JOHANNES KEPLER
UNIVERSITÄT LINZ
Netzwerk für Forschung, Lehre und Praxis

Der PageRank-Algorithmus

MAGISTERARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

in der Studienrichtung

INFORMATIK

Eingereicht von:
Peter J. Zehetner Bakk.techn., 0355200

Angefertigt am:
Institut für Informationsverarbeitung und Mikroprozessortechnik

Betreuung:
o.Univ.-Prof. Dr. Jörg Mühlbacher
Mag. Dipl-Ing. Dr. Michael Sonntag

Linz, Mai 2007

Zusammenfassung:

Der PageRank-Algorithmus ist die Basis für die Bewertung von Webseiten durch moderne Suchmaschinen. Mit ihm ist es möglich, eine effiziente und rasche Suche im World Wide Web durchzuführen. Diese Arbeit beschreibt dabei den Algorithmus in seiner klassischen Form, in der er erstmal 1998 veröffentlich wurde. Weiters werden seine einzelnen Komponenten näher analysiert und es wird auf deren Bedeutung und Manipulationsmöglichkeiten eingegangen. Zusätzlich dazu werden exemplarisch einige alternative Algorithmen bzw. Erweiterungen des PageRank-Verfahrens zur Reihung von Ergebnissen bei Web-Suchmaschinen aufgezeigt. Auch die geschichtliche Entwicklung der Suchmaschinen und Suchmaschinenbewertung werden skizziert. Um die Funktion dieses Algorithmus zu verdeutlichen, wird er in einem Java-Applet visualisiert. In diesem kann man die Berechnung der PageRanks auf einem Graphen aus Webseiten beobachten. Daneben ist es möglich, mit Hilfe einer Applikation die PageRank-Berechnung für reale Webseiten zu simulieren. Dabei wird für eine Gruppe von Webseiten, die man selbst angibt oder durch einen Webcrawler auffinden lässt, der PageRank berechnet.

Abstract:

The PageRank algorithm is the basis for the ranking of websites by state-of-the-art search engines. With this algorithm an efficient and fast search on the World Wide Web can be performed. This thesis describes the algorithm in its classic form in which it was published in 1998. Furthermore its components are analyzed in detail and it is gone into their impact and the possibility of manipulating them. Additionally some alternative ranking algorithms and extensions of the PageRank-method for ranking the results of a web search engine are exemplarily shown. The chronological development of search engines and search engine ranking is also described briefly. To illustrate this algorithm it is visualized in a Java Applet. In this it is possible to watch the calculation of the PageRanks on a graph of websites. Furthermore it is possible to simulate the PageRank calculation by an application. There, the PageRank is calculated for a group of websites which one can specify or which is found by a webcrawler.

Danksagung

An dieser Stelle möchte ich einigen Personen speziell danken, die mir einerseits bei meinem Studium und insbesondere bei der Anfertigung dieser Magisterarbeit eine große Hilfe waren. Allen voran danke ich natürlich meinen Eltern, die mir durch die Unterstützung und Hilfe bei meiner Ausbildung dieses Studium erst ermöglicht haben.

Weiters danke ich allen Institutsmitarbeitern, welche mich hierbei unterstützt haben. In erster Linie gebührt dies natürlich meinen Betreuern, die durch ihre fachliche Expertise eine große Unterstützung für mich waren. Ein besonderer Dank gilt auch meinem Kollegen DI Andreas Putzinger, der stets ein offenes Ohr für meine Anliegen hatte und mir zahlreiche hilfreiche Tipps gab.

Last but not least bedanke ich mich natürlich auch bei allen Freunden und Kollegen, die mir während meines Studiums eine Bereicherung und Unterstützung waren.

INHALTSVERZEICHNIS

1. Aufgabenstellung

Diese Magisterarbeit beschäftigt sich mit einer Analyse des PageRank-Algorithmus, alternativer Verfahren der Suchmaschinenbewertung, sowie der Implementierung einer Simulationen sowie einer Visualisierungen dieses Algorithmus. So wird folgend der zu erarbeitende und implementierende Inhalt dieser Arbeit näher skizziert.

1.1. Analyse des PageRank-Algorithmus

Ziel dieses ersten Teils der Magisterarbeit ist es, die theoretischen Grundlagen des PageRank-Algorithmus zu erarbeiten. Um den Algorithmus und dessen Funktion auch in den Kontext der Zeit zu setzen, ist es dabei unerlässlich, zuerst die historische Entwicklung der Suchmaschinenbewertungen näher zu beleuchten. Dies soll mit jener Zeit und jenen Technologien abschließen, welche zur Zeit der Entwicklung bzw. Publikation des PageRank-Algorithmus gebräuchlich waren.

Darauf aufbauend soll dann die Theorie zum eigentlichen Kern dieser Magisterarbeit erarbeitet werden:

Der Aufbau, die Analyse und die Funktionsweise des PageRank-Algorithmus.
Danach sollen die Vorteile und Schwächen dieses Algorithmus aufgezeigt und bewertet werden. Die Recherche nach Lösungsansätzen für diese Schwächen und alternativer aktueller Algorithmen für die Suchmaschinenbewertung, die entweder auf dem PageRank-Algorithmus aufbauen oder eine ähnlich Funktionsweise haben, sollen ebenso Bestandteil dieses Kapitels sein.

Aufbauend auf der Theorie und den dadurch gewonnen Erkenntnissen, kann der praktische Teil dieser Magisterarbeit abschließend evaluiert werden. Dieser besteht aus insgesamt zwei Projekten.

1.2. Visualisierung des PageRank-Algorithmus mittels Java Applet

Das erste Projekt hat zum Ziel, die Funktionsweise des PageRank-Algorithmus bzw. die Berechnung der PageRanks von Webseiten in einem Graphen von verlinkten Webseiten durch eine visuelle Darstellung einfach und verständlich zu erklären.
Dies soll didaktisch zweckmäßig durch eine schrittweise Simulation des Algorithmus implementiert werden. Diese Simulation soll daher in einem Java Applet stattfinden, welches auch in das Framework des IT-MATH Projekts des Instituts [FIM_1] eingebettet werden soll.

Der Algorithmus soll, wie im realen Fall, auch auf das World Wide Web angewendet werden. Da eine visuelle Darstellung dieses gesamten Netzwerks jedoch unmöglich ist, soll die PageRank-Berechnung in einem stark reduzierten und vereinfachten Graphen stattfinden.

Der Graph selbst soll bearbeitbar sein. Folgende Operationen sind dabei zu implementieren:

- Einen neuen Knoten mit einem selbst gewählten Namen einfügen.
- Mehrere Knoten uni- und bidirektional mit gerichteten Kanten verbinden können.
- Sowohl Knoten als auch Kanten löschen.
- Der gesamte Graph soll in der Größe skalierbar sein.

Neben den Operationen auf den Graphen selbst soll es auch möglich sein, diesen zu speichern und wieder zu laden.

Auf Basis des erzeugten Graphen soll im nächsten Schritt die schrittweise Simulation des PageRank-Algorithmus erfolgen. In jedem Simulationsschritt soll dabei die detaillierte Berechnung des PageRanks einer einzelnen Webseite angezeigt werden.
Wurden alle Webseiten des Knoten durchlaufen, ist eine vollständige und sortierbare Liste aller Knoten mit deren dazugehörigen PageRanks auszugeben.

1.3. Simulation des PageRank-Algorithmus unter realen Bedingungen

Bei diesem Projekt steht weniger die visuelle Darstellung der Funktionsweise des PageRank-Algorithmus im Vordergrund. Vielmehr soll der Algorithmus auf ein bestimmtes Set von Webseiten angewendet werden können. Damit sollen die PageRanks realer Webseiten berechnet werden.

Dem Benutzer dieser Simulation sollen dafür mehrere Möglichkeiten geboten werden:

- Verwendung eines Webcrawlers, um ein Set realer Webseiten zu erhalten.
- Angabe von URLs zu Webseiten. Anhand der auf diesen Webseiten gefundenen Links soll der PageRank errechnet werden.
- Auswahl lokal gespeicherter HTML-Dateien. Damit können beispielsweise die PageRanks einer Homepage mit ihren Unterseiten berechnet werden.

2. Geschichte der Suchmaschinen und Suchmaschinenbewertung

Dieses Kapitel beschäftigt sich mit der historischen Entwicklung von Suchmaschinen, der damit verbundenen Technik und, falls verfügbar, mit der von diesen Suchmaschinen verwendeten Bewertung der Webseiten. Den Beginn dieses geschichtlichen Rückblicks stellen erste Gedanken zur Möglichkeit der Sammlung und Abfrage von Informationen dar. Er endet mit der Veröffentlichung der Suchmaschine Google.

2.1. Die Anfänge

Die Anfänge der Suchmaschinen gehen bereits viel weiter zurück, als man vermuten würde. So kam diese Idee erstmals im Jahre 1945 in [BUSH_1] auf. Der Autor koordinierte damals zum Zwecke der Kriegsführung mehrere tausend Wissenschaftler. Er plädierte dafür, dass nach dem Ende des zweiten Weltkriegs die Zusammenarbeit der Forscher weiter aufrechterhalten werden soll und dabei vor allem eine Zusammenarbeit aller Disziplinen sehr wichtig sei. Dafür sei es unerlässlich, eine Datenbank des menschlichen Wissens zu schaffen. *„Diese ist aber wissenschaftlich nur dann interessant, wenn sie ständig erweitert, gespeichert und abgefragt wird“ [BUSH_1].* Er stellte dabei einen Apparat, „Memex“ genannt vor, *„mit dem es in Zukunft jedem Menschen möglich sein*

sollte, seine Bücher, Aufzeichnungen und Kommunikationen zu speichern" [BUSH_1]. Durch eine Mechanik sollte es dann auch die Möglichkeit geben, diese Daten mit außergewöhnlicher Geschwindigkeit und Flexibilität wieder abzufragen.

Obwohl in diesem Artikel nichts über die Reihung von Suchergebnissen – zu dem Zeitpunkt stellte sich diese Frage nicht – des Memex erwähnt wird, stellt diese Vision einer hypermedialen Suche doch einen bedeutenden Schritt und den theoretischen Beginn der Geschichte der Suchmaschinen dar.

Seit den sechziger Jahren des 20. Jahrhunderts arbeitete der Philosoph und Soziologe Ted Nelson im Rahmen des Projekts XANADU [NELSON_1] an dem Konzept einer dezentralen Datenbank, welche weltweit verfügbar sein sollte. Darin sollten Dokumente gespeichert und bidirektional verlinkt werden. Er legte damit den Grundstein für ein weltweites Netzwerk aus vernetzten Dokumenten und führte auch den Begriff „Hypertext" ein. Aufbauend auf seine Arbeit entwickelten sich das WWW und Auszeichnungssprachen. Damit war die theoretische Grundlage für ein Netzwerk geschaffen, welches dann in weiterer Folge auch durchsucht werden kann.

2.2. Archie

Archie kann als die erste eigentliche Suchmaschine bezeichnet werden und wurde 1990 entwickelt. Zu diesem Zeitpunkt war das WWW noch nicht existent und Dokumente im Internet waren meist auf FTP-Servern gespeichert. Die Idee hinter Archie, welches eine Abkürzung für „archives" ist, war: Die Dateien, welche auf den anonymen FTP-Servern im Internet frei verfügbar sind, sollen mittels eines Suchroboters gesammelt und auch indiziert werden. Damit sollte ein Verzeichnis geschaffen werden, welches alle weltweit verfügbaren Dateien auf den anonymen FTP-Servern im Internet enthält. Ein Benutzer konnte dann nach Begriffen suchen, die in Dateinamen oder Verzeichnissen vorkommen. Diese Suche selbst konnte auch durch reguläre Ausdrücke unterstützt werden. Dafür standen folgende Operatoren zur Verfügung [NETPLANET_1]:

- „$“: Die Zeichenkette vor dem „$“ soll am Ende eines Dateinamens vorkommen.
- „^“: Die Zeichenkette nach dem „^“ soll am Anfang eines Dateinamens vorkommen.
- „.“: Das Zeichen „.“ stellt einen Platzhalter für genau ein Zeichen dar.
- „*“: Das Zeichen „*“ stellt einen Platzhalter für beliebig viele Zeichen dar.
- „\“: Mit diesem „\“ konnten andere Operatoren auch als Teil einer Suchtextes verwendet werden.
- „[…]“: Alle Zeichen, welche zwischen „[„ und „]“ stehen, können in einem Dateinamen vorkommen.
- „[^ …]“: Alle Zeichen, außer jene zwischen „[„ und „]“, können in einem Dateinamen vorkommen.

Weiters konnte man auch nach einem exakten Dateinamen oder einem Teil eines Dateinamens, mit oder ohne Unterscheidung der Groß-, Kleinschreibung, suchen.

Vor Archie gab es kein Verzeichnis der Dateien im Internet. Man musste sich direkt zu einem FTP-Server verbinden und konnte dort im Anschluss nach Dateien suchen. Man war auch oft darauf angewiesen, dass jemand eine Datei auf einem Server gefunden hatte und dies dann in Foren oder Listen veröffentlichte.

Nun war es erstmals möglich nach bestimmten Dateien in einer großen Anzahl an Dokumenten und anderen Dateien in einem dezentralen Netzwerk von Datenservern zu suchen. Der Nachteil war natürlich, dass lediglich nach den Dateinamen gesucht werden konnte und auf den Inhalt der Dateien keine Rücksicht genommen wurde. Eine Reihung der Suchergebnisse nach einer eigenen Methode fand ebenfalls nicht statt. Die Dateien des Suchergebnisses wurden einfach in der Reihenfolge ihres Fundes ausgegeben.

2.3. Gopher

Bei Gopher handelt es sich um einen Informationsdienst, der über das Internet verfügbar ist. Dieser wurde 1991 entwickelt und sollte das System der anonymen FTP-Server ablösen. Diese waren oft in der Bedienung nicht einfach und man musste sich immer erst direkt mit dem Server verbinden und anmelden.

Das durchforsten eines FTP-Servers wurde normalerweise über Befehle in einer Konsole gemacht, wodurch die Handhabung natürlich sehr umständlich war. Mit Gopher wollte man durch eine einfache Syntax und eine übersichtliche Baumstruktur auch diese verbessern. Die Baumstruktur, auch Gopherspace genannt, konnten Dateien, Verzeichnisse oder Links enthalten. Weiters sollte man Informationsdienste einfach, rasch und kostengünstig erstellen können.

Die Daten auf einem Gopher Server konnten mittels eines Clients abgerufen werden. Dieser konnte, beispielsweise durch Symbole, dem Benutzer ein ansprechendes und übersichtlicheres Interface bieten. Weiters enthielt Gopher auch Suchmaschinen, mit denen nach bestimmten Dateien oder Verzeichnissen gesucht werden konnte. Zwei der wichtigsten Suchmaschinen in Gopher waren Veronica und Jughead.

Gopher kann damit als direkter Vorgänger des WWW bezeichnet werden. Durch das Aufkommen und den Erfolg des WWW in der Mitte der neunziger Jahre des 20. Jahrhunderts ist Gopher rasch zurückgedrängt worden und ist heutzutage kaum mehr verbreitet.

2.4. Veronica, Jughead

Diese Suchprogramme bauten auf dem Prinzip von Archie auf und waren sozusagen dessen Pendant für den Gopherspace.

Veronica bedeutet „Very Easy Rodent-Oriented Netwide Index to Computerized Archives“. Es wurden dabei die Titel der Dateien und die Verzeichnisse aller Gopher-Server mit Hilfe einer sogenannten „Spider“ indiziert [NETPLANET_3]. Diese Datenbank diente dann als Grundlage für die Suche nach Dateien im Gopherspace. Der Vorteil gegenüber Archie war, dass man mit Veronica nicht nur nach dem Vorkommen eines Wortes in einem Dateinamen, sondern auch in einem Titel suchen konnte. Diese Titel konnten aus ganzen Sätzen bestehen und boten somit eine größere „Angriffsfläche“ für die Suchanfrage. Weiters konnten einzelne Wörter einer Suchanfrage bereits mit den booleschen Operatoren „UND“ und „ODER“ verknüpft werden.

Die Dateien des Ergebnisses dieser Suchanfrage, die zu Anfangs selbst über einen Gopherclient gestellt wurde, wurden auch wieder in der gleichen Struktur und Übersicht wie ein Gopher Verzeichnis dargestellt. Die Bedienung bei der Suche nach Dokumenten im Gopherspace war daher gleich wie die Navigation in diesem [NETPLANET_3].

Jughead steht für „Jonzy's Universal Gopher Hierarchy Excavation And Display“ und hatte eigentlich dieselbe Funktionsweise wie Veronica. Der Unterschied bestand darin, dass mittels Jughead einzelne Server schneller indiziert werden konnten, jedoch Veronica einen größeren Index besaß.

2.5. WAIS

WAIS wurde 1991 veröffentlicht und steht für „Wide Area Information Server“. Es sollte damit ein Informationssystem geschaffen werden, indem jeder Server über eine oder mehrere Datenbanken verfügt. Jede Datenbank beschäftigt sich mit eigenen Themengebieten und stellt eine eigene Einheit dar. Das bedeutet, dass zwischen den Servern und Datenbanken, im Gegensatz zu Gopher, keine Vernetzung bestand. Die Liste der Server bzw. Datenbanken wurde in einem eigenen Verzeichnis festgehalten. In diesem wurde neben der Adresse des Servers bzw. der Datenbank auch jeweils eine kurze Beschreibung der Datenbank angegeben.

WAIS gliederte sich in eine Client-Server-Architektur. Der zentrale Server übernahm dabei hauptsächlich die gesamte Organisation der Indizierung und die Verarbeitung der Suchanfragen. Über einen speziellen Client konnten Suchanfragen an den Server gestellt werden. Ein Novum war dabei, dass hier erstmals eine Volltextsuche in den Dokumenten selbst implementiert wurde und man nicht mehr nur nach Dateinamen, wie bei Archie, oder Dateititeln, wie bei Veronica und Jughead, suchen konnte.

Weiters wurde jeder Datei im Suchergebnis ein spezieller numerischer „Score“ zugeordnet und mit Hilfe dieses gereiht. Dieser Score wurde berechnet, indem man die Anzahl des Vorkommens des Textes der Suchanfrage in einer Datei in Relation zu ihrer Dateigröße setzte. Dieser Wert wird dann anschließend so normalisiert, dass die höchst gereihte Seite stets den Wert 1000 hat.

Da das WWW sich immer weiter verbreitete, wurde rasch eine Verbindung damit geschaffen. Somit war es dann möglich, auf einen WAIS auch über einen Browser zuzugreifen, und man musste nicht einen eigenen Client dafür verwenden. Dies ist auch mit ein Grund, dass sich WAIS nach der Ausbreitung des WWW noch viel länger hielt als beispielsweise Gopher.

2.6. World Wide Web Wanderer

Dieses, auch nur als „Wanderer“ bekannte Programm, wurde 1993 entwickelt. Seine ursprüngliche Aufgabe war es, die Größe des noch relativ kleinen WWW zu messen. Dazu wurde einfach die Anzahl der aktiven Server im WWW gezählt. Dieses in Pearl verfasste Programm stellt somit den ersten wirklichen Webcrawler dar.

Relativ bald wurde der Wanderer nicht nur im Rahmen seiner ursprünglichen Aufgabe verwendet, sondern er speicherte auch die URLs, welche er besucht hatte. Die Datenbank, in der diese gespeichert wurden, bekam den Namen „Wandex“.

Ein Problem gab es aber: Da die damals vorhandenen Bandbreiten noch sehr klein waren, verursachte der Wanderer durch das oft hundertmalige Besuchen der Server von Webseiten eine hohe Auslastung der Bandbreite. Dies war für die Betreiber des Servers einer Webseite natürlich ein Problem, da die Bandbreite dann für „echte Besucher“ oft nur mehr sehr gering war.

Als Lösung für unter Anderem dieses Problem wurde 1994 ein Standard für Robot Exclusion in [KOSTER_1] geschaffen. Die Intention war, einem Webcrawler (auch Webspider oder Webroboter genannt) durch eine einfache Textdatei bestimmte Restriktionen vorzugeben. Diese Datei ist unter dem Namen „robots.txt“ bekannt und hat folgende Syntax [KOSTER_1]:

```
ROBOTS = ( USERAGENT+ DISALLOW+ )*
USERAGENT = "User-agent: " ( "*" | name ) newline
DISALLOW = "Disallow: " ( ϵ | "/" | path | file ) name newline
```

Code 1: Syntax einer robots.txt Datei

Die robots.txt-Datei enthält demnach eine oder mehrer USERAGENT-Zeilen, gefolgt von einer oder mehreren DISALLOW-Zeilen. In eine USERAGENT-Zeile steht zunächst immer der String „User-agent: “. Mit dieser Zeile lassen sich bestimmte Webcrawler durch einen Namen definieren. Weiters können durch die Angabe eines „*“ auch Regeln für alle Webcrawler erstellt werden.

Für die in der USERAGENT-Zeile definierten Webcrawler gelten dann die folgenden DISALLOW-Zeilen. In diesen kann entweder der Pfad eines Verzeichnisses oder eine Datei angegeben werden. Diese sollen dann von den als Useragent definierten Webcrawlern nicht besucht werden. Weiters kann auch ein „/“ angegeben werden, was bedeutet, dass keine weiteren Webseiten dieses Servers besucht werden sollen. Wird nach dem Disallow nichts angegeben, dürfen die angegebenen Webcrawler alles besuchen.

Diese Syntax soll durch folgende Beispiele verdeutlicht werden:

```
User-agent: somerobot
Disallow: /
```

Code 2: Beispiel einer robots.txt Datei

Bei diesem Beispiel darf der Webcrawler mit dem Namen „somerobot“ keine der Webseiten dieses Servers besuchen.

```
User-agent: *
Disallow: /home/privatefile.html
Disallow: /private/

User-agent: somerobot
Disallow:
```

Code 3: Beispiel einer robots.txt Datei

In dem zweiten Bespiel dürfen von keinem Webcrawler die Datei „privatefile.html“ und das Verzeichnis „/private/“ besucht werden. Der Webcrawler mit dem Namen „somerobot“ darf hingegen jede Seite besuchen.

Ein Problem bei diesem Standard ist allerdings, dass pro Server nur eine einzelne robots.txt-Datei vorhanden sein kann. Diese muss im Wurzelverzeichnis des Webservers abgelegt werden, damit sie von den Webcrawlern erkannt wird. Wenn man also für seine Webseite nicht einen eigenen Webserver zur Verfügung hat, sondern nur ein Unterverzeichnis, kann diese Datei auch nicht genutzt werden.

Da Webcrawler auch noch heute für Suchmaschinen eine bedeutende Rolle spielen, war dies ein weiterer wichtiger Schritt in Richtung der ersten Suchmaschine für das WWW.

2.7. Aliweb

Nur wenige Monate nach dem World Wide Web Wanderer wurde die Suchmaschine Aliweb („Archie Like Indexing for the Web“) 1994 auf der ersten internationalen Konferenz über das World Wide Web präsentiert. Wie der Name schon sagt, stellt es eine Art Pendant zu Archie für das WWW dar. Im Gegensatz zu seinen Vorgängern indizierte Aliweb jedoch Webseiten nicht automatisch. Die Idee war, dass Leute ihre eigene Webseite dort eintragen können. Weiters können diese Einträge mit bestimmten Schlüsselwörtern und kurzen Beschreibungen versehen werden.

Der Vorteil dieser Methode ist, dass die Bandbreite der Server nicht durch Webcrawler ausgelastet wird. Dem gegenüber stehen leider etliche Nachteile: Viele Leute wussten nicht, dass sie ihre Webseite dort eintragen können/müssen. Weiters musste dies mittels einer speziellen Datei gemacht werden, was für viele Inhaber von Webservern eine große Hürde darstellte. Sie wussten oft einfach nicht, wie dies richtig gemacht wird. Dies führte natürlich dazu, dass die Datenbank, welche durchsucht werden konnte, nur sehr klein und deshalb für viele Suchende im WWW wenig interessant war.

Weiters ist die Möglichkeit, dass jemand seiner Webseite selbst Schlüsselwörter geben darf, natürlich immer für Manipulationen anfällig. Diese frühen Arten der Suchmaschinen suchten ihre Datenbanken oftmals ausschließlich nach der Übereinstimmung von Wörtern in Suchanfragen mit Schlüsselwörtern ab. Durch das Hinzufügen beliebig vieler Schlüsselwörter zu dem Eintrag seiner Seite konnte es vorkommen, dass unter den

Suchergebnissen viele Webseiten waren, die nichts mit dem Thema der eigentlichen Suchanfrage zu tun hatten.

Dieser doch überwiegende Teil an Nachteilen führte auch dazu, dass Aliweb nur wenig genutzt wurde und sich so nie wirklich durchsetzte.

2.8. RBSE-Spider

Durch die Probleme, welche bei Aliweb auftraten, entstanden relativ rasch neue Suchmaschinen, welche ihre Datenbanken wieder mit Hilfe von Webcrawlern aufbauten.

Exemplarisch für einen solchen Webcrawler, die zwischen 1993 und 1994 entwickelt wurden, sei der RBSE-Spider angeführt. Dieser Webcrawler wurde 1994 in [EICHMANN_1] vorgestellt. Die Abkürzung RBSE steht dabei für „Repository Based Software Engineering“.

Dieser Webcrawler verwendet dabei einen Suchmechanismus, welcher ähnlich zu jenem in WAIS funktioniert. Dabei wird der Text einer gesamten Webseite indiziert. Andere in dieser Zeit entwickelte Webcrawler indizierten oft nur den Titel und die Überschriften der obersten Ebene.

Es gab mehrere Anforderungen an RBSE-Spider, die unter Anderem die Nachteile anderer Webcrawler, wie z.B.: jene beim World Wider Web Wanderer, auszugleichen sollten [EICHMANN_1]:

- Das Untersuchen der Struktur und die Indizierung sollten getrennt werden, damit bekannte Strukturen bei einem Update der Indizes erhalten bleiben.
- Die RBSE-Spider soll an jeder beliebigen Stelle des WWW neu gestartet werden können, damit bekannte Strukturen nicht nochmals besucht werden müssen.
- Die RBSE-Spider soll sich eindeutig als Webcrawler und nicht als Benutzer gegenüber den Servern identifizieren.

- Durch das Suchen nach Webseiten und das Indizieren dieser, sollte das WWW möglichst wenig beeinflusst werden. Außerdem soll die Suche auf HTML-Dokumente beschränkt sein.

Die in [EICHMANN_1] präsentierten Ergebnisse dieses Webcrawlers waren zwar wenig aussagekräftig, da sie nur auf einen kleinen Teil des WWW angewendet wurden und Tests nur in geringem Ausmaß durchgeführt wurden. Dennoch sind die Anforderungen, welche an diesen RBSE-Spider gestellt wurden, eine wichtiger Schritt für die weitere Entwicklung effizienter und ressourcensparender Webcrawler, welche für zukünftige Suchmaschinen unerlässlich waren.

2.9. Excite

Mit der Entwicklung der Suchmaschine Excite wurde bereits 1993 durch sechs Studenten unter dem Namen „Archtitext“ begonnen. Hier ist bereits anzunehmen, dass dabei nicht mehr der Forschungsgedanke überwog, sondern dies aufgrund von kommerziellen Interessen gemacht wurde. Nach der Umbenennung von Architext in Excite 1994, wurde die Suchmaschine erst 1995 offiziell als Webservice gestartet.

Zur Sammlung von Webseiten wurden einerseits Webcrawler verwendet, andererseits konnten Benutzer auch Webseiten direkt an Excite melden. Dies führte sehr rasch zu einer großen Datenbank. Für die Indizierung der Webseiten wurde ein spezielle Algorithmus verwendet: "Intelligent Concept Extraction" [NICHOLSON_1]. Um die Schlüsselwörter für die Indizierung der Webseiten zu bekommen wurde nicht nach bestimmten Tags (Marken), wie beispielsweise HTML-Tags von Überschriften oder Meta-Tags, gesucht. Es wurde vielmehr versucht, mit Hilfe statistischer Analysen der Beziehung der Wörter auf einer Webseite einzelne Themen zu erkennen. Indiziert wurden - anstelle des gesamten Texts - dann schließlich der Titel der Webseite und die wichtigsten Schlüsselwörter, welche sich aus den Themen der Webseite ergaben.

Das Ergebnis einer Suchanfrage bot zwei besondere Funktionen:
Zum einen wurde eine kurze Zusammenfassung der Webseite gegeben. Da diese jedoch von Excite automatisch erstellt wurden, war sie oft von schlechter Qualität. Erzeugt wurde sie, indem man nach den Sätzen oder Zeilen auf einer Webseite suchte, in denen die gesuchten Schlüsselwörter am öftesten vorkommen. Dies führte unter Umständen zu einer ziemlich verwirrenden oder sinnlosen Zusammenfassung, konnte jedoch auch oft aussagekräftiger sein, als beispielsweise die ersten Zeilen auf einer Webseite.

Die zweite Funktion bot dem Benutzer zu jeder Webseite im Suchergebnis an, ähnliche Seiten zu finden. Dazu wurden einfach die für die Seite in der Datenbank gespeicherten Schlüsselwörter genommen und es wurde eine neuerliche Suche mit diesen gestartet.

Die Suchergebnisse wurden dann nach der Relevanz der Schlüsselwörter einer Webseite zu den Wörtern der Suchanfrage gereiht. Angegeben wurde diese Relevanz in Prozent.
Je größer also die Übereinstimmung der Wörter der Suchanfrage mit den Schlüsselwörtern einer Webseite war, desto höher wurde eine Webseite unter den Suchergebnissen gereiht.

