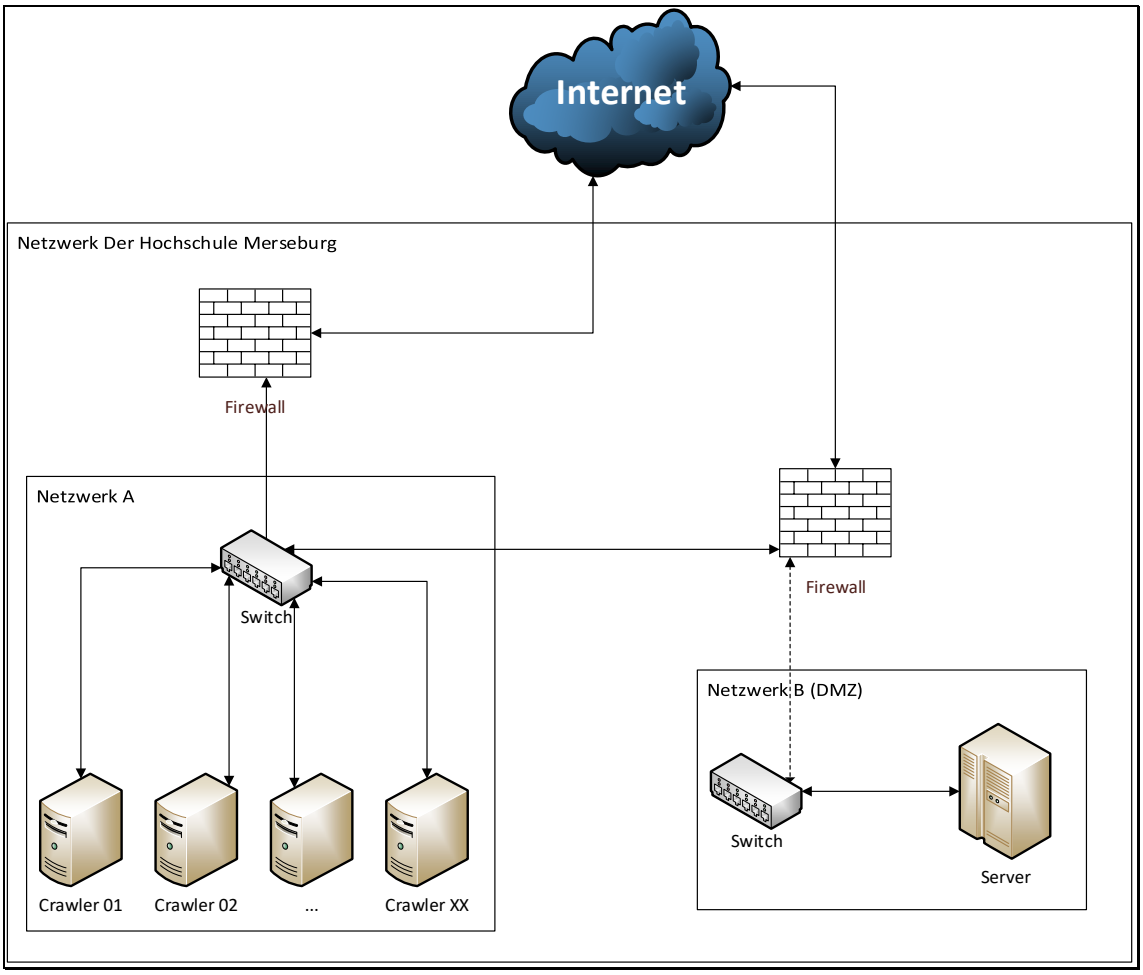

Abbildung 21: Netzwerkdiagramm

Aus dem Netzdiagramm aus Abbildung 21 geht der Netzwerkaufbau des Prototypen innerhalb des Netzwerkes der Hochschule Merseburg hervor. Je nach Netzwerkkonfiguration der Organisation kann sich dessen Aussehen ändern beziehungsweise abweichen.

5.3 Hardwareumsetzung

Ein wichtiger Teil des Prototyps ist die Wahl der richtigen Hardwarekonfiguration. Dank des Netzwerkbetreuers des Fachbereiches stehen dem Projekt zahlreiche alte Server des Fachbereiches zur Verfügung. Für den Server wird ein Dell PowerEdge T300 eingesetzt. Dieser Verfügt über einen Intel Xeon X3363 mit 4 Kernen bei 2.83 GHz Leistung und 20 GB Arbeitsspeicher. Dieser Prozessor ist für die ihm aufzutragenden Datenbanktasks gut geeignet und mit den 20GB Arbeitsspeicher besteht ausreichend Platz für einen hohen Datenbankcache, um deren Performance zweckmäßig zu

verbessern. Während der ersten Testläufe mit ca. 20-Millionen Datensätzen ist aufgefallen, dass die HDD des Servers ein großer Flaschenhals war, da MySQL viele Datensätze von der Festplatte immer wieder lesen und schreiben musste. Um diesen Flaschenhals zu umgehen wurde zunächst eine 500GB SSD eingebaut, auf der sich nur die Datenbanken befinden. Wie sich herausgestellt hat, neigt MySQL dazu, viele Datensätze vor ihrer Verarbeitung in temporäre Datenbanken zu schreiben. Da dies häufig auch bei Massenoperationen auftritt, wurden zwei kleinere SSD´s in einem RAID0 eingebunden. Entschieden wurde sich für ein RAID0, da dieser RAID-Verbund die besondere Eigenschaft aufweist, dass Daten abwechselnd auf beide Platten geschrieben werden. Dies hat den Effekt, dass sich die Lese- und Schreibgeschwindigkeiten der ohnehin schon schnellen Festplatten verdoppelt (EK1). Standardmäßig verfügt jede Platte über eine Lesegeschwindigkeit von 520 MB/s und eine Schreibgeschwindigkeit von 420MB/s (Intenso1). Somit haben beide im Raid0-Verbund theoretisch eine Lesegeschwindigkeit von 1040 MB/s und eine Schreibgeschwindigkeit von 840MB/s. Im Vergleich dazu hatte die zuvor verwendete 4TB IronWolf-Festplatte von Seagate nur Datenübertragungsraten von 210 MB/s (Seagate1). So konnte hierdurch ein großer Flaschenhals beseitigt werden.

Da die Softwareschnittstelle des Servers keine genaue Anzahl an Crawlern festlegt und somit theoretisch beliebig viele (innerhalb des Möglichen der Netzwerkkonfiguration) angeschlossen werden können, war hierfür die Wahl der Hardware nicht von essenzieller Bedeutung. Die Software der Crawler arbeitet hauptsächlich prozessororientiert, daher sollte für den Hauptbetrieb auf Rechner mit einem leistungsstarken Prozessor gebaut werden. Innerhalb des Versuchsaufbaus werden zwei ehemalige Hochschulserver verwendet und 6 Poolrechner. Aus den Serverlogs ist deutlich abzulesen, dass die meisten Webseiten von den beiden ehemaligen Servern analysiert werden, aufgrund ihrer höheren CPU Leistung.

Für die Netzwerkkommunikation werden zwei TP-Link Switche (TL-SG108 V3 8-Port) eingesetzt. Diese Switche verfügen jeweils über eine Datendurchsatzrate von 2000Mbit/s, was für den hier vorliegenden Anwendungsfall sehr ausreichend ist.

5.4 Datenbankumsetzung

Als Datenbanksystem wird im Rahmen des Prototyps eine relationale MySQL-Datenbank verwendet. Diese ist zwar im BigData Bereich nicht das schnellste Datenbanksystem, jedoch ist der Umgang mit diesem sehr benutzerfreundlich.

In der Datenbank existieren drei Tabellen. In der Tabelle **Hosts** sind Webseiten hinterlegt, die als Startbedingung für die Crawler dienen, wenn in der **Link**-Tabelle noch

keine neuen Links vorhanden sind. URLs aus der Host-Tabelle besitzen eine höhere Priorität als URLs aus der Link-Tabelle. Sollten die Crawler keinen neuen Links mehr auf Basis der vorhanden Daten aus der Link-Tabelle finden, greifen diese automatisch wieder auf URLs, aus der Host-Tabelle zurück. Die Struktur der Hosttabelle ist sehr einfach gehalten. Es existiert für die Indexierung nur eine Spalte `id` (int), eine Spalte `url`(varchar) und eine Spalte `TimeStamp` (timestamp) in denen die entsprechenden Daten hinterlegt sind.

In der Tabelle **Links** sind alle URLs hinterlegt, die während des crawlens gefunden wurden, sind. Wenn diese noch nicht analysiert wurden enthalten alle Spalten, außer die Spalte `link`, ihren Default-Wert. In der Tabelle Links existieren insgesamt acht Tabellen, in denen neben dem Link und dem Indizierungszeitpunkt alle Dateiinformation und Inhalte des hinter dem Link stehenden Dokuments bzw. Datei abgespeichert sind. Die gesamte Datenbankstruktur kann in Abbildung 22: Datenbankmodell (UML) eingesehen werden.

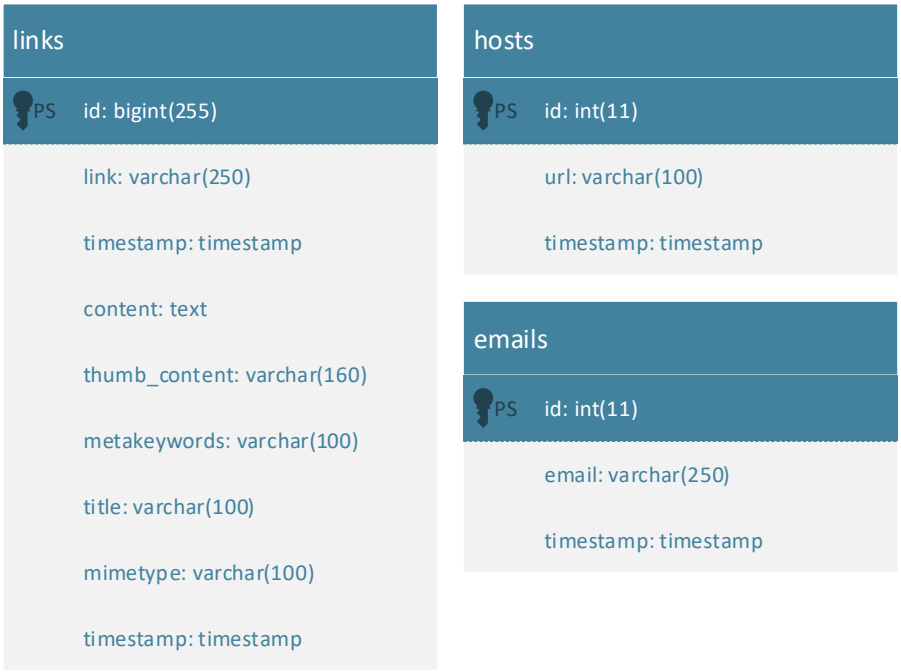

Abbildung 22: Datenbankmodell (UML)

Neben den beiden gerade genannten Tabellen, existiert noch eine weitere Tabelle, **Emails**. Deren Bedeutung ist aber weniger relevant für das Projekt, als die beiden anderen. In dieser Tabelle werden nur Emailadressen abgespeichert die auf einer Webseite mit dem HTML-Code „mailto:“ eingebunden wurden. In der frühen Entwicklungsphase war es geplant, diese bei der Suche mit auszugeben. Da dies aber unter Umständen nicht immer mit der vorliegenden Datenschutzerklärung vereinbar ist, wurde diese Funktion im Frontend nie umgesetzt. Bei der Modellierung des

Datenbankmodels war es nicht nötig Fremdschlüsselbeziehungen zu nutzen. Alle Tabellenfelder die häufig in SQL-Statements verwendet werden, wurden indexiert, um die Performance zu erhöhen.

Wie in Abbildung 22 zu sehen ist, verfügt das System über eine sehr einfache Datenbankstruktur. Grund hierfür ist, dass durch den einfachen Aufbau Speicher gespart wird und Suchanfragen im Anschluss darauf schneller verarbeitet werden können.

5.5 Softwareumsetzung

5.5.1 Serversoftware

Für die Umsetzung der Softwarekomponenten wurde auf verschiedenen Programmiersprachen aufgebaut. So sind in gewissen Softwareteilen JavaScript oder auch C# zu finden. Alle wichtigen Kernbestandteile wurden mit Java umgesetzt.

Die Kommunikation zwischen den Crawlern und dem Server, wurde mithilfe von Sockets in Java umgesetzt. Wie in Listing 1 zu sehen ist, wartet der Server in einem eigenständigen Thread, in dem sich eine Endlosschleife befindet, die auf eine eingehende Anfrage eines Clients auf der Socket Schnittstelle wartet. Diese eingehende Verbindung erkennt er, sobald die zuvor instanziierte Klasse `*ServerSocket* ` an der Methode „*accept()"* ein Socket-Objekt zurückgibt. Dies ist in Zeile 10 zu sehen. Solange keine Verbindung eingegangen ist, wartet der ServerSocket bei dem „accept()" oder bis der Timeout des Sockets abgelaufen ist. Sollte nun der Socket-Timeout erreicht worden sein, wird eine „IOException" geworfen. Diese wird im Programm mit dem in Zeile 9 beginnenden „try-Block" abgefangen. Um nun Memory-Leaks zu verhindern wird in Zeile 14 versucht die Verbindung korrekt zu beenden. Dies ist nötig, da es auch vorkommen kann, dass eine „IOExeption" von einem anderen Grund verursacht werden könnte als durch den „Socket-Timeout". Ein Beispiel hierfür wäre eine Protokollverletzung eines Clients. Im Anschluss wird die Socket-Schnittstelle erneut aufgebaut. Beim ersten Durchlauf der Schleife muss überprüft werden, ob dies der erste Durchlauf ist und wenn ja, muss zunächst der Socket gestartet werden. Zu sehen ist dieses Vorgehen in den Zeilen 5 bis 8. In der Methode „start_socket()", passiert nicht viel. In dieser wird mit "new ServerSocket(int port)" ein neuer Socket erstellt. Dabei wird im Konstruktor ein gewünschter Port übergeben. Im Anschluss wird überprüft, ob der Socket erfolgreich erstellt werden konnte. Dies wird mit der Method „isBound()" überprüft. Sollte der Port, der bei der Erstellung des Socket übergeben wurde, belegt sein, kann keine Verbindung hergestellt werden und „isBound()" gibt ein „false" zurück. In diesem Fall wird die Anwendung geschlossen.

```
1   Thread thread;
2   public int d = 0;
3   public void run() {
4       while (true) {
5           if (d == 0) {
6               start_socket();
7               d = 1;
8           }
9           try {
10          thread = new Thread(new Socket_Client_Connection(serversocket.accept()) );
11              thread.start();
12          } catch (IOException e) {
13              try {
14                  serversocket.close();
15              } catch (IOException e1){
16              e1.printStackTrace();
17              }
18          start_socket();
19          e.printStackTrace();
20          LogManager.writeErrorLog(e);
21          }
22      }
23  }
```

Listing 1: Serverseitige Annahme der Client-Connection

Eine der Hauptaufgaben des Servers ist die Datenbankkommunikation, da es zu unsicher wäre die Crawler direkt mit der Datenbank zu verbinden. Damit man in Java MySQL verwenden kann muss zunächst in das Projekt die entsprechende MySQL-Connector Libary eingebunden werden. Diese kann auf der offiziellen MySQL Webseite heruntergeladen werden (MySQL1). Der Prototyp baut darauf auf, dass für jeden Request eine eigene Verbindung zum Datenbank-Server erstellt wird. So ist eine andere Verbindung nicht direkt vom Abbrechen eines anderen Tasks betroffen und die Parallelisierung der Tasks verläuft flüssiger. In Listing 2 ist zu sehen, wie eine Verbindung zum MySQL-Server hergestellt wird. Da die gesammte Kommunikation über den „localhost“, also innerhalb der Maschiene erfolgt kann auf SSL verzichtet werden, da hier kein Sicherheitsrisiko betseht und so ein geringfügiger Overhead vermieden wird.

```
1   public Connection InitDataBaseConnection() {
2       try {
3           return DriverManager.getConnection("jdbc:mysql://"+SQL_Host+"/"+
4           SQL_DB+"?autoReconnect=true&useSSL=false&"+"user="+
5           SQL_User+"&password="+SQL_PW+"");
6           } catch (Exception e) {
7               System.err.println("Fehler beim Aufbau der Datenbankverbindung");
8               e.printStackTrace();
9               LogManager.writeErrorLog(e);
10              return null;
11          }
12  }
```

Listing 2: Aufbau Datenbankverbindung

Alle Aufgaben, die in Verbindung mit der Datenbank stehen, wurden zur besseren Organisation in einer separaten Klasse („DatabaseRequests“) gekapselt. Diese basieren

alle auf demselben Grundgerüst. Im Folgenden wird eine dieser Methode genauer erläutert (siehe Listing 3).

