Bauhaus-Universität Weimar
Fakultät Medien
Studiengang Mediensysteme

Near-Duplicates im ClueWeb12

Eine Pilotstudie zur Übertragbarkeit von Forschungsergebnissen

Bachelorarbeit

Jan Graßegger
Geboren am 18.05.1987 in Kassel

Matrikelnummer 80483

1. Gutachter: Junior-Prof. Dr. Matthias Hagen
Betreuer: Junior-Prof. Dr. Matthias Hagen & Dr. Martin Potthast

Datum der Abgabe: 29. Juli 2014

Erklärung

Hiermit versichere ich, daß ich diese Arbeit selbständig verfaßt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Weimar,

den 29. Juli 2014 .....

Jan Graßegger

Zusammenfassung

In dieser Arbeit stellen wir die Ergebnisse zu der Frage vor, ob Forschungsergebnisse zwischen Korpora übertragbar sind. Eine Übertragbarkeit von Ergebnissen würde es Forschungsgruppen ermöglichen, alte Ergebnisse mit neuen zu vergleichen und Relevanzurteile, die für alte Korpora erstellt wurden, weiterhin zu nutzen. Dazu haben wir erstmalig eine Pilotstudie zu diesem Thema, anhand der beiden ClueWeb-Korpora, durchgeführt. Wir haben 155.419 Relevanzurteile der TREC-Konferenzen, die für das ClueWeb09 erstellt wurden, im ClueWeb12 gesucht.

Wir erforschten dazu unter anderem, wie viele Dokumente eine gleiche URL besitzen, welches Verfahren sich am besten zur Suche nach Near-Duplicates in Web-Korpora eignet und welche Parametrisierung für dieses Verfahren optimal ist. Dabei fanden wir mithilfe des SimHash-Verfahrens heraus, dass eine Übertragbarkeit von Forschungsergebnissen nur sehr eingeschränkt möglich ist, weil weniger als 10 Prozent der Relevanzurteile Duplikate im ClueWeb12 besitzen.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Diese Arbeit ist eine Pilotstudie, die abschätzt, in welchem Umfang sich Forschungsergebnisse, die mit älteren Web-Korpora erstellt wurden, auf aktuelle Korpora übertragen lassen.

Web-Korpora sind statische Sammlungen von Dokumenten aus dem Web. Jedes Dokument eines Web-Korpus bekommt eine ID zugeordnet, die es eindeutig im Korpus identifiziert. Web-Korpora werden in Workshops genutzt, welche im Rahmen von Konferenzen stattfinden. Bekannte Konferenzen sind die Text REtrieval Conference und das Cross-Language Evaluation Forum, in dessen Rahmen der PAN Workshop veranstaltet wird. Die Workshops dienen dazu, eigene Forschungsergebnisse mit denen anderer Forschungsgruppen zu vergleichen und die Performance eigener Verfahren zu evaluieren. Die Aufgabe der Teilnehmer ist es, gegebene Fragestellungen mithilfe eigener Retrieval-Verfahren zu beantworten. Die Teilnehmer reichen als Lösung Listen von Dokument-IDs ein. Die Dokumente müssen deshalb Teil des verwendeten Korpus sein. Die Veranstalter des Workshops überprüfen, inwieweit die eingereichten Dokumente der Fragestellung entsprechen. Die Erstellung dieser Relevanzurteile bedeutet einen sehr großen Aufwand, weil dies nicht maschinell erfolgen kann.

Wird ein neues Korpus veröffentlicht, bekommen die Dokumente in diesem Korpus eine neue ID zugeordnet, auch wenn diese Dokumente bereits in existierenden Korpora enthalten waren. Ergebnisse, die mit einem neuen Korpus erstellt werden, können nicht mit existierenden verglichen werden. Die Relevanzurteile, die mit sehr viel Aufwand erstellt wurden, sind nicht auf den neuen Korpus anwendbar und sind damit verloren.

In unserer Pilotstudie untersuchen wir erstmals, ob und in welchem Umfang Verfahren zur Erkennung von Duplikaten von Web-Dokumenten dazu verwendet werden können, die Relevanzurteile eines alten Korpus auf ein neues zu übertragen. Dazu führen wir eine Reihe von Untersuchungen zwischen

den ClueWeb-Korpora durch. Die ClueWeb-Korpora sind die derzeit größten, in Workshops genutzten Korpora und beinhalten jeweils über eine halbe Milliarde Web-Dokumente. Des weiteren ist das ClueWeb12 der direkte Nachfolger des ClueWeb09. Es besteht somit einerseits der Bedarf nach der Übertragbarkeit von Forschungsergebnissen und andererseits eine Chance für gemeinsame Dokumente in beiden Korpora. Wir nutzen diese beiden Korpora, um die folgenden Forschungsfragen zu klären:

1. Wie viele URLs von Dokumenten des ClueWeb09 finden sich auch im ClueWeb12?
2. Welche Verfahren zur Erkennung sind gegenwärtig für die Erkennung von duplizierten Web-Inhalten einsetzbar?
3. Welche Parametrisierung des besten Verfahrens erzielt die besten Ergebnisse?

Unsere Ergebnisse zeigen, dass 20 Prozent der untersuchten Relevanzurteile gemeinsame URLs in beiden Korpora besitzen. 12,8 Prozent des ClueWeb09 befindet sich im ClueWeb12. Bei der genaueren Untersuchung dieser Ergebnisse fiel auf, dass der Anteil der Wikipedia im ClueWeb12 deutlich geringer ist, als im ClueWeb09. Liegt der Anteil im ClueWeb09 noch bei 1,2 Prozent, sind dies im ClueWeb12 nur noch 0,0026 Prozent. Selbstverständlich bedeutet die Gleichheit einer URL nicht, dass die Dokumente, die jeweils im ClueWeb09 und ClueWeb12 unter dieser URL zu finden sind, auch inhaltlich gleich sind.

Wir vergleichen das SimHash-Verfahren, dass mit großem Erfolg zur Erkennung von sehr ähnlichen Objekten aller Mediengattungen eingesetzt wird, mit bekannten Verfahren aus der Literatur, wie MinHash, das spezifisch zur Erkennung von Web-Duplikaten eingesetzt wird. Es zeigt sich, dass SimHash, gegenüber den existierenden Verfahren, einen geringeren Speicheraufwand benötigt und durch seine Parametrisierungsmöglichkeiten deutlich flexibler ist.

Wir führen mehrere Experimente durch, die verschiedene Parameter von SimHash für die Erkennung von Duplikaten von Web-Dokumenten, sowie die Skalierbarkeit des Verfahrens auf das gesamte ClueWeb betreffen. Wir zeigen, dass wir die Ergebnismenge von SimHash mit der Kombinationen verschiedener N-Grammlängen bei der Erzeugung des Algorithmus deutlich reduzieren können, ohne einen größeren Qualitätsverlust hinnehmen zu müssen. Wir zeigen aber auch, dass eine eindeutige Aussage, über die Parametereinstellungen des Algorithmus nicht getroffen werden kann, weil die Parameter zu einem großen Teil von der Vergleichsmenge und den Kapazitäten des genutzten Systems abhängen.

Insgesamt deuten die erzielten Ergebnisse jedoch darauf hin, dass für weniger als 10 Prozent aller 155.000 Dokumente, für die Relevanzurteile für das

ClueWeb09 vorliegen, auch entsprechend Duplikate im ClueWeb12 aufzufinden sind. Es handelt sich dabei um eine Schätzung basierend auf der Analyse von einem Viertel des ClueWeb09s mit SimHash. Ausgehend von dieser Erkenntnis schlagen wir weitere mögliche Schritte vor, um die Zahl der Dokumente mit Relevanzurteilen aus dem ClueWeb09 für die Duplikate im ClueWeb12 gefunden werden können zu erhöhen. Ein Möglichkeit besteht darin, die Dokumente nicht vollständig zu betrachten, sondern ihren Inhalt zunächst zu filtern, um zum Beispiel nur die Hauptinhalte einer Webseite für die Feststellung, ob ein Web-Dokument ein Duplikat eines anderen ist, heranzuziehen.

Der Hauptteil der Arbeit ist wie folgt gegliedert: In Kapitel 2 betrachten wir zunächst Forschungsergebnisse zur Veränderung des Web, sowie die Entstehung der verwendeten Web-Korpora. In Kapitel 3 stellen wir die Near-Duplicate Suchalgorithmen MinHash und SimHash vor und vergleichen diese miteinander. In Kapitel 4 betrachten wir die Ergebnisse zu Experimenten der URL-Überschneidung und in Kapitel 5 stellen wir unsere Ergebnisse mit dem SimHash-Algorithmus vor. In Kapitel 6 fassen wir unsere Ergebnisse abschließend zusammen und geben einen Ausblick auf zukünftige Forschungsfragen.

Kapitel 2

Verwandte Arbeiten

Diese Arbeit beschäftigt sich mit der Frage, inwieweit Web-Korpora, die zeitlich weit auseinanderliegend gecrawled wurden, übereinstimmen. Somit sind Arbeiten, die sich mit der Veränderung von Seiten sowie Near-Duplicates im Internet beschäftigen, ein möglicher Indikator für die Entwicklung, die man zwischen Web-Korpora beobachten kann. Eine weitere Rolle spielen die Verfahren mit denen die Korpora erstellt wurden, worauf ebenfalls in diesem Kapitel eingegangen wird. Ein grundlegendes Element dieser Arbeit sind Algorithmen zur Erkennung von Near-Duplicates. Deshalb ist diesem Themenkomplex ein eigenes Kapitel gewidmet (siehe Kapitel 3).