2.10. Yahoo!

Die Entwicklung, durch welche in späterer Folge die Suchmaschine Yahoo! entstand, begann bereits Anfang 1994. Damals handelte es sich zuerst um ein einfaches Verzeichnis mit dem Namen „Jerry's Guide to the World Wide Web“. In diesem wurde von zwei Studenten eine Vielzahl an Webseiten gesammelt, welche sie für besonders interessant befanden. Weiters erstellten sie eigene Webseiten, auf denen sich Links zu anderen populären Webseiten befanden. Dieses Verzeichnis war hierarchisch strukturiert und erfreute sich bald großer Beliebtheit. Deshalb wurde nur wenige Monate später der Name in „Yahoo!“ umgewandelt und es wurde auch die Möglichkeit eingebaut, dieses Verzeichnis durchsuchen zu können.

Über die Bedeutung des Namens selbst gibt es im WWW verschiedene Erklärungen: Eine ist, dass den beiden Entwicklern die Bedeutung des englischen Wortes „yahoo“ so gut gefiel. Die Andere ist dass es eine Abkürzung für „Yet Another Hierarchical Officious Oracle“ ist.

Die Webseiten wurden am Anfang weiterhin manuell in die Datenbank aufgenommen oder es wurden „What's new"-Seiten von Webcrawlern durchsucht [NICHOLSON_1]. Andere Leute konnten ebenfalls ihre Webseite eintragen lassen.

Alle Webseiten wurden in der Datenbank immer einem bestimmten Thema zugeordnet. Zu Beginn handelte es sich daher bei Yahoo! nicht um eine wirkliche Suchmaschine, sondern lediglich um ein Verzeichnis, welches durchsucht werden konnte.

Bei der Indizierung, welche meistens von Menschen gemacht wurde, wurden von den Webseiten die URL, der Titel und eventuell eine kurze Beschreibung gespeichert. Diese Einträge wurde auch nur dann wieder erneuert, wenn ein Benutzer auf ein Problem aufmerksam machte [NICHOLSON_1]. Dies ist natürlich für eine Suche keine gute Grundlage: Der Titel einer Homepage war oft nicht definiert, schlecht gewählt oder wenig aussagekräftig.

Wenn einer Webseite dann auch noch keine Beschreibung gegeben wurde, konnte sie praktisch nicht gefunden werden. Die Suchergebnisse wurden dabei nach ihren Themengebiete gruppiert. Die einzelnen Themenblöcke waren alphabetisch sortiert. Einen eigenen Ranking-Algorithmus innerhalb dieser Blöcke gab es nicht.

Die Stärke von Yahoo! war jedoch nicht die Möglichkeit ihre Datenbank zu durchsuchen, sondern eine Baumstruktur, welche in Themen gegliedert war. So konnten vor allem viele Neulinge im WWW sehr einfach und rasch durch dieses navigieren.
Später wurde von Yahoo! auch die Suchmaschine AltaVista genutzt, falls in dem eigenen Verzeichnis keine Suchergebnisse erzielt werden konnten.

2.11. WebCrawler

Diese Suchmaschine wurde etwa zeitgleich mit der Umbenennung von „Jerry's Guide to the World Wide Web" in „Yahoo!" veröffentlicht. Es war die erste Suchmaschine, die eine Volltextsuche unterstützte. Dies bedeutet, dass jedes Wort einer Webseite indiziert werden

musste. Davor war es den verschiedenen Webcrawlern meist nur möglich, einzelne Abschnitte einer Webseite zu sammeln bzw. diese später zu indizieren.

Ursprünglich handelte es sich um eine Desktopapplikation, welche jedoch relativ bald mit einem Webinterface ausgestattet wurde. Durch die Möglichkeit, komplette Dokumente zu durchsuchen, wurde die Suchmaschine rasch sehr beliebt und auch sehr oft frequentiert. Dies führte unter Anderem dazu, dass die Bandbreite vor allem unter Tags oft dermaßen ausgelastet war, dass kaum ein Zugriff auf das Webinterface möglich war. Dies änderte sich erst, als WebCrawler 1995 von AOL übernommen wurde und von da an über deren Netzwerk lief.

Im Gegensatz zu den vorher beschriebenen Suchmaschinen wurden von WebCrawler nicht nur Webseiten durchsucht, sondern es wurde auch nach FTP- und Gopher-Seiten gesucht. Beschreibungen oder Zusammenfassungen einer Webseite, wie beispielsweise bei Excite oder Yahoo!, fehlten jedoch gänzlich. Dadurch war es oft schwierig zu erkennen, welche Relevanz ein Link zu einer Webseite im Suchergebnis hatte. Als Ranking-Algorithmus wurde zu Beginn einfach die Anzahl der Übereinstimmung der auf einer Webseite vorkommenden Wörter mit jenen einer Suchanfrage gezählt.

2.12. Lycos

Diese Suchmaschine wurde kurz nach WebCrawler veröffentlicht. Obwohl anfänglich nur eine vergleichsweise kleine Datenbank vorhanden war, entwickelte sich diese bis 1996 zu der größten aller damals existierenden Suchmaschinen-Datenbanken und war somit der am schnellsten wachsende Suchmaschinenindex in dieser Zeit.

Der Name leitet sich von der Wolfspinne (Lycosidea lycosa), welche ihre Beute durch Jagd fängt und nicht mit Hilfe eines Netzes [KNOBLOCK_1]. Diese Analogie sollte auch für diese Suchmaschine gelten: Webseiten sollen „gejagt“ werden, d.h. von Webcrawlern aufgespürt werden.

In den Index wurde jede Webseite, dir irgendwo im WWW gefunden wurde, aufgenommen. Indiziert wurden dabei der Titel, die Überschriften und Unterüberschriften,

die ersten 20 Zeilen und jene 100 Schlüsselwörter, welche die größte Häufigkeit in dem Dokument hatten [KNOBLOCK_1]. Bei der Sammlung dieser Wörter gab es auch bereits sogenannte „Stopwords“, welche nicht in den Index aufgenommen wurden. Darunter fallen vor allem Artikel und Bindewörter. Ein Nachteil im Vergleich mit der Volltextindizierung von Webcrawler war daher, dass von Webseiten, die sehr viel Text enthielten, nur 100 Wörter in den Index aufgenommen wurden. Dies war oft nur ein Bruchteil aller auf einer Webseite vorkommenden Wörter [NICHOLSON_1].

Zwei neue Funktionen wurden ebenfalls erstmals für die Suche im WWW eingeführt:

Prefix Matching

Durch den Operator „$“ am Ende eines Wortes der Suchanfrage wurden alle Wörter gesucht, welche mit den Zeichen vor diesem „$“ anfingen. Dies schuf zwar einerseits eine gute Möglichkeit um nach Begriffen zu suchen, die verschiedenste Endungen haben können. Andererseits wurden damit oft auch viele Begriffe gesucht, welche mit dem Prefix eines vom Benutzer gesuchten Wortes wenig zu tun hatten.
Weiters konnte mit dem Operator „.“ am Ende eines Suchbegriffs dafür gesorgt werden, dass nur genau jene vor dem Punkt geschriebene Zeichenkette gefunden werden soll.

Word Proximity

Es gab die Möglichkeit, die Genauigkeit der Übereinstimmung der Wörter einer Suchanfrage mit den Wörtern einer Webseite einzustellen. Dafür waren folgende Einstellungen Möglich: „loose“, „fair“, „good“, „close“ und „strong“. Die Einstellung „loose“ bedeutete dabei, dass die Übereinstimmung auch sehr gering sein durfte, wohingegen „strong“ für die höchste Genauigkeit, d.h. eine exakte Übereinstimmung, stand.

Bei den Suchergebnissen wird auch eine kurze Beschreibung ausgegeben, die eine Liste aus den 100 wichtigsten Wörtern einer Webseite, welche mit den Suchbegriffen übereinstimmten, enthält. Dadurch ist es, im Vergleich zu der Seitenbeschreibung eines Suchergebnisses bei Yahoo!, für den Benutzer leicht nachvollziehbar, woher diese Beschreibung kommt. Weiters wurde für sie Suche nach Schlüsselwörtern die Technik der

sogenannten „Invertierten Datei Indizierung“ [KNOBLOCK_1] verwendet. Dabei wird in der Datenbank für jedes dort vorhandene Schlüsselwort eine Liste von Webseiten erstellt, welche dieses enthielten. Bei der Suche nach einem bestimmten Wort, musste dann nur diese Liste ausgegeben werden. Der Vorteil dieser Methode ist natürlich, dass sie sehr viel schneller ist, als die Datenbank bei jeder Suchanfrage nach dem Vorkommen eines Wortes zu durchsuchen (wie es z.B.: bei Aliweb gemacht wurde).

Der Algorithmus für die Reihung der Webseiten im Suchergebnis beachtete dabei insgesamt fünf Faktoren [KNOBLOCK_1]:

1. Die Anzahl der übereinstimmenden Wörter einer Suchanfrage mit den Wörtern auf einer Webseite.
2. Die Häufigkeit dieser Wörter auf einer Webseite.
3. Die syntaktische Nähe der einzelnen Wörter der Suchanfrage auf einer Webseite.
4. Die Position der Wörter auf einer Webseite (z.B.: im Titel oder in einer Überschrift).
5. Die Genauigkeit der Übereinstimmung der Wörter der Suchanfrage mit den Wörtern auf einer Webseite. So war z.B.: die Genauigkeit der Wörter „glow“ und „glows“ größer, als die Genauigkeit der Wörter „glow“ und „glowing“.

2.13. AltaVista

Als eine der ersten größeren Suchmaschinen des WWW wurde AltaVista nicht von einer Universität oder Studenten, sondern von einer bereits bestehenden Firma - Digital Equipment Corporation – entwickelt. Dieses eigentlich auf Hardware spezialisierte Unternehmen wollte mit diesem Projekt, welches 1995 veröffentlich wurde, die Leistungsfähigkeit seiner Rechner demonstrieren.

Durch diese starke Hardwareunterstützung war es auch möglich, neue Technologien für das Auffinden von Webseiten im WWW zu nutzen. So entwickelte man erstmals einen Webcrawler, genannt „Scooter“, welcher durch Multithreading-Unterstützung in kürzester Zeit eine wesentlich größere Anzahl an Webseiten finden konnte als seine Vorgänger.

Dadurch wurden rasch mehr Seiten im WWW gefunden, als man zum damaligen Zeitpunkt glaubte dass überhaupt existierten.

Indiziert wurden sowohl Webseiten als auch Newsgroups. Gopher- und FTP-Seiten gingen dabei in den Index nicht ein.

AltaVista setzte, wie auch schon sein Vorgänger WebCrawler, auf Volltextsuche. Neben allen Wörtern die auf einer Webseite zu finden waren, konnte der Erzeuger einer Homepage auch sogenannte Meta-Tags definieren. Mit diesen konnten bestimmte Schlüsselwörter für eine Webseite angegeben werden, welche die Reihung unter den Suchergebnissen beeinflusste.

Bei den Suchergebnissen wurden neben der URL und dem Titel einer Webseite auch die ersten 25 Wörter als Beschreibung, die Größe des Dokuments und das letzte Datum der Indizierung angegeben. Die Reihung der Suchergebnisse war von verschiedenen Faktoren abhängig:

- Die Anzahl der gefundenen Wörter in einem Dokument.
- Die syntaktische Nähe der Wörter auf einer Webseite.
- Je mehr Wörter der Suchanfrage in den ersten Wörtern, also meistens dem Titel eines Dokuments, vorkamen, umso höher wurde dies gewichtet.
- Der Grad der Übereinstimmung zwischen der Suchanfrage und den Schlüsselwörtern in den Meta-Tags.

Die Nachteile waren, dass die Reihung der Webseiten dem Benutzer nicht dargestellt wurde. Man wusste also nicht ob das Dokument, welches an der dritten Stelle gereiht wurde noch eine Relevanz von 98% oder nur mehr von 21% hatte. Weiters gestaltete sich aufgrund der Volltextsuche das Suchen, beispielsweise ohne boolesche Operatoren oder gesonderte Einstellungen, sehr schwierig. Die Suchergebnisse waren oft, vor allem bei aus mehreren Wörtern bestehenden Suchanfragen, sehr breit gestreut. Dazu kommt noch, dass die als Beschreibung verwendeten ersten 25 Wörter einer Webseite oft wenig aussagekräftig waren und es für den Benutzer deshalb noch undurchsichtiger war, welche tatsächliche Relevanz eine Webseite mit der Suchanfrage hat.

2.14. Die weitere Entwicklung von Suchmaschinen bis hin zu Google

Betrachtet man die Geschichte der Suchmaschinen erkennt man rasch, dass vor allem Mitte der neunziger Jahre des 20. Jahrhunderts eine Vielzahl dieser für das WWW entwickelt und veröffentlich wurden. Die hier Vorgestellten stehen dabei für die meisten und wichtigsten Entwicklungsschritte in dieser Zeit. Viele Nachfolger bauten auf diesen Technologien auf. Einige bekannte Beispiele für - nach AltaVista - in dieser Zeit entwickelte Suchmaschinen sind: InfoSeek, Inktomi, Magellan, HotBot und GoTo.

Folgend werden nun einige weitere Entwicklungen und Technologien vorgestellt, welche für diese Zeit wichtig waren.

2.14.1. Manuell verwaltete Verzeichnisse

Neben dem Sammeln und der Indizierung von Webseiten durch Webcrawler, gab es auch einige Verzeichnisse, welche auf eine qualitativ hochwertige Datenbank setzten. Ein Beispiel dafür ist das 1998 veröffentliche Open Directory Projekt (damals noch unter dem Namen „GnuHoo“ bzw. „NewHoo“). Hier wurde die Datenbank der Webseiten durch freiwillige Helfer aufgebaut und verwaltet. Dem Nachteil, dass diese Datenbank natürlich im Vergleich mit jenen anderer Suchmaschinen sehr klein war, wollte man durch eine höhere Qualität der Webseiten ausgleichen. Damit konnte auch die eingebaute Suchfunktion ein Suchergebnis liefern, in dem die einzelnen Webseiten eine aussagekräftige Beschreibung enthalten.

Webseiten mit ähnlichen Themen wurden weiters immer zu Gruppen zusammengefasst. Somit war es auch möglich sich einfach in einem bestimmten Themengebiet umzusehen.

2.14.2. Meta-Suchmaschinen

Eine andere Art von Suchmaschinen wurde 1995 erstmals entwickelt: MetaCrawler. Dabei handelte es sich um die erste Metasuchmaschine. Diese sind dadurch gekennzeichnet, dass sie keine eigene physische Datenbank besitzen, sondern dem Benutzer als Suchergebnis einfach die Suchergebnisse mehrerer anderer Suchmaschinen präsentieren.

Der Grundgedanke war, durch die Vernetzung von Suchmaschinen ein besseres Suchergebnis zu bekommen bzw. die Möglichkeit zu haben, wirklich beinahe das gesamte WWW durchsuchen zu können.

Das Ergebnis dieser Suche konnte einerseits eine noch undurchschaubarere Datenmenge sein, andererseits hat man bei sehr speziellen Suchanfragen mehr Chancen überhaupt ein Ergebnis zu bekommen.

Die Implementierungen und Funktionen dieser Suchmaschinen sind sehr unterschiedlich. Welche Suchmaschinen verwendet werden (bekannte oder weniger bekannte), die Anzahl und die Reihung der Ergebnisse sind meistens bei jeder Metasuchmaschine anders gestaltet.

2.14.3. Torrent Tracker

Dabei werden keine Webseiten, sondern Dateien über das Internet gesucht. Die Grundlage dafür stellt das BitTorrent Protokoll [COHEN_1] dar, welches 2001 entwickelt und veröffentlich wurde. Mit diesem Peer-to-Peer Kommunikationsprotokoll ist es möglich, eine große Anzahl an Dateien über ein Netzwerk so zu verteilen, dass für einen einzelnen Peer - das sind Computer auf denen ein Client läuft und Dateien zum Download bereitgestellt werden - die Ressourcenlast möglichst gering gehalten wird.

Die Dateien können mit Hilfe von Clients, die meist als eigene Applikation laufen, heruntergeladen werden. Dafür benötigt man zuerst eine „.torrent“-Datei, welche Metadaten über eine Datei enthält. Danach muss man sich mit einem Server (= Torrent Tracker) verbinden, der unter Anderem die Kommunikation zwischen verschiedenen Peers steuert. Von diesen können dann einzelne Teile einer Datei heruntergeladen werden.

Für die Dateien selbst gibt es keinen eigenen Index. Man ist darauf angewiesen, dass die „.torrent“-Dateien dafür auf Webseiten angegeben sind, oder man verbindet sich zu einem bekannten Torrent Tracker, der eine große Anzahl an „.torrent“-Dateien indiziert hat. Die Indizierung wird hier meist dynamisch aufgrund von Informationen der Clients, welche an den Torrent Tracker gesendet werden, durchgeführt. Auf diesen Servern bzw. Webseiten

kann man über den Client meistens eine Suche in der Liste der „.torrent“-Dateien durchführen. Die Suchergebnisse dabei unterliegen keiner besonderen Reihung. Im Client selbst können sie dann beispielsweise nach Dateinamen oder Bandbreiten sortiert werden.

2.14.4. Google

Eine weitere „Revolution“ brachte erst die Suchmaschine Google, welche 1998 veröffentlicht wurde [PAGE_2]. Für den enormen Erfolg dieser waren vor allem drei Punkte ausschlaggebend:

2.14.4.1. Ein neuer Ranking-Algorithmus

Der PageRank-Algorithmus ermöglichte erstmals die Bedeutung von Webseiten nicht nur augrund ihres Inhalts, sondern auch aufgrund ihrer Einbindung in die Struktur des WWW zu beurteilen. Diese neue Gewichtung ermöglichte es, dem Benutzer ein Suchergebnis zu präsentieren, welches eine bei weitem bessere Reihung relevanter Seiten als die meisten anderen Suchmaschinen vornahm.

Einer der Gründe für die oftmals schlechte Reihung der Suchergebnisse bei Suchmaschinen in dieser Zeit war, dass Leute versuchten den Rang ihrer Webseite durch sogenannte Suchmaschinenoptimierung zu erhöhen. Dabei wurden die Webseiten an den Ranking-Algorithmus einer Suchmaschine angepasst, um eine möglichst hohe Position unter den Suchergebnissen zu erhalten. Sehr oft wurden auch die Schwächen dieser Ranking-Algorithmen ausgenutzt, indem man beispielsweise eine große Anzahl von häufig gesuchten Schlüsselwörtern in die Meta-Tags aufnahm.

Eine andere Möglichkeit war, diese populären Suchanfragen direkt als Text in eine Webseite aufzunehmen, wobei dieser oft die gleiche Farbe hatte wie der Hintergrund. Damit war er für den Benutzer im ersten Augenblick unsichtbar und es wurde oft nicht gleich erkannt, dass sie sich auf Webseiten befanden, die eigentlich keine Relevanz mit ihrer Suchanfrage hatten, sondern lediglich zu Werbezwecken dienten.

Viele Suchmaschinen wollten diesen irrelevanten Webseiten durch neue Ranking-Algorithmen entgegenwirken. So wurden beispielsweise die Anzahl der Klicks auf einen Link als Qualitätskriterium gewertet. Doch auch dies konnte rasch manipuliert werden, da alles was ein Benutzer macht oder machen kann, meist auch automatisiert von Programmen durchgeführt werden kann.

Der PageRank-Algorithmus ist zwar ein wichtiger, aber nicht der einzige Faktor wie Webseiten bei Google gewichtet werden. Natürlich ist neben der Struktur vor allem auch der Inhalt einer Webseite in Relation zu einer Suchanfrage wichtig. Weitere Faktoren, die das Ranking einer Webseite unter den Suchergebnissen beeinflussen können, sind unter Anderem laut [PAGE_2], [HEISE_1] und [HEISE_2]:

- Das Vorkommen von Suchwörtern in der URL einer Webseite selbst.
- Die Position, wo Wörter einer Suchanfrage auf einer Webseite vorkommen. Dabei kommt es darauf an, ob diese beispielsweise bereits im Titel, in den Überschriften, in den Ankern der Links, oder lediglich im Text stehen.
- Die Häufigkeit, mit der die Wörter einer Suchanfrage auf einer Webseite vorkommen.
- Die syntaktische Nähe einzelner Wörter einer Suchanfrage auf einer Webseite.
- Die Schlüsselwörter, mit denen die Links auf eine Webseite verbunden sind.
- Von Google als besonders wichtig oder nützlich eingestufte Seiten zu bestimmten Themen werden bevorzugt gereiht (z.B.: Wikipedia).
- Der Aktualisierungsgrad einer Webseite. Je öfter diese aktualisiert wird, desto wahrscheinlicher ist es, dass die Inhalte darauf aktuell und gepflegt sind.
- Die Lebensdauer einer Webseite. Dies kann als Maß dafür gesehen werden, dass es sich um eine bewährte Webseite handelt, deren langes Bestehen durch gute Inhalte gerechtfertig ist.
- Die Herkunft der Links auf eine Webseite. Dabei spielen vor allem die Anzahl der IP-Adressen, Netzwerkklassen und Domains eine Rolle.

2.14.4.2. Wenig bis keine Werbung

Viele Suchmaschinen in dieser Zeit waren gekennzeichnet durch ein sehr hohes Werbeaufkommen. Einerseits waren die Startseiten und Webinterfaces mit Werbung, vor allem in Form von Werbebannern, regelrecht zugepflastert, andererseits waren auch die ersten Suchergebnisse bezahlte Links oder Werbungen, die oft wenig bis nichts mit der eigentlichen Suchanfrage zu tun hatten. Google bot hingegen ein schlichtes und übersichtliches Webinterface an, welches mit keiner einzigen Werbung versehen war.

Finanziert wurde diese Webseite zu Anfangs hauptsächlich durch sogenannte „AdWords“. Das sind Links, welche als rein textuelle Werbung neben den Suchergebnissen in Abhängigkeit von den Suchanfragen angezeigt werden [GOOGLE_1]. Da diese Werbung in die Suchergebnisse selbst nicht einfloss, war eine klare Trennlinie zwischen echten Suchergebnissen und Werbung vorhanden. Somit konnte sich der Benutzer ganz auf die relevanten Seiten seiner Suchanfrage konzentrieren.

2.14.4.3. Caching von Webseiten

Alle Webseiten, die durch den Webcrawler von Google gefunden wurden, wurden als komplette Kopie in einen Cache aufgenommen. Die Beurteilung der Webseiten und der Struktur des WWW wurde ausschließlich aufgrund dieses Cache gemacht. Dies hatte den großen Vorteil, dass man Webseiten auch dann betrachten konnte, wenn sie kurz- oder langfristig nicht verfügbar waren [GOOGLE_2]. Weiters können Suchanfragen damit sehr rasch bearbeitet werden, da die Reihung nicht erst im Zeitpunkt der Anfrage erstellt werden muss.

3. Der PageRank-Algorithmus

Dieses Kapitel beschäftigt sich mit der Theorie zur Suchmaschinenbewertung mit Hilfe des PageRank-Algorithmus. Nach einer generellen Einleitung wird die Funktionsweise dieses Algorithmus näher beschrieben. Abschließend werden verschiedene Arten der Berechnung der PageRanks vorgestellt und durch Beispiele untermalt.

3.1. Einleitung

Der PageRank Algorithmus wurde an der Stanford University (USA) entwickelt und erstmals in [PAGE_1] vorgestellt. Besonders die beiden Mitentwickler Lawrence Page und Sergey Brin haben heute einen sehr hohen Bekanntheitsgrad erreicht, da sie zur Anwendung des PageRank-Algorithmus die Suchmaschine Google [Page_2] entwickelt haben. Diese zählt heute wohl zu einer der bekanntesten Suchmaschinen und wurde weiters im April 2007 zur wertvollsten Marke der Welt [BRANDZ_1]erkoren. Diese Umstände sind ein eindeutiges Zeichen dafür, welche Bedeutung der PageRank-Algorithmus bzw. dessen Anwendung in der Suchmaschinenbewertung für das tägliche Leben und die Menschen hat.

3.1.1. Motivation

Das WWW wuchs in den Neunziger Jahren des 20. Jahrhunderts mit einer enormen Geschwindigkeit an. Die Tatsache, dass es nun sehr einfach möglich war, Informationen mit relativ geringem Aufwand einer breiten Öffentlichkeit verfügbar zu machen, war für viele Menschen sehr reizvoll: Einerseits konnten Firmen ihre Produkte rasch über das Internet im Rahmen des WWW präsentieren und vermarkten. Andererseits war es aber auch für Privatpersonen möglich, sich oder ihre Interessen auf eigenen Webseiten zu präsentieren. Daneben gibt es natürlich noch weitere Organisationen, die im WWW ein enormes Potential für die Verwirklichung ihrer Ziele oder die Verbreitung ihrer Informationen sahen. So gab es im Jahr 1998 Schätzungen zufolge ungefähr 150 Millionen Webseiten im WWW [PAGE_1], wobei viele dieser Seiten nur eine sehr kurze Lebensdauer hatten. Heutzutage ist anzunehmen, dass sich diese Zahl bestimmt schon vervielfacht hat.

Der Inhalt von Webseiten kann natürlich sehr verschiedenartig und auch unstrukturiert sein. Hinzu kommt, dass es keine Qualitätskontrolle für Webseiten gibt, d.h. jeder kann grundsätzlich Informationen über alles in jeder beliebigen Struktur und Form auf einer Webseite im Internet anbieten.

Eine große Herausforderung ist, dass man bei der enormen Anzahl an Webseiten und deren Informationsumfang im WWW etwas suchen und finden kann. Frühere Suchmaschinen (siehe Kapitel 2) versuchten durch verschiedenste Ansätze dem Benutzer die Möglichkeit zu bieten, Informationen aus der Masse an Webseiten zu finden. Mit der Zeit wurde jedoch, angetrieben von kommerziellem Interesse, versucht auf die Suchergebnisse der Suchmaschinen selbst, aber auch deren Reihung, Einfluss zu nehmen. Das Ziel vieler Organisationen mit starken finanziellen Interessen an der Auffindung ihrer Webseiten war es dabei, durch verschiedenste „Tricks“ die oberste Position bei den Ergebnissen einer Suchanfrage zu erreichen. Jene Webseiten, die sich an den ersten Positionen eines Suchergebnisses finden, werden natürlich von den Benutzern am Häufigsten angeklickt. Dies führte bei einigen früheren Suchmaschinen manchmal dazu, dass bei der Suche nach verschiedensten Begriffen immer dieselben Webseiten an der Spitze der Suchergebnisse standen. Das Auffinden von konkreten Informationen im WWW wurde daher manchmal beinahe unmöglich.

Dieses Problem verlangte also nach einer Lösung. Diese wurde vor allem durch die Entwicklung des PageRank-Algorithmus gefunden.

3.1.2. Grundidee zur Entwicklung des PageRank-Algorithmus

Das WWW ist grundsätzlich eine Sammlung von Webseiten und anderen Dateien. Diese können untereinander durch Links verbunden sein und stellen somit eine Hypermediastruktur dar. Eine solche Struktur kann mehr Informationen bieten, als es eine Webseite alleine kann. Genau diese Informationen über die Einbettung von Webseiten in eine Linkstruktur ist der zentrale Punkt des PageRank-Algorithmus. Mit Hilfe dieser werden unter Anderem die Wichtigkeit und Relevanz einer Webseite bestimmt.

Angeregt wurde diese Art der Suchmaschinenbewertung durch bereits bestehende Zitierungsverzeichnisse. Dabei gilt ein wissenschaftliches Dokument in einer Datenbank als umso wichtiger, je öfter es in anderen Dokumenten zitiert wird.

Dieser Bewertungsalgorithmus eines Dokuments kann jedoch nicht einfach auf eine andere Hypermediastruktur, in diesem Fall das WWW, übernommen werden:
Wissenschaftliche Arbeiten haben normalerweise bestimmte Eigenschaften, die Webseiten in vielen Fällen fehlen. So werden wissenschaftliche Publikationen auf ihre Qualität geprüft und haben meist ein ähnliches Erscheinungsbild. Durch diese Heterogenität kann auch die oben genannte Bewertungsmethode sinnvoll angewendet werden. Webseiten hingegen unterliegen keine Qualitätsprüfung, haben höchst unterschiedliche Inhalte und auch Erscheinungsbilder. Weiters können sie beliebig reproduziert werden, da es sehr einfache Möglichkeiten gibt, eine große Anzahl von Webseiten automatisch generieren zu lassen. Auf diese Unterschiede musste durch eine Weiterentwicklung des Algorithmus zur Gewichtung von wissenschaftlichen Dokumenten eingegangen werden.

3.2. Eine erste Definition des PageRank

Zuerst soll eine vereinfachte Definition des PageRank angegeben sein: Der PageRank einer Webseite ist ein numerischer Wert, der sich aus der Verlinkung einer Webseite berechnet.
Zu einer genaueren Definition sei folgende Formel angeführt [PAGE_1]:

$$PR(u) = c \sum_{v \in B_v} \frac{PR(v)}{N_v}$$

Dabei sind u und v zwei unterschiedlichen Webseiten im WWW, wobei v einen Link auf u hat. Man sagt auch v ist ein Backlink von u bzw. v hat einen Forwardlink auf u. PR(u) steht für den PageRank der Seite u. N_v ist die Anzahl der Links auf der Webseite v. B_v ist die Menge aller Webseiten, welche einen Link auf u enthalten.