Die Methode „updateMimeTypeForUrl()“ ist eine sehr häufig verwendete Methode. Diese wird von den Crawlern nach der ersten Analyse der Webseiten auf dem Server aufgerufen, um für die spätere Verwendung den MimeType zu setzen. Da zwingend davon ausgegangen werden muss, dass sich auf den zu untersuchenden Webseiten SQL-Injections befinden können oder auch die Kommunikation zwischen Clients und Server korrumpiert werden kann, ist es in diesem Anwendungsfall zwingend erforderlich, die übergebenen Daten auf Korrektheit zu überprüfen. Hierfür stehen in der „Java-MySQL-API“ PreparedStatment´s zur Verfügung. Bei diesen werden zunächst die SQL-Statements mir Stellvertreteroperatoren für die Variablen erstellt, wie dies in Zeile 4 von Listing 3 zu sehen ist. Im Anschluss daran können die Stellvertreteroperatoren durch ihren eigentlichen Wert ersetzt werden wie dies in Zeile 6 und 7 zu sehen ist. Sollte nun in einer der Variablen versucht wurden sein SQL-Code einzubringen wird eine SQL-Exception geworfen und so verhindert, dass das Statement auf dem Server ausgeführt wird. Abhängig von dem übergeben SQL-Query muss das Statement unterschiedlich ausgeführt werden. In dem in Listing 3 aufgeführten Fall ist es erforderlich, das Statement mit „executeUpdate()“ auszuführen.

```
1   public void updateMimeTypeForUrl(String url, String mimetype) {
2       try {
3           Connection con = database.InitDataBaseConnection();
4           PreparedStatement statement = con.prepareStatement("UPDATE `links` SET
5           `mimetype`=? WHERE `link`=?;");
6           statement.setString(1, mimetype);
7           statement.setString(2, url);
8           statement.executeUpdate();
9           database.CloseDataBaseConnection(con);
10      } catch (SQLException e) {
11          e.printStackTrace();
12          LogManager.writeErrorLog(e);
13      }
14  }
```

Listing 3: Beispielausführung eines Datenbankquerys

Bei Querys, bei denen eine Antwort zurückerwartet wird, muss das Statement mit „executeQuery()“ ausgeführt werden (siehe Listing 4). Diese Methode liefert dann ein „ResultSet“ zurück, welches alle Ergebnisse enthält. Um alle Ergebnisse aus dem „ResultSet“ auszulesen, nutzt man die Methode „next()“ innerhalb des Kopfes einer While-Schleife. Dann ist es möglich wie in Zeile 3 und Zeile 4 zu sehen ist die benötigten Datensätze zu extrahieren.

```
1   ResultSet rs = stmt.executeQuery();
2   while (rs.next()) {
3       Host host = new Host(rs.getInt("id"), rs.getString("url"),
4       rs.getTimestamp("timestamp"));
5       hostliste.add(host);
6   }
```

Listing 4: SQL-Query mit Rückgabe

Nachdem, wie in Listing 1 zu sehen war, für einen Client ein neuer Thread für die Verbindung angelegt wurde, werden in diesem, wie in Listing 5 zu sehen ist, alle Input- und OutputStreams korrekt konfiguriert. Im Anschluss darauf sendet der Server dem Client eine Willkommensnachricht, damit dieser weiß, dass die Verbindung korrekt hergestellt worden ist. Danach beginnt eine Schleife, die solange läuft, bis die Verbindung zwischen Client und Server unterbrochen wurde. In dieser findet der Großteil der Kommunikation statt. Nachdem der Client dem Server mitgeteilt hat, welche Methode er auf dem Server verwenden möchte, wird die entsprechende Unterfunktion aufgerufen. In dieser teilt der Client dem Server alle nötigen Daten mit. Hierbei ist es wichtig, dass alle Daten im korrekten Format und in der richtigen Reihenfolge übermittelt wurden, da sonst der Server die Daten nicht verarbeiten kann und die Verbindung somit abbrechen muss. Der Server übergibt nach Verarbeitung des Inputs seine Antwort dem Client über den „ObjectOutputStream“.

```
1   client_input = new ObjectInputStream(socket.getInputStream());
2   client_output = new ObjectOutputStream(socket.getOutputStream());
3   client_output.flush();
4   ip = socket.getInetAddress().getHostAddress();
5   port = socket.getPort()+"";
6   System.out.println("\tClient connectet [IP, Port] " + ip + " " + port);
7   ClientManager.AddClientToList(ip, port);
8   client_output.writeUTF("Willkommen auf dem Server");
9   client_output.flush();
10      while (socket.isConnected()) {
11          switch(client_input.readUTF()) {
12              case "requestNextWebpage":
13                  requestNextWebpage();
14              break;
15              case "requestNextHtmlWebpage":
16                  requestNextHtmlWebpage();
17              break;
18              case "deleteUrlFromServer":
19                  deleteUrlFromServer();
20              break;
                ...
            }
23      client_output.flush();
24      }
25
```

Listing 5: Serverseitiger Clientlistner

Neben dieser Kernfunktion verfügt der Server noch über weitere kleine Funktionen, die nun kurz erläutert werden sollen. Bevor Content in die Datenbankmethoden übergegeben wird, überprüft der Server, ob der übergebene String nur gültige Zeichen

enthält. Dafür wurde eine Whitelist mit Chars auf dem Server hinterlegt. Alle nicht gültigen Chars werden aus der Zeichenkette entfernt. Auch Satzzeichen, Klammern und ähnliche Sonderzeichen werden entfernt, um Speicher in der Datenbank zu sparen. Diese Zeichen spielen für die spätere Suche keine Rolle.

Im Rahmen der Serveradministration war es noch nötig, einige Konsolenkommandos in das Programm zu integrieren. Hierfür wurde ein eigner Thread angelegt, der die ganze Zeit auf die Tastaturinputs wartet. In diesem Thread befindet sich ein „Scanner“ der alle eingegebenen Zeilen auf der Konsole einlesen kann. Das Programm stellt so nun 5 verschiedene Konsolenkommandos zur Verfügung:

1. help,
2. show,
3. hide,
4. clients,
5. allclients.

Das Kommando „help“ gibt einfach nur eine Erklärung über alle anderen Kommandos in der Konsole aus. Die beiden Kommandos „hide“ und „show“ dienen dazu, Low-Prio-Ausgaben anzeigen zu lassen beziehungsweise nicht anzeigen zu lassen. Low-Prio-Ausgaben sind in diesem Fall Ausgaben, die nicht wichtig für das Programm sind, aber eine Information zum aktuellen Stand ausgeben. Im weiteren Sinn können diese auch als Debug-Ausgaben bezeichnet werden. In diesen Ausgaben enthalten sind meist nur Informationen, welche Webseite gerade neu hinzugefügt wurden ist und welche Datensätze gerade aus der Datenbank gelöscht wurden. Da diese Ausgaben jedoch die Konsole sehr stark überfluten können, können diese mit den beiden Kommandos gesteuert werden. Das Kommando „clients“ gibt alle Verbindungen zu den aktuelle verbunden Clients an mit den Entsprechenden Endpoint-Informationen. Da es möglich ist das ein Client mit mehreren Instanzen auf den Server zugreift greifen gibt es noch den Command „allclients“. Dieser gibt auch alle weiteren Clientinstanzen aus.

In einem Testdurchlauf mit einer erhöhten Anzahl von Crawlern sind serverseitig einige Performance Probleme aufgetreten. Das Hauptproblem war, dass jeder Request eines Crawlers der einen Link verlangte, immer eine Datenbankverbindung aufbauen musste. Aus diesem Grund wurde Serverseitig ein Cache angelegt, in dem von einem statischen Thread automatisch Links aus der Datenbank hinzugefügt werden. Dieser Thread holt mit jedem Request eine hohe Anzahl von nicht gecrawlten Links aus der Datenbank und fügt diese dem Linkcache hinzu, solange in diesem Cache ein gewünschter Sollwert an Links nicht erreicht ist. Gleichzeitig wird der Linkcache als Ansatz für eine Blacklist genutzt. Wie sich in Testdurchläufen gezeigt hat, ist es empfehlenswert diverse Seiten

wie Ebay vom Crawling auszuschließen, da dort auf vielen Unterseiten nur temporärer Content erzeugt wird, der nach einer bestimmten Zeit nicht mehr verfügbar ist und nur die Suchergebnisse negativ beeinflussen würde. Dabei ist die Blacklist sehr trivial gehalten. Es existiert ein Listenobjekt in dem Strings mit verbotenen Begriffen enthalten sind. Beim Hinzufügen eines Links in den Cache wird überprüft ob der übergebene String des Links einen dieser verbotenen Begriffe enthält und löscht diesen dann gegebenen falls direkt aus der Datenbank.

Auch hat sich somit die Methode „GetOldestLink()“ mit der eben genannten Änderung stark verändert. Wie in Listing 6 zu sehen ist kann nun aus der „ConcurrentLinkedQueue“ mit der Methode „poll()“ das oberste Element aus der Liste geholt werden. Sollte kein Element vorhanden sein, gibt die Methode ein ‚null‘ zurück. Infolge dessen muss in einer While-Schleife darauf gewartet werden, bis ein Link vorhanden ist. Sobald ein Link vorhanden ist, wird dieser von der Methode zurückgegeben.

```
1   public Link GetOldestLink() {
2       Link link = null;
3       link = oldLinkCache.poll();
4       if (link != null) {
5           return link;
6       }
7       while (link == null) {
8           try {
9               Thread.sleep(1);
10          } catch (InterruptedException e) {}
11          link = oldLinkCache.poll();
12      }
13      return link;
14  }
```

Listing 6: Einen Link aus dem Cache holen

Da ähnliche Probleme beim Hinzufügen eines Links und beim Updaten dessen Informationen entstanden sind, wurden diese Methoden auch stark überarbeitet.

So werden neue Links nun auch nicht mehr direkt in die Datenbank gespeichert, sondern werden zunächst gecacht. Das Caching übernimmt in diesem Fall ein „Set“. Ein Set besitzt den wesentlichen Vorteil gegenüber anderer Listentypen, dass in diesem jedes Element nur einzigartig sein darf. Da anlegen eine Set´s in Java ist Listing 7 in zu sehen. So wird beim Hinzufügen zu diesem bereits überprüft ob ein Link innerhalb dieses Caches schon bekannt ist und einige doppelte Insert Versuche bleiben damit der Datenbank erspart. Sobald Eintausend Links in dem Set enthalten sind werden diese automatisch von einem Thread zur Datenbank übertragen und aus dem Set gelöscht.

```
1   public static Set<String> url_cache = Collections.newSetFromMap( new
2   ConcurrentHashMap<String,Boolean>());
```

Listing 7: Anlegen eines Set

Ähnlich dazu werden nun Updates an den Link Informationen ausgeführt. An dieser Stelle wird jedoch anstatt eines Set´s eine „ConcurrentHashMap“ verwendt. Diese bietet die Möglichkeit, mehrere Objekte verknüpft abzuspeichern wie auch bei normale Listen, jedoch sind Hashmaps dabei deutlich schneller. Sobald eintausend Updates eingegangen sind, wird ein entsprechender Query zur Datenbank gesendet. Zu sehen ist diese HashMap in Listing 8.

```
1   public static ConcurrentHashMap<String, Link> update_cache =
2   new ConcurrentHashMap<String, Link>();
```

Listing 8: Anlegen einer Threadsicheren HashMap

All diese kleineren Änderungen haben dazu geführt, dass das Java Programm jetzt keinen Flaschenhals mehr darstellt und die Crawler viel schneller und effizienter arbeiten können.

5.5.2 Crawler Implementierung

Bevor die Software auf einem Rechner lauffähig ist, müssen auf diesem zuvor einige Konfigurationsschritte getan werden. Diese Konfigurationsschritte unterscheiden sich abhängig nach dem jeweils verwendeten Betriebssystem. Im Folgenden wird die verwendete Standardkonfiguration für ein Ubuntu 16.04-System erläutert, wie sie auf allen Crawler-Rechnern des Projektes verwendet wird. Als Basis wird auf dem Rechnern ein Installiertes Ubuntu 16.04 (64 Bit) benötigt. Als erstes müssen die Programme Screen (lässt Anwendungen im Hintergrund weiterlaufen), Htop (Taskmanager für Unix Systeme) und Java (Runtime Environment) installiert werden. Diese sind alle in den Paketlisten vorhanden und können einfach mit apt installiert werden. Da der Autoupdater unabhängig von der Java-JVM, in einer C#-Anwendung läuft, wird außerdem eine dotnet Installation benötigt. Leider stellt Microsoft dieses nicht in den Paketlisten von Ubuntu zur Verfügung, weshalb es nötig ist dieses manuell aus dem Internet zu laden. Ein eleganter weg dieses Problem zu lösen ist es Dotnet den lokalen Paketlisten hinzuzufügen. Dazu holt man zunächst mit:

`wget -q HTTPs://packages.microsoft.com/config/ubuntu/16.04/packages-microsoft-prod.deb`,

das benötigte Paket und fügt es mit:

`sudo dpkg -i packages-microsoft-prod.deb`,

den lokalen Paketlisten hinzu. Im Anschluss dran kann man einfach mit apt die Pakete apt-transport-HTTPs und dotnet-hosting-2.0.6 installieren. Da das Programm nach einem Computerneustart automatisch wieder starten soll, muss man einen Autostart anlegen. Unter Unix gibt es hierfür verschiedene Methoden. Für den Prototypen wurden

ein Service angelegt, der gestartet wird, sobald alle Netzwerkkomponenten, Dotnet und Java geladen sind. Um dies zu erreichen, wurde ein entsprechendes SH-File (Shell Script) unter „/etc/init.d/“ angelegt. Im Anschluss darauf muss dieses mit

```
update-rc.d crawlerautoupdater defaults
```

und

```
update-rc.d crawlerautoupdater enable
```

aktiviert werden. Nun kann dieser Service mit

```
start crawlerautoupdater
```

beziehungsweise mit

```
stop crawlerautoupdater,
```

gestartet und gestoppt werden.