2.1 Beobachtungen im World Wide Web

Sowohl der Anteil von Near-Duplicates im Web als auch die Lebensspanne und Veränderungsrate von Webseiten sind schon lange Themen, mit denen sich Forschungsgruppen beschäftigen. Ergebnisse aus diesem Bereich sind vor allem für Suchmaschinenanbieter höchst interessant, da sie beispielsweise klären, wie häufig eine Seite gecrawled werden muss, um die neuste Version vorhalten zu können, welche Dokumente man bei der Präsentation von Suchergebnissen ignorieren kann, oder wie viel gespart wird, wenn man Near-Duplicates nicht mehrfach speichert.

Für diese Arbeit ist interessant, wie langlebig Webseiten sind und wie groß der Anteil von Near-Duplicates im Web ist. In den letzten 20 Jahren gab es diverse Studien zu diesem Thema. Bereits 1997 evaluierten Broder et al. [BGMZ97] den MinHash Algorithmus (siehe Abschnitt 3.1) mit 30 Mio. Dokumenten, die für die AltaVista-Suchmaschine gecrawled wurden, und fanden heraus, das nahezu 50 Prozent der analysierten Dokumente Duplikate im gleichen Korpus besaßen. Im gleichen Jahr analysierten Douglis et al. [DFKM97] die Zugriffsprotokolle zweier Unternehmen. Ihre Ergebnisse zeigten eine Kor-

relation zwischen den Zugriffsintervallen der Nutzer und der Veränderungsrate der Webseiten: Je größer die Zugriffszahlen einer Seite, desto häufiger ändert sich der Inhalt. Die Häufigkeit von Änderungen steht ebenfalls im Zusammenhang mit der Top-Level-Domain: Eine Webseite deren Domain mit `.edu` endet ist im Durchschnitt älter als eine, die mit `.com` endet. Interessant ist ebenfalls, dass 18 Prozent der Zugriffe auf Dokumente erfolgte, die auch unter anderen URLs aufgerufen wurden, also Duplikate waren.

Um Veränderungen sehen zu können, ist es wichtig ein Verhalten über längere Zeiträume zu beobachten. Cho et al. [CGM00] beobachteten 720.000 Webseiten auf über 250 Domains über einen Zeitraum von 50 Tagen. Die Dokumente wurden aufgrund hoher PageRank-Werte ausgewählt. 40 Prozent der Seiten veränderten sich innerhalb einer Woche. 10 Prozent der Seiten waren weniger als eine Woche erreichbar. Jedoch waren mehr als 70 Prozent noch nach mehr als einem Monat erreichbar. Die Beobachtung, dass die Domain eine Rolle für die Veränderungsrate spielt, wurde ebenfalls bestätigt: `.gov` und `.edu` waren langlebiger als Seiten die auf `.com` endeten.

Über einen deutlich längeren Zeitraum hinweg beobachtete Koehler [Koe02] insgesamt 361 URLs. Von Dezember 1996 bis Februar 2001 wurde jede Woche versucht diese Seiten herunterzuladen. Die URLs verteilten sich über mehrere Top-Level-Domains. Am Ende dieses Versuchs waren noch 3 Prozent der URLs erreichbar. 7 Prozent der URLs änderten sich mit einer Weiterleitung auf eine andere URL und war eine URL nicht erreichbar, tauchte sie mit einer Wahrscheinlichkeit von 85 Prozent wieder auf. Die Halbwertszeit einer Domain lag bei diesem Versuch bei etwa 2 Jahren. Es ließ sich aber ebenfalls feststellen: Je älter eine Webseite ist, desto wahrscheinlicher ist es, dass es sie auch noch länger geben wird.

Eine vom Umfang der Webseiten her, deutlich größere Studie führten Fetterly et al. [FMN03, FMNW03] durch. Über einen Zeitraum von elf Wochen luden sie 150 Mio. Webseiten herunter und analysierten sie in Hinsicht auf Veränderung und Duplikate. Als Ähnlichkeitsmaß wurde MinHash genutzt. Zwei Dokumente galten als gleich, wenn zwei Super Shingles (siehe Kapitel 3) übereinstimmten. Als Basis für den ersten Crawl diente die Yahoo! Webseite. Es wurde eine Breitensuche durchgeführt, welche Seiten mit einem hohen PageRank bevorzugte. Neben den bereits zuvor erzielten Erkenntnissen in Bezug auf Top-Level Domain und Häufigkeit von Änderungen stellten sie ebenfalls eine Korrelation zwischen der Menge an Seiten, die zu einer Domain gehören, und der Änderungsrate fest: Domains mit mehr Seiten ändern sich öfter als solche mit weniger. Die meisten Änderungen sind aber nur unwesentlich. 57 Mio. (29,2 Prozent) Dokumente waren Near-Duplicates der restlichen Dokumente. Das größte Cluster bildeten eine Million Near-Duplicates einer pornographi-

schen Webseite. Von den 150 Mio. Seiten waren 88 Prozent nach elf Wochen noch erreichbar.

Ntoulas et al. [NCO04] beobachteten 150 Domains über einen Zeitraum von einem Jahr. Ihr Fokus lag dabei nicht nur auf der Veränderung des Inhalts, sondern auch auf der Veränderung der Linkstruktur. Die Domains wählten sie anhand ihrer Popularität aus. Dies waren für sie die Top 5 jeder Kategorie von Google Directory.[1] Sie stellten fest, dass der Umfang der Änderung einer Webseite nicht mit der Häufigkeit von Änderungen in Zusammenhang steht. Wie in anderen Studien zuvor waren auch in dieser die meisten Änderungen nur minimal. 8 Prozent der URLs, die jede Woche heruntergeladen wurden, waren neu. Nach sechs Wochen waren nur noch die Hälfte der URLs des ersten Crawls erreichbar, nach einem Jahr noch 40 Prozent.

Kim und Lee [KL05] überwachten über einen Zeitraum von 100 Tagen zwischenzeitlich 3 Mio. URLs. Dazu wählten Sie die 4.000 beliebtesten Domains aus Korea und dazu 30.000 zufällige koreanische Domains. Sie crawlten pro Seite maximal 3.000 URLs mit einer maximimalen Tiefe von neun. Pro Crawl fanden sie im Durchschnitt 1,3 Prozent neue URLs. Nach 100 Tagen bestand ihr Korpus zu 40 Prozent aus neuen URLs. In dem Zeitraum des Crawls veränderten sich 73 Prozent der Webseiten nicht. Weniger als 5 Prozent veränderten sich alle zwei Tage. Es wurden aber keine festen Aktualisierungszyklen erkannt. Die Erreichbarkeit von URLs war ebenfalls sehr hoch: Wenn der Download einer URL erfolgreich war, setzte sich dies auch in neun von zehn darauffolgenden Versuchen fort.

Einen anderen Ansatz zur Untersuchung von Veränderungen von Webseiten wählten Olston und Pandey [OP08]. Sie versuchten die Langlebigkeit von Informationen zu analysieren und konstruierten daraus einen Algorithmus zur Optimierung von Crawls. Dieser Algorithmus beobachtet, wie lange einzelne Fragmente auf einer Seite erhalten bleiben. Sind Fragmente nur sehr kurz auf einer Seite präsent, handelt es sich vermeintlich nicht um relevanten Inhalt, sondern um Werbung oder ähnliches. Kommen jedoch regelmäßig neue Fragmente hinzu und bleiben für einen längeren Zeitraum erhalten, handelt es sich wahrscheinlich um neue relevante Inhalte und die Seite muss neu heruntergeladen werden. Fragmente sind für diesen Algorithmus gleichbedeutend mit Shingles. Zur Evaluation ihres Algorithmus beobachteten sie für einen Monat 10.000 URLs des OpenDirectory. Sie stellten dabei fest, dass die Langlebigkeit von Informationen nicht mit der Häufigkeit von Änderungen einer Webseite zusammenhängt. Ein Großteil der Seiten ändert sich nur minimal. Es findet aber keine Änderung am Inhalt statt.

[1] *Google Directory* ist seit 2011 abgeschaltet. Der genutzte Datensatz kann aber noch im Webarchiv der UCLA heruntergeladen werden: `http://webarchive.cs.ucla.edu`

Adar et al. [ATDE09] wählten wieder einen anderen Ansatz bei der Auswahl der analysierten Webseiten. Ziel war es Seiten zu beobachten, die regelmäßig besucht werden. Dazu nutzten sie die Protokolle von Nutzern der Live Search Toolbar. Sie wählten 55.000 URLs aus einem Pool von Seiten, die in einem Zeitraum von 5 Wochen von 612.000 Nutzern besucht wurden. Die Auswahl wurde anhand der Anzahl der Besucher, der Regelmäßigkeit der Besuche und der Anzahl an wiederholten Besuchen getroffen. Die Veränderung von Webseiten wurde mit dem Dice-Koeffizienten gemessen. 34 Prozent der Seiten veränderten sich innerhalb des Beobachtungszeitraums überhaupt nicht. Die anderen Seiten veränderten sich im Durchschnitt alle 123 Stunden mit einer durchschnittlichen Ähnlichkeit von 0,8 zu ihrer vorigen Version. `.gov` und `.edu` Domains veränderten sich im Durchschnitt, wie schon zuvor beobachtet, seltener als andere. URLs mit einem tieferen Pfad veränderten sich seltener als mit einem kürzeren und Unternehmensseiten veränderten sich seltener als andere Kategorien von Seiten. Aber, wenn sie sich veränderten, dann deutlich stärker als der Rest. Ansonsten konnte aber keine Aussage über den Zusammenhang von Häufigkeit von Änderungen und ihrem Umfang getroffen werden.