Der PageRank einer Webseite u wird demnach aus der Summe der PageRanks jener Webseiten B_v, die auf u einen Link haben, berechnet. Wie viel von dem PageRank einer Seite v aus B_v an u weitergegeben wird, ist dabei von der Anzahl der gesamten Links auf v abhängig. Damit die Summe der PageRanks aller Seiten konstant bleibt, benötigt man einen Normalisierungsfaktor. Dieser ist in obiger Formel als c angegeben ist. Für eine weiterführende Beschreibung dieses Faktors und dessen Bedeutung siehe Kapitel 4.6.

An dieser Stelle ist anzumerken, dass in dieser Arbeit die Begriffe Knoten und Webseite bzw. Link und Kante synonym verwendet werden.

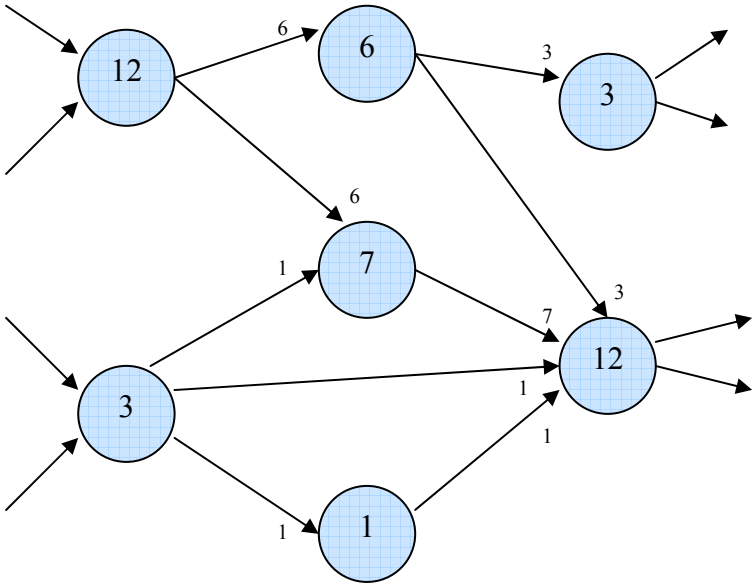

Abbildung 1: Beispielgraph zur PageRank-Weitergabe

Abbildung 1 zeigt den Ausschnitt aus einem Beispielgraphen. Die Knoten stellen dabei die Webseiten dar, in denen jeweils ihr PageRank angegeben ist. Am Ende der gerichteten Kanten, die für die Links zwischen den Webseiten stehen, sieht man den PageRank, der auf den zeigenden Knoten weitergegeben wird.

Eine weitere Möglichkeit zur Definition des PageRank bietet die Darstellung als Matrix, wie auch in [PAGE_1] beschrieben:
Dabei stellt man zuerst eine quadratische Matrix A auf, bei welcher die Zeilen und Spalten für die Knoten des Graphen stehen. Sei nun der Wert an der Stelle $A_{i,j}$ in der Matrix gleich $\frac{1}{N_i}$, falls die Webseite i einen Link auf die Webseite j besitzt. N_i gibt dabei die Anzahl der ausgehenden Links auf i an. Existiert kein solcher Link zwischen i und j ist $A_{i,j} = 0$.

Hat man diese Matrix A aufgestellt, berechnet man nun einen Eigenvektor R, welcher die PageRanks aller Webseiten enthält. Diese Berechnung wird mit Hilfe folgender Formel gemacht [PAGE_1]:

$$\vec{R} = cA\vec{R}$$

Obwohl diese Formel rekursiv angegeben ist, kann sie durch die Belegung von R mit Startwerten auch sehr einfach iterativ berechnet werden.

3.3. Eine erweiterte Definition des PageRank

Die in Kapitel 3.2 vorgestellten Formeln stellen eine gute Basis für die Berechnung des PageRank dar, sind jedoch für zwei Spezialfälle fehlerhaft:

3.3.1. Das Problem der „Dangling Links“

Bei diesem Problem handelt es sich um Verweise zu Webseiten, welche selbst keine ausgehenden Links haben. Oft handelt es sich dabei einfach um Links zu Dateien, welche natürlich oftmals keine eigenen Links haben können. Ein anderer Grund kann sein, dass die Webseite, auf welche ein Link zeigen würde, noch nicht vom Webcrawler der Suchmaschine erfasst wurde oder nicht (mehr) existiert. Aufgrund dieser Nichtexistenz des Zieles eines Links wird dann die Webseite so behandelt, als würde sie gar keine Links enthalten.

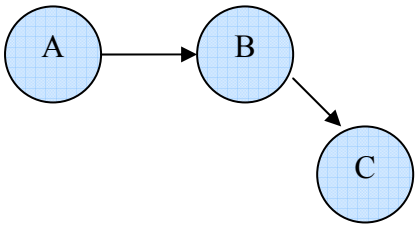

Abbildung 2: Beispielgraph für das "Dangling Links"-Problem

In Abbildung 2 ist die eben beschriebene Situation dargestellt. Dabei hat der Knoten C keine ausgehenden Links. Das Problem dabei ist, dass diese Webseite zwar einen PageRank erhalten würde, diesen jedoch aufgrund der fehlenden Links nicht verteilen

kann. Für eine vollständige und korrekte Berechnung der PageRanks aller Webseiten ist es jedoch unerlässlich, dass jede Webseite ihren eigenen PageRank wieder weitergibt.

Eine Lösung für dieses Problem ist rasch gefunden und wurde auch bereits in [PAGE_1] beschrieben:
Da diese Knoten die PageRank-Berechnung anderer Seiten nicht oder kaum beeinflussen können, werden diese einfach vor der Berechnung der PageRanks entfernt. Wurde nun die Berechnung für alle noch verbleibenden Webseiten abgeschlossen, werden die Dangling Links wieder hinzugefügt und der PageRank wird einfach an die neuen Knoten angepasst. Auswirkungen für andere Seiten gibt es dabei kaum. Der PageRank jener Webseite, die einen Dangling Link enthält verändert sich zwar etwas, da sie durch dessen Löschung einen Link weniger hat. Dies hat jedoch insgesamt keine signifikanten Auswirkungen.

Eine weitere Lösung für dieses Problem bietet auch das Zufallssurfer Modell, welches in Kapitel 3.4.1 vorgestellt wird.

3.3.2. Das Problem des „Rank Sink“

Zur Darstellung dieses Problems bzw. der Problemstellung sei folgender Beispielgraph in Abbildung 3 angeführt:

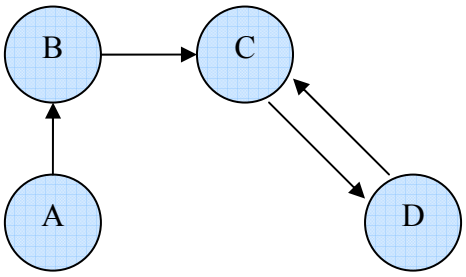

Abbildung 3: Beispielgraph für das "Rank Sink"-Problem

In diesem Graphen verlinken sich die Knoten C und D gegenseitig. Dadurch, dass die beiden Knoten keinen Link auf eine andere Seite haben, werden ihre PageRanks nur zwischen diesen beiden Knoten hin und her geschoben. Diese beiden Webseiten können

ihren PageRank somit nie an Andere weitergeben. Dies hat zur Folge, dass die PageRanks anderer Webseiten auch nicht korrekt bewertet werden.

Berechnet man die PageRanks für den Graphen in Abbildung 3 würden sich folgende Gleichungen ergeben:

$$PR(A) = c * 0$$

$$PR(B) = c\frac{PR(A)}{1}$$

$$PR(C) = c(\frac{PR(B)}{1} + \frac{PR(D)}{1})$$

$$PR(D) = c\frac{PR(C)}{1}$$

Nach dem wiederholten Einsetzen in diese Gleichungen ergeben sich folgende PageRanks: PR(A) = PR(B) = 0 und PR(C) = PR(D) = c^i, i = Anzahl der Iterationen.

Man sieht also, dass die Webseite B den PageRank 0 hat, obwohl sie einen eingehenden Link besitzt. Die Seiten C und D schieben ihre PageRanks im Laufe der Iteration, jeweils multipliziert mit dem Faktor c, nur hin und her. Nimmt man für den Normalisierungsfaktor c einen Wert < 1, würde sich der PageRank jeder Seite immer verringern und schließlich gegen 0 gehen. Bei einem Faktor > 1 würden die beiden Webseiten ihren PageRank immer weiter steigern.

Zur Kompensation dieser Probleme wurde die PageRank-Formel um einen konstanten Wert erweitert, der auch gewährleistet, dass jede Webseite einen bestimmten Grund-PageRank besitzt. Die erweiterte Formel hat folgende Form [PAGE_2]:

$$PR(u) = (1-c) + c\sum_{v \in B_v} \frac{PR(v)}{N_v}$$

Berechnet man nun die PageRanks aus dem vorherigen Beispiel (mit c = 0.85), erhält man folgendes Ergebnis: PR(A) = 0.15, PR(B) = 0.2775, PR(C) = 1.85, PR(D) = 1.7225
Man erhält nun also eine korrekte Reihung aller Webseiten.

Auch die Formel zur Berechnung der PageRanks als Eigenvektor wird erweitert und lautet daher:

$$\vec{R} = cA\vec{R} + \vec{E}$$

Die Variable E steht dabei für einen Vektor mit n (= Anzahl der Webseiten) Einträgen, wobei jeder Eintrag gleich 1 – c ist.

3.4. Die Berechnung des PageRank

Die PageRanks von Webseiten in einem Graphen können auf verschiedene Weise berechnet werden [PAGE_1]:

3.4.1. Das Zufallssurfer Modell

Bei diesem Modell geht man davon aus, dass ein imaginärer Benutzer durch das WWW surft. Mit einer bestimmten Wahrscheinlichkeit α folgt er zufällig einem Link auf der Webseite, auf der er sich gerade befindet. Wie bei einem realen Benutzer kann sich auch dieser langweilen und besucht periodisch mit einer Wahrscheinlichkeit (1 – α) eine neue Webseite im Graphen. Sollte der Benutzer beispielsweise in einer Endlosschleife gefangen sein, wo eine bestimmte Anzahl von Webseiten sich nur im Kreis verlinken, kann sich der Zufallssurfer dadurch aus diesem Zirkel beifreien.

Folgendes Beispiel soll das Zufallssurfer Modell verdeutlichen:

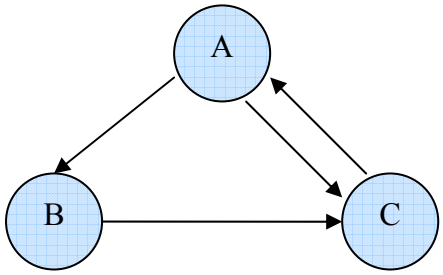

Abbildung 4: Beispielgraph zur PageRank-Berechnung

Der Graph in Abbildung 4 besteht also aus 3 verbundenen Knoten. Als Wahrscheinlich α, dass der Zufallssurfer einen Link auf der Webseite, auf der er sich gerade befindet verfolgt, sei 0.85 definiert. Demnach ist die Wahrscheinlichkeit, dass er eine neue Webseite direkt aufruft mit (1 – α), also 0.15 definiert.
Man kann nun folgende quadratische Matrix aufstellen, die zu den Knoten des Graphen korrespondierende Zeilen und Spalten besitzt:

$$\begin{pmatrix} 0 & \frac{\alpha}{2}+\frac{1-\alpha}{2} & \frac{\alpha}{2}+\frac{1-\alpha}{2} \\ \frac{1-\alpha}{2} & 0 & \frac{\alpha}{1}+\frac{1-\alpha}{2} \\ \frac{\alpha}{1}+\frac{1-\alpha}{2} & \frac{1-\alpha}{2} & 0 \end{pmatrix}$$

Die Diagonale der Matrix hat stets den Wert 0, da angenommen wird, dass der Zufallssurfer immer eine andere Webseite des Graphen auswählt, als jene auf der er sich gerade befindet.

In der ersten Zeile sieht man die Möglichkeiten, wie sich der Zufallssurfer auf der Webseite A verhalten kann. Er kann einem der Links zu den Webseiten B oder C folgen, oder direkt auf eine neue Webseite springen. Die Wahrscheinlichkeit, dass er dem Link von A nach B folgt ist demnach $\frac{\alpha}{2}$. Der Divisor 2 ergibt sich dadurch, dass die Wahrscheinlichkeit einen Link zu benutzen natürlich immer durch die Anzahl der Links

auf der Webseite dividiert werden muss. Die Wahrscheinlichkeit, dass er zufällig einen anderen Knoten des Graphen auswählt ist $\frac{1-\alpha}{2}$ und gilt für alle anderen Knoten, egal ob ein Link vom Knoten A dorthin führt oder nicht. Der Divisor 2 ist in diesem Fall die Anzahl aller Knoten des Graphen minus eins für sich selbst. Man fügt sozusagen neue imaginäre Links einer Webseite hinzu, die mit einer geringeren Wahrscheinlichkeit besucht werden, als die realen Links.

Setzt man nun in die Matrix die Werte für α ein, ergibt sich folgende Matrix A:

$$A = \begin{pmatrix} 0 & 0.5 & 0.5 \\ 0.075 & 0 & 0.925 \\ 0.925 & 0.075 & 0 \end{pmatrix}$$

Aus dieser Matrix können nun folgende Gleichungen für die Berechnung der PageRanks aufgestellt werden:

$$PR(A) = 0.075\ PR(B) + 0.925\ PR(C)$$
$$PR(B) = 0.5\ PR(A) + 0.075\ PR(C)$$
$$PR(C) = 0.5\ PR(A) + 0.925\ PR(B)$$

Nun wird der PageRank jeder Seite in Abhängigkeit von den anderen Seiten iterativ solange berechnet, bis sich die PageRanks nicht mehr verändern. Für den PageRank jeder Seite wird zum Zeitpunkt t_0 ein konstanter Wert φ gewählt. Dadurch ist die Summe der PageRanks aller Knoten immer gleich der Anzahl der Knoten multipliziert mit φ.
Wird φ = 1 definiert, ist im Zeitpunkt t_0 PR(A) = PR(B) = PR(C) = 1. Im Zeitpunkt t_1, welcher der ersten Iteration entspricht, sieht die Berechnung wie folgt aus:

$$PR(A) = 0.075 + 0.925 = 1$$
$$PR(B) = 0.5 + 0.075 = 0.575$$
$$PR(C) = 0.5 + 0.925 = 1.425$$

Nach einigen Iterationen konvergiert der Algorithmus schließlich bei:

$$PR(A) = 1{,}148624325$$
$$PR(B) = 0{,}663409617$$
$$PR(C) = 1{,}187966058$$

3.4.2. Die iterative Berechnung

Die in Kapitel 3.2 vorgestellten Formeln zur Berechnung des PageRanks einer Webseite sind zwar grundsätzlich rekursiv, können jedoch auch iterativ berechnet werden. Dazu wird jeder Webseite am Beginn der Berechnung ein bestimmter PageRank (z.B.: 1) zugewiesen. Durch mehrere Iterationen über alle Knoten im Graphen der Webseiten nähert sich der PageRank immer weiter an den tatsächlichen PageRank an. Der Algorithmus konvergiert, sobald sich die PageRanks der Webseiten nicht mehr verändern.

Anhand dieser Formeln können nun durch Setzen eines Start-PageRanks nach mehreren Iterationen die PageRanks aller Webseiten berechnet werden. Für das Beispiel in Abbildung 4 setzt man den Normalisierungsfaktor c gleich 0.85 und den Start-PageRank gleich 1. Daraus ergeben sich durch Einsetzen in die erste Formel nach einer Iteration folgende Gleichungen:

$$PR(A) = (1 - 0.85) + 0.85 \frac{1}{1} = 1$$

$$PR(B) = (1 - 0.85) + 0.85 \frac{1}{2} = 0.576$$

$$PR(C) = (1 - 0.85) + 0.85 \left(\frac{1}{2} + \frac{1}{1}\right) = 1.425$$

Nach einigen Iterationen konvergiert der Algorithmus schließlich bei:

$$PR(A) = 1.163369135$$
$$PR(B) = 0.644431882$$
$$PR(C) = 1.19219898$$

Für die PageRank-Berechnung mittels Eigenvektor ergibt sich folgende Berechnung:

$$R = 0.85AR + E$$

Dabei haben A, R und E zu Beginn folgende Werte:

$$A = \begin{pmatrix} 0 & 0 & \frac{1}{1} \\ \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & \frac{1}{1} & 0 \end{pmatrix} \quad R = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \quad E = \begin{pmatrix} 0.15 \\ 0.15 \\ 0.15 \end{pmatrix}$$

Das Ergebnis ist folgender Eigenvektor R, der gleich dem Ergebnis obiger Berechnung ist:

$$R = \begin{pmatrix} 1.163369135 \\ 0.644431882 \\ 1.19219898 \end{pmatrix}$$

Es ist auch zu sehen, dass die Werte für die PageRanks im Vergleich zum Zufallssurfer Modell dieselbe Reihung repräsentieren und nur gering davon abweichen.

Abschließend sei noch angemerkt, dass die iterative PageRank-Berechnung laut [PAGE_1] 1998 für 75 Millionen URLs in ungefähr fünf Stunden gemacht werden konnte. Eine Konvergenz bei einer annehmbaren Toleranz wurde dabei bei 52 Iterationen erreicht, wobei anzunehmen ist, dass der Algorithmus bei ungefähr 100 Iterationen vollständig konvergiert.

4. Analyse des PageRank-Algorithmus

Nachdem nun der eigentliche Algorithmus, wie er in [PAGE_1] und [PAGE_2] veröffentlich wurde, eingehend beschrieben wurde, sollen nun einzelne Faktoren dieses Algorithmus näher analysiert werden. Weiters soll aufgezeigt werden, wie sich die

Berechnung der PageRanks bei einer Änderung dieser Faktoren verändert. Auch eine mögliche Manipulation dieses Algorithmus wird näher betrachtet.

4.1. Die Konvergenz des Algorithmus

Bei der Berechnung der PageRanks wird eigentlich der Eigenvektor mit dem größten Eigenwert einer Matrix A berechnet [FARAHAT_1]. Gemäß dem Theorem von Perron und Frobenius hat eine $n^x n$-Matrix, welche ausschließlich positive Einträge enthält, die Eigenschaft, dass es einen positiven Eigenwert λ [MUEHLBACHER_1] dieser Matrix gibt, sodass alle anderen Eigenwerte $|\lambda_i| < \lambda$ sind [MACCLUER_1].
Dadurch wird die Existenz des größten Eigenwerts bestätigt. Durch eine geeignete Wahl des Normalisierungsfaktors $0 < c < 1$ existiert laut [FARAHAT_1] auch der korrespondierende Eigenvektor. Dadurch ist gewährleistet, dass der Algorithmus stets konvergiert. Für den genauen Beweis sei auf [MACCLUER_1] verwiesen.

4.2. Der Einfluss eingehender Links

Die Anzahl der eingehenden Links sind eines der wichtigsten Kriterien in der Formel des PageRank-Algorithmus. Jeder zusätzliche eingehende Link erhöht - bei gleichem Normalisierungsfaktor und gleicher Anzahl ausgehender Links - den PageRank einer Webseite.

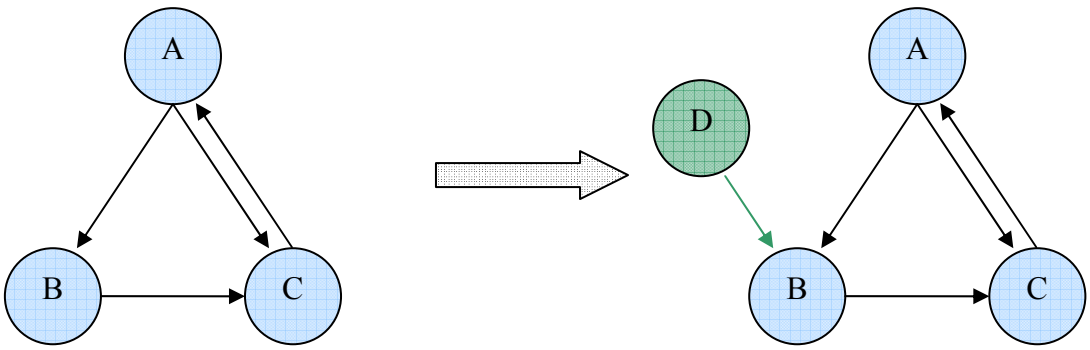

Abbildung 5: Beispielgraph zum Hinzufügen eines eingehenden Links

Bei dem Beispielgraphen in Abbildung 5 wird zu den bestehenden drei Webseiten A, B und C eine neue Webseite D hinzugefügt. Diese hat einen Link auf die Webseite B.

Vor dem Hinzufügen von D lautete die Formel zur Berechnung des PageRanks von B folgendermaßen:

$$PR(B) = (1-c) + c\frac{PR(A)}{2}$$

Nach dem Hinzufügen der Webseite D ergebt sich für B folgende Formel:

$$PR(B) = (1-c) + c(\frac{PR(A)}{2} + \frac{PR(D)}{1})$$

Der gesamte PageRank der Webseite B erhöht sich somit um den Faktor $c\frac{PR(D)}{1}$. Generell findet also mit jedem hinzugefügten Link eine Erhöhung des PageRanks, auf den diese Webseite zeigt, um den Wert $c\frac{PR(NeueSeite)}{\#Links(NeueSeite)}$ statt.

Diese Erhöhung des PageRanks einer Webseite kann jedoch auch die PageRanks anderer Webseiten beeinflussen. Dies ist genau dann der Fall, wenn ein Link von der Webseite mit dem erhöhten PageRank zu einer anderer Webseite besteht. In dem Beispiel aus Abbildung 5 würde sich primär der PageRank der Seite C auch erhöhen, da B seinen höheren PageRank - bei einer gleichbleibenden Anzahl an ausgehenden Links – an C, und in weiterer Folge an die Links von C usw., weitergibt.

Die Summe aller PageRanks im Graphen muss immer konstant bleiben. Das bedeutet, dass wenn durch einen neuen Link eine neue Webseite hinzugefügt wird, erhöht sich auch die Summe aller PageRanks genau um 1. Wird durch einen Link keine neue Webseite hinzugefügt, verändern sich nur die PageRanks jener Webseite, auf die der neue Link zeigt. Da die Summe konstant bleibt, müssen dafür die PageRanks anderer Seiten geringer werden.

4.3. Der Einfluss ausgehender Links

Nicht nur eingehende Links beeinflussen die PageRanks anderer Webseiten, sondern auch zusätzliche ausgehende Links. Dies erscheint vor allem im Bezug auf das Zufallssurfer Modell logisch: Umso mehr Links sich auf einer Webseite befinden, umso kleiner ist die Wahrscheinlichkeit, dass der Zufallssurfer einen bestimmten dieser Links wählt. Damit verringert sich für jeden zusätzlich ausgehenden Link auch der PageRank jener Webseite, auf der sich der zusätzliche Link befindet.

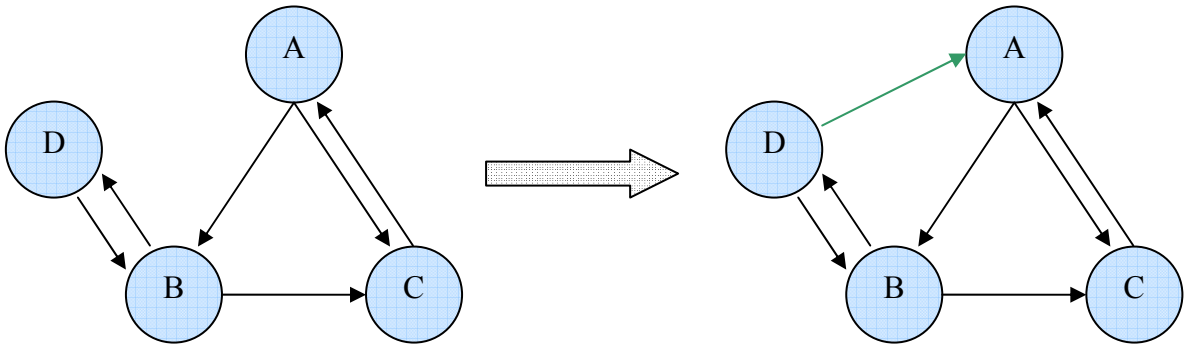

Abbildung 6: Beispielgraph zum Hinzufügen eines ausgehenden Links

Wie im Beispiel in Abbildung 6 ersichtlich, bekommt die Webseite D einen neuen Link auf A. Dadurch muss sich auch der PageRank von D verringern. Durch Einsetzen in die Formel lauten die PageRanks des Graphen vor dem Hinzufügen des neuen Links:

PR(A) = 1,09050049
PR(B) = 1,16001989
PR(C) = 1,10647116
PR(D) = 0,64300846

Nach dem Hinzufügen des Links zwischen D und A ergeben sich folgende PageRanks:

PR(A) = 1,34925268
PR(B) = 0,96071077
PR(C) = 1,13173447
PR(D) = 0,55830208

Man kann beobachten, dass sich wie erwartet der PageRank der Webseite D verringert. Dieser geringere PageRank hat natürlich einen unmittelbaren Einfluss auf die Webseite B, welche von D damit auch einen geringeren Wert erhält. Hingegen erhöht sich jener von A, da sie ja einen neuen eingehenden Link bekommt und nun auch vom PageRank der Seite D profitiert. Die Summe der PageRanks im Graphen ist dabei weiterhin konstant.

Hätte D keine eingehenden Links, würde sich der PageRank von D nicht verringern, sondern konstant den Wert 1 – c haben. Die PageRanks von B und C würden sich aber trotzdem verringern, da der PageRank von D durch eine höhere Anzahl an Links geteilt wird. Der PageRank von A steigt wieder aufgrund des zusätzlich eingehenden Links.

4.4. Der Effekt eines Linktausches

Oft tauschen mehrere Webseiten untereinander Links, in der Absicht damit ihren PageRank steigern zu können. Ob dieser Effekt auch wirklich eintritt, soll folgend untersucht werden. Dazu sei folgender Beispielgraph angeführt:

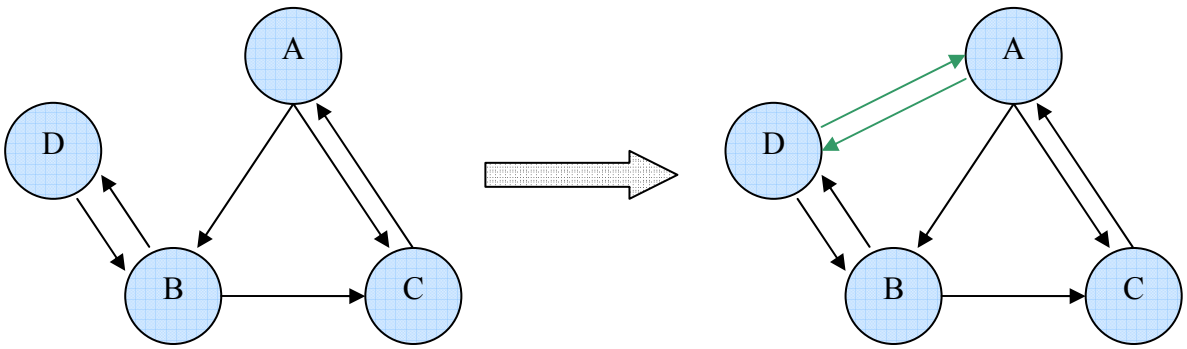

Abbildung 7: Beispielgraph zum Linktausch

Wie in Abbildung 7 ersichtlich, tauschen die Webseiten D und A jeweils einen Link, indem sie sich gegenseitig verlinken.

Die PageRanks vor dem Linktausch sind:

PR(A) = 1,09050049
PR(B) = 1,16001989
PR(C) = 1,10647116
PR(D) = 0,64300846

Nach dem Linktausch ergeben sich folgende PageRanks:

PR(A) = 1,2982456
PR(B) = 0,9005848
PR(C) = 0,9005848
PR(D) = 0,9005848

Es ist ersichtlich, dass die beiden tauschenden Webseiten tatsächlich ihren PageRank erhöhen. Im Gegenzug sinkt jedoch der PageRank der anderen Seiten, da die Summe aller PageRanks konstant bleiben muss. Dieser Effekt gilt natürlich nur, wenn wenige Seiten einen Linktausch machen. Würden alle Webseite in Abbildung 7 dies tun, würde die Reihung unverändert bleiben.

4.5. Der Start-PageRank der Webseiten

Hier stellt sich die Frage, ob unterschiedliche Startwerte auch die PageRanks der einzelnen Webseiten bzw. die Summe aller PageRanks beeinflussen können. Dies ist im Allgemeinen nicht der Fall. Egal welcher Startwert für die PageRanks angenommen wird, es bleiben sowohl die Summe aller PageRanks, als auch die einzelnen PageRanks selbst immer konstant.

Der einzige Unterschied ist die Anzahl der Iterationen bis zur Konvergenz des Algorithmus. So führen unterschiedliche Startwerte auch zu einer geringeren oder höheren Anzahl an Iterationen. Für das Beispiel aus Abbildung 6 ergibt sich in Abhängigkeit der

verschiedener Startwerte folgende Anzahl an notwendigen Iterationen bis zur Konvergenz des konvergiert (Tabelle 1):

Startwert	Anzahl der Iterationen
1	37 Iterationen
0.75	117 Iterationen
1.25	117 Iterationen
0.25	123 Iterationen
0	127 Iterationen
-1	131 Iterationen
100	155 Iterationen
-100	155 Iterationen

Tabelle 1: Anzahl der Iterationen in Abhängigkeit des Start-PageRanks

Es ist ersichtlich, dass je näher ein Wert bei 1 gewählt wird, der Algorithmus desto schneller konvergiert. Daher sollte als Start-PageRank stets 1 gewählt werden, um somit eine rasche Konvergenz zu gewährleisten. Die Summe aller PageRanks entspricht dabei immer der Anzahl der Webseiten.