Da es im Rahmen des Prototypens auch zu Abstürzen kommen kann und eine Fernwartung der Rechner nur bedingt möglich ist, wurde ein täglicher Neustart der Systeme mit crontab eingerichtet. Für die Einrichtung muss in crontab einfach nur der Eintrag: `0 6 * * * /sbin/shutdown -r +5`, hinterlegt werden. Dies hat zur Folge das der entsprechende Rechner jeden Morgen um 06:05 Uhr neugestartet wird. Nun müssen nur noch die Tensorflowlibarys in den Libary Ordner geladen werden und in die Root Verzeichnis der Autoupdater und die Labelstrings für das OCR. Nach einem Neustart des Systems ist dann das Programm lauffähig.

Die Aufgabe der Crawler besteht darin, Webseiten auf Inhalt, Titel, Host und enthaltende Links zu untersuchen und das Ergebnis dem Server zu übermitteln. Da während der Entwicklungsphase schon mehrere Crawler in Verwendung sind und es zu zeitintensiv ist, nach jedem Softwareupdate, die Crawler manuell zu updaten, wurde eine parallellaufende C#-Anwendung implementiert, die regelmäßig überprüft, ob der Client über die aktuellste Version verfügt. Sollte dessen Version veraltet sein, wird das Java-Programm gestoppt und die aktuellste Version vom Server heruntergeladen. Nach erfolgreichem Download wird das „jar“-File ausgeführt. Der sogenannte Autoupdater befindet sich dabei auch im Autostart der Crawler und startet somit automatisch nach einem Neustart des Rechner. In Listing 9 ist aufgeführt wie der Start eines Prozesses in C# funktioniert. Hierfür wird zunächst die Klasse „Process“ instanziert. Im Anschluss daran wird, wie in Zeile 2 zu sehen ist, der Name des zu startenden Programmes und die jeweiligen Argumente übergeben. Der im Listing 9 gezeigte Prozessstart ist für Unixsysteme vorgesehen. Unter Windows muss als erstes Argument von

„ProcessStartInfo“, „java.exe“ übergeben werden. Im Anschluss wird die ProcessStartInfo auf das Feld „StartInfo“ des zuvor instanzierten Processes gelegt. Infolgedessen kann der Prozess und damit das Programm mit dem Methoden-Aufruf „process.Start()“ gestartet werden.

```
1   process = new Process();
2   var processStartInfo = new ProcessStartInfo("java", "-jar crawler.jar");
3   process.StartInfo = processStartInfo;
4   process.Start();
```

Listing 9: Programmlaunch mithilfe von C#

Zum Stoppen des Programmes ist es nötig, von der instanziierten Klasse „Process“ die Methode „Kill()“ aufzurufen. Da dies aus einer anderen Methode geschieht, wurde das „Process“-Objekt global innerhalb der Klasse definiert.

Die Crawler-Software lädt zunächst während des Startens die für den Betrieb benötigte TensorFlow Libary. Diese steht für alle Betriebssysteme zur Verfügung und ermöglicht so eine plattformunabhängige Anwendbarkeit des Programmes. Hierfür ist es jedoch erforderlich, dass die entsprechende Bibliothek von Tensorflow auf dem System installiert ist. Bei Windows muss Besipielsweise die Tensorflow-Libary im Win-32 Ordner hinterlegt werden. Im Anschluss bestimmt es, wie viele Prozessorkerne der Java-Runtime zur Verfügung stehen. Dies ist mit *„Runtime.getRuntime(). Available-Processors()“* möglich. Ziel ist es, aus den Crawlern durch Auslastung aller Prozessorkerne die bestmögliche Leistung heraus zu holen. Für jeden verfügbaren Kern wird ein Thread erzeugt und der Scheduler der JavaRuntime sorgt dann dafür, dass alle Kerne gleichmäßig ausgelastet werden.

```
1   ThreadPoolExecutor threadPool = (ThreadPoolExecutor)
2   Executors.newFixedThreadPool(Runtime.getRuntime().availableProcessors());
3   for (int i = 0; i < Runtime.getRuntime().availableProcessors(); i++) {
4       WebCrawler webcrawler = new WebCrawler();
5       threadPool.submit(webcrawler);
6   }
7   while(true) {
8       int thread_count = threadPool.getActiveCount() ;
9       System.out.println(thread_count + " active");
10      while(thread_count < Runtime.getRuntime().availableProcessors()) {
11          WebCrawler webcrawler = new WebCrawler();
12          threadPool.submit(webcrawler);
13          thread_count++;
14      }
15      try {Thread.sleep(5000);} catch (InterruptedException e) {
16          LogManager.writeErrorLog(e);
17      }
18  }
```

Listing 10: Thread-Factory

Wie in Listing 10 zu sehen wird der eben beschriebene Prozess in den Zeilen 3 bis 6 abgearbeitet. Jeder erstellte Thread wird dem ThreadPoolExecuter hinzugefügt und von

diesem verwaltet. Sollte ein Thread abstürzen, wird dieser vom ThreadPoolExecuter automatisch gelöscht. Zwischen den Zeilen 7 bis 18 wird überprüft, wie viele Threads aktuell noch aktiv sind, sollten weitere Threads benötigt werden, werden diese innerhalb der While-Schleife in Zeile 10 beginnend erstellt.

Die Klasse WebCrawler ist der bedeutendste Kernbestandteil der Crawler. Diese Klasse erbt von der Runnable Klasse und kann somit als Thread ausgeführt werden. Ihre Aufgabe besteht darin, einen Link über die Socketschnitstelle vom Server anzufragen und anschließend diesen auf seinen Inhalt zu untersuchen. Dazu öffnet der Client einen InputStream zu der vom Server erhalten Webaddresse. Im Header jeder Webresponse wird ein MimeTyp übertragen. Dieser enthält Informationen, über welchen Dateitypen es sich bei der aktuellen Webresponse handelt und der Crawler kann diesen korrekt verarbeiten. Auch wird der MimeType direkt zum Server gesendet, damit dieser abspeichert werden kann. Da mit dem Frontend der Suchmaschine nur nach Bildern und Texten gesucht werden können soll, werden nur Links mit dem entsprechenden Mimetypen analysiert. Das heißt es werden nur Mimetypes mit den Typen:

- image/png,
- image/jpeg,
- image/x-icon,
- image/gif,
- image/svg,
- application/pdf,
- text/html,
- text/plain,
- application/vnd.openxmlformats-officedocument.wordprocessingml.document (.docx),
- application/vnd.openxmlformats-officedocument.spreadsheetml.sheet (.xlsx),
- application/vnd.openxmlformats-officedocument.presentationml.presentation (.pptx),

auf ihren Inhalt analysiert.

Bei Dokumenten mit den Typen „**text/html**“ und „**text/plain**“ handelt es sich um einfache Webseiten beziehungsweise Textdateien, die direkt verarbeitet werden können. Verarbeiten heißt, dass der Titel der Webseite, der Content der Seite, alle auf der Seite eingebetteten Links und alle auf der Webseite hinterlegten Emails zum Server übertragen werden. Der Titel der Webseite kann aus dem vorliegenden HTML-Text extrahiert werden. Dazu wird einfach die in Listing 11 gezeigte Methode

„getWebpageTitle()“ aufgerufen. Diese sucht im HTML-Dokument nach dem Tag <title> und gibt den Inhalt dessen zurück.

```
1   public String getWebpageTitle(String webpage) {
2       String pagetitle = "";
3       if (webpage.contains("<title")) {
4           char[] html_frag = webpage.toCharArray();
5           int i = webpage.indexOf("<title>") + 7;
6           if (i < html_frag.length) {
7               while (html_frag[i] != '<' ) {
8                   pagetitle=pagetitle + html_frag[i];
9                   i++;
10              }
11          }
12      }
13      return pagetitle;
14  }
```

Listing 11: Methode getWebpageTitle()

Danach wird der Inhalt für die Datenbank präpariert. Hierfür wird zunächst der Head entfernt und damit nicht bei jedem Suchergebnis nur der Titel im Beschreibungsfeld angezeigt wird, muss dieser auch aus dem Content entfernt werden. Außerdem wird der Zeichensatz decoded, damit dieser einheitlich auf der Serverseite ankommt.

Im Anschluss daran werden mit der Methode „findLinksOnWebpage()“ alle auf der Webseite eingebunden Links gesucht (Siehe dazu Listing 9). Zunächst wird in dieser Methode geschaut, ob diese „href=“ enthält. Wenn dies so ist, kann angenommen werden, dass sich noch mindestens ein Link auf der Webseite befindet. Dann wird der übergebene String in ein Char-Array geparst, damit man mit einer Schleife leichter darüber laufen kann. Nun wird der Index bestimmt an dem ein Link beginnt. Dieser Index wird dazu genutzt, eine Schleife zu starten, die jedes Zeichen in einen Buffer speichert, bis der Link zu Ende ist. Dies ist in Listing 12 in Zeile 9 bis 12 zu sehen. Im Anschluss darauf wird die aktuelle Methode rekursiv aufgerufen, jedoch wird dieses Mal der bereits analysierte HTML-Bereich entfernt. Dies hat zur Folge, dass der gesamte Inhalt der Seite analysiert wird. Bevor nun ein Link der Liste hinzugefügt wird, wird überprüft, dass dieser eine gewisse Länge nicht überschreitet. Dies ist nötig, da diverse Webseiten *„Fallen“* für Crawler erstellen indem sie Webseiten generieren auf denen immer länger werdende Links hinterlegt sind. Aus diesem Grund wird in Zeile 16 auf „%3Bamp%3Bamp“ und „&amp“ geprüft. Beides wird von einigen Webseiten automatisch erstellt und unendlich nach jedem Aufruf an den aktuellen Link angehangen, bis der Crawler irgendwann abstürzt. Auch kann es vorkommen, dass der Link leer ist. Dies wird in Zeile 18 abgefangen. Da sehr häufig immer wieder die gleichen Webseiten verlinkt werden, wurde ein Lokaler Cache eingerichtet, der alle zum Server gesandten Links zwischen speichert und es nur zulässt, Links die noch nicht zum Server gesandt wurden sind, zu

diesem zu senden. So werden unnötiger Traffic und unnötige Datenbankabfragen auf Serverseite vermieden.

```
1   public List <String> findLinksOnWebpage(String htmlline, String currenthost, String
2   cur_url, String pagetitel) {
3       List <String> linkliste = new  ArrayList<String>();
4       if (htmlline.contains("href=\"")) {
5           char[] html_frag = htmlline.toCharArray();
6           int i = htmlline.indexOf("href=\"") + 6;
7           if (i < html_frag.length) {
8           String url = "";
9           while (html_frag[i] != '"' && html_frag[i] != '\'' ) {
10              url = url + html_frag[i];
11              i++;
12          }
13          linkliste.addAll(findLinksOnWebpage(htmlline.substring(i,html_frag.length),
14          currenthost, cur_url, pagetitel));
15          if (url.length() < 7000) {
16              if (!(url.contains("%3Bamp%3Bamp")||url.contains("&amp"))) {
17                  String new_link = ValidateLink(currenthost, cur_url, url);
18                  if (!new_link.equals("") && !new_link.equals(" ")) {
19                      if (!Cache.CheckUrlIsAllreadyInList(new_link)) {
20                          System.out.println("Found Link: "+new_link);
21                          Cache.AddLinkToCache(new_link);
22                          linkliste.add(new_link);
23                      }
24                  }
25              }
26          }
27          }
28      }
29      return linkliste;
30  }
```

Listing 12: Methode findLinksOnWebpage()

Wie in Listing 13 zu sehen ist, handelt es sich bei dem Cache um eine sehr triviale Struktur. Wenn ein Link hinzugefügt wird, wird überprüft, ob dieser sich bereits im Cache befindet und wenn nicht wird er der HashMap hinzugefügt. Außerdem wird überprüft, ob noch mindestens 15% Arbeitsspeicher, welcher der **JVM** (**J**ava **V**irtual **M**achine) zur Verfügung steht, frei ist. Sollte kein Speicher mehr frei sein, wird vermieden den Link hinzuzufügen und somit riskiert, dass ein Link mehrmals zum Server gesandt wird. Anderenfalls kann es passieren, dass der Crawler mit einem „OutOfMemory“-Fehler abstürzt. An dieser Stelle wird eine threadsichere Hashmap verwendet, da auf diese von verschieden Threads zugegriffen wird und diese in Bezug auf Listen schnelle Zugriffe verarbeitet und dabei weniger Speicher benötigt.

```
1   public class Cache {
2
3       private static Set<String> cache =  Collections.synchronizedSet(new
4       HashSet<String>());
5
6       public static boolean CheckUrlIsAllreadyInList(String url)
7       {
8           return cache.contains(url);
9       }
10
11      public static void AddLinkToCache(String url)
12      {
13          if (!CheckUrlIsAllreadyInList(url)) {
14              double s = (double)1-(double)Runtime.getRuntime().freeMemory()/
15                      (double)Runtime.getRuntime().totalMemory();
16              if (s > 0.20) {
17              cache.add(url);
18              }
19          }
20      }
21  }
```

Listing 13: Funktionsweise des Caches auf Clientseite

Nachdem der Inhalt einer Webseite nach Links durchsucht wurde, wird diese auch noch nach eingebetteten Dokumenten und Emails durchsucht. Für das Finden von Dokumenten wurde eine zu der in Listing 12 ähnliche Methode erstellt mit dem Namen „findFilesOnWepage()“. Das Vorgehen ist fast identisch wie der in Listing 12 gezeigte Vorgang. Nur wird dieses Mal anstatt nach „href“ nach „src“ gesucht, da Files damit auf Webseiten eingebunden werden. Auch werden Bilder und Iframes auf Webseiten mit einem solchen Attribut eingebunden. Alle gefundenen Dokumente werden wie Links behandelt und später vom Programm analysiert, da diese unter Umständen in Form von PDF-Files oder Office-Dokumenten relevant sein können. In einem weiteren Schritt wird die Webseite mit der Methode „findEmailOnWebpage()“ dahingehend analysiert, ob Emailaddressen mit dem Attribut „mailto:“ eingebunden wurden. Dazu wird wieder eine abgewandelte Form des Codes aus Listing 12 eingesetzt der nach „mailto“ filtert. Wenn nun alle Emails, Links und eingebettete Dokumente auf der Webseite gefunden wurden, wird das Ergebnis über die Socketschnittstelle zum Server gesendet und von diesem dann in der Datenbank gespeichert.

Wie schon erwähnt, sind die Crawler in der Lage auch andere Dateitypen auf ihren Inhalt hin zu analysieren. Im Folgenden soll erläutert werden, wie die Crawler ein PDF-File analysieren. Dazu dient Listing 14 als Referenz. Erkannt wird ein PDF-File als solches, wenn im Mime Type „application/pdf“ als Dateityp hinterlegt ist. Leider ist Java standardmäßig nicht in der Lage, PDF-Files zu lesen. Im Rahmen des Prototypens wird die von Apache erstellte Libary „PDFBox“ verwendet, um die Dokumente zu analysieren [Apache1]. Diese ist unter anderem in der Lage, Text aus einem PDF-File zu extrahieren. Auch wäre es möglich, mit dieser selbst PDF-Files zu erstellen oder eine vorhandene

PDF in ein Bild umzuwandeln. Da dies aber im Rahmen des Prototypens nicht nötig war, wird es nicht weiter erläutert. Leider ist „PDFBox“ nicht in der Lage, PDF-Files aus einer Websource zu analysieren und benötigt unbedingt ein lokales File für die Analyse dessen. Eine elegantes Vorgehen hierfür ist es, ein temporäres File zu erstellen, welches nach Abschluss der Prozedur automatisch gelöscht wird. In Zeile 68 von Listing 11 ist zu sehen, wie ein solches File erstellt werden kann. Hierfür muss einfach nur die statische Methode „createTempFile()“ des Fileobjekts aufgerufen werden. Als Übergabeparameter erwartet diese Methode, ein Präfix für den Filenamen und eine Dateiendung. Der Dateiname selbst wird dann durch interne Prozeduren automatisch erzeugt. In der Regel werden alle temporären Files vom Betriebssystem nach dem Neustart automatisch gelöscht. Da es aber nicht zwingend notwendig ist, dass die Crawler neugestartet werden, muss noch wie in Zeile 4, von Listing 14 zu sehen ist, das File so geflaggt werden, dass dieses automatisch nach dem Schließen gelöscht wird.