Diese Studien zur Veränderung des World Wide Webs, die einen Zeitraum von über zehn Jahren abdecken, unterscheiden sich zwar in ihrem Ansatz, den verwendeten Seiten und in der Länge des untersuchten Zeitraums, aber dennoch lassen sich einige Gemeinsamkeiten erkennen:

- Die Top Level Domains korrelieren mit der Langlebigkeit von URLs. `.gov` und `.edu` sind deutlich langlebiger als URLs von `.com` Domains.
- Die Häufigkeit von Änderungen korreliert mit der Anzahl der Besucher. Je häufiger eine Seite besucht wird, desto häufiger verändert sie sich auch.
- Die meisten Änderungen von Webseiten sind nur minimal und betreffen selten den eigentlichen Inhalt.
- Je länger eine Seite bereits online ist, desto wahrscheinlicher ist es, dass sie auch weiterhin erreichbar bleibt.

Diese Beobachtungen können ein Indikator für die möglichen Veränderungen zwischen Web-Korpora sein. Jedoch bilden Web-Korpora nicht das ganze Web ab, sondern sind nur ein Ausschnitt. Deshalb ist das Vorgehen bei der Konstruktion von Web-Korpora ein wichtiger Faktor für deren Ähnlichkeit.

2.2 Enstehung von Web-Korpora

Um einschätzen zu können, inwieweit Teile älterer Web-Korpora im ClueWeb12 enthalten sind, ist es zunächst notwendig sich damit auseinanderzu-

Name	Anzahl der Webseiten	Crawl-Zeitraum
ClueWeb12	733.019.372	2012
ClueWeb09 (Kategorie A)	503.903.810	Januar bis Februar 2009
Common Crawl	3.8 Milliarden	Stand Mai 2013
.GOV[2]	1.246.518	2002
DOTGOV2	25.205.179	2004
WT10G[3]	1.692.096	2000

Tabelle 2.1: Bekannte Web-Korpora, ihre Größe und der Zeitraum des Crawls.

setzen, wie sich diese Korpora zusammensetzen. Eine Übersicht über die bekannten Web-Korpora findet sich in Tabelle 2.1.

ClueWeb09 Das ClueWeb09 ist ein Korpus von 1 Milliarde Webseiten, die Hälfte davon in englischer Sprache.[4] Gecrawlt wurde mit dem Sapphire Web Crawler, der am Language Technology Institute der Carnegie Mellon Universität entwickelt wird.[5] Dieser basiert auf Nutch,[6] einem Java Web-Crawler. Erstellt wurde das Korpus von Januar bis Februar 2009.

Als Startseiten für den Crawl dienten 33 Mio. URLs. 28 Mio. davon sind englischsprachig.[7] Die restlichen 5 Mio. sind Dokumente in anderen Sprachen. 20 Mio. dieser URLs stammten aus einem ein Jahr zuvor durchgeführten rein englischsprachigen Crawl. Dieser Crawl beinhaltet 200 Mio. Webseiten. In der Dokumentation heißt es, diese Webseiten seien nicht repräsentativ. Deshalb wurden davon nur 20 Mio. Seiten mit der höchsten On-line Page Importance [APC03] genutzt. Die restlichen wurden mithilfe folgender Verfahren erzeugt:

- Aus dem AOL Query Log wurden die 1050 häufigsten sowie 1050 zufällige Suchanfragen extrahiert. Diese Suchanfragen wurden an Google, Yahoo und Bing gestellt und die insgesamt 500 häufigsten Ergebnisse wurden genutzt.
- 2000 weitere Suchanfragen wurden aus DMOZ Kategorienamen erzeugt und ebenfalls an die Suchmaschinen gestellt.

[2] http://ir.dcs.gla.ac.uk/test_collections/govinfo.html
[3] http://ir.dcs.gla.ac.uk/test_collections/wt10g.html
[4] http://boston.lti.cs.cmu.edu/classes/11-742/S10-TREC/TREC-Nov19-09.pdf
[5] http://boston.lti.cs.cmu.edu/crawler/index.html
[6] http://nutch.apache.org
[7] http://boston.lti.cs.cmu.edu/Data/web08-bst/planning.html

- Um Suchanfragen in anderen Sprachen zu erzeugen, wurden die aus dem AOL Query Log und DMOZ erzeugten englischsprachigen Suchanfragen mithilfe von Google Translate in neun weitere Sprachen übersetzt.
- Für alle Sprachen außer Englisch wurden zudem die 1000 häufigsten Suchanfragen an Yahoo verwendet.
- Um eine breitere Auswahl an chinesischen URLs zu haben, wurden ebenfalls Suchanfragen der Sogou Suchmaschine genutzt.

Die nicht englischen Suchanfragen wurden an die Suchmaschine gestellt, für die jeweilige Sprache am relevantesten ist. Die 200 besten Ergebnisse wurden genutzt. Die Wikipedia, die im ClueWeb09 enthalten ist, gehörte nicht zu den Startseiten, sondern wurde im Nachhinein hinzugefügt[4].

Details über die Einstellungen des Crawlers sind nicht eindeutig dokumentiert. In einem ersten Projektentwurf[8] sind jedoch einige Einstellungen festgehalten (inwieweit diese beim finalen Crawl zum Einsatz kamen, ist jedoch nicht bekannt):

- Dynamische Webseiten wurden ab einer Tiefe größer als fünf ignoriert
- Statische Webseiten wurden ab einer Tiefe größer als 15 ignoriert
- Maxmimal 25.000 Seiten pro Top Level Domain
- Seiten größer als 100 MB wurden abgeschnitten
- Spam Seiten wurden nicht entfernt
- CSS, JavaScript und Mediendateien wurden ignoriert

Für diese Arbeit spielen Relevanzurteile eine wichtige Rolle. Im Rahmen von Information Retrieval Workshops werden oft Wettbewerbe abgehalten, in denen es beispielsweise darum geht, Dokumente zu einem vorgegeben Thema zu finden. Die Workshopteilnehmer reichen die IDs der Dokumente ein, die ihr Algorithmus gefunden hat und die Veranstalter müssen anschließend manuell überprüfen, welche Dokumente relevant oder nicht relevant sind. Am Ende werden diese Urteile dann veröffentlicht, sodass die Teilnehmer ihre Algorithmen weiter optimieren können.

Damit sichergestellt ist, dass alle Teilnehmer auf denselben Daten arbeiten und jedes Dokument auch eine ID besitzt, wird vorgegeben, was für ein Korpus für die Versuche genutzt werden muss. Die Relevanzurteile, die für diese

[8] `http://boston.lti.cs.cmu.edu/Data/web08-bst/web08-bst-Sep26-08.pdf`

Arbeit genutzt werden, beziehen sich alle auf Dokumente aus dem ClueWeb09. Aus den Relevanzurteilen verschiedener Workshops der TREC Konferenz (sogenannte TREC Tracks) wurden 155.419 Dokument-IDs und Relevanzurteile extrahiert (siehe Tabelle 2.2).

TREC Track	Anzahl Dokument-IDs
Web '09	21.855
Web '10	24.470
Web '11	19.220
Web '12	16.028
Session '10	24.287
Session '11	18.509
Session '12	15.080
Gesamt	155.419

Tabelle 2.2: TREC Workshops und die Anzahl extrahierter Relevanzurteile

Diese Relevanzurteile beinhalten neben der ID des Dokuments und der ID des Themas, auf das sie sich beziehen, auch das Urteil wie relevant das Dokument für dieses Thema ist. Es wird nicht nur zwischen relevant und nicht relevant unterschieden, sondern es werden feinere Aussagen getroffen. Die möglichen Relevanzurteile sind $-2, 0, 1, 2$ und 3. -2 und 0 sind nicht relevant, 1 ist relevant und 2 sehr relevant. Das Relevanz-Level 3 hat in manchen Workshops eine spezielle Bedeutung und kann weder als relevant, noch als nicht relevant angesehen werden. In Abbildung 2.1 ist die Verteilung der Werte bei den genutzten Relevanzurteilen zu sehen. Dabei fällt auf, dass der Großteil der Dokumente als nicht relevant eingestuft wurde.

ClueWeb12 Dieses Korpus ist der direkte Nachfolger des ClueWeb09 und enthält 733 Mio. Webseiten.[9] Es enthält ausschließlich Webseiten in englischer Sprache. Gecrawlt wurde mit Heritrix.[10] Heruntergeladen wurden die Seiten von Februar bis Mai 2012.