In der Literatur findet man jedoch oft noch eine etwas andere Möglichkeit, die PageRanks zu berechnen:

$$PR(u) = \frac{(1-c)}{n} + c \sum_{v \in B_v} \frac{PR(v)}{N_v}$$

Die Variable n steht dabei für die Anzahl aller Knoten im Graphen. Durch diese kleine Modifikation bleibt die Reihung der Webseiten gleich, es verringern sich jedoch sowohl die PageRanks der einzelnen Webseiten, als auch die Summe aller PageRanks. Diese ist mit dieser Formel immer gleich 1.

Durch die veränderte Formel, ergibt sich auch ein veränderter optimaler Start-PageRank, wie in Tabelle 2 festgehalten:

Startwert	Anzahl der Iterationen
0.25	37 Iterationen
0	117 Iterationen
0,75	122 Iterationen
1	125 Iterationen
0	127 Iterationen
-1	129 Iterationen
100	155 Iterationen
-100	155 Iterationen

Tabelle 2: Anzahl der Iterationen in Abhängigkeit des Start-PageRanks

Es ist zu sehen, dass hier der beste Startwert $\frac{1}{n}$ ist. Bei dieser Formel sollte als Startwert daher stets $\frac{1}{n}$ verwendet werden, um eine möglichst rasche Konvergenz zu gewährleisten.

Abschließend sei noch anzumerken, dass es egal ist, mit welchem Knoten man die Berechnung beginnt, da die Startwerte in der ersten Iteration ja für alle Knoten gleich sind.

4.6. Der Normalisierungsfaktor c

Ein weiterer wichtiger Faktor im PageRank-Algorithmus ist der Normalisierungsfaktor c (in der Literatur auch oft Dämpfungsfaktor d genannt). Dieser muss stets einen Wert $0 < c < 1$ haben [PAGE_1]. Es stellt sich die Frage, ob die Wahl von c auch einen Einfluss auf die PageRank-Berechnung hat.

Aus der Sichtweise des Zufallssurfer Modells stellt c dabei die Wahrscheinlichkeit dar, mit der der Surfer einem Link auf der Webseite, auf der er sich befindet auswählt. Hingegen gibt $(1 - c)$ an, mit welcher Wahrscheinlichkeit der Surfer eine neue Webseite besucht und keinem Link auf der Webseite folgt.

Geht der Wert von c gegen 1 ergibt sich folgende Berechnung:

$$PR(u) = 0 + 1 \sum_{v \in B_v} \frac{PR(v)}{N_v}$$

Dies würde bedeuten, umso näher der Wert von c gegen 1 geht, umso wahrscheinlicher ist es, dass der Zufallssurfer nur Links folgt und kaum mehr zufällig eine Seite auswählt. Man nutzt also beinahe ausschließlich die ursprüngliche Linkstruktur, wodurch auch das Problem des Rank Sink wieder wahrscheinlicher auftritt.

Würde der Wert für c gegen 0 gehen, würde sich folgende Berechnung ergeben:

$$PR(u) = 1 + 0 \sum_{v \in B_v} \frac{PR(v)}{N_v}$$

Daraus folgt, dass auch jeder PageRank gegen 1 gehen würde. Dies impliziert, dass der Zufallsserver fast keinen Links auf den Webseiten folgen würde und somit die Linkstruktur des Graphen vernachlässigbar bzw. irrelevant wird.

Zusammenfassend kann man also sagen, dass für Werte $0 < c < 0.5$ die Linkstruktur weniger wichtig ist und für Werte $0.5 < c < 1$ die Linkstruktur immer wichtiger wird.
Da man natürlich bei diesem Algorithmus ein stärkeres Gewicht auf die Linkstruktur legen möchte, soll auch ein Wert für c genommen werden, der zwischen 0.5 und 1 liegt. In [PAGE_1] und [PAGE_2] wird dafür immer wieder 0.85 angegeben, mit welchem die Besten Ergebnisse erzielt werden können.

Eine weitere Möglichkeit, wie man den Normalisierungsfaktor c nutzen könnte, wäre ein „personalisierter PageRank“ [PAGE_1]. Dabei wird c, genauer gesagt der Teil (1 – c) nur einer einzigen oder eine Gruppe von Webseiten zugeordnet. Man simuliert damit, dass ein Benutzer auf einer dieser Webseiten startet (z.B.: wurde diese als Startseite eingestellt). Sie hat dabei für den Benutzer selbst einen höheren Stellenwert und soll demnach auch besser

gereiht sein. Ebenso sollen alle Webseiten, zu denen ein Link auf dieser „Startseite“ besteht, eine bessere Reihung erhalten.

Als Beispiel sei wieder folgender Graph angeführt:

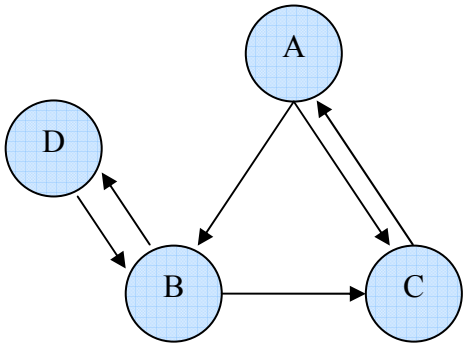

Abbildung 8: Beispielgraph für den personalisierten PageRank

Nach der normalen PageRank-Berechnung würde sich für den Graphen in Abbildung 8 folgendes Ergebnis einstellen:

PR(A) = 1,09050049

PR(B) = 1,16001989

PR(C) = 1,10647116

PR(D) = 0,64300846

Möchte man nun beispielsweise C das größte Gewicht zusprechen, müsste man folgende Gleichungen aufstellen:

$$PR(A) = c\frac{PR(C)}{1}$$

$$PR(B) = c(\frac{PR(A)}{2} + \frac{PR(D)}{1})$$

$$PR(C) = (1-c) + c(\frac{PR(A)}{2} + \frac{PR(B)}{2})$$

$$PR(D) = c\frac{PR(B)}{2}$$

Als Ergebnis würde man mit c = 0.85 erhalten:

PR(A) = 0,32003979
PR(B) = 0,21294233
PR(C) = 0,37651740
PR(D) = 0,09050049

Man sieht also, dass C nun den größten PageRank hat. Jene Webseiten, auf die C einen Link hat (hier nur Webseite A), werden ebenfalls höher gereiht, da sie ja von dem Rang von C profitieren.

Durch diese Eigenschaft des Normalisierungsfaktors ist es also möglich einen auf den Benutzer zugeschnittenen PageRank zu schaffen, bei dem von diesem bevorzugte Webseiten höher gereiht werden bzw. einen höheren Wert haben.

Wichtig hier zu erwähnen ist, dass es sich bei der Berechnung dieser personalisierten PageRanks vor allem um eine theoretische Möglichkeit handelt. Zur Verwirklichung müsste man immerhin für jeden einzelnen Benutzer die PageRanks aller Webseiten im WWW in Abhängigkeit seiner Interessen neu berechnen. Dies ist derzeit jedoch äußerst ressourceintensiv und deshalb nicht wirtschaftlich zu implementieren.

4.7. Die Skalierbarkeit

Da der PageRank-Algorithmus vor allem nicht für sehr kleine, sondern gerade für den extrem großen Graphen des WWW funktionieren muss, stellt sich die Frage ob er für diesen Graphen auch in vernünftiger Zeit berechnet werden kann.

Wie in den vorherigen Beispielen aufgezeigt bleibt die Summe aller PageRanks in einem Graphen stets konstant. Dies hat zu Folge, dass die PageRanks aller Seiten mit zunehmender Größe des Graphen zwar kleiner werden, die Reihung unter den einzelnen Webseiten jedoch immer gleich bleibt. Vorraussetzung dafür ist natürlich, dass die PageRanks mit entsprechender mathematischer Genauigkeit berechnet werden können.

Wie in [PAGE_1] beschrieben, *„konvergiert der Algorithmus für eine Datenbank mit 322 Millionen Webseiten mit einer angemessenen Genauigkeit nach 52 Iterationen. Bei einer halb so großen Datenbank geschieht dies nach 45 Iterationen. Der Skalierungsfaktor beträgt daher log n, wodurch der Algorithmus natürlich besonders für extrem große Graphen geeignet ist.“*

4.8. Manipulation des PageRank-Algorithmus

Zu Beginn des Einsatzes des PageRank-Algorithmus zur Suchmaschinenbewertung galt dieser als ziemlich „fälschungssicher“. Doch die ersten Manipulationsversuche ließen nicht lange auf sich warten. Man versuchte die Suchmaschine Google oder den PageRank-Algorithmus selbst, beispielsweise durch Suchmaschinenoptimierung (Search Engine Optimization = SEO) oder andere Tricks, den PageRank und damit die Reihung einer einzelnen Webseite innerhalb der Suchergebnisse zu beeinflussen. Folgend werden einige der bekanntesten Manipulationsmethoden erklärt.

4.8.1. Google Bomben

Google Bomben, oder auch als Link Bomben bekannte, Suchmaschinenmanipulation hatten Anfangs eher humoristische bzw. politische Hintergründe. Im Gegensatz zu früheren Manipulationsarten, die hauptsächlich darauf abzielten den Benutzer auf Webseiten mit starkem Werbeaufkommen zu führen, traten diese eigentlich nur vereinzelt auf.

Die erste wirklich Aufsehen erregende Google Bombe wurde 2003 bekannt [HEISE_3]. Dabei wurde bei der Suchanfrage „miserable failure“ als erstes Suchergebnis bei Google und zahlreichen anderen Suchmaschinen ein Link zur Biografie des amerikanischen Präsidenten George W. Bush angezeigt. In der Folge traten weitere solcher Bomben auf, durch welche meist nach ironischen oder beleidigenden Suchanfragen Politiker oder andere Institutionen an erste Stelle der Suchergebnisse gereiht wurden. In Österreich wurde der damalige Finanzminister Karl-Heinz Grasser „Opfer“ einer Google Bombe: Bei der Suchanfrage „völlige Inkompetenz“ erschien seine private Homepage an erster Stelle [DIEPRESSE_1].

Realisiert wurden diese durch das Ausnutzen einiger Schwächen im Ranking-Algorithmus von Google. So war ein Kriterium bei der Reihung der Suchergebnisse, ob die Links, welche auf eine Webseite verweisen, bestimmte Schlüsselwörter im Ankertext enthielten. Dadurch kann man, meist in einer Gruppe organisiert, Links in Foren, Blogs, Webseiten, etc. auf eine bestimmte Webseite setzen. Diese Links werden mit einem oder mehrerer Schlüsselwörter verknüpft. Durch die große Anzahl an Links zu dieser Webseite, welche aus Sicht des Indexers offensichtlich genau diese Schlüsselwörter als Inhalt hat, wird sie bei den Suchmaschinen an erster Stelle gereiht.

Die meisten Google Bomben wurden von den Suchmaschinen bereits wieder entfernt. Da jedoch auch damit begonnen wurde, diese Manipulationsmethode für Werbung und kommerzielle Zwecke zu verwenden, wurden zusätzlich die entsprechenden Ranking-Algorithmen der Suchmaschinen so abgeändert, dass eine solche Manipulation nicht mehr möglich ist.

4.8.2. Link Farmen

Dabei handelt es sich um eine große Anzahl von meist automatisch generierten Webseiten. Diese enthalten alle Links zu einer bestimmten Webseite. Erstmals trat dies 1999 auf. Damit versucht man bei der Platzierung dieser Webseite unter den Suchergebnissen möglichst den ersten Rang zu erhalten. Damals benutzten einige Suchmaschinen die sogenannte „Link Popularity“, mit der einfach die Anzahl der auf eine Webseite zeigenden Links gezählt wurde. Durch die in einer Link Farm enthaltenen enormen Anzahl an Webseiten, erreichte man bald das gewünschte Ziel.

Da auch für den PageRank-Algorithmus eines der wichtigsten Kriterien die Anzahl der eingehenden Links ist, war dieser natürlich auch anfällig für diese Manipulationsart. Dem wirkten die Suchmaschinen entgegen, indem sie nicht nur die Anzahl der eingehenden Links oder den PageRank als einziges Bewertungskriterium heranzogen, sondern auch die Qualität der Webseiten als wichtiges Kriterium in ihren Ranking-Algorithmus aufnahmen. Auch die Anzahl an verschiedenen Webservern und Domainnamens jener Webseiten, von denen aus eine andere Seite verlinkt wurde, war dabei sehr bedeutend, da Link Farmen oft auf einem einzigen Server oder über eine einzige Domain angelegt wurden.

Eine ähnliche Art die Popularität seiner Seite durch mehr eingehende Links zu steigern, sind sogenannte Link Kampagnen. Hier soll durch den direkten Kontakt mit anderen Betreibern von Webseiten versucht werden, einen Link auf seine eigene Seite zu erhalten. Teilweise wird dafür auch, vor allem um einen Link auf Webseiten mit einem hohen PageRank zu erhalten, ein nicht unerheblicher Geldbetrag bezahlt. Dies entspricht zwar auch nicht unbedingt den Qualitätsrichtlinien von vielen Suchmaschinen (z.B.: Google), wird aber eher toleriert als jene Webseiten, die durch Link Farmen gepusht werden. Diese werden als „Bestrafung“ oft komplett von den Suchergebnissen ausgeschlossen. [GOOGLE_3]

4.8.3. Wiki/Gästebuch/Blog/Forum Spam

Bei dieser Manipulationstechnik werden vor allem bei frei bearbeitbaren Webseiten wie Wikis, Gästebüchern, Blogs oder Foren Links zu bestimmten Webseiten hinzugefügt. Da diese bearbeitbaren Webseiten auch oft von den Webcrawlern der Suchmaschinen besucht werden, entdecken diese natürlich auch alle dort eingetragenen Verweise. Dadurch kann sich auch wieder der PageRank jener Webseite erhöhen, auf die diese Links verweisen.
Die Einträge selbst werden oft von Programmen oder „Robotern“ ohne wirklichen Inhalt oder mit irgendwelchen automatisch generierten Texten hinzugefügt.

Eine hier wichtige Unterscheidung ist jene, ob es sich um Spam auf einer bearbeitbaren Webseite handelt oder ob es sich um eine Spam-Webseite selbst handelt.
Bei Ersteren handelt es sich um reale Webseiten, wie eben z.B. Blogs, in denen einfach automatisch Links eingetragen werden. Bei den Spam-Webseiten handelt es sich eigentlich um keine „realen Webseiten“, sondern um automatisch generierte oder von anderen Webseiten kopierte Seiten. Diese dienen somit nur zur Verbreitung von Links.

Um dem Problem von Spam auf bearbeitbaren Webseiten entgegenzuwirken, haben die Betreiber von Wikis, Gästebücher, Blogs oder Foren sich zahlreiche Möglichkeiten einfallen lassen, um dieses Problem gleich direkt bei der Wurzel, also dem Spam-Eintrag selbst, zu vermeiden:

CAPTCHA

Dabei muss der Benutzer zum Absenden eines Beitrages noch ein zusätzliches Feld ausfüllen, welches testen soll, ob der Beitrag von einem Menschen oder einer Maschine abgesendet wird. Dieser Test wurde im CAPTCHA Projekt durch vier Arten implementiert [CAPTCHA_1]:

1. Gimpy

Dabei handelt es sich um ein dynamisch generiertes Bild, welches stark verzerrte Buchstaben und/oder Zahlenkombinationen enthält. Diese müssen dann in ein Textfeld eingegeben werden.

2. Bongo

Hier werden zwei Muster dargestellt, die sich in einer Charakteristik unterscheiden. Der Benutzer muss entscheiden, welches diese Charakteristik ist.

3. Pix

Es werden 4 verschiedene Bilder dargestellt, welche einem gemeinsamen Begriff oder Thema zugeordnet werden können. Der Benutzer muss aus einer Liste möglicher Begriffe den richtigen auswählen.

4. Sounds

Dabei wird eine zufällige Buchstaben und/oder Zahlenkombination in einer verzerrten Sound-Datei wiedergegeben. Diese Kombination muss dann vom Benutzer in ein Textfeld eingegeben werden.

Derzeit ist es bereits möglich, dass einige Roboter - vor allem bei manchen Formen von Gimpy - mit einer hohen Wahrscheinlichkeit diesen Test bestehen. Die Meisten dieser Tests können jedoch von Robotern noch nicht oder nur mit einer sehr geringen Wahrscheinlichkeit und Effizienz bestanden werden. Somit ist meist sichergestellt, dass Beiträge nur von Menschen abgesendet werden können.

Schlüsselwörter ausschließen

Einträge, welche bestimmte Schlüsselwörter, welche von den Spammern oft verwendet werden, enthalten, dürfen nicht eingetragen werden.

Neue Attribute für Links

Es wird ein neues Attribut zum HTML-Tag eines Links hinzugefügt. Dieser rel="nofollow"-Tag wurde von Google eingeführt und hat zur Folge, dass ein Link, der diesen Tag enthält, nicht zur PageRank-Berechnung herangezogen wird. Bei den bearbeitbaren Webseiten werden daher oft standardmäßig alle Links, welche von Benutzern in einem Beitrag eingetragen werden, mit diesem Attribut versehen.

Verbot von Links

Die sicherste Methode hierbei ist es, ganz einfach das Eintragen von Links generell zu verbieten. Dies hat aber den großen Nachteil, dass damit vor allem Wikis und Blogs einer großen Einschränkung unterliegen und für viele Benutzer unattraktiv werden.

4.8.4. Abgelaufene Domains kaufen

Dabei handelt es sich zwar nicht um eine direkte Manipulationsmethode des PageRank-Algorithmus, hat jedoch eine gewisse Verbindung dazu. Dabei wird beobachtet, welche Domains demnächst ablaufen werden. Hier hat man es natürlich vor allem auf Domains abgesehen, welche einen hohen PageRank haben. Sobald diese abgelaufen sind, z.B.: weil vom Inhaber der Domain die Verlängerung vergessen wurde oder nicht gewollt war, wird diese gekauft. Danach setzt man von dieser Seite Links zu einer oder mehrerer seiner eigenen Webseiten und profitiert damit von dem PageRank der alten Seite.

Dieser PageRank wird sich zwar langfristig nicht halten, kann jedoch kurzfristig trotzdem zu einer guten Position unter den Suchergebnissen beitragen.

5. Alternative Suchmaschinenbewertungen

Dieses Kapitel beschäftigt sich mit Suchmaschinenbewertungen, welche entweder alternativ zum PageRank-Algorithmus verwendet werden können, oder aufbauend auf

[PAGE_1] bzw. [PAGE_2] entwickelt wurden. Aufgrund dessen, das es heutzutage bereits eine sehr große Anzahl an möglichen Bewertungsmethoden gibt, werden hier nur einige exemplarische angeführt. Damit soll aufgezeigt werden, dass dieser Algorithmus in seiner klassischen Form noch viele Möglichkeiten zur Verbesserung bietet und die Suchmaschinenbewertung noch nicht ihr Optimum erreicht hat.

5.1. Gerichtetes Zufallssurfer Modell

Dabei handelt es sich um eine Weiterentwicklung des in Kapitel 3.4.1 vorgestellten Zufallssurfer Modells. Die Grundidee ist, das vorhandene Modell soweit abzuwandeln, dass sich die Wahrscheinlichkeit, mit der der Zufallssurfer einem Link folgt, in Abhängigkeit von der Relevanz des Links zu einer Suchanfrage ändert.
Dazu wurde folgende Formel aufgestellt [RICHARDSON_1]:

$$P_q(j) = (1-\beta)P_q^{'} + \beta \sum_{i \in B_j} P_q(i) P_q(i \rightarrow j)$$

$P_q(j)$ ist die Wahrscheinlichkeit, mit der ein Knoten besucht wird und ist in diesem Fall mit dem PageRank vergleichbar. Der Term $(1-\beta)P_q^{'}$ gibt die Wahrscheinlichkeit an, dass der Zufallssurfer auf der Webseite j einen anderen Knoten des Graphen besucht und keinem Link auf der Webseite j folgt. $P_q(i \rightarrow j)$ gibt die Wahrscheinlichkeit an, dass der Zufallssurfer einem Link von der Seite i zu j mit der Suchanfrage q folgt. B_j ist die Menge aller Webseiten, welche auf die Seite j zeigen.
$P_q^{'}$ und $P_q(i \rightarrow j)$ werden folgendermaßen berechnet [RICHARDSON_1]:

$$P_q^{'} = \frac{R_q(j)}{\sum_{k \in W} R_q(k)} \qquad P_q(i \rightarrow j) = \frac{R_q(j)}{\sum_{k \in F_i} R_q(k)}$$

$R_q(j)$ repräsentiert dabei die Relevanz der Seite j in Abhängigkeit der Suchanfrage q. W ist die Menge aller Knoten des Graphen. F_i stellt das Set aller Links auf Webseite i dar.

Da die Relevanz einer Seite für eine spezifische Suchanfrage nicht zur Zeit der Suchanfrage berechnet werden kann – dies würde mehrere Stunden benötigen -, ist es notwendig diese Relevanz davor zu berechnen und lokal zu speichern. Dies wird für jedes Wort q in einem Lexikon L = $\{q_1, q_2, \ldots, q_n\}$ gemacht.

Für alle in einer Suchanfrage vorkommenden Wörter wird der sogenannte QD-$\text{PageRank}_Q(j)$ (= Query Dependant PageRank = QD-PR) berechnet [RICHARDSON_1]:

$$\text{QD - PageRank}_{\text{Q}}(j) \equiv P(j) = \sum_{q \in Q} P(q) P_q(j)$$

Der QD-PR entspricht dabei der Wahrscheinlichkeit dass die Seite P(j) besucht wird. Q ist die Menge aller in einer Suchanfrage enthaltenen Wörter.

Es kann angenommen werden, dass jeder Suchanfrage ein Wort aus dem Lexikon zugeordnet werden kann. Sollte dies nicht der Fall sein, wird der normale PageRank zur Bewertung der Webseiten herangezogen.

Nach [RICHARDSON_1] liefert dieses Modell qualitativ bessere Ergebnisse als das Zufallssurfer Modell des klassischen PageRank-Algorithmus. Der Nachteil ist, dass dabei mehr Ressourcen benötigt werden:

So wurden für eine Datenbank mit 1.7 Millionen Webseiten etwa 2.3 Millionen einzigartige Wörter gefunden, von denen bereits ungefähr 100 der gebräuchlichsten Wörter (z.B.: Artikel und Bindewörter) entfernt wurden, da diese für die Suchanfrage unerheblich sind. Dies ergibt einen Speicherverbrauch, der 163-mal höher und einen Rechenzeit, die 124-mal höher ist, als jene um nur nach beliebigen Wörtern in diesem Set von Webseiten zu suchen.

5.2. Link Popularity

Die Link Popularity stellt sozusagen die Oberklasse vieler heute verwendeter Algorithmen zur Suchmaschinenbewertung dar. Hierbei ist eine Webseite umso wichtiger bzw. besitzt umso mehr Link Popularity, je mehr Webseiten auf diese Webseite verweisen. Folgende Formel kann dafür verwendet werden:

$$LP(j) = |I|$$

LP(j) ist die Link Popularity der Webseite j. I beschreibt eine Gruppe von Webseiten, die einen Link auf j besitzen.

Eine genaue Definition der Link Popularity ist nicht verfügbar. In dieser Form wird sie auch von keiner Suchmaschine angewendet, da diese Art der Bewertung von Webseiten sehr manipulationsanfällig ist. Dennoch wird die Grundidee, dass die Anzahl der auf eine Webseite zeigenden Links einen großen Einfluss auf dessen Rang hat, von vielen Algorithmen bei der Suchmaschinenbewertung genutzt. Ein Beispiel dafür ist der klassische PageRank-Algorithmus, dessen Formel jedoch zusätzlich noch weitere Faktoren berücksichtigt. Daher wird er auch oft als Algorithmus zur Bewertung der Link Popularity bezeichnet. Diese kann deshalb auch so verstanden werden, dass der Rang einer Webseite j nicht nur durch die Anzahl, sondern auch durch den Rang der auf j verweisenden Webseiten beeinflusst wird.

5.3. Click Popularity

Diese Art der Suchmaschinenbewertung reiht die Seiten basierend auf dem Verhalten der Benutzer einer Suchmaschine. Demnach ist eine Seite umso wichtiger, je mehr Leute diese nach einer bestimmten Suchanfrage anklicken. Ein weiteres Kriterium, das bei dieser Bewertung mit einfließt ist, wie lange der Benutzer auf der Seite, die er angeklickt hat verbleibt. Wenn er nur kurz auf dieser Seite bleibt, wird angenommen, dass für diese Seite für die Suchanfrage des Benutzers nur eine geringe Relevanz hatte. Die Zeit, die er auf einer Webseite verbringt, wird gemessen indem man die Zeit zwischen zwei Clicks auf

eine URL des Suchergebnisses oder zwischen dem Click auf ein URL bis zur Rückkehr auf die Suchergebnisseite aufzeichnet. Welche Seiten angeklickt wurden, wird mit Hilfe von Cookies gespeichert.

Die Grundidee, die hinter dieser Bewertungsmethode steht, ist durchaus vielversprechend: Man überlässt den Benutzer selbst die Bewertung der Webseiten. Dieser beeinflusst durch sein Verhalten die tatsächliche Relevanz einer Webseite in Abhängigkeit einer speziellen Suchanfrage. Bei dieser Bewertungsmethode kann es aber durchaus zu einigen Problemen kommen:
Das Verhalten der Benutzer ist oft nicht vorhersagbar. So kann z.B. ein Benutzer den Computer einfach verlassen oder aus Interesse auf einen Link zu einer Webseite klicken - und auch dort längere Zeit verharren - obwohl dieser mit seiner eigentlichen Suchanfrage wenig zu tun hat. Weiters müssen sich die Relevanz von Suchergebnissen bei gleichen Anfragen nicht unbedingt auf verschiedene Benutzer übertragen zu lassen. Wird ein Homonym wie beispielsweise „Leiter“ als Suchanfrage eingegeben, kann ein Benutzer damit einen elektrischen Leiter meinen, ein anderer Benutzer sucht vielleicht nach dem Gerät zum Hinauf- und Hinabsteigen oder einem Synonym für „Chef“. Eine Reihung aufgrund der Click Popularity würde hier falsche Ergebnisse liefern, da je nach der gemeinten Suchanfrage des Benutzers die Wichtigkeit der Webseiten anders bewertet wird und somit einzelnen Begriffen keine eindeutige Reihung zugeordnet werden kann.

Ferner ist zu beachten, dass alles, was ein einzelner Benutzer machen kann, normalerweise auch zur Manipulation von vielen Benutzern, Programmen oder Robotern gemacht werden kann. Dies hat zur Folge, dass die Bewertung nach Click Popularity grundsätzlich sehr manipulationsanfällig ist.

Das ist auch der Grund, dass diese Methode als einziges Kriterium zur Suchmaschinenbewertung heutzutage von keiner Suchmaschine verwendet wird. Jedoch verwenden manche Suchmaschinen dieses System als zusätzliches Bewertungskriterium zu einem anderen Hauptalgorithmus. Dies hat den Vorteil, dass der Einfluss der Click Popularity auf die Suchergebnisse einer Suchanfrage nicht einen zu großen Einfluss hat und sich deshalb eine Manipulation oft nicht lohnen würde.

5.4. HITS

Der HITS (Hypertext-Induced Topic Selection) Algorithmus wurde 1999 in [KLEINBERG_1] veröffentlicht. Ähnlich wie der PageRank Algorithmus baut die Suchmaschinenbewertung mittels HITS ebenfalls auf der Analyse der Linkstruktur auf. Der Grundgedanke dabei ist, dass man - in Abhängigkeit von einer bestimmten Suchanfrage - die besten Webseiten, die sich mit dem Thema der Suchanfrage befassen, findet. Dabei versucht man immer eine gewisse Anzahl an Webseiten zu einem bestimmten Thema zusammenzufassen. Die wichtigsten Seiten aus diesem Thema werden dann dem Suchenden als Ergebnis präsentiert.

Dazu seien zuerst zwei Begriffe diniert:

- Authorities: Dabei handelt es sich um Webseiten, die von sehr vielen anderen Webseiten verlinkt wurden.
- Hubs: Dies sind Webseiten, welche zu besonders vielen, mit einem bestimmten Thema verwandten, Authorities Links besitzen.

Das Ranking der Ergebnisses einer Suchanfrage wird durch die Beziehung zwischen Hubs und Authorities berechnet. Man versucht dabei dem Benutzer wenige, aber dafür sehr gute Ergebnisse zu liefern. Dazu wird in einem ersten Schritt die Anzahl der Suchergebnisse reduziert.