Im Anschluss daran wird das File nun in das eben erstellte File heruntergeladen. Dazu wird in einen BufferdOutputStream der Input des Inputstreams abgespeichert. Dabei wird Byteweise vorgegangen und der Inhalt schrittweise abgespeichert, bis das Fileende erreicht ist. Nach Abschluss des Prozesses, wird der BufferdOutputStream geschlossen. Nun wird das heruntergeladene File mit der Klasse *PDDocument* geöffnet, die von der Apache Libary zur Verfügung gestellt wird. Da nur nicht verschlüsselte Files geöffnet werden können, muss nun zunächst geprüft werden ob das vorliegende File verschlüsselt ist. Dies geht recht einfach mit der Methode „isEncrypted()“ der Klasse *PDDocument*. Diese liefert ein „true“ zurück, wenn das Dokument verschlüsselt ist. Mithilfe des *PDFTextStripper* ist es nun möglich, den Textinhalt des Dokumentes zu extrahieren. Dafür muss nur die Methode „getText()“ auf das *PDDocument*, einer Klasse instanziierten Klasse, angewandt werden. Im Anschluss drauf erhält man den Inhalt des Dokumentes. Dieser wird im Prototypen zum Server gesendet. Nun muss nur noch das *PDDocument* mit der Methode „close()“ geschlossen werden, um MemoryLeaks zu vermeiden.

```
1   String readywebpage = "";
2   InputStream in = new URL(Url).openStream();
3   File tempFile = File.createTempFile("tmp_", ".pdf");
4   tempFile.deleteOnExit();
5   BufferedOutputStream bw = new BufferedOutputStream(new
6   FileOutputStream(tempFile));
7   byte[] input = new byte[1024*1024];
8   int amountRead = 0;
9   while((amountRead = in.read(input)) != -1) {
10      bw.write(input, 0, amountRead);
11      bw.flush();
12  }
13  bw.close();
14  PDDocument document = PDDocument.load(tempFile);
15  if (!document.isEncrypted()) {
16      PDFTextStripper stripper = new PDFTextStripper();
17      readywebpage = stripper.getText(document);
18  }
19  document.close();
20  server.updateLink(Url, "[PDF]", readywebpage);
```

Listing 14: Verarbeitung eines PDF-Files

Wie auch Google soll der Prototyp in der Lage sein, verschiedene Office Dokumente zu indexieren. Hierfür ist es notwendig, diese zuvor zu lesen. Alle üblichen Microsoft-Office Produkte lassen sich mithilfe der Open-Source Bibliothek „Apache POI“ öffnen (apache2). Diese steht kostenfrei für Java zur Verfügung. Hierbei muss aber klar zwischen dem alten Office-Speicherformat, welches bis Office 2003 genutzt wurde und dem aktuellen Speicherformat unterschieden werden. Zu erkennen ist das neue Speicherformat an dem angehängten ‚x‘. So heißt die alte Dateiendung für Word-Dateien einfach nur ‚.doc‘ und die neue ‚.docx‘. Das ‚x‘ in der Dateiendung steht für XML, da im Gegensatz zu dem alten Speicherformat die neuen Dateien mit XML-Code abgespeichert werden. Für den Prototypen ist dies nun dahingehend wichtig, dass sich nicht nur die Mime-Types unterscheiden, sondern auch die Zugriffsweisen auf die einzelnen Dateien. In der Apache POI Bibliothek stehen nun für die alten Dokumenten-Typen die sogenannten HWPF Klassen zur Verfügung und für die neuen die XWPF Klassen. Dabei unterscheiden sich beide in ihrer Anwendung nicht wesentlich voneinander. Beide nutzen identische Klassennamen, nur das entweder HWPF oder XWPF vor diesen gegangen wird. Methoden innerhalb der Klassen sind weitestgehend äquivalent. Wie schon erwähnt können beide Dokumententypen anhand ihres Mimetype unterschieden werden. So lautet der offizielle MimeType für *‚doc‘*-Dateien *„application/msword“* und für *‚docx‘*-Dateien *„application/vnd.openxmlformats-officedocument.wordprocessingml.document“*. Unabhängig von dem Dateiformat wird nun zunächst, wie schon zuvor bei den PDF-Files erläutert, das Dokument in ein temporäres File heruntergeladen. Im Folgenden soll nun anhand einer ‚doc‘- und einer ‚docx‘-Datei gezeigt werden, wie die Extraktion des Textes funktioniert. In Listing 15 ist nun zu sehen wie, eine Solche Textextraktion aussehen kann. Wie in Zeile 136 zusehen

ist wird zunächst die Klasse **HWPFDocument** instanziiert indem der FileInputstream dem Konstruktor übergeben wird. Im Anschluss darauf kann der WordExtractor auf dieses Dokument angewandt werden, wie dies in Zeile 3 zu sehen ist. Die Klasse WordExtractor enthält einige nützliche Methoden, so ist es möglich aus dem Header der Datei mit „getHeaderText()“ den Title des Dokumentes zu extrahieren. Hierbei ist aber darauf zu achten, dass diese Herangehensweise veraltet ist. Den gesamten Text aus der Datei kann dann mithilfe der Methode „getText()“, extrahiert werden.

```
1   FileInputStream fis = new FileInputStream(tempFile);
2   HWPFDocument doc = new HWPFDocument(fis);
3   WordExtractor extractor = new WordExtractor(doc);
4   if (extractor.getHeaderText() != null) {
5       title = extractor.getHeaderText();
6   } else {
7       title = extractor.getText().substring(100);
8   }
9   If (title.equals("")) {
10      title = extractor.getText().substring(100);
11  }
12  readywebpage = extractor.getText();
13  extractor.close();
```

Listing 15: Text aus einer doc-Datei extrahieren

Nun wird im nachfolgenden Listing 16 die gleiche Analyse an einer ‚docx‘-Datei gezeigt. Wie zu sehen ist, ist das Analyseverfahren fast identisch mit dem aus Listing 15. Wie jedoch in Listing 16 deutlich zu erkennen ist, steht dort vor jedem Klassennamen XWPF, für den entsprechenden Dateitypen. Da es für die neue Dateien keine Methode „getHeaderText()“ mehr gibt, werden hier einfach, falls möglich, die ersten 100 Zeichen aus dem Textfeld des Dokumentes als Titel abgespeichert, wie dies in den Zeilen 4 bis 8 zu sehen ist. Der Text selbst wird wieder einfach mit der Methode „getText()“ aus dem Dokument extrahiert.

```
1   FileInputStream fis = new FileInputStream(tempFile);
2   XWPFDocument doc = new XWPFDocument(fis);
3   XWPFWordExtractor extractor = new XWPFWordExtractor(doc);
4   if(extractor.getText().length() > 100) {
5       title = extractor.getText().substring(100);
6   } else {
7       title = extractor.getText();
8   }
9   readywebpage = extractor.getText();
10  extractor.close();
```

Listing 16: Text aus einer docx-Datei extrahieren

Da wie eben gezeigt, die Analyse zwischen dem alten und dem neuen Format sich programmtechnisch nicht wesentlich voneinander unterscheiden, wird im Folgenden für Excel-Dateien und PowerPoint-Dateien dies nur an dem alten Typen erklären.

Zunächst die Analyse eines Excel-Files. Bei einem Excel-File handelt es sich um einen, sehr speziellen Dateitypen, da dieser aus mehreren Arbeitsmappen bestehen kann,

wobei sich in jeder dieser Mappe wiederum Tabellen befinden können. Nun ist es das Ziel, den Inhalt aus jeder Arbeitsmappe und jeder Zelle zu extrahieren. Dazu wird wie in Listing 17 gezeigt vorgegangen. Zunächst muss die Klasse HSSFWorkbook initialisiert werden mit dem FileInputStream auf das Excel-File. Im Anschluss darauf ist man in der Lage mithilfe der Methode „getNumberOfSheets()“ zu bestimmen, aus wie vielen Arbeitsmappen das Excel-File besteht. Nun muss man innerhalb einer Schleife durch jede Arbeitsmappe durchlaufen, wie dies in Listing 17 zu sehen ist. Innerhalb dieser Schleife wird zunächst die aktuelle Arbeitsmappe mit der Klasse HSSFSheet verarbeitet. So ist es nun möglich, wie in Zeile 6 zu sehen ist, mithilfe eines Iterators über alle Zeilen hinweg zu laufen.

```
1   FileInputStream fis = new FileInputStream(tempFile);
2   HSSFWorkbook workbook = new HSSFWorkbook(fis);
3   for(int i = 0; i < workbook.getNumberOfSheets();i++) {
4       content += "[Sheet" + i + "]:\n";
5       HSSFSheet sheet = workbook.getSheetAt(i);
6       Iterator<Row> rowIterator = sheet.iterator();
7       ...
```

Listing 17: ExcelFile lesen

Wie in Listing 18 zu sehen ist, wird zunächst ein Iterator für die Spalten initialisiert. So ist es möglich mit 2 While-Schleifen die gesamte Mappe zu analysieren. Dies ist in den Zeilen 3 bis 6 zu sehen. In Zeile 7, wird eine Zelle mithilfe der Klasse Cell initialisiert wird. Da es nun verschiedene Typen von Zellen gibt, muss an diesem Punkt mithilfe der Klasse CellType überprüft werden, um welchen Typen es sich dabei handelt. Sollte an einer Zelle kein Type definiert sein, wird diese einfach übersprungen. Sollte die Tabelle eine Booleschen Wert enthalten, wird bestimmt, ob dieser ‚true‘ oder ‚false‘ ist und wird entsprechend hinterlegt. Wenn der Typ Blank hinterlegt ist, wurde nichts in die Zelle eingetragen und kann somit übersprungen werden. Wenn der Type Formular hinterlegt ist, muss wie in den Zeilen 21 bis 25 vorgegangen werden, um den Inhalt dessen zu extrahieren. Bei numerischen Feldern kann der Zahlenwert mit der Methode „getNumericCellVallue()“ entnommen werden. Analog dazu wird bei String-Werten vorgegangen nur dass diese mit der Methode „getStringCellValue()“ extrahiert werden.

```
1   HSSFSheet sheet = workbook.getSheetAt(i);
2   Iterator<Row> rowIterator = sheet.iterator();
3   while (rowIterator.hasNext()) {
4       Row row = rowIterator.next();
5       Iterator<Cell> cellIterator = row.cellIterator();
6       while (cellIterator.hasNext()) {
7           Cell cell = cellIterator.next();
8           CellType cellType = cell.getCellTypeEnum();
9           switch (cellType) {
10          case _NONE:
11              content += "\t";
12          break;
13          case BOOLEAN:
14              content += cell.getBooleanCellValue();
15              content += "\t";
16          break;
17          case BLANK:
18              content += "\t";
19          break;
20          case FORMULA:
21              FormulaEvaluator evaluator =
22              workbook.getCreationHelper().createFormulaEvaluator();
23              content += cell.getCellFormula();
24              content += evaluator.evaluate(cell).getNumberValue();
25              content += "\t";
26          break;
27          case NUMERIC:
28              content += cell.getNumericCellValue();
29              content += "\t";
30          break;
31          case STRING:
32              content += cell.getStringCellValue();
33              content += "\t";
34          break;
35          case ERROR:
36              content += "!";
37              content += "\t";
38          break;
39          }
40      }
41  }
```

Listing 18: Analyse eines Excel-Files

Nach erfolgreicher Analyse des Files wird der zusammengefasste Content zum Server gesendet. Um MemoryLeaks zu verhindern muss das Workbook am Ende geschlossen werden.

Ein weiteres weit verbreitetes Dateiformat sind PowerPoint-Dateien. Diese sind im Vergleich zu Excel-Files viel einfacher aufgebaut. In Apache POI werden diese als SlideShows bezeichnet, wie in Zeile 2 von Listing 19 zu sehen ist. Nun besteht jede Seite einer PowerPoint aus mehreren Shapes. Shapes können Textbereiche oder auch andere Einbindungen wie Bilder oder Grafiken sein. So ist es zunächst nötig, über alle Seiten der PowerPoint hinweg zu laufen. Umgesetzt wird dies mit einer einfachen For-Schleife, die in Zeile 3 beginnt. Mit der Methode „getSlides()“ ist es möglich eine Liste mit allen Slides zu erhalten. Daraufhin ist es möglich wie in Zeile 4 aufgeführt, von der

aktuellen Slide alle Shapes zu extrahieren. Wieder erhält man eine Liste, über die man mit einer For-Schleife iterieren kann. Im Anschluss wird in Zeile 6 geprüft, ob die übergebene Shape eine TextShape ist und wenn dies so ist, wird der Inhalt von dieser mit „getText()“ extrahiert und dem Content-Buffer hinzugefügt. Abschließend ist es noch erforderlich die Slide zu schließen, um Memory-Leaks zu verhindern.

```
1   FileInputStream fis = new FileInputStream(tempFile);
2   HSLFSlideShow ppt = new HSLFSlideShow(fis);
3   for (HSLFSlide slide : ppt.getSlides()) {
4       List<HSLFShape> shapes = slide.getShapes();
5       for (HSLFShape shape : shapes) {
6           if (shape instanceof HSLFTextShape) {
7               HSLFTextShape textShape = (HSLFTextShape)shape;
8               content += textShape.getText() + "\n";
9           }
10      }
11  }
12  ppt.close();
```

Listing 19: Analyse einer PowerPoint-Datei

Ein besonderer Teil der Suchmaschine soll die Möglichkeit sein, auch nach Bildern suchen zu können. Hierbei wird der Ansatz verfolgt, dass der Inhalt des Bildes erkannt werden soll, mithilfe von der Bilderkennung TensorFlow und des Textes auf dem Bild mithilfe von OCR erkannt wird. Somit ist es jedoch nicht möglich den Kontext, von der Seite auf dem das Bild eingebunden war, in der Suche einzubinden. Zunächst sei noch klarzustellen, dass nicht alle Bildformate mit den hier verwendeten Methoden analysiert werden können, dadurch beschränkt sich die Suche auf Bilder mit den Formaten (MimeType)

- image/gif,
- image/x-icon,
- image/jpeg,
- image/png.

Als Beispiel lassen sich Vektorgrafiken (image/svg) nicht analysieren. Wie auch die anderen Dateitypen zuvor werden auch Bilder mit einem gültigen MimeType zunächst in ein temporäres File heruntergeladen, damit diese analysiert werden können. Nachdem der Download des Bildes erfolgreich war, wird das Bild mit den statischen Methoden aus der selbst erstellten Klasse *ImageChecker* analysiert. Diese enthält die Methode „CheckImage()“. Diese Methode erwartet die Übergabe eines Pfades zu einem Bild und gibt als Ergebnis die Zusammenfassung von OCR und der TensorFlow Bilderkennung als String zurück.