Als Startseiten des Crawls dienten 2,8 Mio. Webseiten.[11] Diese bildeten sich aus den 10 Mio. Seiten des ClueWeb09 mit dem höchsten PageRank, aus denen der Spam herausgefiltert wurde. Hinzu kamen außerdem, Alexa[12] zufolge, die 262 meist besuchten Seiten in englischsprachigen Ländern, sowie 5.920 zum

[9] `http://www.lemurproject.org/clueweb12/specs.php`
[10] `https://webarchive.jira.com/wiki/display/Heritrix/Heritrix`
[11] `http://boston.lti.cs.cmu.edu/clueweb12/`
[12] `http://www.alexa.com`

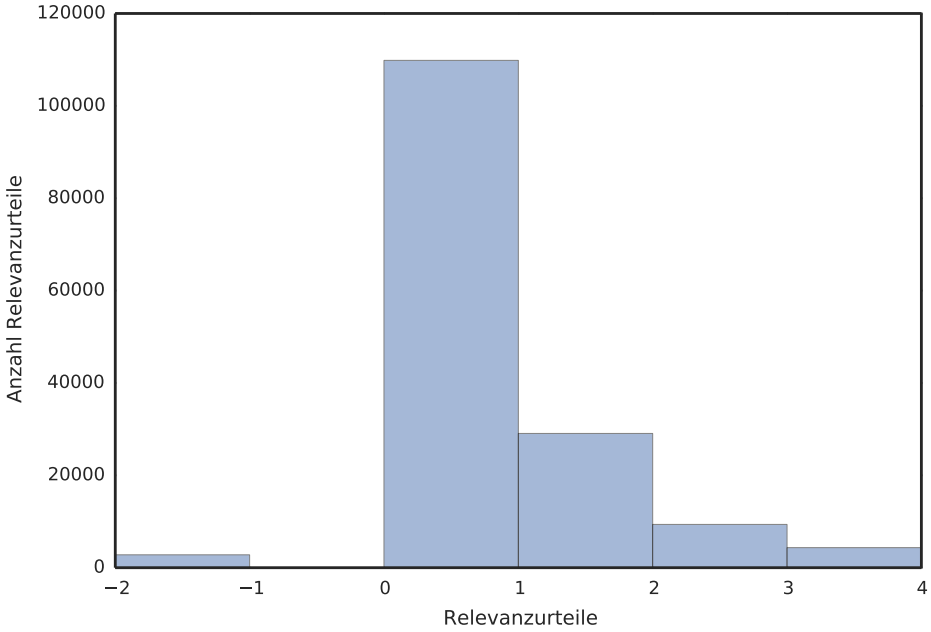


Abbildung 2.1: Verteilung der extrahierten Relevanzurteile.

Thema Reisen. Anschließend wurden die Seiten noch gefiltert. Dazu dienten Blacklists des Anbieters URLBlacklist.com. Auf das manuelle Hinzufügen der Wikipedia wurde dieses mal verzichtet.

Der Crawler wurde dazu konfiguriert alle Inhalte außer Video, Audio und Dateiarchive herunter zu laden. Er beinhaltete außerdem einen Klassifizierer, sodass nicht-englische, pornographische und Spamseiten ignoriert wurden. Es wurden ebenfalls die Bilder zu den Webseiten heruntergeladen, jedoch sind diese nicht Teil des veröffentlichten Korpus.

Ursprünglich enthielt das ClueWeb12 140 Mio. Seiten mehr, bei diesen handelte sich aber um unbeabsichtigte Duplikate. Deshalb wurden sie in späteren Versionen entfernt. Für diejenigen, die den Korpus bereits erhalten hatten, wurde ein Patch bereitgestellt.

Kapitel 3

Near-Duplicate Hashingverfahren

Near-Duplicate Hashingverfahren haben die Aufgabe hochähnliche Dokumente zu finden. Die Schwierigkeit besteht für diese Verfahren zum einen darin, dass nicht nur exakt gleiche Dokumente, sondern auch leicht unterschiedliche gefunden werden sollen und zum anderen in der Menge der nötigen Vergleiche. Für den Vergleich der englischen Wikipedia mit dem ClueWeb12 sind theoretisch mehr als 10^{15} Kreuzvergleiche notwendig. In diesem Kapitel betrachten wir zunächst die verschieden Ansätze für Near-Duplicate Hashingverfahren. Dann betrachten wir, wie die beiden derzeit relevantesten Verfahren (MinHash und SimHash) arbeiten. Zum Schluss vergleichen wir noch diese Verfahren miteinander.

Interessant ist zunächst, wie Near-Duplicate Hashingverfahren die für performante Vergleiche notwendige Komplexitätsreduktion der Dokumente realisieren. Davon abhängig ist auch die Art, wie Vergleiche der Dokumentrepräsentationen durchgeführt werden. Eine Dokumentrepräsentation ist eine unidirektionale Abbildung eines Dokuments auf ein Modell. Ziel einer solchen Abbildung ist die Reduktion auf das Notwendigste. Ein solches Modell ist beipielsweise das Vektorraummodell, welches die Häufigkeit von Worten im Dokument als Vektor repräsentiert (siehe Abbildung 3.1). Im Allgemeinen kann man zwischen zwei unterschiedlichen Ansätzen zur Komplexitätsreduktion von Near-Duplicate Hashingverfahren unterscheiden: Projektion und Einbettung [PS08].

Verfahren mit Projektionen erzeugen Hashes mithilfe gewählter Datenpunkte. Diese Datenpunkte unterscheiden sich je nach gewähltem Verfahren. Dies können zum Beispiel die Worte des Dokuments sein. Für ein Dokument entstehen somit Hashwerte, entsprechend der Datenpunkte des Dokuments. Zur Bestimmung der Ähnlichkeit wird überprüft, wie viele der Hashwerte identisch sind. Dieses Ähnlichkeitsmaß entspricht der Jaccard-Ähnlichkeit.

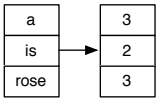


Abbildung 3.1: Repräsentation des Dokuments `(a, rose, is, a, rose, is, a, rose)` im Vektorraummodell.

Das in diesem Kapitel untersuchte MinHash-Verfahren fällt unter diese Kategorie. Weitere Verfahren mit diesem Ansatz sind rare chunks [Hei96], SPEX [BZ04] und I-Match [CFGM02, CGS03, KCA04].

Verfahren, die die Informationen des Dokumentmodells einbetten, erhalten die Eigenschaften des Dokumentmodells und bilden das Dokumentmodell als Ganzes auf einen oder mehrere Hashes ab. Das zugrundeliegende Ähnlichkeitsmaß entspricht somit ebenfalls dem ursprünglichen Ähnlichkeitsmaß des Dokumentmodells. In den bisherigen Verfahren dieser Art ist dies der Winkel zwischen den Dokumentrepräsentationen im Vektorraummodell.

Zu dieser Kategorie von Verfahren gehört das in diesem Kapitel vorgestellte SimHash-Verfahren, sowie Fuzzy-Fingerprinting [Ste05].

3.1 MinHash

MinHash [BGMZ97, Bro97, Bro00] wurde entwickelt um hochähnliche Dokumente aus der Ergebnisliste der AltaVista Suchmaschine herauszufiltern. Es basiert auf zwei Kernkonzepten: Shingles und Min-wise Independent Permutations.

Shingles sind im Dokument auftretende Wortfolgen. Die Menge der Shingles eines Dokuments enthält jedes Shingle aber nur einmal. Dabei sind die w-Shingles eines Dokuments alle Shingles mit der Länge w. Betrachtet man beispielsweise ein Dokument mit der tokenisierten Wortfolge `(a, rose, is, a, rose, is, a, rose)`, so ergeben sich für dieses Dokument die 4-Shingles:

{`(a,rose,is,a)`, `(rose,is,a,rose)`, `(is,a,rose,is)`}

Die Ähnlichkeit für zwei Dokumente d und d' bestimmen wir mit der Jaccard-Ähnlichkeit für die Mengen der Shingles S_d und $S_{d'}$:

$$sim(d, d') = \frac{|S_d \cap S_{d'}|}{|S_d \cup S_{d'}|}. \tag{3.1}$$

Jedoch ist eine solche Berechnung nicht effizient und für den Vergleich vieler Dokumente ungeeignet, weil sie den Speicheraufwand und die Komplexität

des Vergleichs nicht reduziert. Um einen effizienteren Vergleich zu ermöglichen und den Speicheraufwand gering zu halten, werden nicht alle Shingles eines Dokuments miteinander verglichen, sondern nur eine zufällige Menge. Die Auswahl dieser zufälligen Menge muss aber möglichst für alle Dokumente gleich sein, damit gleiche oder hochähnliche Dokumente korrekt identifiziert werden können. Diese Auswahl geschieht mithilfe der Min-wise Independent Permutations [BCFM00].

Definition 3.1. Wir wählen eine zufällige Permutation σ für alle Shingles $S_d \cup S_{d'}$. Diese Permutation erzeugt eine Reihenfolge für das gesamte Vokabular, dass aus der Menge aller Shingles der zu vergleichenden Dokumente besteht (diese Reihenfolge kann, muss aber nicht, der natürlichen Reihenfolge der Elemente entsprechen). Deshalb gilt:

$$sim(d, d') = \mathbf{Pr}[\min\{\sigma(S_d)\} = \min\{\sigma(S_{d'})\}]. \tag{3.2}$$

Die Wahrscheinlichkeit, dass mit derselben Permutation, bei zwei unterschiedlichen Dokumenten, das gleiche Shingle als kleinstes Element ausgewählt wird, entspricht der Ähnlichkeit dieser beiden Dokumente. Min-Wise Independent Permutations müssen außerdem das Unabhängigkeitskriterium erfüllen:

$$\mathbf{Pr}[\min\{\sigma(S_d \cup S_{d'})\} = \sigma(s)\}] = \frac{1}{|S_d \cup S_{d'}|} \quad \forall\ s \in S_d \cup S_{d'}. \tag{3.3}$$

Bei einer beliebigen Permutation soll somit auch jedes Element gleich wahrscheinlich sein.

Mithilfe der Min-wise Independent Permutations ist es möglich die Ähnlichkeit für zwei Dokumente abzuschätzen, ohne die vollständigen Dokumentrepräsentationen zu betrachten. Es wird für jede Permutation nur das kleinste Element gespeichert. Dies reduziert nicht nur die Laufzeit des Verfahrens, sondern auch den Speicheraufwand.

Ein Problem der Min-wise Independent Permutations ist aber die Auswahl der Permutationen. Bei großen Korpora ist es zu aufwendig das ganze Vokabular herauszufinden und speichern zu müssen. Möchte man dynamisch neue Dokumente mit bestehenden vergleichen, ist das Vokabular zuvor meist nicht bekannt. Dies wäre aber für die Konstruktion perfekter Permutationen notwendig.