Bei diesem Verfahren legt man wieder dem WWW eine Graphenstruktur zugrunde, bei der die Webseiten als Knoten und die Links zwischen den Webseiten als gerichtete Kanten definiert sind. Da man jedoch nur eine geringe Anzahl an Links im Suchergebnis haben möchte, wird nicht der gesamte Graph des WWW, wie es beispielsweise beim PageRank-Algorithmus gemacht wird, betrachtet. Für eine bestimmte Suchanfrage mit dem Text σ wird ein Subgraph des WWW gebildet. Dieser könnte beispielsweise aus einer Gruppe von Webseiten Q_σ, in denen der Text σ vorkommt, bestehen. Diese Gruppe kann jedoch unter Umständen wieder sehr viele Webseiten enthalten. Weiters werden oft die für eine bestimmte Suchanfrage relevantesten Webseiten nicht gefunden, da sie σ gar nicht enthalten. Ein Beispiel wäre ein Automobilhersteller wie Mercedes Benz: Sucht man nach dem Text „Automobilhersteller“ wird die Webseite von Mercedes Benz nicht in Q_σ

enthalten sein, da sie selbst diesen Text nicht auf ihrer Webseite hat. Daher wurden in [KLEINBERG_1] drei Eigenschaften definiert, die eine Webseitengruppe S_σ erfüllen muss:

(i) S_σ soll ziemlich klein sein
(ii) S_σ hat enthält eine große Anzahl relevanter Webseiten
(iii) S_σ enthält sehr viele der stärksten Authorities

Um den Subgraph S_σ zu bekommen, werden zuerst die t (t sollte dabei eher klein gewählt werden, z.B.: 200) am höchsten gelisteten Links des Suchergebnisses einer text-basierten Suchmaschine als Basis R_σ genommen. Alle in R_σ enthaltenen Webseiten enthalten den Suchtext σ. Dadurch sind die Bedingungen (i) und (ii) für den Subgraphen S_σ bereits erfüllt. Wie bereits in dem obigen Beispiel beschrieben, müssen in diesem Set jedoch nicht alle, oder gar die wichtigsten Authorities, enthalten sein. Zur Kompensation dieses Problems werden nun zu S_σ all jene Webseiten hinzugefügt, welche mindestens einen Link auf eine Webseite in R_σ besitzen. Man vertraut also darauf, dass die Webseiten in R_σ aufgrund des Vorkommens des Suchtexts σ Links auf die wichtigsten Authorities besitzen, die sich mit dem Thema des Suchtextes beschäftigen.

Der Algorithmus für das Hinzufügen der Authorities zu S_σ lautet formal folgendermaßen:

```
For each website p in Rσ
  Sσ := Sσ + Γ+(p)
  If |Γ-(p)| ≤ d
    Sσ := Sσ + Γ-(p)
  Else
    Sσ := Sσ + {d beliebige Elemente aus Γ-(p)}
End
```

Algorithmus 1: Hinzufügen von Authorities bei HITS

Für jede Webseite p in R_σ fügt man zu S_σ zuerst alle ausgehenden Links $\Gamma^+(p)$ dieser Webseite p hinzu. Ist die Anzahl der auf p zeigenden Links $\Gamma^-(p)$ kleiner als ein selbst gewählter Parameter d (z.B.: mit dem Wert 50 [KLEINBERG_1]), werden diese ebenfalls zu S_σ hinzugefügt. Ist dies nicht der Fall, werden d beliebige Webseiten aus $\Gamma^-(p)$ zu S_σ hinzugefügt. Der Grund dafür ist, dass viele Webseiten von tausenden anderen Webseiten

referenziert werden. Will man nun Bedingung (i) erfüllen, also dass der Subgraph klein sein soll, können diese Webseiten nicht alle zu S_σ hinzugefügt werden.

Anzumerken ist auch, dass dieser Algorithmus zwischen zwei Arten von Links unterscheidet:

- Transverse Links: Dabei handelt es sich um Links, welche zwischen Webseiten verschiedener Domain Names (oberster Level der URL) bestehen.
- Intrinsic Links: Diese Art von Links bestehen zwischen Webseiten des gleichen Domain Namens. Oft handelt es sich dabei um Navigationslinks zwischen verschiedenen Subseiten oder andere Links, welche sich auf jeder Seite einer Homepage befinden, z.B.: Link zu Copyrighthinweisen.

Bei diesem Algorithmus werden lediglich die Transverse Links beachtet. Eine andere Möglichkeit wäre auch, dass man eine geringe Anzahl (zwischen 4 und 8) von Intrinsic Links für Seiten des gleichen Domain Namens zulässt. Dies wurde jedoch in [KLEINBERG_1] nicht weiter getestet.
Dadurch sind nun alle drei Anforderungen an den Subgraphen G, welcher nun als Knoten jene Webseiten in S_σ enthält, erfüllt.

Zur Berechnung der Ränge der Hubs und Authorities:
Ein Ansatz wäre nach [KLEINBERG_1] einfach die Anzahl der eingehenden Links zu zählen. Dies könnte in diesem Fall als Maß für die Relevanz einer Seite gesehen werden, da sich diese nur auf den Subgraphen S_σ bezieht, und nicht auf alle Webseiten, die den Suchtext σ enthalten. Das Problem dabei ist, dass man für Homonyme eine kontext-falsche Bewertung erhält. So wurde in [KLEINBERG_1] der Suchtext „Java“ als Beispiel angegeben: Dies lieferte zwar gute Suchergebnisse für Seiten, die sich mit der Programmiersprache Java beschäftigen. Es wurden jedoch auch Seiten, deren Inhalt sich mit der Insel Java beschäftigt, sehr weit oben gereiht.
Die Lösung für dieses Problem ist, dass man Hubs, welche auf mehrere gemeinsame Authorities zeigen, dasselbe Themengebiet zuordnet.

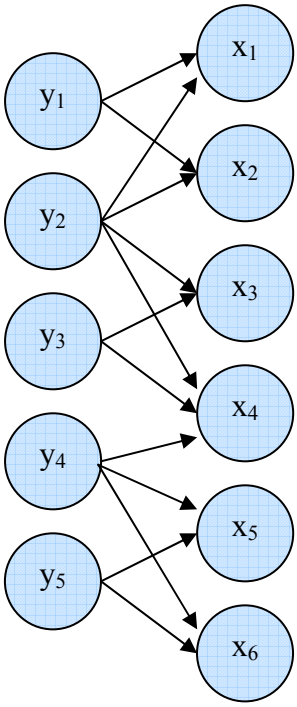

Abbildung 9: Hubs und Authorities zweier Themengebiete

Wie in Abbildung 9 dargestellt, verweisen die Knoten y_1, y_2 und y_3 zu denselben Authorities x_1, x_2, x_3 und x_4. Die Knoten y_4 und y_5 verweise auf die Knoten x_5 und x_6. Daher kann angenommen werden, dass die Hubs y_1, y_2 und y_3 bzw. y_4 und y_5 jeweils Links zu den gleichen Themengebieten haben und somit zwei Themengruppen bilden.

Nun, da klargestellt ist wie man die Suchergebnisse auch kontext-spezifisch sortieren kann, muss man sich überlegen, was überhaupt einen guten Hub bzw. eine gute Authority ausmacht. Hierzu definiert [KLEINBERG_1] dies rekursiv so: Ein guter Hub ist jener, der auf viele gute Authorities verweist. Eine gute Authority ist jene, auf die von vielen guten Hubs verwiesen wird.
Jeder Webseite im Graphen werden nun zwei Gewichte zugeordnet: $x^{<p>}$ („authority weight“) und $y^{<p>}$ („hub weight“). Durch die Normalisierung dieser Gewichte gilt stets:

$$\sum_{p \in S\sigma} (x^{<p>})^2 = \sum_{p \in S\sigma} (y^{<p>})^2 = 1$$

Dabei gilt, dass umso besser der x- und y-Wert einer Webseite sind, umso höher ist auch ihr Rang.

Für die Beziehung zwischen x- und y-Werten gilt folgendes:

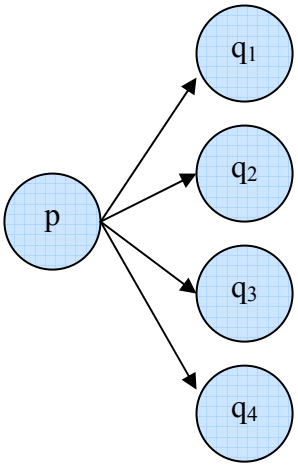

Abbildung 10: Einfluss der x-Werte auf Hubs

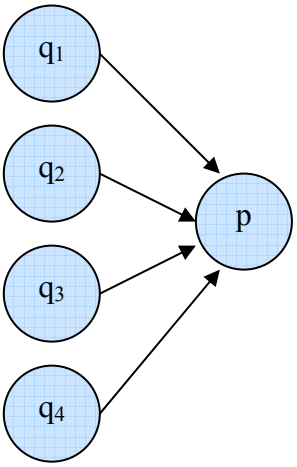

Abbildung 11: Einfluss auf y-Werte auf Authorities

In Abbildung 10 berechnet sich der y-Wert der Webseite p aus der Summe der x-Werte von q_1, q_2, q_3 und q_4. Umgekehrt wird der x-Wert einer Webseite p (siehe Abbildung 11) aus der Summe der y-Werte von q_1, q_2, q_3 und q_4 berechnet. Diese zwei Aussagen lassen sich zu folgenden Operationen $\mathcal{I}$ und $\mathcal{O}$ für die Berechnung von $x^{<p>}$ und $y^{<p>}$ formulieren:

$$\mathcal{I}: x^{<p>} \leftarrow \sum_{q:(q,p)\in E} y^{<q>} \qquad \mathcal{O}: y^{<p>} \leftarrow \sum_{q:(p,q)\in E} x^{<q>}$$

$x^{<p>}$ ist demnach die Summe aller $y^{<q>}$, von denen eine Webseite q einen Link auf die Webseite p hat. $y^{<p>}$ ist die Summe aller $x^{<q>}$, jener Webseiten auf die die Webseite p einen Link hat. Mit Hilfe des folgenden Algorithmus werden nun die x- und y-Werte jeder Webseite berechnet [KLEINBERG_1]:

```
For i = 1 .. k
  xi' := I(xi-1,yi-1)
  yi' := O (xi',yi-1)
  xi := Normalize(xi')
  yi := Normalize(yi')
End
```

Algorithmus 2: Berechnung der x- und y-Werte bei HITS

Als Parameter müssen der Subgraph G, welcher aus n verbundenen Knoten besteht, und eine natürliche Zahl k, welche die Anzahl der Iterationen für die Berechnung bestimmt, angegeben werden. Für die Zahl k reicht meist der Wert 20 aus, damit die Berechnung konvergiert oder zumindest ein durchaus brauchbares Ergebnis liefert. Die Variablen x und y stehen hier für Vektoren, in denen für jeden Knoten in S_σ der $x^{<p>}$ bzw. $y^{<p>}$-Wert gespeichert ist. Als initialen Wert wird für jeden Knoten 1 angenommen.

Es werden nun in jeder Iteration durch die Operationen $\mathcal{O}$ und $\mathcal{I}$ die Werte $x^{<p>}$ und $y^{<p>}$ aller Webseiten normalisiert berechnet. Aus dem Ergebnis dieser Berechnung können nun die c besten Hubs und die c besten Authorities gefiltert werden. Der Wert c wird dabei üblicherweise mit $c \approx 5\text{-}10$ angegeben.

Ein weiteres interessantes Anwendungsgebiet dieses Algorithmus ist das Suchen ähnlicher Seiten. Dabei sucht man nicht nach Webseiten, die Authorities für einen Suchtext σ darstellen, sondern nach Authorities, welche sich mit dem gleichen oder einem ähnlichen Themengebiet wie eine Webseite p befasst. Man sucht daher zuerst nach einem Set von t Webseiten, welche auf p einen Link haben. Diese t Webseiten werden wieder zu einer Basis R_p zusammengefasst. Diese wird dann wieder zu einer Webseitengruppe S_p erweitert, welche nun auch alle Links der in R_p befindlichen Webseiten enthält. Damit kann man wieder einen Subgraphen konstruieren, indem man genauso wie in dem oben beschriebenen Algorithmus nach Hubs und Authorities sucht.

Abschließend kann man sagen, dass dieser Algorithmus einen interessanten Ansatz zur Suchmaschinenbewertung darstellt, jedoch auch einige Probleme mit sich bringt:
Dadurch, dass die Hub- bzw. Authoritywerte in wechselseitiger Abhängigkeit stehen, ist der Algorithmus für Manipulationen anfällig. Weiters muss für jede einzelne Suchanfrage ein Subgraph erstellt werden, was unter Umständen zu einem hohe Rechenaufwand oder einer langen Wartezeit für den Benutzer führt.

5.5. SALSA

Dieser Algorithmus wurde 2000 in [LEMPEL_1] vorgestellt, und stellt gewissermaßen eine Verschmelzung des PageRank- und des HITS-Algorithmus dar. Die Idee dabei ist, dass man zuerst eine Menge von Webseiten findet, die sich mit einem bestimmten Thema beschäftigen. Danach soll die Wahrscheinlichkeit berechnet, wie oft ein Zufallssurfer eine Webseite in dieser Menge besuchen würde. Daraus kann man folgern, dass eine Webseite umso mehr Gewicht hat, je öfter sie der Zufallssurfer besucht bzw. umso leichter sie erreichbar ist. Eine Webseite mit hoher Relevanz ist demnach jene, die für den Zufallssurfer im Kontext eines bestimmten Themas unübersehbar ist.

Wie bei HITS wird auch bei SALSA (Stochastic Approach for Link Structure Analysis) zu Beginn ein Set C von n Webseiten angelegt. Dieses Set enthält dabei ausschließlich Webseiten, welche sich mit dem Thema t einer Suchanfrage beschäftigen. In C sollen möglichst viele Hubs und Authorities vorhanden sein, die relevant für das Thema t sind und wenige Hubs und Authorities, welche sich mit einem anderen Thema t' beschäftigen.
Ein Hub ist bei diesem Algorithmus eine Webseite, die mindestens einen ausgehenden Link hat. Eine Authority ist eine Webseite, die mindestens einen eingehenden Link hat.

An dieser Stelle wird nun das Zufallssurfer Modell des PageRank-Algorithmus angewendet. Dieses wird jedoch, im Gegensatz zu dem des PageRank-Algorithmus, nicht auf das ganze WWW, sondern eben nur auf C angewendet. Ein weiterer Unterschied ist, dass bei SALSA in jedem Schritt zwei Zufallswege gegangen werden, im Gegensatz zum PageRank-Algorithmus, bei welchem in jedem Schritt nur ein Zufallsweg eingeschlagen wird. Bei SALSA führt ein Weg immer von einem Hub zu einer Authority und von dieser wieder zu einem Hub, et vice versa.

Für die Berechnung wird nun aus C ein ungerichteter bipartiter Graph [MUEHLBACHER_1] $\tilde{G} = (V_h, V_h, E)$ gebildet:

$$V_h = \{s_h \mid s \in C \text{ and out-degree}(s) > 0 \}$$

V_h beschreibt dabei die Hub-Seite von $\tilde{G}$ und enthält alle Webseiten aus C, die mindestens einen ausgehenden Link haben.

$$V_a = \{s_a \mid s \in C \text{ and in-degree}(s) > 0 \}$$

V_a beschreibt dabei die Authority-Seite von $\tilde{G}$ und enthält alle Webseiten aus C, die mindestens einen eingehenden Link haben.

$$E = \{(s_h, r_a) \mid s \longrightarrow r \in C\}$$

E repräsentiert die Menge aller ungerichteter Kanten, welche zwischen den Webseiten in C bestehen.

Um die Umwandlung von C in einen in den Graphen $\tilde{G}$ zu verdeutlichen, wird der aus C aufgebaute in Abbildung 12 dargestellte Graph in den ungerichteten bipartiten Graphen $\tilde{G}$ in Abbildung 13 umgewandelt:

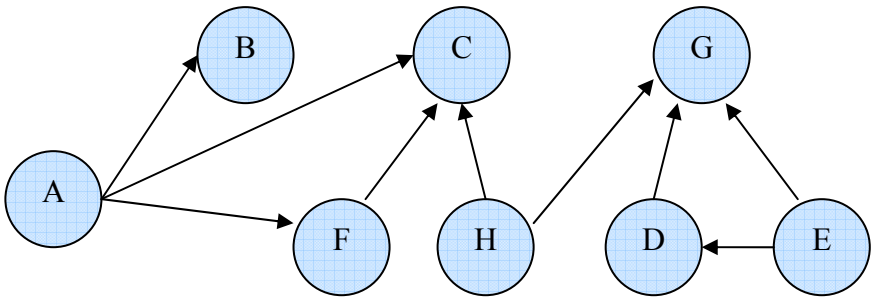

Abbildung 12: Gerichteter Graph zu Beginn des SALSA-Algorithmus

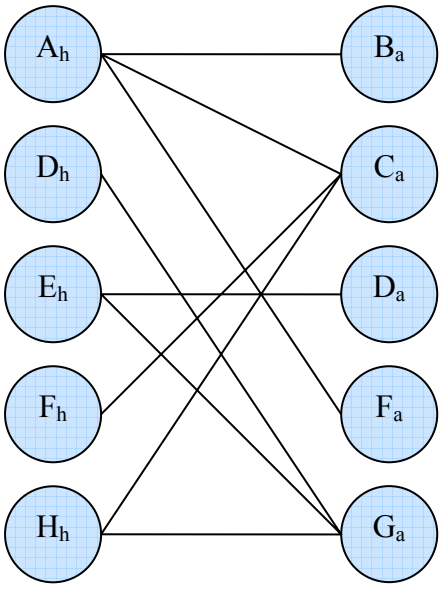

Abbildung 13: Ungerichteter bipartiter Graph des SALSA-Algorithmus

Die beiden Zufallssurfer starten jeweils auf einer Seite des bipartiten Graphen und machen in jedem Schritt einen zufälligen Übergang auf die andere Seite bzw. wieder zurück auf die ursprüngliche Seite. Da es bei bipartiten Graphen definitionsgemäß zwischen den Knoten einer Seite keine Kanten gibt, erreichen die Zufallssurfer nach jedem zweiten Schritt stets dieselbe Seite, auf der Sie ursprünglich waren.

Als beste Authorities für ein Thema t sind jene Knoten im Graphen definiert, welche vom Zufallssurfer, der auf der Authority-Seite gestartet ist, am öftesten besucht wurden. Als beste Hubs für ein Thema t sind jene Knoten im Graphen definiert, welche vom Zufallssurfer, der auf der Hub-Seite gestartet ist, am öftesten besucht wurden.
Die Zustandsübergänge der Zufallssurfer werden mathematisch als zwei Markow-Ketten modelliert: Eine Kette für die Zustandsübergänge der auf der Authority-Seite, eine für die Zustandsübergänge auf der Hub-Seite.

Es werden für die beiden Markow-Ketten zwei stochastische Übergangsmatrizen definiert:

$$\tilde{h}_{i,j} = \sum_{\{k|(i_h,k_a),(j_h,k_a)\in\tilde{G}} \frac{1}{\deg(i_h)} \frac{1}{\deg(k_a)}$$

$$\tilde{a}_{i,j} = \sum_{\{k|(k_h,i_a),(k_h,j_a)\in\tilde{G}} \frac{1}{\deg(i_a)} \frac{1}{\deg(k_h)}$$

Ein Eintrag $\tilde{h}_{i,j}$ in der Übergangsmatrix der Hubs enthält demnach die Wahrscheinlichkeit, dass man in zwei Schritten von eine Hub-Knoten i_h über einen Authority-Knoten k_a zu einem Hub-Knoten j_h gelangen kann. Es wird also die Wahrscheinlichkeit berechnet, dass man zuerst vom Knoten i_h eine Kante zum Knoten k_a wählt und vom Knoten k_a zum Knoten j_h eine Kante zum Knoten j_h wählt. Diese Erklärung gilt auch für die Einträge in der Übergangsmatrix der Authority-Seite sinngemäß.

Das Ergebnis dieser Matrix sind jene k Seiten des Eigenvektors jeder Matrix, die darin die höchsten Einträge haben.

Eine Weiterentwicklung dieser Berechnung wäre, eine Gewichtung der Links zwischen den Webseiten vorzunehmen. Die vorherige Beschreibung des SALSA-Algorithmus bezieht sich ausschließlich auf ungewichtete Links.

Mögliche Gewichtungsfaktoren könnten laut [LEMPEL_1] sein:

- Der Text eines Ankers, der für die Suchanfrage relevant ist.
- Der Benutzer selbst kann bestimmen, dass ein Link zu einer Webseite für ein bestimmtes Thema eine hohe Relevanz hat. Links, welche auf gute Authorities verweisen, sollen ein höheres Gewicht bekommen. Links, welche von guten Hubs weggehen, sollen ebenfalls höher gewichtet werden.
- Die Position des Links auf einer Webseite. So wird von vielen Suchmaschinen angenommen, dass Links, die sich weiter oben auf einer Webseite befinden, mehr über ihren Inhalt reflektieren, als Links, die sich weiter unten befinden. Deshalb sollten diese oberen Links auch eine höhere Gewichtung bekommen.

Es wird nun ein Graph G = (H; A; E) als positiv gewichteter, gerichteter bipartiter Graph definiert, der aus einer Menge H von Hubs, einer Menge A von Authorities und einer Menge E von Kanten besteht. Alle Kanten in E sind von Knoten in H zu Knoten in A gerichtet. Die Gewichtung der Kanten wird formal wie folgt definiert:

$$d_{in}(i) = \sum_{\{k \in H | k \to i\}} w(k \to i)$$

Dabei gibt d_{in} den gewichteten Eingangsgrad eines Knoten $i \in A$ an. Es werden dafür die Gewichte aller Kanten, welche vom Knoten $k \in H$ zum Konten i führen, summiert.

$$d_{out}(k) = \sum_{\{i \in A | k \to i\}} w(k \to i)$$

Dabei gibt d_{out} den gewichteten Ausgangsgrad eines Knoten $k \in H$ an. Es werden dafür die Gewichte aller Kanten, welche vom Knoten h zu Knoten $i \in A$ führen, summiert.

Dabei gibt d_{out} die Gewichtung eines Links auf einer Webseite $k \in H$ an. Es werden die Gewichte aller Links $k \longrightarrow i$, zu welcher die Webseite k einen Link auf die Webseite i hat summiert.

$$W = \sum_{i \in A} d_{in}(i) = \sum_{k \in H} d_{out}(k)$$

Die Summe W aller Kantengewichte ist dabei gleich der Summe der Kantengewichte der eingehenden und der ausgehenden Kantengewichte.

Man kann nun wieder zwei Übergangsmatrizen der Markow-Ketten für die Hub- und die Authority-Seite bilden [LEMPEL_1]:

$$P_A(i,j) = \sum_{\{k \in H | k \to i, k \to j\}} \frac{w(k \to i)}{d_{in}(i)} \frac{w(k \to j)}{d_{out}(k)}$$

$$P_H(i,j) = \sum_{\{i \in A | k \to i, l \to i\}} \frac{w(k \to i)}{d_{out}(k)} \frac{w(l \to i)}{d_{in}(i)}$$

Die Wahrscheinlichkeit P_A, gibt an, wie wahrscheinlich ein Übergang vom Knoten $i \in A$ zum Knoten $j \in A$ ist. Dabei wird das Gewicht der Kante zwischen $k \in H$ und j auf das gesamte Eingangsgewicht des Knoten i verteilt und mit dem Gewicht der Kante zwischen k und j, welches auf das gesamte Ausgangsgewicht des Knoten k verteilt wird, multipliziert. Analog gilt dies für die Wahrscheinlichkeit $P_H(i,j)$.

Um nun ein Ranking der Webseiten zu erhalten, muss man nur die Summe der Eingangs- und Ausgangsgewichte jedes Knoten berechnen. Diese Methode hat auch den Vorteil, dass man kein iteratives Berechnungsverfahren benötigt.

5.6. Weighted PageRank

Bei diesem in [XING_1] vorgestellten Algorithmus handelt es sich um eine Mischung aus dem klassischen PageRank Algorithmus und dem HITS Algorithmus. Dabei sind für die Popularität einer Webseite u nicht nur die Anzahl der ausgehenden Links einer anderen Webseite v, welche auf u verlinkt, ausschlaggebend, sondern direkt auch die Anzahl der eingehenden Links von v. Es wird also mehr Wert auf die Anzahl der eingehenden Links der auf u verweisenden Webseiten gelegt. Dass dies vom klassischen PageRank oft übersehen wird lässt sich anhand eines Beispiels zeigen: Eine Webseite, welche sehr viele Links zu einem bestimmten Thema enthält, kann manchmal einfach aufgrund genau dieser großen Anzahl an Links keinen sehr hohen PageRank weitergeben. Da diese Seite aber durchaus sehr populär sein kann, weil sie auch auf viele andere populäre Webseiten zeigt, sollte sie auch dementsprechend gereiht werden.

Man definiert hier zuerst die Gewichte für einen Link zwischen zwei Seiten v und u [XING_1]:

$$W^{in}_{(v,u)} = \frac{I_u}{\sum_{p \in R(v)} I_p} \qquad W^{out}_{(v,u)} = \frac{O_u}{\sum_{p \in R(v)} O_p}$$

$W^{in}_{(v,u)}$ gibt dabei das Gewicht eines Links zwischen v und u durch die Anzahl der eingehenden Links von u I_u und die Anzahl der eingehenden Links I_p aller Webseiten R(v), auf welche v einen Link hat, an.

$W^{out}_{(v,u)}$ gibt dabei das Gewicht eines Links zwischen v und u durch die Anzahl der ausgehenden Links von u O_u und die Anzahl der ausgehenden Links O_p aller Webseiten R(v), auf welche v einen Link hat, an.

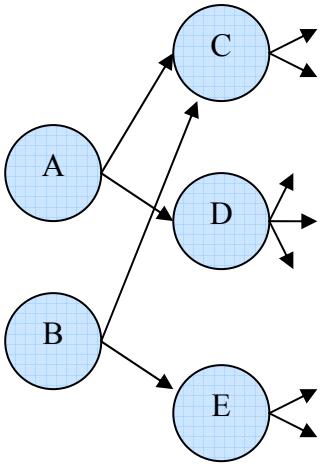

Abbildung 14: Beispielgraph zur Linkgewichtung des WPR-Algorithmus

Bei diesem in Abbildung 14 dargestellten Beispielgraphen würden sich folgende Gewichte der Links von A und B ergeben:

$$W_{(A,C)}^{in} = \frac{2}{2+1} = \frac{2}{3} \qquad W_{(A,C)}^{out} = \frac{2}{2+3} = \frac{2}{5}$$

$$W_{(A,D)}^{in} = \frac{1}{2+1} = \frac{1}{3} \qquad W_{(A,D)}^{out} = \frac{3}{2+3} = \frac{3}{5}$$

$$W_{(B,C)}^{in} = \frac{2}{2+1} = \frac{2}{3} \qquad W_{(B,C)}^{out} = \frac{2}{2+2} = \frac{1}{2}$$

$$W_{(B,E)}^{in} = \frac{1}{2+1} = \frac{1}{3} \qquad W_{(B,E)}^{out} = \frac{2}{2+2} = \frac{1}{2}$$

Wobei $I_A = 0$, $I_B = 0$, $I_C = 2$, $I_D = 1$, $I_E = 1$, $O_A = 2$, $O_B = 2$, $O_C = 2$, $O_D = 3$ und $O_E = 2$ sind.

Der Weighted PageRank (WPR) einer Webseite wird dann mit Hilfe folgender abgewandelter Formel des PageRank-Algorithmus berechnet [XING_1]:

$$PR(u) = (1-c) + c \sum_{v \in B_v} PR(v) W_{(v,u)}^{in} W_{(v,u)}^{out}$$

Würde man nun für den Graphen aus Abbildung 14 in diese Formel einsetzen, würde sich folgendes Ergebnis ergeben:

PR(A) = 0,15 = 0,17177212%

PR(B) = 0,15 = 0,17177212%

PR(C) = 0,2265 = 0,25937589%

PR(D) = 0,1755 = 0,20097338%

PR(E) = 0,17125 = 0,1961065%

Mit dem klassischen PageRank-Algorithmus würde man folgende PageRanks erhalten:

$$PR(A) = 0,15 = 0,149253731\%$$

$$PR(B) = 0,15 = 0,149253731\%$$

$$PR(C) = 0,2775 = 0,276119403\%$$

$$PR(D) = 0,21375 = 0,212686567\%$$

$$PR(E) = 0,21375 = 0,21\%$$

Vergleicht man nun die beiden Ergebnisse, ist ersichtlich, dass zwar dieselbe Reihung vorliegt, die Webseite A und B jedoch prozentuell einen höheren PageRank erhalten.

Dieser Algorithmus findet laut [XING_1] etwas mehr relevante Webseiten, als es der ursprüngliche PageRank Algorithmus kann und bringt somit eine kleine Verbesserung mit sich. Dabei wurde die Relevanz einer Webseite in Abhängigkeit von der Suchanfrage von einer bestimmten Personengruppe bestimmt. Zur genaueren Evaluierung des WPR-Algorithmus sei auf [XING_1] verwiesen.

6. Die Visualisierung des PageRank-Algorithmus in einem Java Applet

Dieses Kapitel beschäftigt sich mit der Entwicklung des Java Applets, bei dem die Berechnung der PageRanks von Webseiten visualisiert wird.

6.1. Implementierung

Dieses Unterkapitel beschäftigt sich mit der Implementierung des Applets selbst und der dabei verwendeten Komponenten. Das Applet wurde mit der Syntax von Java 1.4 entwickelt. Obwohl bei der Implementierung bereits Java in der Version 1.6 verfügbar war, wurde aus Rückwärts-Kompatibilitätsgründen zu der noch immer sehr weit verbreiteten Version von Java gewählt. Ein weiterer Grund war, dass das Framework, in welches das gesamte Applet eingebettet werden sollte, ebenfalls in dieser Version verfasst wurde.