Zu Beginn wird die OCR-Texterkennung auf das Bild angewendet. Ziel hierbei ist es, zum Beispiel Firmennamen aus Logos zu extrahieren. Um die OCR-Erkennung in Java

umzusetzen wird für den Prototypen Tess4J eingesetzt. Um diese einsetzen zu können, ist es auf Windowsgeräten jedoch zuvor erforderlich „VC++ 2017 Redistributable“ zu installieren oder auf Unix Maschinen „tesseract-ocr“ aus den Paketquellen (Tess4J1). In Listing 20 ist nun zu sehen wie im Prototypen die OCR-Analyse umgesetzt wurde. Zunächst muss die Klasse Tesseract, wie in Zeile 2 zu sehen, ist instanziiert werden. Im Anschluss darauf muss auf diese Instanz ein Sprachpaket gesetzt werden mit der Methode „setLanguage()“. Dies ist erforderlich, denn Tesseract ist auch in der Lage, in einem gewissen Rahmen sprachentypische Buchstabenverknüpfungen zu erkennen oder auch Schriftzeichen aus anderen Zeichensätzen (zum Beispiel chinesische Schriftzeichen). Im Paket von Tess4J sind zahlreiche Sprachpakete enthalten, die so gut wie alle Sprachen abdeckt. Da die Suchmaschine in der Lage sein soll, deutsch- und englischsprachige Begriffe zu finden, wird das OCR einmal auf die deutsche Sprache und einmal auf die englische Sprache angewandt. Zu sehen ist dies in Zeile 4, in der mit der Methode „setLanguage()“ „eng“ als Parameter übergeben wird, was eine Abkürzung für Englisch ist. In der darauffolgenden Zeile wird nun mit der Methode „doOCR()“ der Text des Bildes mit der zuvor gesetzten Sprache analysiert. Dessen Ergebnis wird dem Result-String hinzugefügt. Im Anschluss darauf wird das Gleiche nochmal für die deutsche Sprache gemacht. Hierfür wird dieses Mal in der „setLanguage()“ Methode, „deu“, für Deutsch übergeben. Damit ist die OCR-Analyse des Bildes abgeschlossen.

```
1  File image = new File(imagepath);
2  ITesseract instance = new Tesseract();
3  instance.setLanguage("eng");
4  result1 += "Englisch:\n";
5  result1 += instance.doOCR(image);
6  result1 += "Deutsch:\n";
7  instance.setLanguage("deu");
8  result1 += instance.doOCR(image);
```

Listing 20: OCR-Analyse

Nachdem das Bild nach Textfragmenten mit OCR durchsucht wurde, wird nun der Inhalt des Bildes untersucht. Dass bedeutet, dass versucht wird, Gegenstände und andere Objekte auf diesem zu erkennen. Damit dies möglich ist, wird ein neuronales Netz benötigt. Für den Prototypen wurde dabei das von Google frei zur Verfügung gestellte „Machine-Learning-Framework“ TensorFlow verwendet.

TensorFlow ist eine OpenSource Software-Bibliothek für die Umsetzung von rechenaufwändigen numerischen Operationen, insbesondere denen von Neuronalen Netzen. Dabei ist es möglich, parallel für verschiedene Plattformen zu entwickeln. Das bedeutet ein Programm ist gleichzeitig auf einem CPU, GPU oder TPU lauffähig. Auch ist die Entwicklung für Cluster Systeme oder mobile Anwendungen möglich. TensorFlow

wurde von Google entwickelt und ursprünglich unter anderem für die Gesichtserkennung in Google Maps eingesetzt. Mittlerweile hat Google dieses Framework veröffentlicht, damit es jedem Entwickler erleichtert wird, mit neuronalen Netzen zu arbeiten. (Tens)

In Listing 21 ist nun zu sehen, wie eine TensorFlow Implementierung in Java aussehen kann. Hier wird gezeigt, wie TensorFlow anhand eines bereits trainierten Netzes Entscheidungen treffen kann. Die Trainingsphase wird komplett vernachlässigt. Wie in Zeile 1 und 2 zu sehen ist, müssen Zunächst die Daten aus dem vortrainierten Netz extrahiert werden. Dazu werden zunächst alle antrainierten Gewichte, des Netzes geladen und in einem Byte-Array gespeichert (zu sehen in Zeile 1). Da das Ergebnis am Ende nur eine ID mit einer Wahrscheinlichkeit ist, ist es außerdem erforderlich, jede ID mit einer Bezeichnung zu versehen.

Die Aufschlüsselungen zwischen ID´s und Bezeichnungen sind in einem Textfile gespeichert. Dieses muss, wie in Zeile 80 zu sehen ist, zunächst in eine String-Liste geladen werden. Es ist darauf zu achten, dass beide Files später dem kompilierten Programm zur Verfügung stehen. Nachdem die beiden Files vollständig geladen wurden, muss nun noch das zu analysierende Bild geladen werden. Hierfür werden alle Bytes von diesem in ein Byte-Array geladen. Dieses Byte-Array wird im Anschluss daran, wie in Zeile 83 zu sehen ist, dazu verwendet, um die Tensor-Klasse zu instanziieren. Die Tensor-Klasse ist in der TensorFlow-API enthalten und kümmert sich im Folgenden um die Analyse des Bildes.

Im nachfolgendem Schritt wird versucht, den eben geladenen Graphen zu instanziieren. Nachdem dieser instanziiert wurde wird der Graph geladen. Sobald dies alles erfolgreich abgeschlossen wurde wird eine Session gestartet mit dem eben geladenen Graphen, die im Folgenden das Bild intern verarbeitet (siehe Zeile 7). Als Bildformat wird JPEG angegeben, da dies mit allen andern gängigen Bildformaten gleichermaßen gut umgehen gehen. Als Vergleichsfunktion wurde „softmax“ festgelegt. Diese ist dazu ausgelegt, zeilenweise das Bild zu analysieren. Nachdem dies nun alles ausgeführt wurde, kann das erlangte Ergebnis aus der Tensor-Klasse auf ein 2-dimensionales Float-Array kopiert werden. Da in diesem jetzt jedoch viele Informationen (Gewichte) hinterlegt sind, die für die eigentliche Bestimmung nicht wichtig sind, ist es ausreichend, alle Gewichte, die sich in der ersten Spalte befinden, in ein anderes Float-Array zu kopieren, da diese nun die Wahrscheinlichkeiten für die einzelnen Labels enthalten. Beginnend in Zeile 17 ist zu sehen, wie über alle Wahrscheinlichkeiten iteriert wird, um alle Labels zu bestimmen, bei denen das Neuronale Netz festgestellt hat, dass es sich bei diesem mit mindestens 2% Wahrscheinlichkeit um das entsprechende Objekt handelt. Die 2% haben sich bei Experimenten als ein guter Wert herausgestellt. Häufig

sind Objekte die mit einer 2% Wahrscheinlichkeit bestimmt wurden noch ausreichend korrekt. Zum Vergleich haben Objekte, die nichts mit einem Bild zu tun haben, eine Wahrscheinlichkeit im Fünf stelligen Kommabereich. Alle Labels, die nun in diese Schranke fallen, werden gemeinsam mit ihrer Wahrscheinlichkeit dem Ergebnis hinzugefügt, welches so nach Abschluss der Analyse in der Datenbank gespeichert wird.

```
byte[] graphDef = readAllBytesOrExit(Paths.get(modelpath,
"tensorflow_inception_graph.pb"));
List<String> labels  = readAllLinesOrExit(Paths.get(modelpath,
"imagenet_comp_graph_label_strings.txt"));
result1 += "\nTensorflow:\n";
byte[] imageBytes = readAllBytesOrExit(Paths.get(imagepath));
Tensor image = Tensor.create(imageBytes);
try (Graph g = new Graph()) {
    g.importGraphDef(graphDef);
    try (Session s = new Session(g);
        Tensor result = s.runner().feed("DecodeJpeg/contents", image).
        fetch("softmax").run().get(0)) {
        float[][] result_array = new float[1][1008];
        result.copyTo(dd );
        float[] labelProbabilities = result_array[0];
        int bestLabelIdx = maxIndex(labelProbabilities);
        for (int i = 0; i < labelProbabilities.length; i++) {
            if (labelProbabilities[i]>0.02f) {
                result1 += (labels.get(i) + " " + labelProbabilities[i] * 100f) + "%\n";
            }
        }
    }
}
```

Listing 21: Anwendung TensorFlow

5.5.3 Weboberfläche

Neben den eben genannten Backendsystemen verfügt der Prototyp auch über ein Frontend, in dem Clients nach Inhalten suchen können. Als Plattform für dieses System wurde ein Tomcat9-Webserver gewählt. Bei diesem handelt es sich um einen Open-Source-Webserver, der Java Servlets und JavaServer Pages implementiert und es so möglich macht, in Java geschriebene Web-Anwendungen auszuführen (Tomcat1). Der Tomcat9-Webserver befindet sich dabei auf dem gleichen Server wie die anderen Komponenten auch, eine Dezentralisierung des Systems ist aber jederzeit möglich. Auch wäre bei einem hohen Nutzeraufkommen ein Clustering über mehrere Systeme möglich.

In der folgenden Abbildung ist ein Screenshot des Frontends der Standardsuche des Prototypen zu sehen. Im oberen Menü kann man zwischen den beiden Suchmodis Standardsuche und Bildersuche wechseln. Bei der Standardsuche werden alle Textinhalte aus Webseiten, PDF-Files und Office-Dokumenten angezeigt, wo hingegen bei der Bildersuche nach Inhalten auf PNG´s, JPG´s oder auch GIF`s gesucht werden kann. Unter den Menüpunkten ist das Logo des Prototypens mit dem Schriftzug „Zypersearch“ zu sehen. Sollte auf dieses geklickt werden, wird ein Nutzer immer

automatisch zurück zur Startseite geleitet und die Suche zurückgesetzt. Direkt unter dem Logo befindet sich das Suchfeld mit dem nach Inhalten gesucht werden kann. Für die Suche muss ein Anwender nur seine Suchbegriffe in das Feld eingeben und kann diese dann entweder mit Enter oder dem Button „Suchen!“ bestätigen. Falls der Anwender die Präferenz haben sollte, nach einem bestimmten Dokumententypen zu suchen, kann er dieses mithilfe der Checkboxen unter dem Suchfeld dem Webserver mitteilen. Sollte keine Auswahl in den Checkboxen getroffen worden sein, werden alle indexierten Dokumente mit dem MimeTypen „text/“ ausgegeben. Wie zu sehen ist, verfügt das Web-Frontend auch über einen Footer. In diesem wird dem Anwender angezeigt, wie viele Links der Suchmaschine aktuell bekannt sind. Bei diesem Wert handelt es sich jedoch nicht um einen jederzeit exakten Wert, da dieser Serverseitig gecacht wird und nur in diversen Zeitabständen synchronisiert wird, da der hierfür zuständige SQL-Query sehr lange benötigt. Die Suchmaske der Bildersuche sieht der Suchmaske der Standardsuche sehr ähnlich. Der einzige optische Unterschied zwischen beiden ist, dass bei der Bildersuche anstatt „Webpages“, „PDF“ und „Office“ als Typenauswahl „PNG“, „JPG“ und „GIF“ zur Auswahl stehen.

Abbildung 23: Screenshot Zypersearch

Nachdem die Suche abgeschlossen wurde ändert sich das Layout der Seite etwas, damit genug Platz für die Suchergebnisse entsteht. So werden alle zuvor zentrierten Elemente, nach oben links in die Ecke verschoben und das Logo verkleinert. Zu sehen ist dies in der nachfolgenden Abbildung. Alle gefunden Suchergebnisse werden dann unter dem Suchfeld eingeordnet. Dabei ist der Seitentitel in einem blauen Farbton hervorgehoben. Unter diesem wird dann noch der Link zu der Seite in grün angezeigt. Zur besseren Verdeutlichung des Suchergebnisses werden auch die ersten 140 Zeichen der Seite als Beschreibung in schwarz angezeigt. Alle Suchergebnisse mit einer Linien voneinander getrennt. Wenn nun ein Anwender eine der Seiten besuchen möchte, kann er dies mit

einem einfachen Klick auf den Titel tun. Auch stehen in dieser Ansicht alle Funktionen für eine nächste Suche zur Verfügung.

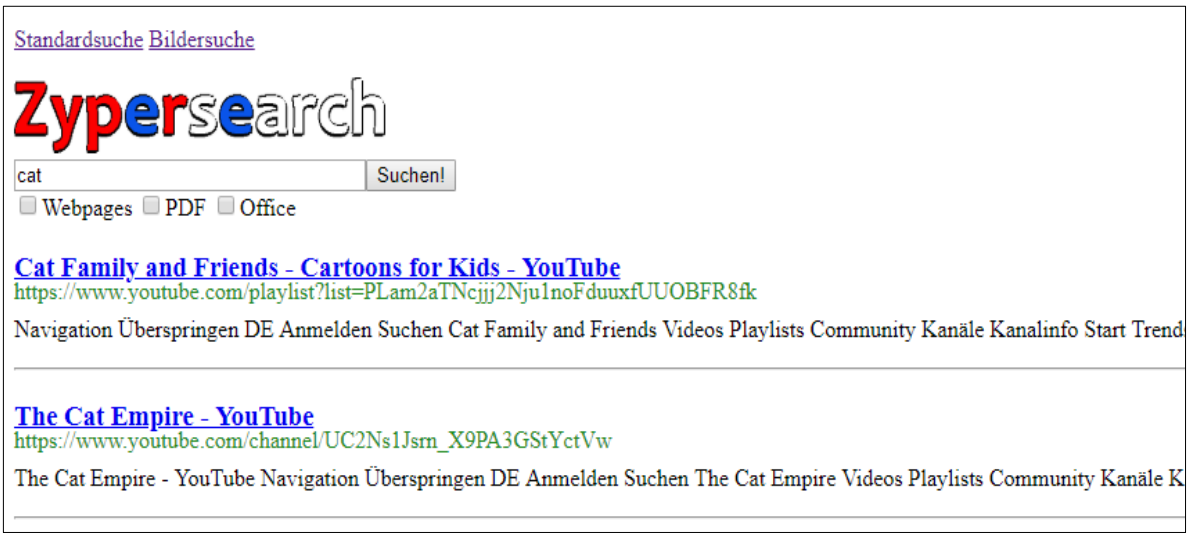

Abbildung 24: Suchergebnisse

In Abbildung 25 ist nun die Ergebnisdarstellung der Bildersuche zu sehen. Wie auch bei der Standardsuche hat bei der Bildersuche der Nutzer die Möglichkeit hier frei nach Suchbegriffen zu suchen. Nach einer erfolgreichen Suche werden die Suchergebnisse, in diesem Fall die Bilder, direkt nebeneinander angezeigt. Dabei werden diese zeilenweise nebeneinander angeordnet und bei mangelndem Platz automatisch in der darauffolgende Zeile angezeigt. Damit die Zeilen einheitlich aufgefüllt werden können, werden alle Bilder unabhängig ihrer vorherigen Größe auf eine Höhe von 100px gestreckt beziehungsweise gestaucht. Die Breite bleibt dabei jedoch proportional zur Höhe erhalten. Für das Beispiel wurde im Screenshoot nach dem Schlagwort „cat“ gesucht und wie klar zu erkennen ist, wurden verschiedene Arten von Katzenbildern gefunden. Dies beweist, dass das Neuronale Netz der Bilderkennung in der Lage ist, Bilder verschiedenster Art zu erkennen. Alle eingebunden Bilder beziehen sich stets auf ihre Herkunftsquelle, aus welcher diese auch geladen werden müssen. Dies hat unter Umständen ein längeres Nachladen zur Folge, wenn die Bilder auf einer langsamen Seite hinterlegt sind.