Diesen Aufwand vermeidet man, indem man zur Berechnung der Min-wise Independent Permutations Hashfunktionen nutzt. Diese sind womöglich nicht perfekt, aber gut genug. Der häufig verwendete Rabin-Fingerprint mit einer Länge von 64 bit besitzt einen Informationsgehalt von $2^{64} \approx 10^{19}$. Das Vokabular der Shingles müsste 1,97 Milliarden Worte umfassen, damit die Wahr-

scheinlichkeit von Hashkollisionen 10 Prozent beträgt.[1] Es wird für jede Hashfunktion das Shingle mit dem kleinsten numerischen Wert ausgewählt. Ersetzt man in Gleichung 3.2 die Permutationen durch Hashfunktionen, gilt:

$$sim(d, d') = \frac{|S_d \cap S_{d'}|}{|S_d \cup S_{d'}|} = \mathbf{Pr}_{h \in \mathcal{F}}[h(d) = h(d')], \tag{3.4}$$

wobei $\mathcal{F}$ eine Familie von Hashfunktionen ist.

Dies beschreibt ein *Locality Sensitive Hashing Scheme*. Das heißt ein Hashing Schema, bei dem die Ähnlichkeit der Hashes auf die Ähnlichkeit der Dokumente schließen lässt. Bei MinHash wird dies über mehrere Hashes mithilfe der Jaccard-Ähnlichkeit realisiert. Je mehr unterschiedliche Hashes gebildet werden, desto genauer wird die Ähnlichkeitsaussage.

Um den Speicheraufwand und die Anzahl der Lookups pro Dokument weiter zu verkleinern, wird noch ein zusätzlicher Schritt durchgeführt: Es werden sogenannte Super Shingles erzeugt. Dies sind Shingles von Shingles, das heißt die zuvor erzeugten Hashes der Shingles werden sortiert und anschließend werden diese zu nicht überlappenden Sequenzen zusammengefasst und erneut gehasht. So ist es möglich nur wenige Super Shingles im Speicher zu halten. Der Vergleich der Mengen der Shingles ist nun ebenfalls überflüssig, da man von einem Near-Duplicate ausgeht, wenn eine bestimmte Menge an Super Shingles übereinstimmt.

Zusammenfassend läuft die Erzeugung von MinHashs für ein Dokument in folgenden Schritten ab:

1. Bestimmung der Startparameter: Dies sind die Länge der Shingles und Super Shingles, eine Familie von Hashfunktionen und die Anzahl der Super Shingles, die übereinstimmen müssen, damit zwei Dokumente als Near-Duplicates gelten.

2. Extraktion der Shingles: Es werden die N-Gramme mit der gewählten Shinglelänge erzeugt. Anschließend wird jedes N-Gramm zur Menge der Shingles hinzugefügt. Jedes Shingle ist in dieser Menge nur einmal enthalten.

3. Bestimmung der minimalen Shingles: Für jede Hashfunktion wird das Shingle mit dem für die Funktion kleinstem Hashwert bestimmt. Diese Shingles werden gepeichert. Alle anderen Shingles werden nicht mehr gebraucht.

4. Erzeugung der Super Shingles: Die zuvor ausgewählten Shingles werden ihrem Hashwert entsprechend sortiert. Diese werden nun in nicht überlappenden Gruppen zusammengefasst und ebenfalls mit den gewählten

[1] `http://preshing.com/20110504/hash-collision-probabilities/`

Hashfunktionen gehasht. Die Hashes der Super Shingles sind die einzigen, die für den Dokumentvergleich notwendig sind.

In der von Fetterly et al. [FMN03] vorgestellten Konfiguration kamen als Hashfunktionen 64-bit Rabin-Fingerprints zum Einsatz. Es wurden 84 Min-Wise Independent Permutations gebildet (in anderen Quellen werden bis zu 200 Min-Wise Independent Permutations gebildet[2]) und daraus wurden wiederum pro Dokument sechs Super Shingles gespeichert. Damit ein Dokument als Near-Duplicate eines anderen gilt, mussten mindestens zwei Super Shingles übereinstimmen. Insgesamt ist somit pro Dokument ein Speicheraufwand von 384 Bit erforderlich ($6 \cdot 64$ Bit).

3.2 SimHash

Das SimHash-Verfahren [Cha02] ist ein Verfahren zur Erkennung hoch ähnlicher Dokumente. Ein SimHash ist dabei eine Bitfolge, die die Terme eines Dokumentes entsprechend ihrer Termhäufigkeit abbildet. Diese Bitfolge ist ein Locality Sensitive Hash, weil ähnliche Dokumente auch ähnliche Bitfolgen produzieren. Ein *Locality Sensitive Hash* ist ein Locality Sensitive Hashing Scheme, das aus nur einem Hash besteht.

Für die Erzeugung eines SimHash-Hashes muss zunächst ein Hashlänge f in Bit gewählt werden. Diese basiert auf der Anzahl der Dokumente n und sollte der Bedingung $2^f \gg n$ genügen. Dies ist nötig, damit die Ähnlichkeitsaussage genau genug getroffen werden kann. Wenn der Informationsgehalt der Kollektion ($\log_2 n$) größer ist als die Kapazität des Hashes, kann dies nicht mehr garantiert werden. Des weiteren wählt man eine zufällig verteilte Hashfunktion h.

Von dem Ausgangsdokument wird zunächst ein Vektor mit den Termhäufigkeiten der Worte erzeugt (siehe Abbildung 3.2). Die Termhäufigkeit ist die Anzahl eines Wortes im Text. Dann werden die Terme im Vektor durch ihre Hashrepräsentation ersetzt. Die Hashes müssen die gewählte Länge f haben. Es wird ein leerer f-dimensionaler Vektor erzeugt. Dieser Vektor wird die Zwischenwerte des SimHashes enthalten. Dann wird für jede Stelle der Termhashwerte die entsprechende Dimension des Vektors gefüllt. Für jede `1` wird der TF-Wert des Terms addiert. Für jede `0` subtrahiert. Auf diese Weise wird der Vektor mit Werten gefüllt. Zuletzt wird jede Dimension des Vektors zu einer Stelle des SimHash. Positive Dimensionen führen zu einer `1`, die anderen zu einer `0`.

SimHash verfolgt das Prinzip der Einbettung. Wie bereits erwähnt, korreliert bei diesen Verfahren die Ähnlichkeit zwischen den Hashes mit dem Win-

[2] `http://blog.cluster-text.com/tag/minhash/`

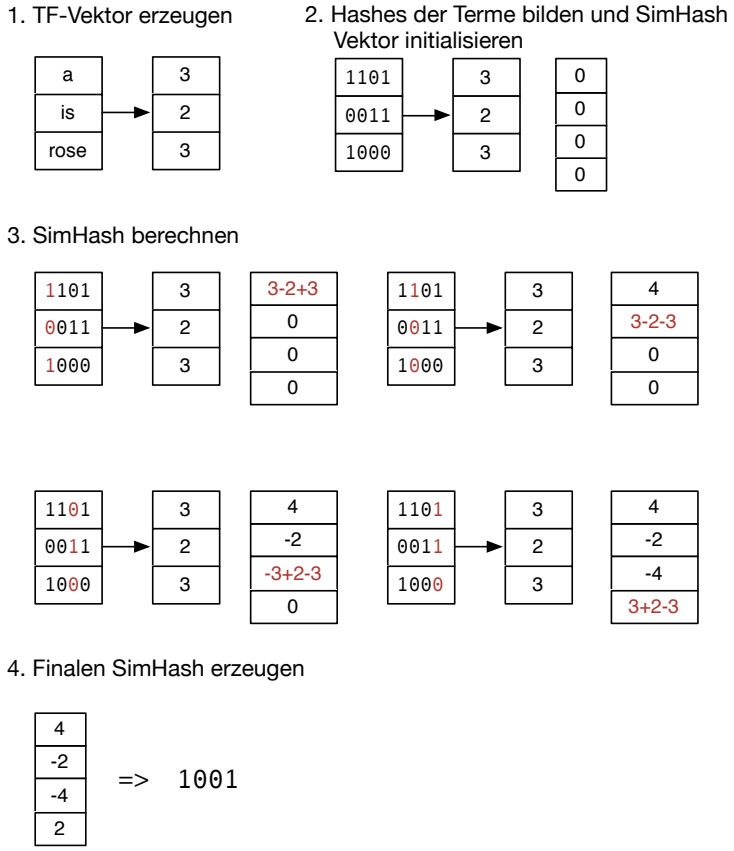


Abbildung 3.2: Ablauf für das Dokument `(a rose is a rose is a rose)` mit $f = 4$

kel zwischen den Dokumentvektoren. Der Nachweis dieser Korrelation folgt aus den Beweisen zu Eigenschaften der semidefiniten Programmierung für das *MAX-CUT* Verfahren von Goemans und Williamson [GW95].