Bei diesem Framework handelt es sich um das Institut FIM entwickelte IT-Math Framework, welches zur *„einheitlichen Visualisierung von und Interaktion mit Algorithmen im E-Learning Bereich dient [FIM_1]“.*

Das zweite bei der Entwicklung des Applets verwendete Framework war JGraph. Dieses Open Source Framework, welches unter der GNU Lesser General Public License LGPL lizenziert ist, dient zur einfachen Visualisierung von und dem Arbeiten mit Graphen.

6.1.1. Das IT-Math Framework

Dieses Framework wurde vorwiegend zur Entwicklung von E-Learning Materialien entworfen. Es soll damit eine Möglichkeit geschaffen werden, Algorithmen oder mathematische Berechnungen einheitlich und anschaulich darzustellen. Ein Beispiel dafür sind die Lernmaterialien in [MUEHLBACHER_1].

6.1.1.1. Die Komponenten des IT-Math Framework

Ein didaktischer Aspekt hinter diesem Framework ist die uniforme Darstellung von Lernmaterialen. So enthält ein mit diesem Framework entwickeltes Applet folgende Komponenten [FIM_1]:

Die Einleitung

Diese wird als erste Seite des Applets dargestellt. Auf ihr werden theoretische Grundlagen, die Ziele und die Handhabung des Applets erklärt. Dadurch können die Lernenden entscheiden, ob sie die eigentliche Visualisierung starten möchten oder nicht. Weiters dient dies für eine problemlose und informierte Handhabung des Applets. Die eigentliche Visualisierung erfolgt in einem separaten Fenster.

Der Hilfe Text

Jedes Applet muss eine ausführliche Hilfe enthalten. Diese gliedert sich einerseits in Hilfe für die Bedienung des Applets selbst, andererseits in die Theorie, die hinter dem im Applet visualisierten Algorithmus steht. Diese beiden Hilfe-Dateien sind als HTML-Dateien verfasst und ermöglichen so eine ansprechende Darstellung der Hilfe.

Ein einfaches und generelles Layout

Dieses Layout bezieht sich auf die Visualisierung und Berechnung des Algorithmus selbst. In einer Werkzeugleiste kann der Ablauf (Start, Pause, Stopp) gesteuert werden und es kann auch die Geschwindigkeit, mit der die Visualisierung durchgeführt wird, skaliert werden. Weiters wird in einer Fortschrittsanzeige der prozentuelle Fortschritt oder, für Algorithmen mit einer ungewissen Anzahl an Schritten, eine andauernde Bearbeitungsanzeige, dargestellt.

Dieses generelle Layout gewährleistet auch, dass Lernende mit der Handhabung der Visualisierung in einem Applet, egal welche Komplexität der dargestellte Algorithmus aufweist, vertraut sind (siehe Abbildung 15).

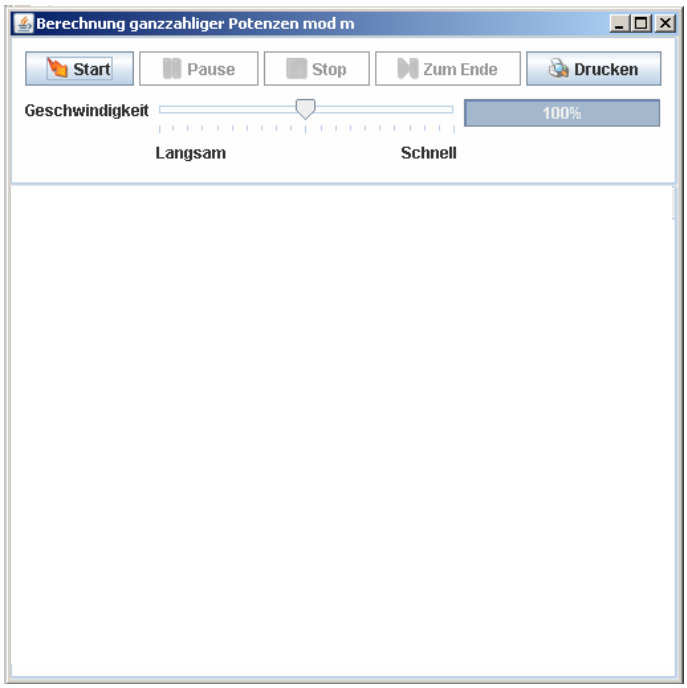

Abbildung 15: Layout eines Applets des IT-Math Frameworks

6.1.1.2. Die Entwicklung eines Applets mit dem IT-Math Framework

Die Entwicklung eines Applets mit Hilfe dieses Frameworks gestaltet sich ebenfalls sehr komfortabel. Es gibt insgesamt 4 Klassen, welche implementiert werden müssen. Dies gewährleistet auch, dass das Applet nach dem Model-View-Controller Architekturmuster aufgebaut ist:

Die Feature-Klasse

Diese enthält die bereits vorhin erwähnte Einleitung. Weiters wird aus dieser Klasse die Visualisierung gestartet.

Die Algorithm-Klasse

In dieser Klasse finden alle eigentlichen algorithmischen Berechnungen, die für die Visualisierung benötigt werden, statt. Die Berechnung selbst wird immer in diskreten Schritten ausgeführt.

Die View-Klasse

Diese Klasse dient zur visuellen Darstellung des Algorithmus bzw. dessen Berechnung. Es werden also die in der Algorithm-Klasse generierten Ergebnisse visualisiert.

Die Model-Klasse

Diese Klasse dient als Bindeglied zwischen der Algorithm- und der View-Klasse. Die Algorithm-Klasse sendet Daten an das Model. Die View-Klasse registriert diese Änderungen und ändert daraufhin die Visualisierung.

6.1.2. Das JGraph Framework

Dieses frei verfügbare Framework ermöglicht es, einen Graphen sehr einfach zu erzeugen, darzustellen, zu bearbeiten und zu analysieren. In der dafür geschaffenen API sind zahlreiche Methoden enthalten, die dies ermöglichen.

Die Konstruktion des Graphen funktioniert nach dem Model-View-Controller Architekturmuster. Die grafische Darstellung selbst kann sehr individuell angepasst

werden. So kann beispielsweise das Aussehen eines Knoten und der Kanten beliebig gestaltet werden.

Für die Bearbeitung des Graphen wurden ebenfalls viele Möglichkeiten geschaffen. Beispielsweise kann der gesamte Graph oder einzelne darin enthaltene Knoten einfach mit der Maus positioniert werden. Weiters stehen natürlich auch zahlreiche Operationen wie etwa das Hinzufügen, Löschen, Bearbeiten, Markieren, Skalieren oder Umbenennen einzelner Knoten und Kanten, als auch des gesamten Graphen, zur Verfügung.

6.1.3. Die Implementierung des Graphen Panels

Mit Hilfe des JGraph Frameworks wird auf einem ersten Panel ein Graph konstruiert, dessen Knoten einzelne Webseiten darstellen. Die gerichteten Kanten zwischen den einzelnen Knoten im Graphen repräsentieren die Links zwischen den Webseiten. Verlinken sich zwei Webseiten gegenseitig, wird dies als ein einziger Pfeil mit zwei Pfeilspitzen dargestellt (siehe Abbildung 16).

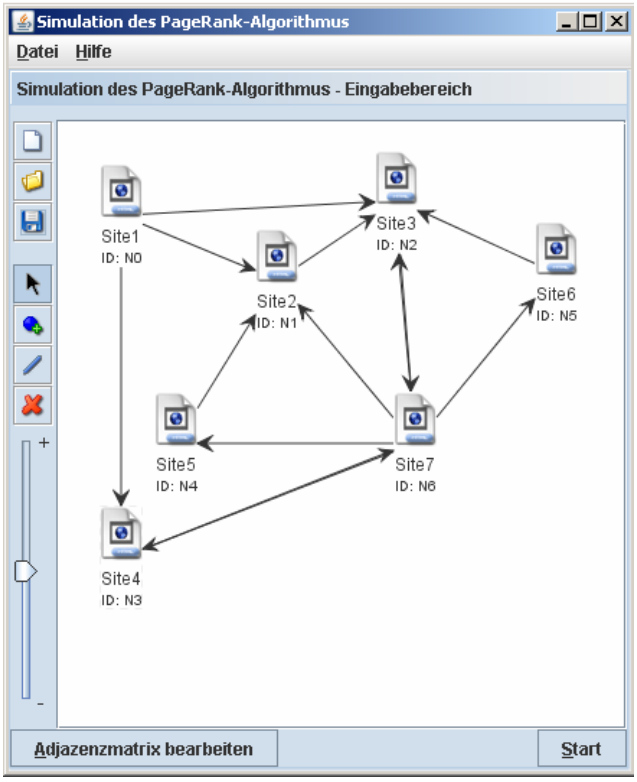

Abbildung 16: Darstellung des Graphen im Applet

6.1.3.1. Die Toolbar im Graphen Panel

Mit Hilfe einer Toolbar ist es möglich den Graphen zu bearbeiten. Es werden dazu folgende Operationen angeboten, die nach Auswahl in der Toolbar ausgeführt werden:

Einzelne Knoten des Graphen markieren und verschieben

Dabei können mit der Maus ein einzelner, oder durch gedrückt halten der STRG-Taste auch mehrere, Knoten gewählt werden. Dieses können dann beliebig angeordnet werden. Sollte ein Knoten über den Rand des Panels geschoben werden, entsteht automatisch eine Scrollbar.

Einen neuen Knoten in den Graphen einfügen

An beliebiger Stelle im Graphen kann ein neuer Knoten durch einen Mausklick eingefügt werden. Bevor dieser jedoch tatsächlich eingefügt wird, wird der Benutzer nach dem Namen des Knoten gefragt. Dafür sollte in erster Line die Adresse oder der Titel einer Webseite verwendet werden. Ist ein Knoten mit demselben Namen bereits im Graphen vorhanden, muss ein anderer Name gewählt werden.

Jeder eingefügte Knoten bekommt auch eine eindeutige ID zugewiesen. Diese dient später bei der Visualisierung der Berechnung dazu, dass in die Formel zur Berechnung des PageRank nicht die oft sehr lange URL einer Webseite eingefügt werden muss, sondern lediglich deren ID.

Anzumerken ist, dass in den Graphen maximal 20 Knoten eingefügt werden können. Technisch wäre es natürlich möglich einen Graphen mit einer wesentlich größeren Anzahl an Knoten darzustellen. Diese Limitation wurde aus Gründen der Übersichtlichkeit willkürlich gewählt und sollte außerdem für die Visualisierung des PageRank-Algorithmus ausreichend sein.

Eine gerichtete Kante erzeugen

Diese kann nur zwischen zwei Knoten erzeugt werden. Der Hintergrund dafür ist, dass Webseiten zwar auch auf sich selbst verweisen können, dies jedoch bei der Berechnung des PageRank nicht berücksichtigt wird.

Beim Erzeugen der Kante zwischen den Knoten wird auch eine Hilfslinie angezeigt, damit der Benutzer stets weiß, wo die neue Kante im Graphen verlaufen wird.

Knoten oder Kanten löschen

Durch einen Mausklick auf einen Knoten oder eine Kante werden diese aus dem Graphen entfernt. Ist der zu entfernender Knoten mit einem anderen Knoten verbunden, werden alle Verbindungen von oder zu dem zu entfernenden Knoten ebenfalls gelöscht, da diese danach zwecklos sind.

Den Graphen skalieren

Der gesamte Graph kann auch mittels eines Schiebreglers skaliert werden. Die Skalierung kann zwischen 50% und 150% liegen.

Neuer Graph, Speichern, Laden

Soll mit einem neuen Graphen begonnen werden, werden einfach alle Knoten und Kanten aus dem aktuellen Knoten entfernt.

Die Speicherung des Graphen erfolgt in einer XML-Datei. Es wird der Skalierungsfaktor (Knoten <zoomfactor>) des Graphen und auch jeder einzelne darin enthaltene Knoten (Knoten <node>) gespeichert. Für jeden Knoten werden seine ID (Attribut ID), sein Name (Attribut name) und seine Position (Knoten <position>) gespeichert. Weiters auch Referenzen auf die IDs jener Knoten (Knoten <targets>), auf welche ein Knoten zeigt. Folgendes Beispiel zeigt einen als XML-Datei abgespeicherten Graphen:

```
<graph>
  <zoomfactor>1.0</zoomfactor>
  <node ID="N0" name="Site1">
    <position>
      <x>29.0</x>
      <y>29.0</y>
    </position>
    <targets>
      <target>N1</target>
    </targets>
  </node>
  <node ID="N1" name="Site2">
    <position>
      <x>137.0</x>
      <y>74.0</y>
    </position>
    </targets>
  </node>
</graph>
```

Code 4: Auszug aus der XML-Datei eines Graphen

Eine solche Datei kann natürlich auch jederzeit wieder in das Applet geladen werden und stellt dadurch ein genaues Abbild des vorher gespeicherten Graphen dar.

6.1.3.2. Die Adjazenzmatrix

Ein weiters Feature in diesem Applet ist, dass sich der Graph nicht nur mit der Maus bearbeiten lässt, sondern auch mit Hilfe einer Adjazenzmatrix:

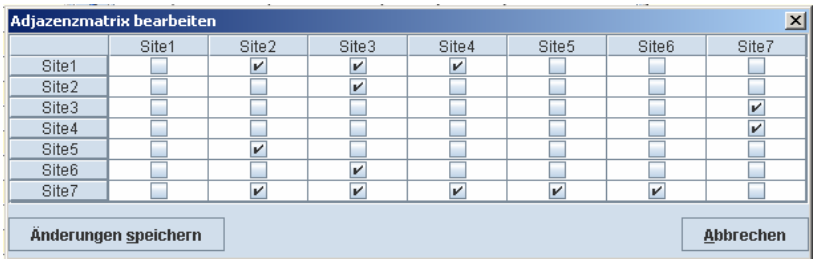

Abbildung 17: Bearbeitung der Adjazenzmatrix des Graphen

Durch Checkboxen kann eine Verbindung zwischen zwei Knoten hergestellt oder entfernt werden. Wird die Adjazenzmatrix verändert, wird dies sofort als Vorschau im Graphen selbst dargestellt. Eine neue Verbindung wird als Kante in grüner Farbe, eine gelöschte als unterbrochene Kante dargestellt (siehe Abbildung 18).

Werden die Änderungen gespeichert, werden diese in den Graphen übernommen, ansonsten verworfen.

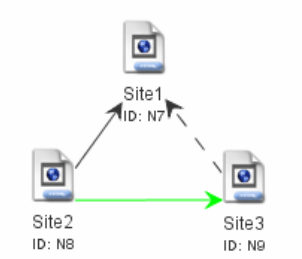

Abbildung 18: Vorschau der Graphenbearbeitung mittels Adjazenzmatrix

6.1.4. Die Visualisierung der PageRank-Berechnung

Wurde ein Graph fertig konstruiert, kann die Visualisierung durch den „Start-Button“ gestartet werden.

6.1.4.1. Die Wahl der Parameter vor der Visualisierung

Bevor mit der tatsächlichen Visualisierung der PageRank-Berechnung begonnen werden kann, müssen zuerst die Parameter für diese ausgewählt werden.

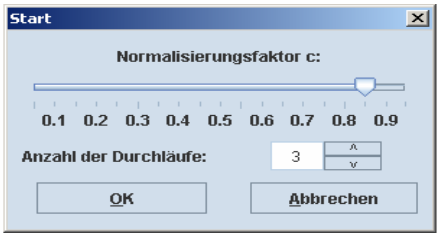

Abbildung 19: Auswahl der Parameter vor der Visualisierung im Applet

Wie in Abbildung 19 ersichtlich, müssen sowohl die Anzahl der Durchläufe, als auch die Normalisierungsfaktor c gewählt werden. Letzterer wird standardmäßig mit 0.85 voreingestellt [PAGE_2], kann jedoch mittels Schieberegler auch verändert werden.

Die Anzahl der Durchläufe gibt an, wie oft der PageRank jedes Knoten im Graphen berechnet wird. Je höher die Anzahl der Durchläufe ist, desto genauer wird der PageRank berechnet. In diesem Applet muss die Anzahl der Durchläufe zwischen 1 und 30 liegen, da für jene Graphen, welche in diesem Applet dargestellt werden können, meist nach nur wenigen Durchläufen der PageRank exakt berechnet wurde und sich nicht mehr verändert. Anzumerken ist, dass bei der Berechnung der PageRanks das Dangling Links Problem ausgeklammert wird und dessen „negativen Effekte“ (siehe dazu Kapitel 3.3.1) daher auftreten können.

Nach der Wahl der Parameter kann die Visualisierung beginnen.

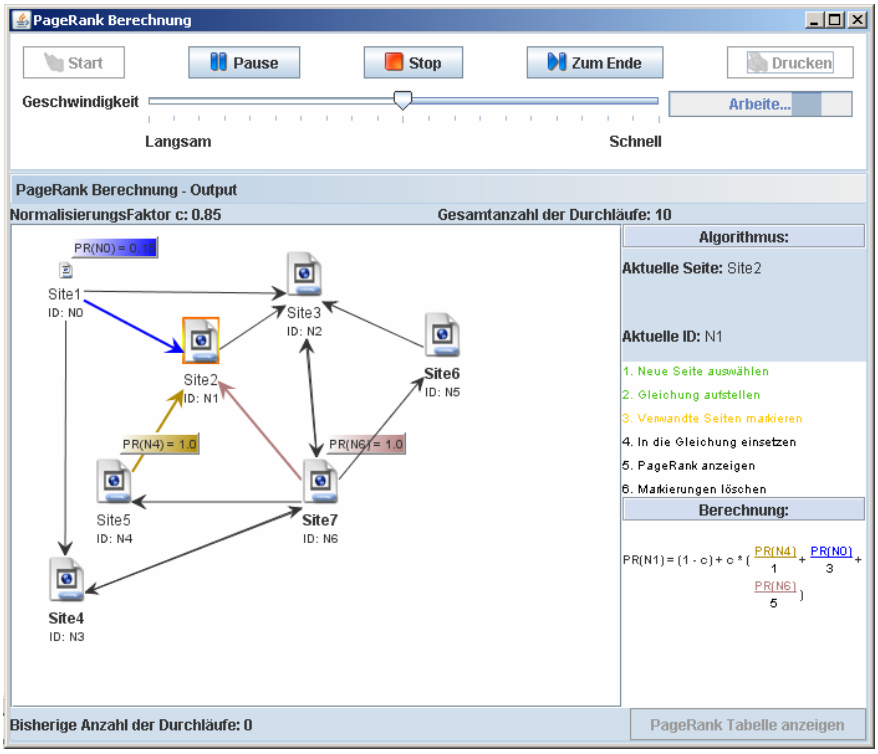

Abbildung 20: Visualisierung der PageRank-Berechnung

In Abbildung 20 ist eine gerade laufende Visualisierung zu sehen. In dem obersten Panel werden nochmals die vorher gewählten Parameter (Normalisierungsfaktor c und die

Anzahl der Durchläufe) angezeigt. Rechts werden der Name und die ID der Seite angezeigt, von der gerade der PageRank berechnet wird. Darunter findet sich der Algorithmus für die Visualisierung. Dieser besteht aus 6 Schritten:

1. Neue Seite auswählen

Es wird eine neue Webseite ausgewählt, für die der PageRank berechnet wird. Diese wird grafisch hervorgehoben indem der Knoten im Graphen orange umrandet wird (siehe Knoten Site2 in Abbildung 20).

2. Gleichung aufstellen

Es wird die Gleichung für die Berechnung des PageRank der aktuellen Webseite aufgestellt. Diese Gleichung wird ebenfalls rechts im Fenster, unter dem Algorithmus für die Visualisierung, angezeigt. Sie hat stets folgende Form:

$$PR(ID_U) = (1 - c) + c * \left(\frac{PR(ID_{V1})}{N_{V1}} + \frac{PR(ID_{V2})}{N_{V2}} + ... + \frac{PR(ID_{Vn})}{N_{Vn}}\right)$$

ID_U entspricht dabei der ID der aktuellen Webseite. Die Variable c steht für den Normalisierungsfaktor. ID_{V1}, ID_{V2} bis ID_{Vn} geben die ID von auf die aktuelle Webseite zeigenden Knoten an. N_{V1},N_{V2} bis N_{Vn} sind die Anzahl der Links auf den Webseiten ID_{V1} und ID_{V2}.

Beispielsweise wäre dies für die Webseite „Site2“ in Abbildung 20:

$$PR(N1) = (1 - c) + c * \left(\frac{PR(N4)}{1} + \frac{PR(N0)}{3} + \frac{PR(N6)}{5}\right)$$

Als zusätzliche visuelle Betonung werden die oberen Teile der Brüche in der Gleichung in der Farbe des Knoten dargestellt. Genaueres zu den Farben der Knoten findet sich in der Beschreibung des nächsten Schritts.

Sollte eine Webseite keine verwandten Knoten haben, d.h. gibt es keine Webseite, auf der sich ein Link zu der aktuellen Webseite befindet, werden die folgenden beiden Schritte „Verwandte Seiten markieren“ und „In die Gleichung einsetzen“ ausgelassen. Die Gleichung wird daher sofort in folgender Form angezeigt:

$$PR(ID_u) = (1 - c) + c * 0$$

Somit hat diese Webseite stets den Normalisierungsfaktor c als PageRank.

3. Verwandte Seiten markieren

Dabei werden alle Webseiten markiert, die einen Link auf die aktuelle Webseite enthalten. Die geschieht jeweils in der Farbe des Knoten:
Vor der Visualisierung wird jedem Knoten eine bestimmte Farbe zufällig aus einem Pool von 6 Farben zugeordnet. Diese Farbe bleibt während der gesamten Visualisierung konstant und soll dabei helfen, die Referenzen auf die aktuelle Webseite besser visuell wahrnehmen zu können.

Weiters wird in diesem Schritt auch eine kleine Zelle oberhalb jedes auf die aktuelle Webseite zeigenden Knoten angezeigt. Diese ist ebenfalls in der Farbe des Knoten gehalten und zeigt den PageRank dieses Knotens an (siehe Abbildung 20 bei den Knoten Site1, Site5 und Site7).

4. In die Gleichung einsetzen

In diesem Schritt des Algorithmus zur Visualisierung werden die Werte für den Normalisierungsfaktor c und die PageRanks jener Knoten, die einen Link auf die aktuelle Webseite haben, in die Gleichung eingesetzt.
Danach hat die Gleichung z.B.: für die Webseite „Site2“ in Abbildung 20 folgende Form:

$$PR(N1) = 0.15 + 0.85 * \left(\frac{1.0}{5} + \frac{1.0}{1} + \frac{0.15}{3}\right)$$

Die Werte für die PageRanks der verwandten Seiten in dieser Gleichung erklären sich dadurch, dass alle Webseiten mit PageRank 1.0 initialisiert werden, sich die Berechnung im ersten Durchlauf befindet, und der PageRank für die Webseite „Site1“ bereits berechnet wurde.

5. PageRank anzeigen

Nachdem in die Gleichung nun vollständig eingesetzt wurde, kann der PageRank für die aktuelle Webseite berechnet werden. Dieser wird dann unter der Gleichung angezeigt.

6. Markierungen löschen

Alle Markierungen im Graphen werden nun wieder gelöscht. Die aktuelle Webseite wird dabei in Abhängigkeit des gerade errechneten PageRanks skaliert. Damit die Skalierung für besonders große und kleine PageRanks die Darstellung des Knoten der Webseite jedoch nicht zu sehr verzerrt, werden die Länge und Breite des Knoten mit der Wurzel des PageRank multipliziert. Für einen PageRank von 0.15 würde die Bedeuten, dass sich der Knoten auf $\sqrt{0.15}\,\%$, also etwa 38%, der ursprünglichen Größe verringert. Da der PageRank einer Webseite nie weniger als 0.15 betragen kann, ist die minimale Größe auch 38% der Standardgröße eines Knotens. Eine Begrenzung der maximalen Größe eines Knotens wurde nicht eingeführt, da die Vergrößerung eines Knotens auf den maximal denkbaren PageRank kein Problem darstellt.

Die Skalierung erfolgt immer auf die Größe des Knoten, die er vor Beginn der Berechnung hatte. Damit bleibt die Skalierung stets einheitlich für alle Knoten des Graphen.

Die einzelnen Schritte des Algorithmus der Visualisierung werden ebenfalls durch eine Farbgebung unterstrichen. So wird der Text bereits absolvierter Schritte grün eingefärbt. Der Text jenes Schritts, in dem sich die Visualisierung gerade befindet, wird orange dargestellt. Der Text jener Schritte, die noch nicht absolviert wurden, ist schwarz gefärbt.

Werden die beiden Schritte „Verwandte Seiten markieren“ und „In die Gleichung einsetzen“ ausgelassen (siehe Beschreibung des Schritts 2 „Gleichung aufstellen“), wird der Text dieser beiden Schritte grau eingefärbt.

Links unten im Visualisierungsfenster wird die Anzahl der bereits absolvierten Iterationen der Berechnung angezeigt. Wenn alle Iterationen abgeschlossen sind, wird der rechts unten im Visualisierungsfenster befindliche Button „PageRank Tabelle anzeigen“ aktiviert. Mit diesem kann ein Dialog geöffnet werden, der die IDs, Namen und PageRanks aller Webseiten des Graphen enthält:

Abbildung 21: Tabelle der berechneten PageRanks im Applet

Durch einen Klick auf die Überschrift einer Spalte, kann nach der ID, dem Namen oder dem PageRank sowohl aufsteigend als auch absteigend sortiert werden.

7. Die Websimulation des PageRank Algorithmus

Dieses Kapitel beschäftigt sich mit dem zweiten Projekt in dieser Magisterarbeit: Der Simulation des PageRank-Algorithmus auf einem Set von realen Webseiten.

7.1. Implementierung

Da hier der Schwerpunkt nicht auf einer visuellen Darstellung liegt, ist es auch nicht notwendig ein weiters Applet, welches in das IT-Math Framework eingebettet ist, zu entwickeln.

Eine Möglichkeit wäre gewesen, die Simulation als ein Panel zu gestalten, auf der dann alle Aktionen ausgeführt werden können. Dies kann aber unter Umständen sehr

unübersichtlich werden. Außerdem benötigt man für die Berechnung der PageRanks realer Webseiten mehrere aufeinander aufbauende Schritte. So muss zuerst eine Gruppe von Webseiten angegeben werden, für die die Berechnung gemacht werden soll. Dies allein kann schon auf mehreren Arten gemacht werden. In Abhängigkeit von dieser Art können dann auch noch unterschiedliche Schritte folgen.

Daher habe ich mich für die Implementierung der Simulation in Wizard-Form entschieden. Dabei werden dem Benutzer die einzelnen Schritte in der korrekten Reihenfolge übersichtlich und einfach präsentiert und auch die Übersichtlichkeit des Programms ist gewährleistet.

Für die Simulation wird also ein Wizard implementiert, der den Benutzer zu Berechnung der PageRanks hinführt.

7.1.1. Der Wizard

Als Erstes musste für dieses Projekt ein Framework geschaffen werden welches die Möglichkeit bietet, den Wizard rasch und effizient zu gestalten. Dieser wurde im Wesentlichen durch zwei Klassen implementiert:

7.1.1.1. Die Klasse „Wizard“

Diese Klasse implementiert sozusagen den Rahmen des gesamten Programms. Der Wizard selbst, genauer gesagt die darauf enthaltenen Komponenten, bleiben in jedem Schritt dieselben. Abbildung 22 zeigt den „rohen“ Wizard, in dem noch kein WizardPanel enthalten ist.

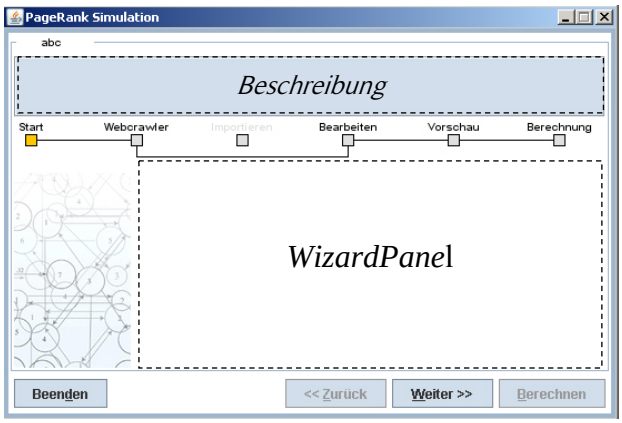

Abbildung 22: Der Wizard

Der Wizard besteht aus einer Vielzahl an Komponenten, die selbst bei Bedarf sehr einfach geändert werden können. Deshalb ist eine hohe Wiederverwendbarkeit dieser Klasse für andere Projekte gewährleistet.

Alle im Wizard verwendeten Farben wurden in einer Klasse ColorConstants angelegt. Dadurch kann auch die komplette farbliche Gestaltungen des Wizards einfach durch das Ändern einiger Konstanten verändert werden.

Der Klasse Wizard ist von javax.swing.JFrame abgeleitet. Dies impliziert bereits, dass dieses Programm nicht für das Web entwickelt wurde, sondern direkt auf dem Computer des Benutzers, wie es für einen Wizard auch üblich ist, verwendet werden soll.
Insgesamt bietet der Wizard sechs Komponenten:

1. Die Überschrift

Diese befindet sich im Wizard ganz oben (in Abbildung 22 durch den Beispieltext „abc“ dargestellt“) und ist Teil eines javax.swing.Border.TitledBorder. Das WizardPanel übergibt den Text für die Überschrift an den Wizard.