Abbildung 25: Ergebnisse der Bildersuche

Im Folgenden soll nun die softwaretechnische Umsetzung des Frontends mit seinen backend Funktionen erläutert werden. Die Basis des Frontends bildet eine HTML-Webseite, die aus den zuvor genannten Komponenten besteht. Neben den sichtbaren Komponenten existieren aber jedoch noch einige unsichtbare Komponenten. So sieht der Nutzer beim ersten Betreten der Webseite eine andere Suchmaske als die, wenn er die Suchergebnisse angezeigt bekommt. Grund hierfür waren Layout Probleme die andernfalls, das Design der Seite zerstört hätten. Jedoch sind zu jeder Zeit beide Suchmasken auf der Seite vorhanden, und es wird nur nach Bedarf das jeweilige Style Attribut angepasst. Alle Funktionen der nicht benötigten Suchmaske werden solange deaktiviert.

Nach der Betätigung des Such-Buttons wird die JavaScript-Methode „StartSearch()“ aufgerufen die in Listing 22 aufgeführt wird. Wie in Zeile 6 zu sehen ist, wird zunächst überprüft ob es bei diesem Aufruf um den Start der ersten Suche handelt. Wenn dies so

ist, muss die erste Suchmaske deaktiviert werden und dessen Inhalt in Suchmaske 2 dupliziert werden. Auch wird an dieser Stelle der Status der Check-Boxen abgefragt und auf Variablen gelegt (siehe Zeile 6 bis 19). Sollte es sich nicht um die erste Suche handeln werden die Inhalte direkt aus der 2.Suchmaske extrahiert und diese vorläufig deaktiviert (siehe Zeile 20 bis 28).

```
1   var first_search = 0;
2   function StartSearch() {
3       var web_search = false;
4       var office_search = false;
5       var pdf_search = false;
6       if (first_search === 0) {
7           first_search = 1;
8           document.getElementById("loading_image").style.visibility = "visible";
9           document.getElementById("search_field1").disabled = true;
10          document.getElementById("search_button1").disabled = true;
11          web_search = document.getElementById('webpages1').checked;
12          office_search = document.getElementById('office1').checked;
13          pdf_search = document.getElementById('pdf1').checked;
14          document.getElementById("search_field2").value =
15                  document.getElementById("search_field1").value;
16          document.getElementById('webpages2').checked = web_search;
17          document.getElementById('office2').checked = office_search;
18          document.getElementById('pdf2').checked = pdf_search;
19      }
20      else
21      {
22          document.getElementById("loading_image").style.visibility = "visible";
23          document.getElementById("search_field2").disabled = true;
24          document.getElementById("search_button2").disabled = true;
25          web_search = document.getElementById('webpages2').checked;
26          office_search = document.getElementById('office2').checked;
27          pdf_search = document.getElementById('pdf2').checked;
28      }
```

Listing 22: Start der Suche (1)

Weiterführend wird nun ein Ajax-Request initiiert, der die serverseitige Java-Schnittstelle des Servlets aufruft und dann den Inhalt der Webseite nachlädt.

Ajax (Asynchronous JavaScript and XML) ist die Bezeichnung für ein Konzept der asynchronen Datenübertragung zwischen Browser und einem Server. Hierdurch ist es möglich, HTTP-Anfragen durchzuführen, während eine HTML-Seite beim Anwender angezeigt wird. Damit kann die Seite verändert werden, ohne sie komplett neu laden zu müssen. (Wiki8) Als Referenz ist in Abbildung 26 ein Beispiel eines Ajax-Requests zu sehen.

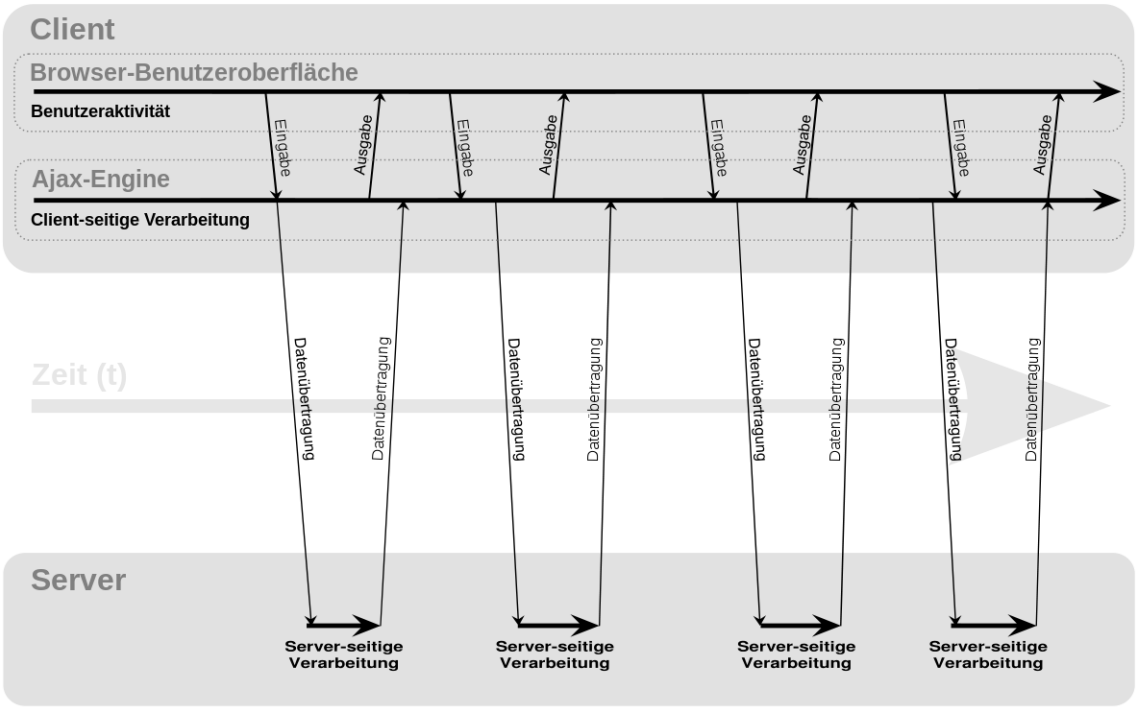

Abbildung 26: Beispiel eines Ajax-Requests (Wiki8)

Um den Ajax-Request umzusetzen, wird zunächst, wie in Zeile 2 von Listing 23 zu sehen ist, ein XML-HTTP-Request initialisiert. Nun wird in diesem ein Post zu dem entsprechenden Servlet, hier ZyperSearch, aufgebaut. In Zeile 6 ist nun zu sehen, wie darauf gewartet wird, dass eine Antwort vom Server zurückgesendet wird. Nun muss nur noch, wie in Zeile 24 zu sehen ist, die Anfrage gesendet werden. Dazu wird dem HTTP-Request die aufzurufende Methode, hier „SearchText“, des Servlets übergeben und alle dazugehörigen Parameter. Wenn nun die Anfrage erfolgreich verarbeitet wurde, ändert sich der „ReadyState“ der Anfrage auf 4 mit dem „Status“ 200. In einem Fehlerfall auf Serverseite können auch entsprechende Fehlercodes im Status zurückgeliefert werden. Diese sollen aber an dieser Stelle nicht weiter aufgeführt werden. Nachdem nun die Suche erfolgreich abgeschlossen wurde wird die Methode „SearchFinished()“ aufgerufen, diese löscht zunächst alle Suchergebnisse einer vorrangegangenen Suche und aktiviert wieder das Suchfeld und den Such-Button. Im Anschluss darauf werden aus der XML-Response alle gesammelten Ergebnisse mit der Methode „AddResElement()“ in das Ergebnisfeld eingeordnet, wie es in den Zeilen 10 bis 15 zu sehen ist.

```
1   var xmlhttp;
2   xmlhttp = new XMLHttpRequest();
3   xmlhttp.open("POST", "ZyperSearch", true);
4   xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
5   xmlhttp.onreadystatechange = function () {
6       if (xmlhttp.readyState === 4) {
7           if (xmlhttp.status === 200) {
8               SearchFinished();
9               var xmlDoc = xmlhttp.responseXML;
10              var x = xmlDoc.getElementsByTagName('result');
11              for (var i = 0; i < x.length; i++) {
12                  AddResElement(x[i].childNodes[0].childNodes[0],
13                    x[i].childNodes[2].childNodes[0], x[i].childNodes[1].childNodes[0],
14                    x[i].childNodes[3].childNodes[0]);
15              }
16              document.getElementById("results").innerHTML += "<br><br><br>";
17          }
18          else {
19              alert('Something is wrong!' + xmlhttp.status);
20          }
21      }
22  };
23  xmlhttp.send("method=SearchText" + "&" +
24      "value=" + document.querySelector("#search_field2").value +
25      "&web_search=" + web_search +
26      "&office_search=" + office_search + "&pdf_search=" + pdf_search + "");
27  }
```

Listing 23: Start der Suche (2)

Bei der Bildersuche läuft dieses Prozedere analog dazu ab. Für die Bildersuche wird serverseitig nur anstatt „SearchText“, „SearchImg“ mit den entsprechenden Parametern für die Bildersuche aufgerufen.

In Listing 24 ist zu sehen, wie der Server die Anfrage des Clients verarbeitet. Der Server empfängt darauf einen Request von dem Client. Wie in Zeile 1 dargestellt, holt sich der Server zunächst den Parameter, der als Methode übergeben wurde, um im nachfolgenden Switch/Case-Block zu bestimmen wie der Request verarbeitet werden soll. Wenn als Methode „SearchText“ übermittelt wurde, weiß der Server somit, dass dieser nach Webseiten suchen soll. Die in den Zeilen 5 bis 9 übermittelten Boolean-Werte bestimmen nun, ob Filter für die Suche des Clients gesetzt worden sind. Im Folgenden wird eine Verbindung zur Datenbank aufgebaut und mit der Methode „GetSearchResults()“ wird nach den übergeben Suchparametern gesucht. Als Rückgabewert gibt diese Methode eine Liste mit den entsprechenden „Results“ zurück. Result ist dabei eine selbsterstellte Klasse die als Container für die Daten aus der Datenbank dient. Das Paket mit den Antworten soll gebündelt im XML Format zurück zu dem Clienten gesendet werden. Dazu wird, wie in Zeile 10 zu sehen ist, zunächst ein outPutWriter initialisiert. Darauffolgend werden dann in Zeile 14 und 15 der Header und der Tag für die Antwort dem Output hinzugefügt. Innerhalb einer For-Schleife (siehe Zeile 17 bis 36) werden nun alle Ergebnisse diesem Output im XML-Format hinzugefügt. Dazu

erhält jedes Suchergebnis seinen eignen Result-Tag und in diesem sind dann alle dazugehörigen Informationen untergebracht. Wie in Zeile 20 zu sehen ist, müssen die Strings zunächst „Escaped“ werden, da im XML-Format nicht alle Zeichen übertragen werden dürfen.

Dabei wandelt die Methode „EscapeText()“ jeden Char der nicht mit XML übertragen werden kann mit *String.format („&#x%04x“, (int)c)+“;“;,(c=ein Char im Text) in Unicode um.*

Zum Ende muss noch der Tag „message_desc“ beendet werden wie es in Zeile 37 zu sehen ist.

```
1   String method = request.getParameter("method");
2       switch(method){
3          case "SearchText":
4            serachtext = request.getParameter("value");
5            boolean web_search =
6            Boolean.valueOf(request.getParameter("web_search"));
7            boolean office_search =
8            Boolean.valueOf(request.getParameter("office_search"));
9            boolean pdf_search = Boolean.valueOf(request.getParameter("pdf_search"));
10           out = response.getWriter();
11           db = new DatabaseConnection();
12           search_results = db.GetSearchResults(serachtext, web_search,
13           office_search, pdf_search);
14           out.append("<?xml version=\"1.0\" encoding=\"utf-8\"?>");
15           out.append("<message_desc>");
16           counter = 0;
17           for (Result result : search_results ) {
18                out.append("<result>");
19                out.append("<title>");/*Title*/
20                out.append(Tools.EscapeText(result.GetTitle()));
21                out.append("</title>");
22                out.append("<url>");/*Link*/
23                out.append(Tools.EscapeText(result.GetUrl()));
24                out.append("</url>");
25                out.append("<mimetype>");/*Mimetype*/
26                out.append(result.GetMimeType());
27                out.append("</mimetype>");
28                out.append("<content>");/*Content*/
29                out.append(Tools.EscapeText(result.GetContent()));
30                out.append("</content>");
31                out.append("<score>");/*Score*/
32                out.append(result.GetScore()+"");
33                out.append("</score>");
34                out.append("</result>");
35                counter ++;
36             }
37             out.append("</message_desc>");
38          break;
```

Listing 24: Serverseitige Implementierung SearchText

Im Folgenden soll nun noch die Methode „GetSearchResults()“ in Listing 24 in Zeile 12 erläutert werden.

Wie in Listing 25 zu sehen ist, muss zu Beginn untersucht werden, ob ein Filter für die Dateitypen gesetzt wurde. Abhängig davon muss der MySQL-Query angepasst werden. Dies ist in den Zeilen 2 bis 29 zu sehen. Wenn ein Anwender angegeben hat, nur nach Webseiten suchen zu wollen, werden diesem auch nur Webseiten angezeigt bei denen der Mime Type „text/html“ hinterlegt ist. Sollte er anderenfalls nach PDF oder Office Dokumenten gesucht haben, müssen diese dem Filter entsprechend angefügt werden.

```
1   String mimetype;
2   if (!web_search && !office_search && !pdf_search) {
3       mimetype = "`mimetype` LIKE 'text%'";
4   } else {
5       boolean first = true;
6       mimetype = "";
7       if(web_search) {
8           mimetype += " `mimetype` LIKE 'text/html%'";
9           first=false;
10      }
11      if(office_search) {
12          if (first) {
13              mimetype += " `mimetype` LIKE 'application/msword%' OR `mimetype` LIKE
14              'application/msexcel%' OR `mimetype` LIKE 'application/mspowerpoint%' OR
15              `mimetype` LIKE 'application/vnd.openxmlformats-officedocument%' ";
16              first = false;
17          } else {
18              mimetype += " OR `mimetype` LIKE 'application/msword%' OR `mimetype`
19              LIKE 'application/msexcel%' OR `mimetype` LIKE 'application/mspowerpoint%'
20              OR `mimetype` LIKE 'application/vnd.openxmlformats-officedocument%' ";
21          }
22      }
23      If (pdf_search) {
24          If (first) {
25              mimetype += " `mimetype` LIKE 'application/pdf%'";
26          } else {
27              mimetype += " OR `mimetype` LIKE 'application/pdf%'";
28          }
29      }
```

Listing 25: Bestimmung des MimeTypes für die Suche

Nachdem der Filter für die Mime-Types korrekt gesetzt wurde, müssen nun noch die eingegebenen Suchbegriffe bestimmt werden. Wie in Listing 26 zu sehen ist, wird dazu der übergebene String aufgesplittet und jedes Suchwort in eine String-Liste eingeordnet. Dies wird gemacht, da in zukünftigen Versionen geplant ist, verschiede Worttrennzeichen zu erlauben. So soll es möglich sein alternativ zu einem Leerzeichen ein ‚+‘ oder ein ‚-‘ verwenden zu können. Dabei soll ein Plus dazu führen, dass verstärkt nach diesem Wort gesucht wird und ein Minus, dass dieses Wort nicht enthalten sein darf. Auch werden so doppelte Worttrennungen durch doppelte Leerzeichen abgefangen und die Suche ein wenig beschleunigt.

```
List<String> search_words = new ArrayList<>();
String buffer = "";
for(int i = 0; i < input.length(); i++) {
    if (input.charAt(i)!=' ') {
        buffer += input.charAt(i) ;
    } else {
        if (!buffer.equals("")) {
            search_words.add(buffer);
            buffer = "";
        }
    }
}
if(!buffer.equals("")) {
    search_words.add(buffer);
}
String search_words_string = "";
for (String search_word : search_words) {
    if(search_word.charAt(0)!='+' && search_word.charAt(0)!='-' &&
    search_word.charAt(0)!='~') {
        search_word = "+"+search_word;
    }
    search_words_string +=" "+search_word;
}
```

Listing 26: Analyse der Suchbegriffe

Für die Suche in der MySQL-Datenbank wird eine MySQL-Full-Text Search verwendet. Damit dies schnell möglich ist, war es nötig, einen Index über die Tabellenspalte „content“ zu definieren. Da es jedoch nicht möglich ist, einen Index über einen Standard-Text-Spalte anzulegen war es nötig dieser den Typen „FULLTEXT“ zuzuweisen, da sonst Suchanfragen sehr lange dauern würden.