Theorem 3.2 ([GW95])**.** *Sei* $\mathbf{r}$ *die Normale einer zufälligen durch den Ursprung verlaufenden Hyperebene. Beliebige Vektoren lassen sich nun in solche die "darüber" (positives Skalarprodukt mit* $\mathbf{r}$*) und solche die "darunter" (negatives Skalarprodukt mit* $\mathbf{r}$*) liegen aufteilen.*

$$\boldsymbol{Pr}[sgn(\mathbf{u} \cdot \mathbf{r}) \neq sgn(\mathbf{v} \cdot \mathbf{r})] = \frac{\varphi(\mathbf{u}, \mathbf{v})}{\pi} \tag{3.5}$$

Die Wahrscheinlichkeit, dass diese Hyperebene zwei gegebene Vektoren $\mathbf{u}$ *und* $\mathbf{v}$ *teilt, ist somit direkt proportional zum Winkel zwischen diesen beiden Vektoren.*

Es besteht also ein Zusammenhang zwischen dem Winkel, der von zwei Vektoren aufgespannt wird und dem Vorzeichen des Skalarproduktes mit der Normalen einer Hyperebene.

Ein weit verbreitetes Maß zur Berechnung der Ähnlichkeit zweier Dokumente ist im Information Retrieval die Kosinusähnlichkeit. Dabei wird der Kosinus des Winkels der beiden Dokumente in ihrer Vektorrepräsentationen berechnet. Je höher dieser Wert ist, desto Ähnlicher sind diese Dokumente.

Theorem 3.3 ([GW95])**.** *Analog zu Theorem 3.2 sei die Wahrscheinlichkeit für gleiche Vorzeichen:*

$$\boldsymbol{Pr}[sgn(\mathbf{u} \cdot \mathbf{r}) = sgn(\mathbf{v} \cdot \mathbf{r})] = 1 - \frac{\varphi(\mathbf{u}, \mathbf{v})}{\pi}. \tag{3.6}$$

Dann verhält sich die Wahrscheinlichkeit proportional zum Kosinus mit einer festen unteren Schranke.

$$1 - \frac{\varphi(\mathbf{u}, \mathbf{v})}{\pi} > 0,878 \cdot \cos(\varphi(\mathbf{u}, \mathbf{v})) \tag{3.7}$$

Das Ergebnis des SimHash-Verfahrens ist deshalb direkt proportional zur Kosinus-Ähnlichkeit.

Theorem 3.2 und 3.3 gelten zwar für den Vergleich zwischen zwei Vektoren, bilden diesen Vergleich aber lediglich auf die Vorzeichen, also ein Bit, ab. SimHash ermöglicht es diesen Vergleich auf einen Hash mit beliebig vielen Bits auszuweiten, um eine Ähnlichkeitsaussage mithilfe der Hamming-Distanz zu ermöglichen. Somit gilt für den Vergleich zweier Dokumente:

$$sim(d, d') = \Delta(h(d), h(d')) = \mathbf{Pr}_{h \in \mathcal{F}}[h(d) = h(d')], \tag{3.8}$$

wobei Δ die Hamming-Distanz der Hashes und $\mathcal{F}$ eine Familie von Hashfunktionen beschreibt. Diese Gleichung beschreibt, genauso wie Gleichung 3.4, ein *Locality Sensitive Hashing Scheme*. Im Fall von SimHash ist es aber die Hamming-Distanz, die die Wahrscheinlichkeit für ähnliche Hashes in eine Ähnlichkeitsaussage abbildet.

Dieser Vergleich zwischen den erzeugten Hashes ist im Gegensatz zu MinHash nicht trivial. Denn es sind nicht nur exakte Treffer zulässig. Auch Hashes, die sich in wenigen Bits voneinander unterscheiden, sind zulässige Ergebnisse, da die Ähnlichkeit der Hashes die Ähnlichkeit der Dokumente abbildet. Um nicht für jedes mögliche Dokumentpaar ein *XOR* ihrer SimHash-Werte berechnen zu müssen, existieren für den effizienten Vergleich mehrerer Hashes verschiedene Ansätze. Einer dieser Ansätze wurde von Sood und Log [SL11] entwickelt. Ein deutlich effizienterer Ansatz sind Lookup-Tabellen.

Lookup Tabellen. Manku et al. [MJS07] haben bei dem Versuch SimHash für größere Datensätze einzusetzen ein Verfahren entwickelt, dass es ermöglicht eine Vorauswahl von möglicherweise relevanten Hashes zu bilden. Diese Vorauswahl kann, im Gegensatz zum direkten Vergleich der Hashes, mit direkten Tabellenlookups getroffen werden.

Betrachten wir eine Kollektion von Dokumenten mit der Größe n. Dann geht dieses Verfahren von zwei Grundannahmen in Bezug auf diese Kollektion aus:

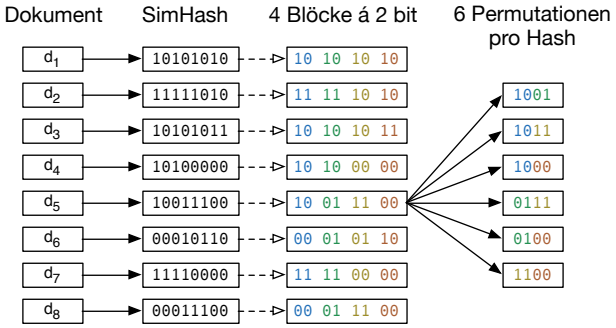
- Es existieren circa $n = 2^l$ unterschiedliche Hashkombinationen.
- Nur sehr wenige Hashkombinationen sind identisch.

Die ersten l Bit können somit als Index und die nachfolgenden $f - l$ Bit als zufällig angesehen werden (f ist die Länge des Hashes). Daraus lässt sich weiterhin ableiten, dass die Ergebnismenge nur sehr klein ist, wenn für einen Hash mit der Länge l' für die gilt, das $|l' - l|$ ebenfalls sehr klein ist, ein Lookup durchgeführt wird. Für diese Länge muss man mit sehr wenigen Kandidaten rechnen, für die direkte Vergleiche notwendig sind. Deshalb bietet es sich an den Hash in Blöcke mit der Länge l' aufzuteilen.

Ein weiterer Faktor ist die maximale Hammingdistanz k. Alle Hashpaare, die sich in mehr als k Bit unterscheiden, werden nicht mehr als Near-Duplicates angesehen und können ignoriert werden.

Die Tabellen, die am Ende dieses Verfahrens entstehen, sind die Permutationen der Blöcke. Es werden dabei immer so viele Blöcke kombiniert, dass nur in k Blöcken Unterschiede auftreten können (Anzahl der Blöcke $- k$). Denn diese Anzahl an Blöcken muss mindestens mit dem Hash übereinstimmen. Zur Ermittlung der nötigen Anzahl an Permutationen σ und somit auch an Tabellen nutzen wir die Formel $\binom{\text{Anzahl der Blöcke}}{\text{Anzahl der Blöcke}-k}$. Die Anzahl der Kandidaten mit dem dann pro Lookup gerechnet werden muss, entspricht $2^{f-min(|\sigma|)}$. Denn die kürzeste Permutation hat den geringsten Informationsgehalt und bildet somit auf die meisten Ergebnisse ab. Anschließend erzeugen wir die entsprechenden Tabellen mit der jeweiligen Permutation als Schlüssel sowie, dem ganzen Hash und der entsprechenden Dokument-ID als Wert.

Beispiel 3.4. Sei $D = \{d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8\}$ eine Menge von acht Dokumenten. Der Informationsgehalt der Dokumente beträgt $d = \log_2(8) = 3$. Die Länge des SimHash soll $t = 8$bit betragen. Die Schwelle für Near-Duplicates ist eine Hammingdistanz von $k = 2$. Die Hashes werden in 4 Blöcke à 2 bit aufgeteilt. Es sind somit $\binom{4}{4-2} = 6$ Tabellen notwendig, um diese Konfiguration zu realisieren. Zunächst wird für jedes Dokument der SimHash berechnet. Anschließend bestimmt man die Permutationen der Hashes, die die Schlüssel zu den Tabellen bilden.



Nun werden vier Tabellen erzeugt, deren Schlüssel die jeweilige Permutation bildet und deren Inhalt die entsprechenden Hashes mit ihren Dokument-IDs bilden.

6 Lookup-Tabellen

t_1

0001	00010110 d_6	00011100 d_8	
1001	10011100 d_5		
1010	10101010 d_1	10101011 d_3	10100000 d_4
1111	11111010 d_2	11110000 d_7	

t_2

0001	00010110 d_6	
0011	00011100 d_8	
1000	10100000 d_4	
1010	10101010 d_1	10101011 d_3
1011	10011100 d_5	
1100	11110000 d_7	
1110	11111010 d_2	

t_3

0000	00011100 d_8	
0010	00010110 d_6	
1000	10100000 d_4	10011100 d_5
1010	10101010 d_1	
1011	10101011 d_3	
1100	11110000 d_7	
1110	11111010 d_2	

t_4

0101	00010110 d_6	
0111	10011100 d_5	00011100 d_8
1000	10100000 d_4	
1010	10101010 d_1	10101011 d_3
1100	11110000 d_7	
1110	11111010 d_2	

t_5

0100	10011100 d_5	00011100 d_8
0110	00010110 d_6	
1000	10100000 d_4	
1010	10101010 d_1	
1011	10101011 d_3	
1100	11110000 d_7	
1110	11111010 d_2	

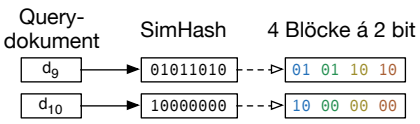
t_6

0000	10100000 d_4	11110000 d_7
0110	00010110 d_6	
1010	10101010 d_1	11111010 d_2
1011	10101011 d_3	
1100	10011100 d_5	00011100 d_8

Mithilfe dieser Tabellen lassen sich nun effizient Vergleiche mit Anfragedokumenten realisieren.