2. Die Beschreibung

Die Beschreibung ist das in blau gehaltene Kästchen unter der Überschrift in Abbildung 22. Mit Hilfe dieser Beschreibung können dem Benutzer Informationen

über das gerade geladene WizardPanel mitgeteilt werden. Außerdem kann es Anweisungen, Hilfestellungen und Erklärungen für die Bedienung enthalten. Dies trägt natürlich in hohem Maße zur Benutzerfreundlichkeit bei und ersetzt damit auch eine eigene Hilfe für den Wizard oder die einzelnen WizardPanel. Die Beschreibung erhält der Wizard ebenfalls von dem gerade geladenen WizardPanel.

3. Die dynamische Fortschrittsanzeige

Die dynamische Fortschrittsanzeige zeigt dem Benutzer den Verlauf des Wizards an. Dadurch sind alle Schritte des Wizards schon beim Start dieses ersichtlich. Ferner kann der Benutzer auch ungefähr abschätzen, welchen Zeitaufwand er für die Benutzung des Wizards benötigen wird.

Es handelt sich dabei um eine dynamische Fortschrittsanzeige. Dies bedeutet, dass je nachdem welchen Weg der Benutzer im Wizard (zu den verschiedenen Wegen siehe das Kapitel: Die WizardPanels) wählt, verändert sich auch der Verlauf der Fortschrittsanzeige. Die Anzahl der Schritte und die Schritte selbst bleiben jedoch stets konstant. Es kann durch die Auswahl der verschiedenen Wege nur dazu führen, dass einzelne Schritte ausgelassen werden. Diese werden dann unter Anderem auch von der Fortschrittslinie umwandert (siehe Abbildung 23, wo der zweite Schritt umgangen wird).

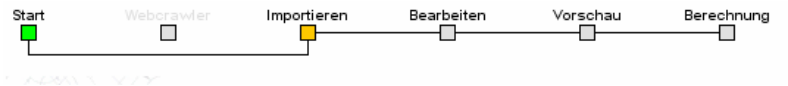

Abbildung 23: Dynamische Fortschrittsanzeige des Wizard

Die als umrandete Rechtecke dargestellten Schritte haben auch eine eigene Farbgebung (siehe Abbildung 23). Dadurch wirkt die Fortschrittsanzeige für den Benutzer visuell noch ansprechender und verdeutlicht ihm auch den Verlauf des Wizards. Bereits besuchte Schritte werden grün eingefärbt, der Schritt, in dem sich der Benutzer gerade befindet wird in orange dargestellt und noch nicht besuchte Zustände oder nicht zu besuchende Zustände werden ausgegraut. Der Unterschied zwischen diesen besteht darin, dass bei letzteren auch der Text des Schritts grau gefärbt wird.

Wie im Beispiel von Abbildung 23 ersichtlich, wurde der erste Schritt („Das Welcome Panel“) bereits absolviert. Es wurde vom Benutzer offensichtlich gewählt, eine Datei zu importieren. Deshalb muss der Schritt, in dem der Webcrawler verwendet wird, nicht besucht werden. Der Benutzer befindet sich gerade im Schritt „Importieren“ und es folgen noch 3 weitere Schritte.

4. Das Bild

Zu einer angenehmen und optisch ansprechenden Darstellung wurde auch ein Bild am linken Rand des Wizard eingefügt. Dieses hat selbst keine besondere Funktion. Es soll lediglich zur optischen Untermalung und Aufbereitung des Wizard dienen.

5. Die Buttons

Es befinden sich insgesamt 4 Buttons auf dem Wizard. Diese steuern den gesamten Ablauf und stellen allen notwendigen Funktionen bereit, um durch den Wizard zu navigieren. Zusätzlich ermöglichen KeyMnemonics eine rasche Navigation:

- Der „Beenden-Button“

Dieser Button dient zum Beenden des Wizards. Bevor sich der Wizard schließt, wird nochmals nachgefragt, ob der Wizard wirklich beendet werden soll. Damit wird ein irrtümliches Betätigen des Button, und ein damit verbundenes ungewolltes Beenden des Wizards, ausgeschlossen.

- Der „Weiter-Button“

Dieser Button dient zur Vorwärtsnavigation im Wizard. Der Button kann jedoch, abhängig von in einem WizardPanel eventuell vorhandenen Bedingungen, deaktiviert werden. Somit ist gewährleistet, dass der Benutzer nicht zu einem WizardPanel gelangen kann, für welches nicht alle erforderlichen Daten vorhanden sind oder falls der vorherige Schritt noch nicht vollständig abgeschlossen ist. Dies unterstützt die Fehlerrobustheit des Programms.

Beim letzten WizardPanel des Wizard wird normalerweise auch dieser Button deaktiviert, da danach kein weiteres WizardPanel mehr kommen kann. Es wird demnach beim Laden jedes WizardPanels überprüft, ob für dieses ein Nachfolger vorhanden ist.

- Der „Zurück-Button“

Dieser Button dient zur Rückwärtsnavigation im Wizard. Um dies zu ermöglichen, wird jedes neu erzeugte Panel auf einen Stack, der durch die Klasse java.util.Stack implementiert ist, gelegt. Dadurch kann durch eine einfache pop-Operation auf den Stack zum vorhergehenden WizardPanel zurückgekehrt werden.
Da das erste Panel des Wizard natürlich keinen Vorgänger hat, ist beim Laden des ersten Panels dieser Button stets deaktiviert.

- Der „Berechnen-Button“

Dieser Button ist grundsätzlich während aller Schritte des Wizards deaktiviert. Die Ausnahme ist dabei das letzte WizardPanel. Da hier kein weiters WizardPanel mehr folgt, ist es eindeutiger und ansprechender mit diesem speziellen Button die Berechnung zu starten und damit den Wizard abzuschließen. Ein weiterer Aspekt ist, dass der Abschluss des Wizard durch einen speziellen Button dem Benutzer gegenüber verdeutlicht wird.

6. Das WizardPanel

Das WizardPanel implementiert den Inhalt des Wizards. Auf diesem Panel werden Komponenten platziert, welche die eigentlichen Aufgaben des Programms übernehmen. Diese Panels stellen damit auch die einzigen Komponenten dar, die ihr Aussehen während dem Ablauf des Wizard komplett verändern können.

7.1.1.2. Die Klasse WizardPanel

Diese abstrakte Klasse wurde von javax.swing.JPanel abgeleitet und implementiert das Interface java.awt.event.ActionListener. Dieser Listener wurde in dieser Oberklasse hinzugefügt, um auf Events der Komponenten auf einem WizardPanel reagieren zu können.

Es werden einerseits für die erbenden Klassen Methoden bereitgestellt, die von mehreren dieser Klassen verwendet werden, andererseits müssen bestimmte Methoden zwingend überschrieben werden, um eine fehlerlose Funktionsweise des Wizards bzw. Kommunikation mit dem Wizard zu gewährleisten.

7.1.2. Die Architektur des Simulationsprogramms

Wie bereits im Kapitel „Der Wizard“ erläutert, besteht das Framework des Wizards eigentlich nur aus zwei Klassen. Im vorherigen Kapitel wurde die Wizard Klasse bereits ausführlich besprochen. Folgendes in Abbildung 24 dargestellte vereinfachte Klassendiagramm skizziert nun den genaueren Zusammenhang beider Klassen und die Architektur des auf Grundlage des Wizard-Frameworks implementierten Simulationsprogramms.

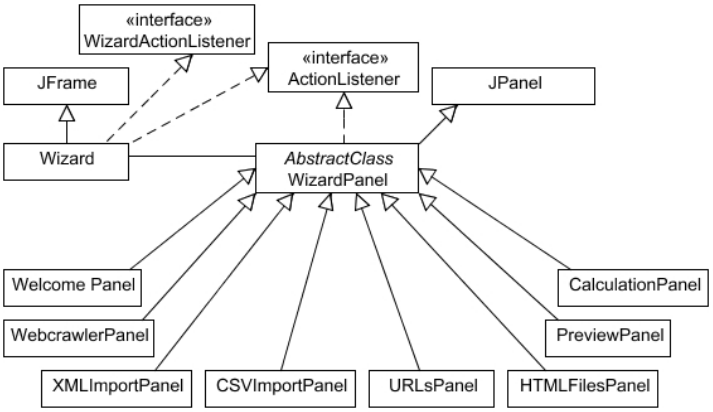

Abbildung 24: Klassenarchitektur des Simulationsprogramms

Im Simulationsprogramm erben insgesamt 8 Klassen von der abstrakten Klasse WizardPanel. Dies sind genau jene, die dann auch in den Wizard geladen werden.

Die beiden Klassen Wizard und WizardPanel implementieren auch das Interface java.awt.event.ActionListener. Der Wizard benötigt dies, um auf das Betätigen seiner Buttons reagieren zu können. Die abstrakte Klasse WizardPanel stellt für ihre Unterklassen damit gleich einen ActionListener für eventuell vorhandene Komponenten, z.B.: Buttons, Radio Buttons oder Tables, bereit.

Der Wizard implementiert jedoch auch noch das Interface WizardActionListener. Mit Hilfe dieses selbst implementierten Listeners steht der Wizard mit den in ihm eingebetteten WizardPanels in Kontakt und kann dadurch auch auf Befehle oder Aktionen dieser reagieren. Benötigt wird dies vor allem für die dynamische Fortschrittsanzeige und um einzelne Buttons des Wizard gegebenenfalls aktivieren oder deaktivieren zu können.

7.1.3. Die WizardPanels

Dieses Unterkapitel beschäftigt sich nun mit der eigentlichen Aufgabe des Simulations-Wizard, welche in den WizardPanels implementiert ist. Dazu wird folgend das Thema und die genaue Funktion jedes der 8 implementierten Wizard Panels erläutert.

7.1.3.1. Das Welcome Panel

Dieses Panel ist, wie der Name bereits vermuten lässt, das erste Panel. Es wird beim Start des Programms in den Wizard geladen.

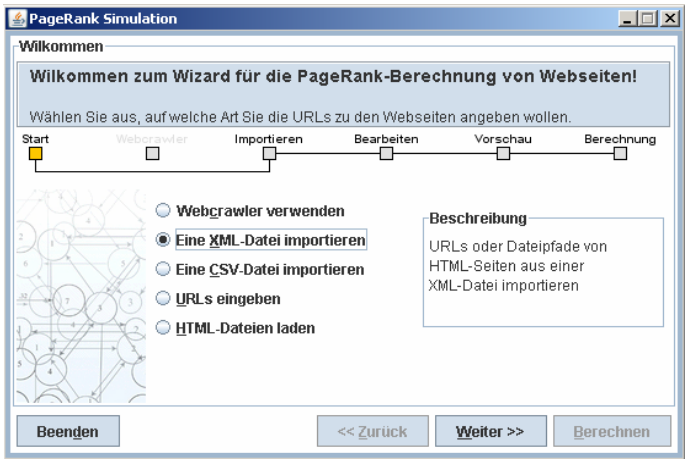

Abbildung 25: Das Welcome Panel

Wie in Abbildung 25 ersichtlich, wird auf diesem Panel eine Auswahl angeboten. Damit kann gewählt werden, wie ein Set von URLs von Webseiten für die Berechnung derer PageRanks gewonnen wird. Sobald einer der RadioButtons ausgewählt wurde, ändert sich die nebenstehende Beschreibung. Damit werden dem Benutzer nähere Details zu dieser Möglichkeit angeboten Ebenso ändert sich die dynamische Fortschrittsanzeige in Abhängigkeit von der Auswahl, wie bereits in Kapitel „Die dynamische Fortschrittsanzeige“ beschrieben. Der Benutzer kann demnach aus fünf verschiedenen Möglichkeiten wählen:

1. Webcrawler verwenden

Die URLs sollen mit Hilfe eines Webcrawlers, ausgehend von einer Start-URL, gefunden.

2. Eine XML-Datei importieren

Ein Set von zuvor als XML-Datei exportierten URLs (siehe das Kapitel „Das URL Panel“) soll importiert werden.

3. Eine CSV-Datei importieren

Ein Set von zuvor als CSV (**C**haracter/**C**olon/**C**omma **S**eparated **V**alues)-Datei exportierten URLs (siehe das Kapitel „Das URL Panel“) soll importiert werden.

4. URLs eingeben

URLs können direkt manuell in eine Liste eingegeben werden.

5. HTML-Dateien laden

Lokal gespeicherte HTML-Dateien können in eine Liste geladen werden. Diese werden dann wie URLs zu Webseiten behandelt.

7.1.3.2. Das Webcrawler Panel

In diesem Panel wird ein Webcrawler verwendet, um ein Set von URLs einzulesen.

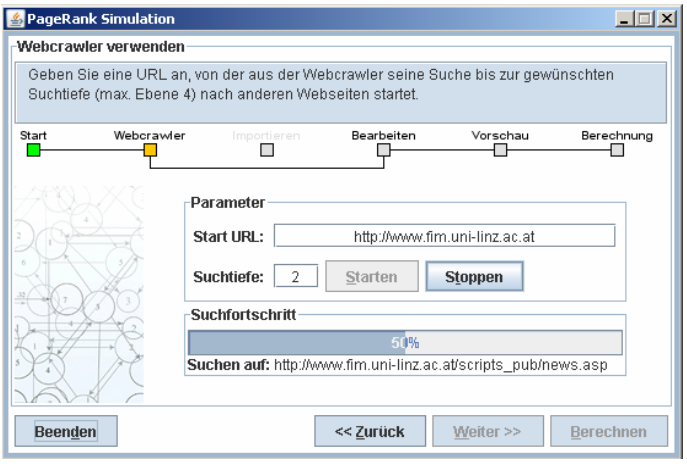

Abbildung 26: Das Webcrawler Panel

Wie in Abbildung 26 ersichtlich erfordert der Webcrawler insgesamt zwei Parameter:

Die Start URL

Diese Seite stellt den Startpunkt für den Webcrawler dar. Beginnend mit dieser Seite sucht er nach Links auf Webseiten. Vor dem Starten des Webcrawlers wird diese URL auf seine Gültigkeit überprüft. Zur genauen Funktion und Beschreibung des Webcrawlers, siehe das Kapitel „Der Webcrawler“.

Die Suchtiefe

Dieser Parameter gibt an, bis zu welcher Ebene der Webcrawler seine Suche nach Webseiten ausführen soll. Wird beispielsweise bis zur Ebene 1 gesucht, wird nur die Start URL und die darauf enthaltenen Links eingelesen. Bei der Ebene 2 werden auch die Webseiten und Links dieser in Ebene 1 gefundenen URLs eingelesen, usw.
Aus Performanzgründen muss die Suchtiefe in diesem Simulationsprogramm zwischen 1 und 4 liegen, da eine Suche bis zu einer höheren Ebene unter Umständen sehr viel Zeit in Anspruch nimmt.

Wurden beide Parameter korrekt gewählt, kann der Webcrawler seine Arbeit aufnehmen. Während der Suche nach Webseiten wird der Suchfortschritt in einem Fortschrittsbalken angezeigt. Darunter wird immer jene Seite angezeigt, auf der gerade nach Links bzw. neuen Webseiten gesucht wird.

7.1.3.2.1. Der Webcrawler

Der in dem Programm verwendete Webcrawler wurde selbst implementiert. Aus Interesse an der Technologie und der Herausforderung, ein solches Werkzeug selbst zu implementieren, wurde deshalb nicht auf bereits entwickelte und eventuell sogar als Open Source verfügbare Webcrawler zurückgegriffen.

Im ersten Schritt holt sich der Webcrawler den HTML-Quelltext der Webseite der Start URL. Dieser Quelltext wird nach Links zu anderen Webseiten geparst.
Das extrahieren der Links aus dem Quelltext stellte die größte Herausforderung dar. Die Fragen vor der Implementierung waren:
Wie erkennt man einen Link im Quelltext einer Webseite bzw. welche Besonderheiten könnte es hierbei geben? Wie findet man eine angemessene Anzahl an Links auf einer Webseite? Welche Art von URLs in den Links kann es geben, und welche sollen ausgelesen werden bzw. welche sind für die PageRank-Berechnung irrelevant oder sollen generell verworfen werden?
Für die Beantwortung dieser Fragen gibt es mehrere Lösungsansätze:

Suche nach dem String „http://“

Dabei wird der Quelltext nach dem Vorkommen der Zeichenkette „http://“ durchsucht. Wird eine solche Zeichenkette gefunden, liest man den Rest der Zeichenkette bis zu einem Leerzeichen oder Anführungszeichen ein. Der Vorteil dieser Methode ist, dass man mit einer einfachen String-Operationen in Java und ohne großen Suchaufwand alle Links findet, die mit dieser Zeichenkette beginnen. Der Nachteil ist, dass dadurch keine URLs gefunden werden die mit „www.“ beginnen oder bei denen einfach nur eine Subseite angegeben wurde. Weiters kann nicht unterschieden werden, ob wirklich ein Link gefunden wurde oder nur eine URL als Text auf einer Seite.

Suche nach dem HTML-Tag „<a href="

Dabei wird ebenfalls der HTML-Quellcode nach einer bestimmten Zeichenkette durchsucht. Wie im HTML-Standard von W3C [W3C_1] festgehalten, beginnt ein Link immer mit dem Tag „<a>", gefolgt von dem Attribut „href". Um einen Link in HTML anzugeben, muss also die Zeichenkette „<a href="*URL einer Webseite*">" gebildet werden.

Bei dieser Methode der Linkextraktion sucht man also nach dem eben beschrieben Tag und extrahiert zwischen den beiden Anführungszeichen stehenden Text. Zum Beispiel würde bei dem Link <a href="www.google.at"> der Text www.google.at extrahiert werden.

Der Vorteil dieser Methode gegenüber der ersten besteht darin, dass nun auch alle Links gefunden werden, die nicht mit „http://" beginnen. Der Nachteil dieser Methode ist, dass zwischen dem Tag „<a" und dem Attribut „href" beispielsweise noch weitere Leerzeichen oder andere Attribute stehen können. Dieses Problem würde sich zwar in Java einfach durch Schleifen, die weitere Leerzeichen überlesen, lösen lassen, verkompliziert jedoch auch den Code selbst.

Die Suche nach Patterns

Diese Suchmethode nach Links ist sehr ähnlich zu der Suche nach dem HTML-Tag „<a href=". Der wesentliche Unterschied besteht jedoch darin, dass es die Nachteile der vorher beschriebenen Methode ausgleicht. Es wird nicht nach dem HTML-Tag mit dem Attribut als Gesamtes gesucht, sondern es wird versucht einen Link mittels eines regulären Ausdrucks zu erkennen. Die Unterstützung regulärer Ausdrücke ist bereits in der Java API standardmäßig im Package java.util.regex implementiert. Als ersten Schritt ist es daher notwendig eine Syntax für einen Link zu definieren, nach der dann im Anschluss im Quelltext einer Webseite nach Links zu Webseiten gesucht werden kann.

Ich habe dafür folgende Syntax in EBNF definiert:

```
ANY = enthält alle Zeichen
S = space
LINK = href S* = S* " ANY "
```

Code 5: Syntax eines Links

Das Zeichen „*“ bedeutet, dass der Token davor beliebig oft vorkommen kann, aber auch nicht vorkommen muss. Diese Syntax kann in Java mit Hilfe der Pattern Klasse in einen regulären Ausdruck umgewandelt werden:

```
Pattern p = Pattern.compile("href\\s*=\\s*\"(.*?)\"");
```

Code 6: Pattern für die Suche nach Links

Ein Link enthält demnach stets zuerst die Zeichenfolge „href“, auf die eine beliebige Anzahl an Leerzeichen folgen kann. Dadurch werden auch Links erkannt, welche zwischen dem „<a“ und „href“ noch andere Attribute besitzen. Als nächstes muss das Zeichen „=“ folgen. Nach einer weiteren beliebigen Anzahl an Leerzeichen kann entweder ein Anführungszeichen und danach die URL, oder gleich die URL kommen. Abgeschlossen wird die URL durch ein weiteres Anführungszeichen.

Ein weiterer Vorteil dieser Methode liegt darin, dass vor allem der Code sehr simpel und übersichtlich gehalten werden kann.

Anzumerken ist, dass vor der Suche nach Links der gesamte Quelltext einer Webseite in Kleinbuchstaben umgewandelt wird. Die Groß- und Kleinschreibung spielt hier also keine Rolle.

Dieses Pattern reicht normalerweise aus, um die meisten Links im Quelltext einer Webseite finden zu können. Natürlich kann ein Link auch anders aufgebaut sein. Beispielsweise wären folgende 4 Links syntaktisch nach [W3C_2] gültig:

```
Link 1: <a    href    =    "text.html"   >test</a>
Link 2: <a name="test" href=    "text.html">test</a>
Link 3: <a type="text/html" name="test"     href="text.html"   >test</a>
Link 4: <a
 type="text/html" name="test" href    =
  "text.html"   >test
</a>
```

Code 7: Beispiele für Links

Da es jedoch nicht das Ziel ist, dass durch den Webcrawler in dieser Simulation wirklich alle Links auf einer Webseite gefunden werden, ist das vorher definierte Pattern ausreichend, da die meisten Links nach dieser Syntax aufgebaut sind.

Wurde nun eine URL in einem Link gefunden, gibt es noch einige Ausnahmen, bei denen diese URL wieder verworfen wird:

- Wenn keine URL vorhanden ist, d.h. einfach nur zwei Anführungszeichen ohne Inhalt angegeben sind.
- Wenn die URL mit „#“ beginnt, handelt es sich nur um einen Link zu einer Sprungmarke auf der aktuellen Seite.
- Wenn der Ausdruck „mailto:“ in der URL enthalten ist, handelt es sich bei der URL um die Angabe einer E-Mail Adresse.
- Links, die den Ausdruck „javascript:“ enthalten, werden ebenfalls verworfen, da diese nur Funktionsaufrufe für eine JavaScript Funktion enthalten.

Da URLs auch oft nicht mit ihren absoluten Pfaden angegeben werden (z.B.: <a href=“/test.html“>), wird geprüft ob die URL mit dem Zeichen „/“ beginnt. Ist dies der Fall wird der URL der Ausdruck „http://“ und der Host Name der Seite, auf der sich der Link befindet, vorangestellt. Dasselbe wird auch für den Fall gemacht, wenn nur eine HTML-Seite als Link angegeben wird (z.B.: <a href=“test.html“>). Enthält die URL am Ende eine Sprungmarke, dann wird diese entfernt, da sie nicht benötigt wird.

Anzumerken ist noch, dass das Attribut rel="nofollow" (näheres dazu siehe Kapitel 4.8.3) nicht beachtet wird, da es für den klassischen PageRank Algorithmus keine Rolle spielt und eine Besonderheit des heute z.B.: bei Google verwendeten Webcrawlers ist.

Am Ende wird nochmals durch das Erzeugen eines java.net.URL-Objekts geprüft, ob die URL selbst eine gültige Syntax hat. Ist dies der Fall wird sie zu einer Liste hinzugefügt. Diese Liste enthält jede gefundene URL nur einmal. Für jede gefundene URL in einem Link auf einer Webseite wird in der nächsten Ebene nach Links auf der Webseite von dieser URL gesucht, usw. Der implementierte Webcrawler realisiert somit eine Breitensuche.

Wurde die angegebene Suchtiefe erreicht, beendet der Webcrawler seine Tätigkeit und es kann der nächste Schritt beginnen. Siehe dazu das Kapitel „Das URL Panel".

7.1.3.3. Das XML-Import und das CSV-Import Panel

Diese beiden Panels haben optisch eine einheitliche Gestaltung, siehe Abbildung 27. Ihre Aufgabe besteht darin, eine Liste von URLs aus einer XML- bzw. CSV-Datei zu importieren.

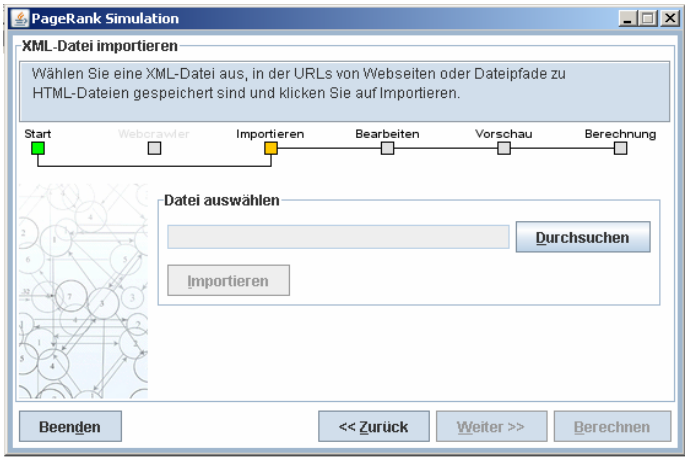

Abbildung 27: Das Import Panel

Mit Hilfe des „Durchsuchen-Button" kann nach einer XML- bzw. CSV-Datei gesucht werden. Wurde eine Datei gewählt, wird der deren Dateipfad in das Textfeld eingefügt.

Danach wird der „Importieren-Button“ automatisch aktiviert. Ein Klick auf diesen Button bewirkt, dass die URLs aus der Datei eingelesen werden. Danach kann der nächste Schritt beginnen. Siehe dazu das Kapitel „Das URL Panel“.

7.1.3.4. Das URL Panel

Dieses Panel kann einerseits direkt nach dem Welcome Panel dazu verwendet werden, um URLs manuell in eine Liste einzugeben. Andererseits ist es auch der Nachfolger des Webcrawler, XML- und CSV-Import und HTML-Dateien Panels. Die mit Hilfe der verschiedenen Möglichkeiten auf diesen Panels gewonnen URLs werden dann in die Liste auf dem URLs Panel aufgenommen. Dies dient als Vorschau der vorher gewonnen Links. Weiters wird dem Benutzer die Möglichkeit geboten, diese Links nochmals zu bearbeiten, zu exportieren oder noch manuell neue Links der Liste hinzuzufügen.

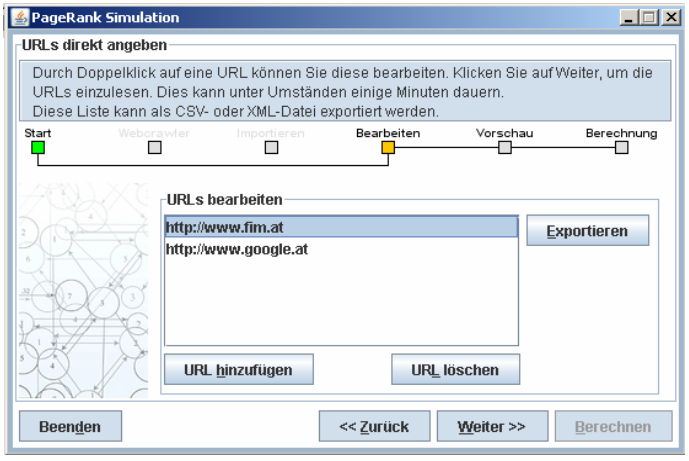

Abbildung 28: Das URL Panel

Wie in Abbildung 28 ersichtlich, besteht der Hauptteil dieses Panels aus einer Liste von URLs. Jede URL darin darf maximal einmal vorhanden sein. Durch einen Klick auf den „URL hinzufügen“-Button kann eine neue URL in die Liste eingefügt werden. Mit Hilfe des „URL löschen“-Button kann eine markierte URL wieder aus der Liste gelöscht werden. Ist in der Liste keine URL vorhanden oder keine markiert, wird dieser Button deaktiviert.
Durch einen Doppelklick auf eine URL in der Liste kann diese bearbeitet werden. Anzumerken ist hier, dass in dieser Liste selbst nicht direkt geprüft wird, ob eine eingegebene URL gültig ist. Dies wird erst beim Einlesen der URL gemacht.

Der „Exportieren-Button“ bietet die Möglichkeit alle in der Liste vorhandenen URLs zu exportieren. Dafür stehen die folgenden zwei Dateiformate zur Auswahl:

Export in eine XML-Datei

Dabei besteht die XML-Datei aus einem „urls“-Knoten, der eine Liste von „url“-Knoten enthält, siehe folgendes Beispiel:

```
<urls>
  <url>http://www.fim.at</url>
  <url>http://www.google.at</url>
</urls>
```

Code 8: Auszug aus einer XML-Datei mit URLs

Export in eine CSV-Datei

Es wird eine CSV (**C**haracter/**C**olon/**C**omma **S**eparated **V**alues)-Datei erzeugt. Dabei handelt es sich grundsätzlich um eine Textdatei. In ihr werden Datensätze gespeichert, die durch ein bestimmtes Zeichen, dem sogenannten Delimiter, getrennt sind. In diesem Programm wurde der CSV-Export so implementiert, dass jeweils 1 URL in eine Zeile geschrieben wird. Dies entspricht damit einem Datensatz pro Zeile. Als Delimiter zwischen den einzelnen URLs wird ein Semikolon verwendet.

Wurde nun eine Liste aus URLs von Webseiten angelegt, kann der nächste Schritt beginnen. Durch einen Klick auf den „Weiter-Button“ werden allerdings, bevor das nächste Panel geladen wird, alle auf den Webseiten enthaltenen Links extrahiert. Der nächste Schritt des Wizard wird im Kapitel „Das Preview Panel“ beschrieben.

7.1.3.5. Das HTML-Dateien Panel

Dieses Panel ist ähnlich wie das URL Panel aufgebaut. Auf diesem ist es möglich, lokal gespeicherte HTML-Dateien in den Wizard zu importieren.