Die MySQL-Full-Text Search wurde dazu entwickelt, in großen Datenmengen Texte schnell nach einzelnen Wörtern oder auch Wortgruppen durchsuchen zu können. Es existieren die folgenden drei verschiedene Methoden der Full-Text Search:

1. Natural Language Mode,
2. Boolean Full-Text Search,
3. Query Expansion. (MYSQL1)

Der Natural Language Mode bietet die Möglichkeit in freien Textphrasen zu suchen, wie es ein Mensch tun würde, das heißt es wird im Wesentlichen nur geschaut ob der Suchbegriff in der Textphrase vorkommt. Dagegen ist die Boolean-Suche ein wenig komplexer. Bei dieser ist es möglich, Operatoren einzufügen, die definieren ob beispielsweise ein Suchbegriff vorkommen muss, kann oder gar nicht vorkommen darf. Die Suche im Query Expansion Modus basiert auf dem Natural Language Mode, nur das hier die Suche zweimal hintereinander ausgeführt wird. Das heißt es wird zunächst wie im Natural Language Mode gesucht, dessen Ergebnisse mit den besten Ergebnissen

zwischengespeichert und auf diese wird erneut der Natural Language Mode angewandt, um genauere Ergebnisse zu erhalten. (MYSQL1)

Für den aktuellen Stand des Prototypen, wurde entschieden zunächst nur den Natural Language Mode zu verwenden, da dieser den aktuellen Funktionsumfang abdecken kann. Im Hinblick auf zukünftige Versionen, sollte in Betracht gezogen werden zu der Boolean-Search zu wechseln, da diese in der Lage ist Worte unterschiedlich zu priorisieren. (MYSQL1)

Die Full-Text Search ist standardmäßig in der Lage einen String innerhalb einer Text Collection zu finden. Diese kann dabei auch aus Einträgen über mehrere Tabellenspalten stammen. Diese Texte müssen nur in der „MATCH()"-Methode eingetragen werden. Das gesuchte Wort beziehungsweise die gesuchten Worte können im Anschluss darauf innerhalb eines String in der Methode „AGAINST()" übergeben und gesucht werden. Die Methode „MATCH()" gibt einen Score zurück, der sich nach Angaben von MySQL aus eine Ähnlichkeitsmessung zwischen dem gesuchten Begriffen und den gegeben Text(en) berechnet. Standardmäßig ist die Suche nicht Case-Sensitive, das heißt es ist unabhängig ob ein Wort groß oder klein geschrieben wurden ist. Die Full-Text Search ist auch in der Lage, eigenständig Trennzeichen und Bedeutung von Worten durch Trennung von beispielsweise Bindestrichen oder Apostrophen zu erkennen und diese dem Score hinzuzufügen. Es ist zu beachten, dass einige Worte bei der Full-Text Search vernachlässigt werden. So werden alle Worte vernachlässigt, die zu kurz sind. Diese länge kann in der Datenbank Konfiguration mit dem Parameter **ft_min_word_len** angepasst werden. Standardmäßig steht dieser Wert, abhängig des Datenbanksystems auf drei oder vier. Außerdem werden alle Worte aus einer „Stopword"-Liste ignoriert. Bei diesen Worten handelt es sich um Worte, die meist für eine Suche nicht relevant sind, wie beispielsweise Prädikate. Diese „Stopword"-Liste kann mit den entsprechenden Parametern:

- **innodb_ft_enable_stopword**,
- **innodb_ft_server_stopword_table,** oder
- **innodb_ft_user_stopword_table**

in der Konfiguration deaktiviert werden. Das Gewicht von jedem einzelne Wort wird demnach bestimmt, wie oft es in der Text-Collection vorkommt. So hat ein Wort, welches in vielen Dokumenten vorkommt, ein geringeres Gewicht als ein anderes was nicht so häufig vorkommt. Somit hat auch ein seltenes Wort ein dementsprechendes höheres

Gewicht. Die Kombination aus allen Gewichten aller Worte des Suchtextes ergibt dann die Wichtigkeit/Score der jeweiligen Tabellenreihe. (MYSQL2)

In Listing 27 ist nun die Implementierung der Full-Text Search zu sehen. In Zeile 1 ist zu sehen, wie die zuvor bestimmten Variablen in den SQL-String eingesetzt werden. Wie zu sehen ist, wird die Full-Text Search innerhalb des SQL-Statements zwei Mal verwendet. Dies ist zum einen nötig, weil die Suchergebnisse geordnet nach dem Score der Full-Text Search zurückgegeben werden sollen und es selbstverständlich nötig ist, in der WHERE-Klausel des Statements nach den Begriffen zu filtern. In den nachfolgenden Zeilen ist zu sehen, wie die Daten aus dem Result-Set extrahiert werden und einer Liste hinzugefügt werden. Diese Liste wird dann im Anschluss von der Methode zurückgegeben. Wie zu sehen ist, wird in dem SQL-Statement für die Full-Text Search der „Natural Language Mode“ auf die Tabellenspalte „content“ angewandt.

```
1   String sql_string = "SELECT `link`, `thumb_content`, `title`, `mimetype`,
    MATCH(`content`) AGAINST('"+search_words_string+"' IN NATURAL LANGUAGE
    MODE)  AS score FROM `links` WHERE  "+mimetype + " AND MATCH(`content`)
    AGAINST('"+search_words_string+"' IN NATURAL LANGUAGE MODE) By `score`
    DESC LIMIT 100";
2   List<Result> search_result = new ArrayList<>();
3   Connection con = InitDataBaseConnection();
4   PreparedStatement stmt;
5   stmt = con.prepareStatement(sql_string);
6   ResultSet rs = stmt.executeQuery();
7   while(rs.next()) {
12      search_result.add(new
13      Result(rs.getString("title"),rs.getString("link"),rs.getString("mimetype"),
14      rs.getString("thumb_content"),rs.getFloat("score")));
15  }
16  CloseDataBaseConnection(con);
```

Listing 27: SQL-Suche

Diese zurückgegebenen Ergebnisse werden nun mithilfe der zuvor dargestellten Struktur in eine XML-Struktur verpackt und mithilfe von Ajax dem Browser des Clients übermittelt, der diese dann dem Nutzer im Browser anzeigen kann.

Aufgrund des gewählten SQL-Statements mit der Full-Text Search im „Natural Language Mode“, wurde am Ende der Suche nur mittelmäßige Ergebnisse angezeigt. Mittelmäßig heißt, dass ein Nutzer Beispielsweise nach *Wikipedia* und *Katze* gesucht hat, nur sehr wenige Ergebnisse gefunden hat und nicht das Ergebnis, dass ein Anwender erwartet hätte. Ein Ergebnis, welches ein Anwender erwartet hätte, wäre zum Beispiel ein Wikipedia Artikel über Katzen gewesen. Grund hierfür war, dass nur der Content der Webseite zur Suche eingeschlossen wurde. Dies hatte jedoch zur Folge, dass das einzige Ergebnis, was man zu Wikipedia gefunden hatte, das Impressum von Wikipedia selbst war, da nur dort Wikipedia niedergeschrieben stand. Aus diesem Grund wurde nach einigen Testdurchläufen das oben genannte Statement angepasst. Die Grundidee

der Optimierung sollte sein, dass der Titel der Seite und der Inhalt des Links der Suche hinzugefügt wird. Hierfür war es nötig, einen Full-Text Index über alle drei Tabellenspalten zu definieren. Um dies zu erreichen musste das Statement:

```
ALTER TABLE links ADD FULLTEXT fulltext_keyword_rows (link, content, title);,
```

ausgeführt werden. Im Anschluss darauf war es möglich, diese im SQL Statement zu verwenden. Auch wurde zur Optimierung der Suchmodus angepasst. Damit man ungewollte Suchergebnisse direkt bei der Suche, wie bei der Google Suche, ausschließen kann, wurde nach einer Möglichkeit gesucht dies zu lösen. Die Boolean-Search ist in der Lage, Wörter unterschiedlich zu priorisieren oder gar auszuschließen. Hierfür muss für Wörter, die gesucht werden sollen, ein + geschrieben werden, vor Wörter, die eine niedrigere Priorität haben, eine ~ und ein -, für Wörter die komplett ausgeschlossen werden sollen. Um die Anwendbarkeit zu vereinfachen wird serverseitig nun automatisch vor jedes Wort, welches kein Vorzeichen enthält, ein + geschrieben, da davon ausgegangen werden kann, dass nach diesem Begriff gesucht werden soll. In Listing 28 ist nun das neue SQL-Statement zu sehen.

```
1   String sql_string = "SELECT `link`, `thumb_content`, `title`, `mimetype`, " +
    "MATCH(`content`, `title`, `link`) AGAINST('"+search_words_string+"' IN BOOLEAN
    MODE) AS score FROM `links` WHERE "+mimetype+" AND MATCH(`content`, `title`,
    `link`) AGAINST('"+search_words_string+"' IN BOOLEAN MODE) ORDER By `score`
    DESC LIMIT 1000";
```

Listing 28: Neues SQL Statement

Der durch diese kleine Anpassung erreichte verbesserte Sucherfolg ist bemerkenswert. Nun erhält ein Anwender ein Ergebnis was dessen Erwartungen erfüllen kann. Hierfür ist es nur erforderlich das ein Entsprechender Datensatz auch indexiert in der Datenbank vorhanden ist.

6 Schlusswort

Innerhalb dieser Arbeit wurden unter anderem die verschiedenen Herangehensweisen des Information Retrieval mit neuronalen Netzen und mit klassischen Textanalyseverfahren betrachtet. Wie sich bei der Entwicklung des Prototypens gezeigt hat, ist es wichtig für eine gute Web Suchmaschine, nicht nur einen der beiden Ansätze zu verfolgen, sondern eine Kombination aus beiden zu wählen, bei der die Stärken der jeweiligen Methode angewandt werden. Es hat sich gezeigt, dass für die Suche nach Textphrasen auf Webseiten Standardtextanalyseverfahren weitestgehend ausreichend sind. An dieser Stelle sind neuronale Netze nur hilfreich, wenn nutzerspezifische Nutzungsdaten vorliegen würden und anhand denen die Priorität der Suchergebnisse modifiziert werden könnte. Dagegen hat sich bei der Bildersuche gezeigt, dass hier neuronale Netze nicht mehr als Ergänzung zur Phrasensuche wegzudenken sind. Grund hierfür ist, dass bei der Bildersuche häufig nach dem Inhalt, der auf dem Bild dargestellt ist, gesucht wird und dieser nur mithilfe eines neuronalen Netzes identifiziert werden kann. Für den Prototypen war es jedoch auch hilfreich den Titel des Bildes für die Suche mit zur Verfügung zu stellen, da das vorhandene neuronale Netz nicht ausreichend genug trainiert war und auch so nach besonderen Begriffen, wie Eigennamen, gesucht werden konnte. Zukunftsweisend gedacht, wäre die Suche nach Musik oder Videos auch mithilfe von neuronalen Netzen umsetzbar.

Der in der Arbeit gezeigte Prototyp kann an einigen Stellen noch optimiert werden. So wäre es für eine größere Skalierung zwingend erforderlich, an vielen Stellen Clustering vorzunehmen und Module voneinander strikter zu kapseln. Auch wäre es ratsam, neue Links zunächst in einer temporären Datenbank zu speichern die dann zunächst optimiert wird und im Anschluss darauf durch die Datenbank, auf der die Suchanfragen angewandt werden,

ersetzt wird.

Wie sich in der Arbeit gezeigt hat, sind die aktuellen Suchmaschinengiganten nicht ohne Grund so erfolgreich. Über viele Jahre haben diese ihre Suchmethoden optimiert und ihre Namen haben sich in die Köpfe der Menschen förmlich „eingebrannt“. So gehört für viele Menschen das Wort „googlen“ mittlerweile zum täglichen Wortschatz dazu. (PH1) Eine neue Suchmaschine, die sich in der heutigen Zeit behaupten wolle, müsste dafür auf andere Strategien setzen. Aktuell herrscht unter vielen Internet Nutzern die Angst hervor, dass ihre Daten und Suchverläufe von den amerikanischen Großkonzernen missbraucht werden könnten.(Zeit1) Aus der Sicht des Autors müsste eine Suchmaschine, die sich vorrangig auf dem Europäischen Markt durchsetzen wollte, vollkommen transparent arbeiten. Dies bedeutet, dass sie zu einhundert Prozent Open-

Source sein sollte und der Quell-Code so jederzeit für jeden frei zur Verfügung stehen müsste. Auch dürfte sie keine Werbung enthalten und müsste sich nur durch freiwillige Spenden, ähnlich wie Wikipedia, finanzieren (Wiki9). Zudem wäre es notwendig, dass alle Server in Europa gehostet werden müssten, um die Vertrauenswürdigkeit von Staatsorganisationen und Ämtern zu gewinnen.

Insgesamt betrachtet ist es jedoch unwahrscheinlich, dass es eine neue Suchmaschine schafft einen der aktuellen Marktführer von ihrem Thron zu stoßen.

Es ist anzunehmen das Google in Zukunft ihre Suchstrategien weiter optimieren werden in Zuhilfenahme von neuronalen Netzen. Jedoch wird deren größtes Problem sein, dass das Internet immer weiterwachsen wird und sie ihre Serverfarmen demnach skalieren müssen.