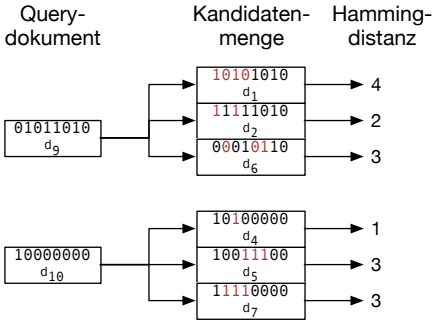
Um Anfragen an diese Tabellen stellen zu können, müssen für die Anfragedokumente zunächst Hashes mit der gleichen Konfiguration erstellt werden. Anschließend müssen die den Tabellen entsprechenden Permutationen gebildet und nachgeschlagen werden. Gibt es für keine Permutation ein Ergebnis, dann gibt es auch kein Near-Duplicate in dieser Kollektion. Für alle gefunden Ergebnisse muss dann noch die Hamming-Distanz mit dem SimHash des Anfragedokuments bestimmt werden. Ist diese kleiner oder gleich k, handelt es sich bei diesem Ergebnis um ein Near-Duplicate des Anfragedokuments. Die Lookup Tabellen liefern somit keine false negatives, aber durchaus false positives. *False negatives* sind Dokumente, die nicht gefunden werden, obwohl es sich um Near-Duplicates handelt und *false positives* sind Dokumente, die gefunden werden, bei denen es sich aber nicht um Near-Duplicates handelt (Hammingdistanz $> k$).

Beispiel 3.5. Gegeben seien zwei Anfragedokumente $D_q = \{d_9, d_{10}\}$. Es sollen Near-Duplicate-Dokumente für den Korpus D aus Beispiel 3.4 gesucht werden. Dazu werden zunächst ebenfalls die SimHashes für D_q berechnet.



Anschließend wird jede Permutation eines jeden Anfragedokumentes in der entsprechenden Tabelle nachgeschlagen.

Die so gefundenen Dokumente bilden die Kandidatenmenge C für das jeweilige Anfragedokument. Die Kanditenmengen für d_9 und d_{10} sind $C_9 = \{d_1, d_2, d_5, d_6\}$ und $C_{10} = \{d_4, d_5, d_7\}$. Für alle anderen Dokumente im Korpus D kann ausgeschlossen werden, dass diese den Threshold von $k = 2$ einhalten. Für die Dokumente in den Kandidatenmengen muss nun noch die Hammingdistanz zum SimHash des Anfragedokumentes berechnet werden.



Ein Near-Duplicate für d_9 ist somit d_2 und für d_{10} wurde d_4 ermittelt. Alle anderen Dokumenten der Kandidatenmengen hatten eine Hammingdistanz, die größer war als der Schwellwert von $k = 2$. In diesem Beispiel wurde für jedes Dokument nur ein Near-Duplicate gefunden, aber es ist durchaus üblich mehrere zu finden.

3.3 Vergleich MinHash & SimHash

SimHash bringt einige Vorteile mit sich, weshalb es sich gut als Near-Duplicate-Erkennungsverfahren in großen Korpora eignet:

1. Ein SimHash repräsentiert durch den Ansatz der Einbettung im Gegensatz zu MinHash das ganze Dokument, inklusive der Termfrequenzen. Diese spielen bei der Erzeugung von MinHash keine Rolle.

2. Der Speicheraufwand für ein Dokument wird auf einige Bit reduziert. Dies ermöglicht es eine große Anzahl von Vergleichsdokumenten im Speicher zu halten. Qualitativ hochwertige Vergleiche sind für SimHash schon mit Hashes von 64 bit möglich. Für MinHash werden mindestens vier Super Shingles gespeichert. Dies entspricht bei gleicher Hashlänge dem vierfachen Speicheraufwand.

3. Die Vergleiche werden mithilfe der Hammingdistanz gemessen und geschehen auf Bitebene. Deshalb sind die Vergleiche sehr schnell. Haben zwei Hashes eine zu große Hammingdistanz, können sie und somit auch ihre zugrunde liegenden Texte nicht mehr als hochähnlich angenommen werden.

Der Vergleich der Hashes mithilfe der Hammingdistanz ist aber auch ein Nachteil von SimHash. Es sind keine *Random Access*-Zugriffe auf Hashes möglich, weil auch leicht unterschiedliche Hashes valide Ergebnisse sein können. Mit dem Verfahren von Manku et al. [MJS07] existiert jedoch ein zwar relativ

komplexes jedoch sehr effizientes Verfahren, das sich sehr gut für große Korpora und Batch-Vergleiche eignet.

Der bisher größte Performancevergleich zwischen den beiden Verfahren wurde 2006 von Henzinger [Hen06] durchgeführt. Der Korpus für diesen Vergleich umfasste 1,6 Milliarden nicht annnotierter Webdokumente. MinHash erzielte eine Precision von 0,38, SimHash von 0,50. Weil der Korpus nicht annotiert war und somit nicht alle relevanten Dokumente bekannt waren, war der Recall nicht messbar.

MinHash wurde in diesem Vergleich mit 8-Shingles gebaut. Es wurden pro Dokument jeweils 84 Min-wise Independent Permutations als 64 bit Rabin Fingerprints erzeugt. Daraus wurden 6 Super Shingles erzeugt, sodass der Speicheraufwand 384 bit pro Dokument betrug. Dokumente galten als Near-Duplicates, wenn mindestens zwei Super Shingles übereinstimmten.

Die SimHashs wurden mit der gleichen Bitlänge wie für MinHash erzeugt. Die maximale Hammingdistanz wurde auf $k = 3$ gesetzt. Über die zugrundeliegende Länge der N-Gramme, mit denen die Hashes erzeugt wurden, wurde keine Aussage getroffen.

Dieser Performancevergleich war lange Zeit die einzige Veröffentlichung dieser Art. Darin werden jedoch nicht alle nötigen Details zu den Einstellungen der Algorithmen erwähnt. Auch bei der Auswertung der Ergebnisse bleibt vieles vage, beziehungsweise schwer nachvollziehbar. Beispielweise wird keine Vermutung dafür geliefert, das SimHash mehr Dokumente findet, die sich lediglich in der URL unterscheiden und MinHash mehr Dokumente, die sich lediglich im Zeitstempel unterscheiden. Ebenfalls interessant wäre zu sehen, wie SimHash mit einer kürzeren Hashlänge oder einem anderen Schwellwert abgeschnitten hätte.

In unserer eigenen Versuchsreihe, die auf den Vergleichen von Near-Duplicate Hashingverfahren von Potthast und Stein [PS08] aufbaut, wurden Precision und Recall für einen SimHash mit unterschiedlichen Schwellwerten gemessen. Als Versuchskorpus diente die Versionshistorie der Wikipedia. Für eine Seite der Wikipedia wurde jeweils das neuste Dokument mit den älteren verglichen. Insgesamt wurden 1,2 Millionen Vergleiche durchgeführt. Die Länge des erzeugten SimHashes betrug 64 bit. In Abbildung 3.3 sieht man die gemessenen Precision und Recall. Die Menge der relevanten Dokumentpaare wird durch die Kosinusähnlichkeit gebildet. Alle Dokumentpaare, deren Ähnlichkeit die angezeigte Kosinusähnlichkeit überschreiten, gelten als relevant. Das SimHash-Verfahren liefert in diesem Versuch eine sehr hohe Precision. Dies kann aber auch daran liegen, dass SimHash mit der Kosinusähnlichkeit korreliert. Deshalb wurde neben der Kosinusähnlichkeit ebenfalls die Jaccardähnlichkeit gemessen.

Die Jaccardähnlichkeit misst die Übereinstimmung des Vokabulars zweier Dokumente und ist ein geläufiges Maß zur Bestimmung von Near-Duplicates.

Schwellwert k	Precision	Recall
1	0,78	0,63
3	0,54	0,81
5	0,38	0,92

Tabelle 3.1: Gemessene Performance des SimHash Algorithmus mit den Schwellwerten 1, 3 und 5. Als relevant gelten Dokumentpaare mit einer Jaccardähnlichkeit $\geq 0,9$

Man spricht bei zwei Dokumenten von Near-Duplicates, wenn die Jaccardähnlichkeit größer oder gleich 0,9 ist. Deshalb werden für die Bestimmung von Precision und Recall alle Dokumentpaare mit dieser Ähnlichkeit als relevant angesehen. Als die gefundenen Dokumentpaare gelten alle, deren Hammingdistanz den entsprechen Schwellwert unterschreitet oder gleich ist. Die Ergebnisse finden sich in Tabelle 3.1.

Die Precision für den Schwellwert von $k = 3$ entspricht der von Henzinger gemessenen. Jedoch ist die von Henzinger verwendete Länge des Hashes mit 384 bit deutlich größer. Das Verhältnis zwischen Länge des Hashes und dem Schwellwert ist somit ein ganz anderes, weshalb sich eher die Ergebnisse für den Schwellwert von $k = 1$ und einer Precision von 0,78 mit den Ergebnissen von Henzinger vergleichen lassen. Der gemessene Recall dieses Schwellwertes ist mit 0,63 jedoch nicht sehr gut.