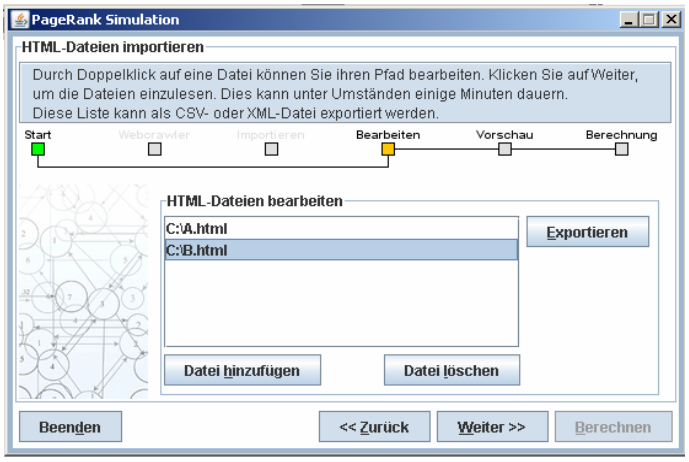

Abbildung 29: : Das HTML-Dateien Panel

Mit einem Klick auf den „Datei hinzufügen“-Button öffnet sich ein Dateidialog. In diesem können HTML-Dateien ausgewählt werden. Diese - eigentlich die Dateipfade zu diesen Dateien - werden dann in die in Abbildung 29 ersichtliche Liste aufgenommen. Der Export der Liste funktioniert grundsätzlich genau wie beim URLs Panel, mit der Ausnahme, dass keine URLs, sondern eben die Dateipfade exportiert werden.
Ein weiterer kleiner Unterschied ist, dass beim Export in eine XML-Datei diese andere Knoten besitzt. Folgendes Beispiel zeigt die Struktur einer solchen XML-Datei.

```
<files>
  <file>http://www.fim.at</file>
  <file>http://www.google.at</file>
</files>
```

Code 9: Auszug aus einer XML-Datei mit Files

Da ansonsten dieselbe Funktionalität wie beim URLs Panel besteht, sei auf dessen Beschreibung im Kapitel „Das URL Panel“ verwiesen.

7.1.3.6. Das Preview Panel

Nachdem nun eine gewisse Liste an Webseiten und der darauf enthaltenen Links eingelesen wurde, wird das Ergebnis dieses Einleseprozesses als Tabelle (siehe Abbildung 30) dargestellt.

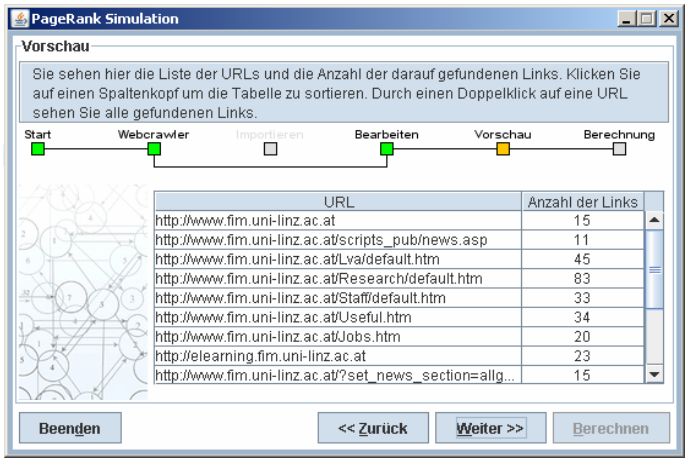

Abbildung 30: Das Preview Panel

In der linken Spalte der Tabelle auf diesem Panel sind die vorher angegebenen URLs bzw. Dateipfade angegeben. In der rechten Spalte wir die Anzahl der auf dieser Webseite gefundenen Links angegeben. Konnte eine URL oder eine Datei nicht eingelesen werden, wird statt der Anzahl der Links der Text „Konnte nicht eingelesen werden“ in der rechten Spalte angezeigt.

Diese Tabelle kann durch einen Klick auf eine Spaltenüberschrift sowohl nach den URLs als auch nach der Anzahl der darauf gefunden Links sortiert werden.
Durch einen Doppelklick auf eine URL oder die Anzahl ihrer Links öffnet sich ein weiteres Fenster, in dem alle auf dieser Webseite gefundenen Links aufgelistet sind. Siehe dazu Abbildung 31.

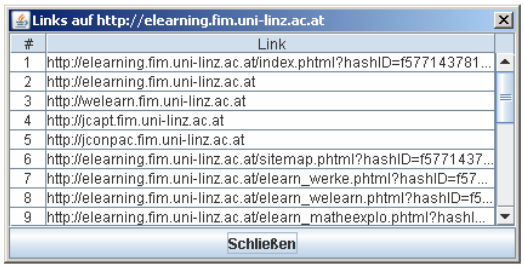

Abbildung 31: Liste der auf einer Webseite gefundenen Links

Auf diesem Panel lässt sich also nichts mehr bearbeiten oder einstellen. Es soll dem Benutzer lediglich als Vorschau vor der Berechnung der PageRanks dienen.

7.1.3.7. Das Berechnungs Panel

Am Ende des Wizard kann nun auf diesem letzten Panel die eigentliche Berechnung der PageRanks beginnen.

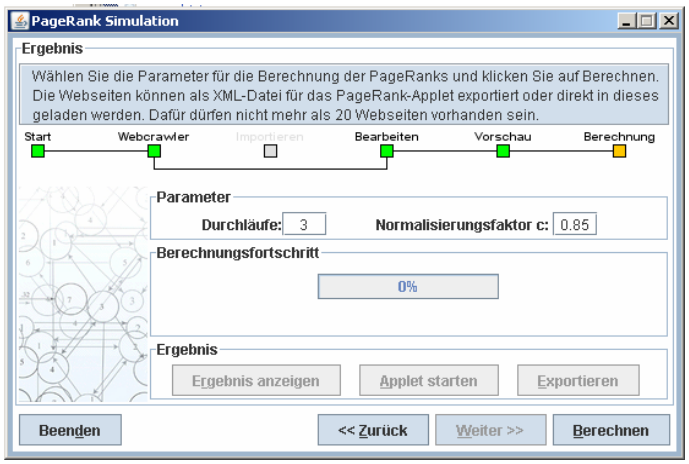

Abbildung 32: Das Berechnungs Panel

Es können insgesamt zwei Parameter für die Berechnung des PageRank eingestellt werden: die Anzahl der Durchläufe (Voreinstellung: 10) und der Normalisierungsfaktor c (Voreinstellung: 0.85). Die Anzahl der Durchläufe kann in dieser Simulation zwischen 1 und 100 liegen, der Normalisierungsfaktor c zwischen 0.05 und 0.95.

Nachdem es sich bei diesem Panel um das letzte des Wizards handelt, ist natürlich der „Weiter-Button“ deaktiviert und der „Berechnen-Button“ aktiviert. Wurden alle Parameter eingestellt, kann der Berechnungsvorgang mit einem Klick auf den „Berechnen-Button“ begonnen werden. Der Berechnungsfortschritt wird in eine Fortschrittsanzeige angezeigt.
Ist die Berechnung beendet, kann man sich die PageRanks aller Seiten in einem Dialog als Tabelle anzeigen lassen. Siehe dazu Abbildung 33.

Abbildung 33: Tabelle der berechneten PageRanks

In der linken Spalte dieser Tabelle wird die Adresse der Webseite angegeben. In der rechten Spalte findet man den errechneten PageRank. Auch diese Tabelle kann wieder nach der Adresse der Webseite oder dem PageRank durch einen Klick auf die Spaltenüberschrift auf- bzw. absteigend sortiert werden.

Da bei der Berechnung der PageRanks das Dangling Links Problem ausgeklammert wird, kann es sein, dass die Summe der PageRanks dabei nicht dem erwarteten Wert entspricht bzw. ein einzelner PageRank auch nicht genau dem tatsächlichen PageRank entspricht.
Neben dem Anzeigen dieser Tabelle können noch zwei weitere Aktionen dem Berechnungs Panel ausgeführt werden:

7.1.3.7.1. Applet starten

Damit wird das in Kapitel 6 beschriebene Applet gestartet. Die Webseiten, für welche im Simulations-Wizard gerade der PageRank berechnet wurde, werden dabei sofort in das Applet geladen und als Graph dargestellt. Dazu wird eine temporäre XML-Datei erzeugt, mit der das Applet gestartet wird.

Ein Problem, welches beim Erzeugen des Graphen im Applet besteht, ist dass man für die Anordnung der Knoten des Graphen keine Koordinaten kennt. Deshalb werden die Koordinaten für die Webseiten so berechnet, dass sie exakt einen Kreis bilden:
Ausgehend von einem Radius r und einer Startkoordinate auf dem Applet, können nun alle Koordinaten berechnet werden, wie folgender Codeausschnitt zeigt:

```
private Coordinate[] createCoordinates() {
    // the number of all nodes in the graph
    int nrOfNodes = this.nodes.size();
    Coordinate[] coordinates = new Coordinate[nrOfNodes];

    double degreeStep = 360 / nrOfNodes;
    double rad;
    double degrees = 0.0;

    String nodeName;
    for (int i = 0; i < this.nodes.size(); i++) {
        // the nodeName is needed to center the node on the circle
        nodeName = ((Node) this.nodes.get(i)).getUrl();
        // get the radian
        rad = Math.toRadians(degrees);
        // set the coordinates for the current url
        coordinates[i] = new Coordinate((START_X
                + (int) (Math.sin(rad) * RADIUS) - this.fm
                .stringWidth(nodeName) / 2), (START_Y
                - (int) (Math.cos(rad) * RADIUS) - 31));
        degrees += degreeStep;
    }
    return coordinates;
}
```

Code 10: Berechnung der Koordinaten für den Export eines Graphen aus dem Wizard

Dabei wird zuerst ein neues Coordinate-Array angelegt. Ein Objekt vom Typ Coordinate besteht nur aus einer double-Variable x und einer double-Variable y. Damit die Knoten gleichmäßig auf einem Kreis verteilt werden, wird die Schrittgröße „degreeStep“ ermittelt, indem man 360 durch die Anzahl der Knoten dividiert.
Für alle Knoten wird eine Koordinate errechnet, wobei anzumerken sei, dass hier natürlich mit Radianten und nicht mit Grad gerechnet werden muss.

Ein weiteres Problem besteht zu diesem Zeitpunkt der Koordinatenberechnung noch: Man kann einen Knoten nur unter Angabe der Koordinaten des linken oberen Eckpunkts in den

Graphen einfügen. Da die einzelnen Knoten durch verschieden lange Namen, die ebenfalls Teil eines Knotens sind, eine unterschiedliche Breite haben können, muss man diese noch zentrieren. Dies wird dadurch gemacht, dass man von der x-Koordinate die Hälfte der Breite des Namens des Knotens abzieht. Bei der y-Koordinate wird die Hälfte der Höhe des Knotens, die stets konstant bleibt, abgezogen. Damit bilden alle Knoten im Graphen einen übersichtliche Ring-Struktur, siehe Abbildung 34.

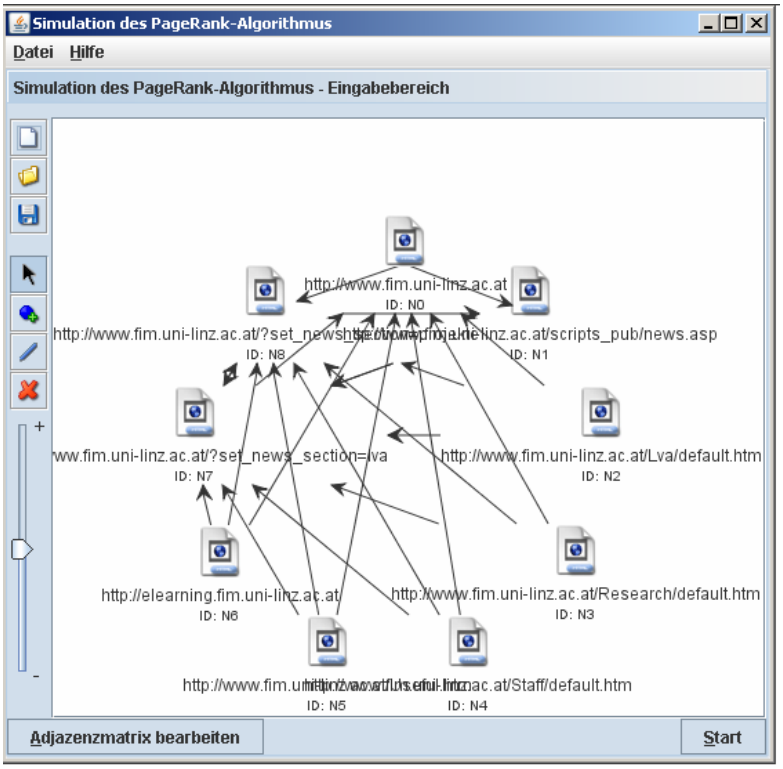

Abbildung 34: Ringförmige Anordnung der Knoten eines Graphen

Die Übersichtlichkeit bezieht sich dabei nur auf die Knoten selbst und nicht auf die Kanten. Diese können sich, wie besonders auch in Abbildung 34 ersichtlich, sehr oft überkreuzen. Eine optimale Anordnung der Knoten in einer Struktur, die möglichst wenige Überschneidungen der Kanten enthält, wurde dabei nicht implementiert. Die Ring-Struktur kann aber natürlich wieder manuell bearbeitet werden.

7.1.3.7.2. Exportieren

Der Export der Knoten kann als XML-Datei erfolgen. Diese XML-Datei sieht äquivalent zu jener aus, die vor dem Laden des Graphen in das Applet temporär erzeugt wird. Diese

XML-Datei enthält ebenfalls dieselben Koordinaten und kann damit auch zu einem späteren Zeitpunkt, ohne vorheriges Durchlaufen des Wizard, in das Applet geladen werden.

Werden die Koordinaten einzelner Knoten im Applet dann verändert und wieder abgespeichert, werden natürlich die veränderten Koordinaten übernommen. Die Koordinaten für die Anordnung in einer Ring-Struktur werden also nur beim erstmaligen Export aus dem Wizard berechnet.

8. Evaluierung der implementierten Simulationen

In diesem Kapitel werden die gefundenen Lösungen und die erstellten Programme nochmals abschließend betrachtet und evaluiert. Dabei sollen vor allem die Grenzen und Erweiterungsmöglichkeiten aufgezeigt werden.

8.1. Evaluierung des Java Applets

Bei dem Applet wurden einige Restriktionen zu einer übersichtlicheren und einfacheren Handhabung eingebaut:

- Im Graphen selbst wurde die Anzahl der Knoten, welche im Graphen enthalten sein können, beschränkt. Dies hat den Sinn, dass die Visualisierung übersichtlich bleibt. Für das Verständnis des PageRank-Algorithmus genügt es, dass dieser auf kleinen Graphen angewendet wird.
- Weiters werden Links einer Webseite auf sich selbst nicht gestattet. Eine Webseite kann zwar grundsätzlich einen Link auf sich selbst besitzen, doch wird dies bei der Berechnung des PageRank nicht berücksichtigt.
- Bei der Visualisierung wurde der Parameter c auf Werte zwischen 0.05 und 0.95 beschränkt. Damit soll gewährleistet sein, dass der Benutzer die Visualisierung nur in einem sinnvollen Wertebereich für c starten kann. Außerdem können allfällig auftretende Effekte (siehe Kapitel 4.6) auch bei diesem Einstellungsbereich beobachtet werden.

- Die Anzahl der Durchläufe für die iterative Berechnung wurde ebenfalls beschränkt. Spätestens nach einer geringen Anzahl an Iterationen ist gewährleistet, dass der Algorithmus konvergiert bzw. die PageRanks mit einer sehr hohen Genauigkeit berechnet wurden. Außerdem würde eine Berechnung der PageRanks mit einer großen Anzahl Iterationen im Applet eine sehr lange Zeit in Anspruch nehmen.

Daneben könnte man noch einige Erweiterungen hinzufügen, um den Lerneffekt noch weiter zu erhöhen oder die Visualisierung besser darzustellen:

- So könnte die Anzahl der Farben, mit denen ein Knoten im Graphen bei der Visualisierung assoziiert wird, erhöht werden. Derzeit werden dabei nur sechs verschiedene Farben verwendet, wodurch eine optimale farbliche Unterscheidung oft nicht klar gegeben ist.
- Bisher sind bei jedem Knoten nur ein Name und eine ID angegeben. Als Name wird normalerweise die URL verwendet, kann jedoch auch jeden beliebigen anderen (aber für alle Knoten im Graphen einzigartigen) Text enthalten. Eine sinnvolle Erweiterung wäre, dass man, neben der URL und der ID, auch den Seitentitel hinzunimmt.
- Eine andere Erweiterungsmöglichkeit wäre, dass der Benutzer auswählt, welcher Ranking-Algorithmus verwendet werden soll. Das bedeutet, dass das Applet nicht nur auf die Visualisierung des klassischen PageRanks beschränkt wäre, sondern eine Vielzahl alternativer Bewertungsmethoden visualisieren kann.
- In der derzeitigen Visualisierung wird die Berechnung des PageRank einer Webseite als Gleichung angegeben. Hier könnte man zusätzlich auch die Berechnung des Eigenvektors mit Matrizen und Vektoren darstellen.
- Der PageRank wird derzeit bis auf zwei Nachkommastellen genau angegeben. Um feine Unterschiede der PageRanks bei großen Graphen besser erkennen zu können, wäre ev. eine Erhöhung der Nachkommastellen sinnvoll. Dies wäre jedoch nur dann relevant, wenn auch die Anzahl der maximalen Knoten im Graphen erhöht wird, da ansonsten für kleinere Graphen kaum ein Unterschied wäre.

- Eine Möglichkeit, den Graphen selbst noch übersichtlicher zu gestalten wäre, dass die Knoten automatisch in verschiedenen Strukturen (Ring, Feld, Baum, etc.) angeordnet werden können.
- Eine andere Erweiterung wäre, bei der Berechnung der PageRanks auf das „Dangling Links“ Problem einzugehen, beispielsweise indem man auch die Entfernung dieser Webseiten vor der Berechnung und das Hinzufügen danach visualisiert.

8.2. Evaluierung der Websimulation

Grundsätzlich gilt für die Restriktionen in der Websimulation dasselbe wie für jene im Applet (Normalisierungsfaktor c, Anzahl der Iterationen bei der PageRank-Berechnung. Anzahl der Nachkommastellen der berechneten PageRanks, Dangling Links Problem). Weitere Restriktionen oder Erweiterungsmöglichkeiten in der Websimulationen sind:

- Die Suchtiefe des Webcrawlers ist beschränkt: Da dieser nur eine kleine Gruppe von Webseiten sammeln können muss, sollte die Suchtiefe bis zur 4. Ebene ausreichen.
- Die maximale Ebene der Suchtiefe wäre aber durchaus erweiterbar. Durch eine effizientere Gestaltung des Webcrawler, beispielsweise durch Multithreading, könnte man rasch eine viel höhere Suchtiefe erreichen, wenn nicht sogar alle verlinkten Webseiten im WWW auffinden. Dazu würde man aber natürlich auch die notwendigen Hardwareressourcen benötigen.
- Danach wäre auch eine Erweiterung des Programms zu einer Art lokaler Suchmaschine möglich, mit der man beispielsweise die Webseiten im Anschluss mit einem personalisierten PageRank (siehe Kapitel 4.6) bewerten könnte. Dies würde jedoch für einen privaten PC mit der heutigen Technologie zu große Ressourcen in Anspruch nehmen.
- Natürlich wäre hier auch eine Erweiterung des Wizard für andere Ranking-Algorithmen denkbar. Dies würde für die Forschung im Bereich der Suchmaschinenbewertung eventuell große Vorteile bringen. So könnten Tests für einen neu entwickelten Ranking-Algorithmus mit diesem Wizard rasch und standardisiert durchgeführt werden.

9. Ausblick

Aus heutiger Sicht bieten Suchmaschinen die Möglichkeit einer raschen und effizienten Suche im WWW. Dies wird vor allem dadurch gewährleistet, dass die Algorithmen, mit denen Webseiten von einer Suchmaschine bewertet werden, ständig weiterentwickelt und angepasst werden. Die genauen Algorithmen sind dabei meist unbekannt und benutzen oft eine Mischung aus mehreren einzelnen Algorithmen.

Es besteht jedoch ein ständiger Wettlauf zwischen den Betreibern der Suchmaschinen und den Personen, die versuchen aus wirtschaftlichen Interessen durch Suchmaschinenoptimierung ihre Webseite(n) an die erste Stelle von Suchergebnissen zu bringen. Dem versucht man auch zu entgegnen, indem man Webseiten, die gegen die Richtlinien (z.B.: [GOOGLE_3]) einer Suchmaschine verstoßen, einfach aus den Suchergebnissen ausschließt. Damit müssen sich die Suchmaschinenoptimierer an sehr strenge Regeln halten, mit denen sie zwar eventuell einen etwas besseren Rang erreichen können, jedoch diesen grundsätzlich nur in geringem Ausmaße beeinflussen können.

Die Technologie zur Suchmaschinenbewertung ist heutzutage bereits sehr ausgereift und auch die Probleme, wie sie Mitte der Neunziger Jahre des 20. Jahrhunderts bestanden, nicht mehr so gravierend. Daher ist aus heutiger Sicht in nächster Zeit nicht mit einer neuen „Revolution“ bei der Suchmaschinenbewertung zu rechnen, wie sie der PageRank-Algorithmus auslöste.

LITERATUR- UND QUELLENVERZEICHNIS

[BRANDZ_1] THE WPP BRAND EQUITY STUDY: BrandZ Top 100 Most Powerful Brands
URL: www.brandz.com (letzter Aufruf: 26.04.07)

[BUSH_1] V. Bush: *„As We May Think“*, The Atlantic Monthly in July 1945
URL: http://www.theatlantic.com/doc/194507/bush (letzter Aufruf: 1.05.07)

[CAPTCHA_1] The CAPTCHA Project: CAPTCHA Tests
URL: http://www.captcha.net/captchas/ (letzter Aufruf: 3.05.07)

[COHEN_1] B. Cohen: The BitTorrent Protocol
URL: http://www.bittorrent.org/protocol.html (letzter Aufruf: 15.05.07)

[DIEPRESSE_1] DiePresse.com: Grasser: *„Opfer von Google-Guerilla“*, 16.12.2004
URL: http://diepresse.com/home/techscience/internet/174570/index.do (letzter Aufruf: 15.05.07)

[EICHMANN_1] D. Eichmann: *„The RBSE Spider - Balancing Effective Search Against Web Load“*

[FARAHAT_1] A. Farahat, T, Lofaro, J. Miller, G. Rae, L. Ward: *„AUTHORITY RANKINGS FROM HITS, PAGERANK, AND SALSA: EXISTENCE, UNIQUENESS, AND EFFECT OF INITIALIZATION“*

[FIM_1] A. Paramythis; S. Loidl; J. Mühlbacher; M. Sonntag: „*A Framework for Uniformly Visualizing and Interacting with Algorithms in E-Learning“,* in: T. C. Montgomerie, J. R. Parker (Eds.): The IASTED International Conference on Education and Technology (ICET) 2005, Calgary, pp. 28 - 33, Anaheim: ACTA Press 2005; ISBN 0-88986-487-X (2005)

[GOOGLE_1] Die Suchmaschine Google: Google Werbeprogramme
URL: http://www.google.com/ads/index.html (letzter Aufruf: 7.05.07)

[GOOGLE_2] Die Suchmaschine Google: Google Funktionen
URL: http://www.google.com/intl/de/help/features.html#cached (letzter Aufruf: 3.05.07)

[GOOGLE_3] Die Suchmaschine Google: Richtlinien für Webmaster
URL: http://www.google.com/support/webmasters/bin/answer.py?answer=35769&hl=de (letzter Aufruf: 3.05.07)

[HEISE_1] J. Bager; J. Becker-Fochler: *„Forellenschwemme“*
URL: http://www.heise.de/ct/SEO-Wettbewerb/Analyse.shtml (letzter Aufruf: 3.05.07)

[HEISE_2] J. Bager; J. Becker-Fochler: *„Dicke Fische“*
URL: http://www.heise.de/ct/SEO-Wettbewerb/Analyse2.shtml (letzter Aufruf: 3.05.07)

[HEISE_3] heise online: *„Google überrascht bei der Suche nach erbärmlichen Versagern“*
URL: http://www.heise.de/newsticker/meldung/42679 (letzter Aufruf: 15.05.07)

[IETF_1] T. Berners-Lee, R. Fielding, L. Masinter: *„Uniform Resource Identifiers (URI): Generic Syntax (1998)“*
URL: http://www.ietf.org/rfc/rfc2396.txt (letzter Aufruf: 20.04.07)

[KLEINBERG_1] J. Kleinberg: *„Authoritive Sources in a Hyperlinked Environment“*

[KNOBLOCK_1] C. Knoblock: „*Searching the World Wide Web“*, IEEE Expert January-February 1997, pp. 8-11

[KOSTER_1] M. Koster: *„A Standard for Robot Exclusion“*
URL: http://www.robotstxt.org/wc/norobots.html (letzter Aufruf: 1.05.07)

[LEMPEL_1] R. Lempel; S. Moran: *„The stochastic approach for link-structure analysis (SALSA) and the TKC effect“*

[MACCLUER_1] C. MacCluer: *„The many proofs and applications of Perron's Theorem“*, SIAM Review 42 2000, pp. 487-498.

[MUEHLBACHER_1] J. Mühlbacher; G. Pilz; M. Widi: *„Mathematik explorativ Ausgewählte Kapitel der Algebra, Zahlentheorie und Graphentheorie für Anwendungen in der Informatik“,* ISBN 3-85499-070-7, Trauner Verlag

[NELSON_1] T. Nelson: PROJECT XANADU
URL: http://xanadu.com/ (letzter Aufruf: 1.05.07)

[NETPLANET_1] Das Internet Lexikon NETPLANET: Archie
URL: http://www.netplanet.org/dienste/archie.shtml (letzter Aufruf: 1.05.07)

[NETPLANET_2] Das Internet Lexikon NETPLANET: Gopher
URL: http://www.netplanet.org/dienste/gopher.shtml (letzter Aufruf: 1.05.07)

[NETPLANET_3] Das Internet Lexikon NETPLANET: Veronica
URL: http://www.netplanet.org/dienste/veronica.shtml (letzter Aufruf: 1.05.07)

[NICHOLSON_1] S. Nicholson: *„Indexing and abstracting on the World Wide Web: An examination of six Web databases“*

[PAGE_1] L. Page; S. Brin; R. Motwani; T. Winograd: *„The PageRank Citation Ranking: Bringing Order to the Web“*

[PAGE_2] L. Page; S. Brin: *„The Anatomy of a Large-Scale Hypertextual Web Search Engine“,* WWW7 / Computer Networks 30 (1-7): pp. 107-117

[RICHARDSON_1] M. Richardson; Pedro Domingos: *„The Intelligent Surfer: Probabilistic Combination of Link and Content Information in PageRank“*

[W3C_1] World Wide Web Consortium: HTML 4.01 Specification
URL: http://www.w3.org/TR/html401/ (letzter Aufruf: 20.04.07)

[W3C_2] World Wide Web Consortium: Markup Validation Service
URL: http://validator.w3.org/ (letzter Aufruf: 20.04.07)

[XING_1] W. Xing; A. Ghorbani: *„Weighted PageRank Algorithm“*, Proceedings of the Second Annual Conference on Communication Networks and Services Research (CNSR'04)

TABELLENVERZEICHNIS

ALGORITHMENVERZEICHNIS

CODEVERZEICHNIS

ABBILDUNGSVERZEICHNIS

Curriculum vitae

Schulische/Universitäre Ausbildung

Seit 2006: Masterstudium Informatik an der JKU Linz

2003 bis 2006: Bakkalaureatsstudium Informatik an der JKU Linz

Seit 2003: Studium der Rechtswissenschaften an der JKU Linz

Seit 2005: Studium der Wirtschaftswissenschaften an der JKU Linz

1993 bis 2002: Bundesrealgymnasium Landwiedstraße, Naturwissenschaftlicher Zweig

1989 bis 1993: Volksschule Keferfeld

Präsenzdienst

2002 bis 2003: Stabskompanie Hörsching

Berufliche Aktivitäten

Seit September 2005: Wiss. Mitarbeiter im Forschungsbetrieb ohne Diplom am Institut für Informationsverarbeitung und Mikroprozessortechnik an der JKU Linz

Sommersemester 2007: Wiss. Mitarbeiter im Lehrbetrieb ohne Diplom am Institut für Systemsoftware an der JKU Linz

Wintersemester 2006/2007: Wiss. Mitarbeiter im Lehrbetrieb ohne Diplom am Institut für Wissensbasierte mathematische Systeme an der JKU Linz

Sommersemester 2006: Wiss. Mitarbeiter im Lehrbetrieb ohne Diplom am Institut für Systemsoftware, Institut für Pervasive Computing und Institut für Integrierte Schaltungen an der JKU Linz

Wintersemester 2005/2006: Wiss. Mitarbeiter im Lehrbetrieb ohne Diplom am Institut für Wissensbasierte mathematische Systeme, Institut für Pervasive Computing und Institut für Systemsoftware an der JKU Linz

Sommersemester 2005: Wiss. Mitarbeiter im Lehrbetrieb ohne Diplom am Institut für Systemsoftware an der JKU Linz

Wintersemester 2004/2005: Wiss. Mitarbeiter im Lehrbetrieb ohne Diplom am Institut für Wissensbasierte mathematische Systeme an der JKU Linz

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Magisterarbeit selbstständig, und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Des Weiteren versichere ich, dass ich diese Magisterarbeit weder im In- noch im Ausland in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

Linz, am 21.05.2007