Abkürzungsverzeichnis

CPU	Central Processing Unit (Prozessor)
DB	Datenbank
DMZ	Demilitarisierte Zone
GPU	Graphics Processing Unit (Grafikprozessor)
HTML	Hypertext Markup Language
Inc.	Incorporated
IR	Information Retrieval
JVM	Java Virtual Machine
TPU	Tensorflow Procesor Unit
XML	Extensible Markup Language (Erweiterbare Auszeichnungssprache)

Glossar

Clustering(Server)	Als Computercluster beschreibt man einen Verbund von Rechnern, die zusammengeschalten sind, um gemeinsam eine Aufgabe zu lösen.
Crawler	Computerprogramm, welches automatisch das World Wide Web durchsucht
CSS	Cascading Style Sheets, beschreibt eine Desing-Sprache für das World Wide Web
HTTP	Hypertext Transfer Protocol
Hyperlink	Querverweis in einem Hypertext
Hypertext	ein Text mit einer netzförmigen, dynamischen Struktur, Gestaltungsanweisungen sind in diesem möglich
MemoryLeak(s)	Ein Fehler in der Speicherverwaltung eines Computers, der dazu führt, dass es einen Teil des Arbeitsspeichers zwar belegt, diesen jedoch weder freigibt noch nutzt. (Wiki12)
Meta.XML	Standard für den Datenaustausch zwischen Softwarekomponenten
Präfixe	ein sprachlicher Baustein, der vor dem Wortstamm vorangestellt wird
Sitemap	Beschreibt eine vollständige hierarchisch strukturierte Darstellung aller Dokumente eines Internetauftritts (Wiki10)
Spider	siehe Crawler
Token	Sicherheitskomponente zur Identifizierung von Benutzern im Rahmen einen Zugriffskontrolle mit Zwei-Faktor-Authentisierung.
URL	Uniform Resource Locator, Indentifizierungsadresse einer Ressource. (Wiki11)

Formelverzeichnis

Listings

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

Apache1	Apache PDFBox https://pdfbox.apache.org/ Online, Abruf: 07.03.2019
Apache2	Apache POI https://poi.apache.org/ Online, Abruf: 07.03.2019
Bing1	Webmaster Help: meet our crawlers https://www.bing.com/webmaster/help/which-crawlers-does-bing-use-8c184ec0 Online, Abruf: 07.03.2019
EK1	Elektronik Kompendium: RAID-Level 0 https://www.elektronik-kompendium.de/sites/com/1312051.htm Online, Abruf: 07.03.2019
FHKOELN1	FH-Köln: Methoden und Verfahren des Information Retrieval https://ixtrieve.fh-koeln.de/lehre/s-020-methoden-und-verfahren-des-Information Retrieval-05.pdf Online, Abruf: 07.03.2019
FreeForm1	FreeFormatter.com: MimeTypes https://www.freeformatter.com/mime-types-list.html Online, Abruf: 07.03.2019
Google1	Google Support: Der Googlebot https://support.google.com/webmasters/answer/182072?hl=de Online, Abruf: 07.03.2019
Google2	Google Support: Google-Crawler (User-Agents) https://support.google.com/webmasters/answer/1061943 Online, Abruf: 07.03.2019
Google3	Google Support: Index https://support.google.com/webmasters/answer/7643011 Online, Abruf: 07.03.2019
Google4	Google Support: Crawling-Frequenz https://support.google.com/webmasters/answer/48620 Online, Abruf: 07.03.2019
Google5	Google Support: robots.txt https://support.google.com/webmasters/answer/6062608 Online, Abruf: 07.03.2019
Google6	Google Support: APIs-Google-User-Agent https://support.google.com/webmasters/answer/7426684 Online, Abruf: 07.03.2019
Google7	Google Support: AdSense – Crawler https://support.google.com/adsense/answer/99376 Online, Abruf: 07.03.2019
Google8	Google Support: Duplizierter Content https://support.google.com/webmasters/answer/66359?hl=de Online, Abruf: 07.03.2019
Google9	Google Support: Richtlinien für Webmaster https://support.google.com/webmasters/answer/35769?hl=de Online, Abruf: 07.03.2019
Hebb1	The Organization of Behaivor (S. 62, Übersetzung nach Kandel et al., 1995, S. 700)

Heise1	Heise online: Yahoo macht Altavista dicht https://www.heise.de/newsticker/meldung/Yahoo-macht-Altavista-dicht-1908789.html Online, Abruf: 07.03.2019
HTMLSemi1	HTML-seminare.de: Textbereiche fett hervorheben mit <b> und https:///www.html-seminar.de/html-texte-formatieren.htm Online, Abruf: 07.03.2019
HTMLSemi2	HTML-seminare.de: Kursiv mit HTML-Befehl <i> oder https://www.html-seminar.de/text-kursiv-html-tag-i-em.htm Online, Abruf: 07.03.2019
HTMLSemi3	HTML-seminare.de: Links https://www.html-seminar.de/html-links.htm Online, Abruf: 07.03.2019
IANA1	IANA: APPLICATION MimeTypes https://www.iana.org/assignments/media-types/application.csv Online, Abruf: 07.03.2019
IANA2	IANA: MIME-Types https://www.iana.org/assignments/media-types/media-types.xhtml Online, Abruf: 07.03.2019
Intenso1	Intenso: SSD Performance https://intenso.dev.die-etagen.de/produkte/solid-state-drives/2%2C5%22%20SSD%20Top%20Performance Online, Abruf: 07.03.2019
IRWiki1	Information Retrieval Wiki: Crawler https://ir.dcs.gla.ac.uk/wiki/Crawler Online, Abruf: 07.03.2019
LP1	Suchmaschinenmarktanteile weltweit 2017 https://www.luna-park.de/blog/9907-suchmaschinen-marktanteile-weltweit-2014/ Online, Abruf: 07.03.2019
MySQL1	MySQL Download https://dev.mySQL.com/downloads/connector/j/5.1.html Online, Abruf: 07.03.2019
MYSQL1	MySQL FulltextSearch https://dev.mySQL.com/doc/refman/8.0/en/fulltext-search.html Online, Abruf: 07.03.2019
MYSQL2	MySQL FulltextSearch Natural language https://dev.mySQL.com/doc/refman/8.0/en/fulltext-natural-language.html Online, Abruf: 07.03.2019
Na1	Learning representations by back-propagating errors https://www.nature.com/articles/323533a0 Online, Abruf: 07.03.2019
NetPlanet1	NetPlanet Archie https://www.netplanet.org/dienste/archie.shtml Online, Abruf: 07.03.2019
NLSEO1	NextLevelSeo: Was sin Meta Tags https://nextlevelseo.de/seo-meta-tags-4371/ Online, Abruf: 07.03.2019
NN1	Neuronales Netz: Einleitung https://www.neuronalesnetz.de/einleitung.html Online, Abruf: 07.03.2019
NN10	Neuronales Netz: Backpropagation und folgende https://www.neuronalesnetz.de/backpropagation1.html Online, Abruf: 07.03.2019
NN11	Neuronales Netz: Competitive Learning https://www.neuronalesnetz.de/competitive.html Online, Abruf: 07.03.2019
NN12	Neuronales Netz: Rekurrente Netz https://www.neuronalesnetz.de/rekurrente1.html Online, Abruf: 07.03.2019

NN13	Neuronales Netz: Pattern Associator https://www.neuronalesnetz.de/pattern.html Online, Abruf: 07.03.2019
NN14	Neuronales Netz: Netztypen https://www.neuronalesnetz.de/netztypen.html Online, Abruf: 07.03.2019
NN15	Neuronales Netz: Kompetitive Netze https://www.neuronalesnetz.de/kompetitive.html Online, Abruf: 07.03.2019
NN16	Neuronales Netz: Kohonennetze https://www.neuronalesnetz.de/kohonennetze.html Online, Abruf: 07.03.2019
NN17	Neuronales Netz: Probleme https://www.neuronalesnetz.de/probleme.html Online, Abruf: 07.03.2019
NN18	Zusammenfassung Lernregeln http://www.neuronalesnetz.de/zfs_lernregeln.html Online, Abruf: 07.03.2019
NN19	Zusammenfassung Netztypen http://www.neuronalesnetz.de/zfs_netztypen.html Online, Abruf: 07.03.2019
NN2	Neuronales Netz: Units http://www.neuronalesnetz.de/units.html Online, Abruf: 07.03.2019
NN3	Neuronales Netz: Verbindungen zwischen Units http://www.neuronalesnetz.de/verbindungen.html Online, Abruf: 07.03.2019
NN4	Neuronales Netz: Input und Netzinput http://www.neuronalesnetz.de/input.html Online, Abruf: 07.03.2019
NN5	Neuronales Netz: Aktivitätsfunktion, Aktivitätslevel und Output http://www.neuronalesnetz.de/aktivitaet.html Online, Abruf: 07.03.2019
NN6	Neuronales Netz Trainings und Testphase http://www.neuronalesnetz.de/training.html Online, Abruf: 07.03.2019
NN7	Neuronales Netz: Lernregeln überblick http://www.neuronalesnetz.de/lernregeln.html Online, Abruf: 07.03.2019
NN8	Neuronales Netz: Hebbregel http://www.neuronalesnetz.de/hebb.html Online, Abruf: 07.03.2019
NN9	Neuronales Netz: Delta-Regel http://www.neuronalesnetz.de/delta.html Online, Abruf: 07.03.2019
OCR1	OCR-Glossar http://www.ocr-systeme.de/index/glossary/ Online, Abruf: 07.03.2019
ORF1	http://orf.at/v2/stories/2189765 Online, Abruf: 07.03.2019
PH1	http://www.phase-6.de/magazin/fakten_der_sprache/der-deutsche-wortschatz/ Online, Abruf: 07.03.2019
Quaero1	http://www.quaero.org/ Online, Abruf: 07.03.2019
ResearchGate	ResearchGate: Skalierbarkeit von Ontology-Matching-Verfahren http://www.researchgate.net/publication/265425141_Skalierbarkeit_von_Ontology-Matching-Verfahren Online, Abruf: 07.03.2019
RyteWiki1	Ryte Wiki: Yahoo! Slurp

	http://de.ryte.com/wiki/Yahoo!_Slurp Online, Abruf: 07.03.2019
RyteWiki2	Ryte Wiki: Bingbot http://de.ryte.com/wiki/Bingbot Online, Abruf: 07.03.2019
RyteWiki3	Ryte Wiki: Duplicate Content http://de.ryte.com/wiki/Duplicate_Content Online, Abruf: 07.03.2019
RyteWiki4	Ryte Wiki: OnPage Optimierung https://de.ryte.com/wiki/OnPage_Optimierung Online, Abruf: 07.03.2019
RyteWiki5	RyteWiki: OffPage Optimierung http://de.ryte.com/wiki/OffPage_Optimierung Online, Abruf: 07.03.2019
RyteWiki6	RyteWiki: Ranking https://de.ryte.com/wiki/Ranking Online, Abruf: 07.03.2019
RyteWiki7	RyteWiki https://de.ryte.com/wiki/Hyperlink Online, Abruf: 07.03.2019
Seagate1	Seagate: Ironwolf 14TB Datasheet https://www.seagate.com/www-content/datasheets/pdfs/ironwolf-14tb-DS1904-10-1807DE-de_DE.pdf Online, Abruf: 07.03.2019
SearchSec1	DMZ (Demilitarisierte Zone) https://www.searchsecurity.de/definition/DMZ-Demilitarisierte-Zone Online, Abruf: 07.03.2019
SEL1	Interview von Danny Sullivan mit Vertretern der Suchmaschinen Google und Bing https://searchengineland.com/what-social-signals-do-google-bing-really-count-55389 Online, Abruf: 07.03.2019
SEOChat1	SeoChart: The Yahoo! Slurp Crawler https://www.seochat.com/c/a/search-engine-spiders-help/the-yahoo-slurp-crawler/ Online, Abruf: 07.03.2019
SZ1	Stuttgarter Zeitung: Und am Anfang war nicht Google https://www.stuttgarter-zeitung.de/inhalt.entwicklung-der-suchmaschinen-und-am-anfang-war-nicht-google.1176b35d-592a-4e24-9a72-0c20fb7e9ec7.html Online, Abruf: 07.03.2019
Tens	Tensorflow https://www.tensorflow.org/ Online, Abruf: 07.03.2019
Tess4J1	Tess4J Tutorial https://tess4j.sourceforge.net/tutorial/ Online, Abruf: 07.03.2019
Tomcat1	Tomcat9 Documentation https://tomcat.apache.org/tomcat-9.0-doc/ Online, Abruf: 07.03.2019
UNIH1	Universität Hildesheim: Das COSIMIR-Modell für Information Retrieval mit neuronalen Netzen https://pdfs.semanticscholar.org/6131/a76a00321e7c7d2517360334ce5637d6621b.pdf Online, Abruf: 07.03.2019
W3.ORG1	The Multipart Content-Type https://www.w3.org/Protocols/rfc1341/7_2_Multipart.html Online, Abruf: 07.03.2019
W3Schools1	W3schools.com: HTML <h1> to <h6> Tags https://www.w3schools.com/tags/tag_hn.asp

	Online, Abruf: 07.03.2019
WebRef1	Webreference: Relative Urls https://webreference.com/html/tutorial2/3.html Online, Abruf: 07.03.2019
Wik1	Wiktionary: googeln https://de.wiktionary.org/wiki/googeln Online, Abruf: 07.03.2019
Wiki1	Wikipedia: Information Retrieval http://de.wikipedia.org/wiki/Information_Retrieval Online, Abruf: 07.03.2019
Wiki10	Wikipedia: Sitemap https://de.wikipedia.org/wiki/Sitemap Online, Abruf: 07.03.2019
Wiki11	Wikipedia: URL https://de.wikipedia.org/wiki/Uniform_Resource_Locator Online, Abruf: 07.03.2019
Wiki12	Wikipedia: Speicherleck https://de.wikipedia.org/wiki/Speicherleck Online, Abruf: 07.03.2019
Wiki2	Wikipedia: User Agent https://de.wikipedia.org/wiki/User_Agent Online, Abruf: 07.03.2019
Wiki3	Wikipedia: Neuronales Netz https://de.wikipedia.org/wiki/Neuronales_Netz Online, Abruf: 07.03.2019
Wiki4	Wikipedia: Aktionspotential von Neuronen https://de.wikipedia.org/wiki/Aktionspotential Online, Abruf: 07.03.2019
Wiki5	Wikipedia: Text Retrieval Conference HTTPs://de.wikipedia.org/wiki/Text_Retrieval_Conference Online, Abruf: 07.03.2019
Wiki6	Wikipedia: Texterkennung https://de.wikipedia.org/wiki/Texterkennung Online, Abruf: 07.03.2019
Wiki7	Wikipedia: Sequentzieller Zugriff https://de.wikipedia.org/wiki/Sequentieller_Zugriff Online, Abruf: 07.03.2019
Wiki8	Wikipedia: Ajax (Programmierung) https://de.wikipedia.org/wiki/Ajax_(Programmierung) Online, Abruf: 07.03.2019
Wiki9	Wikipedia Spenden https://de.wikipedia.org/wiki/Wikipedia:Spenden Online, Abruf: 07.03.2019
Zeit1	Zeit: Datenschützer wollen gegen Google vorgehen https://www.zeit.de/news/2018-11/27/datenschuetzer-wollen-gegen-google-vorgehen-181127-99-977357 Online, Abruf: 07.03.2019

Bildquellen

Abbildung 5: Precison und Recall Messung in Abhängigkeit von τ	https://www.researchgate.net/figure/Abbildung-419-Recall-Precision-und-F-Measure-in-Abhaengigkeit-von-t-am-Beispiel-von_fig10_265425141
Abbildung 13: Schematische Darstellung des Patter Associator (nach NN13	http://www.neuronalesnetz.de/
Abbildung 12: Gewichtsmatrix eines neuronalen Netzes (NN7)	http://www.neuronalesnetz.de/nnbilder/large/matrix_large.gif

Anhang A

Anlagen

CD-ROM mit folgendem Inhalt:

- Bachelorarbeit im PDF-Format
- Quellcode des Crawler-Server, Crawler-Clients und der Suchmaschine

Anhang B

Selbstständigkeitserklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegeben Literatur und Hilfsmitteln angefertigt habe. Insbesondere versichere ich, dass ich alle wörtlichen und sinngemäßen Übernahmen aus anderen Werken als solche kenntlich gemacht habe.

Ort, Datum eigenhändige Unterschrift