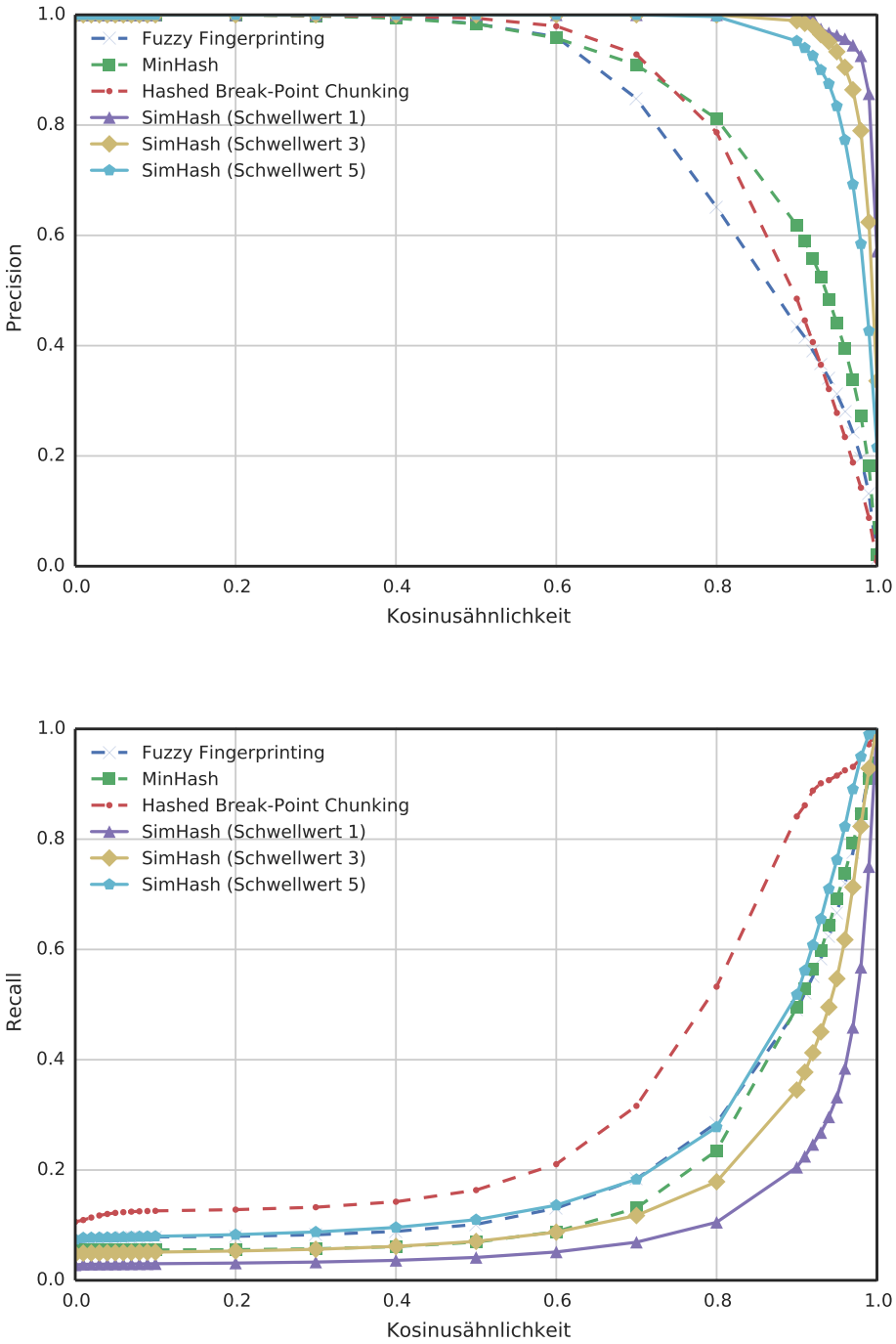


Abbildung 3.3: Precision und Recall drei verschiedener SimHash Einstellungen im Vergleich mit anderen Near-Duplicate Hashingverfahren

Kapitel 4

Überschneidung von URLs

Wie in Kapitel 2 beschrieben, verändert sich der Inhalt der meisten Webseiten nur minimal. Bleibt eine Webseite unter derselben URL erreichbar, so ist auch ihr Inhalt meist relativ konstant. Deshalb werden zunächst die URLs der Dokumente mit Relevanzurteilen mit denen des ClueWeb12 abgeglichen. In einem nächsten Schritt wird dies für Dokumente der Wikipedia wiederholt und zuletzt werden alle URLs der Korpora abgeglichen.

Für die Experimente nutzen wir Listen mit URLs und den korpuseigenen Dokument-IDs. Diese Listen werden mit jedem Web-Korpus veröffentlicht. Es muss nur abgeglichen werden, ob sich die gleiche URL in beiden Listen befindet. Um diesen Abgleich effizient und schnell durchzuführen, werden alle Experimente auf einem Hadoop-Cluster durchgeführt. Für die URLs des ClueWeb12 wird ein MapFile erstellt, das einen wahlfreien Zugriff auf URLs ermöglicht.

4.1 Relevanzurteile

Für den Vergleich der URLs der Relevanzurteile mit den URLs des ClueWeb12 werden zunächst die URLs der 155.000 Relevanzurteile bestimmt und anschließend mit denen des ClueWeb12 abgeglichen. 34.319 Dokumente enthalten demnach eine URL, die sowohl im ClueWeb09 als auch im ClueWeb12 vorhanden ist.

Bei der Betrachtung der häufigsten Top-Level-Domains (siehe Tabelle 4.1) fällt uns zunächst auf, dass die Domainendung `.com` im ClueWeb12 sowohl in den ursprünglichen, als auch in den wiedergefundenen Relevanzurteilen am häufigsten vorkommt. Dies könnte dazu führen, dass Relevanzurteile mit gleicher URL einen veränderten Inhalt im ClueWeb12 haben. Wie in Kapitel 2 beschrieben, verändert sich der Inhalt dieser Seiten am häufigsten. Bei der Domainendung `.org` fällt auf, dass sie mit über 20 Prozent in den ursprünglichen Relevanzurteilen deutlich stärker als im ClueWeb09 repräsentiert ist.

Bei den gemeinsamen URLs fällt dieser Anteil aber wieder auf 12 Prozent zurück. Dies hängt mit der fehlenden Wikipedia im ClueWeb12 zusammen. Ist `en.wikipedia.org` noch mit Abstand die häufigste Domain der Relevanzurteile, so finden sich von diesen 18.000 URLs nur noch 165 im ClueWeb12 wieder. Die dritte auffällige Top-Level-Domain ist `.gov`. Diese Domain ist im ClueWeb12 mit einem Anteil von 0,8 Prozent nur die elfthäufigste. Bei den Relevanzurteilen steigt dieser Anteil deutlich auf 2,6 Prozent und ist mit 3,8 Prozent die Domainendung, von der anteilig die meisten URLs im ClueWeb12 gefunden werden. Dies entspricht auch den Ergebnissen anderer Arbeiten: Domains, die auf `.gov` oder `.edu` enden, sind am langlebigsten. Auch ihr Inhalt ist sehr konstant. Deshalb kann bei einem Großteil dieser URLs davon ausgegangen werden, dass es sich um die gleichen Dokumente handelt.

Wir vergleichen die Dokumente mit gemeinsamen URLs mithilfe der Jaccardähnlichkeit. Nur 23 Prozent der Dokumente haben eine Jaccardähnlichkeit größer oder gleich 0,9 und können als Near-Duplicates gelten. Die Dokumente haben eine durchschnittliche Ähnlichkeit von 0,6. Der Median beträgt 0,63.

4.2 Wikipedia

Im ClueWeb09 ist die ganze Wikipedia mit über 6 Mio. Dokumenten enthalten. Aus den Recherchen zur Entstehung des ClueWeb12 ist bereits bekannt, dass nicht die ganze Wikipedia Teil des Korpus ist. Die meisten Seeds rekrutierten sich aus den 10 Mio. Seiten des ClueWeb09 mit den höchsten PageRanks. Weil die Wikipedia kein Teil des Crawls des ClueWeb09 war, ist es sehr wahrscheinlich, dass viele Wikipedia-Seiten nicht in diesen 10 Mio. Seiten enthalten sind. Jedoch ist der PageRank der Wikipedia im Web allgemein sehr hoch, was wahrscheinlich der Grund dafür ist, dass sich unter den neu gecrawlten Seiten des ClueWeb12 Wikipedia-Seiten befinden. Eine Vorhersage über den Umfang, in dem Wikipedia im ClueWeb12 enthalten ist, ist somit sehr schwer zu treffen. Die Ergebnisse zu den Relevanzurteilen zeigen uns, dass nur sehr wenige Wikipedia-Seiten im ClueWeb12 wiedergefunden werden.

Zunächst wurden alle URLs mit der Domain `en.wikipedia.org` aus der URL-Liste des ClueWeb12 extrahiert. Insgesamt haben Wikipedia und ClueWeb12 lediglich 19.166 gemeinsame URLs. Wenn man bedenkt, dass die englischsprachige Wikipedia über 4,5 Mio. Artikel beinhaltet, dann kann man nicht davon sprechen, dass die Wikipedia im ClueWeb12 enthalten ist, im Gegensatz zum ClueWeb09, in dem sich mehr als 6 Mio. URLs der Wikipedia finden.

4.3 Korpora

Neben den Untersuchungen, zu den Relevanzurteilen und der Wikipedia, ist es für die Übertragbarkeit von Forschungsergebnissen, aber auch für die Wiederholung von Versuchen, interessant zu wissen, wie viele Dokumente alter Korpora insgesamt im ClueWeb12 enthalten sind. Deshalb wurden die URL-Listen des ClueWeb09, sowie der älteren Korpora WT10G und .GOV, analog zu den anderen Untersuchungen in diesem Kapitel mit der URL-Liste des ClueWeb12 abgeglichen.

Insgesamt befinden sich 64,5 Mio. URLs des ClueWeb09 im ClueWeb12. Diese stammen von fast 1,5 Mio. unterschiedlichen Domains. Die häufigste Domain ist `en.wikipedia.org` mit 13.758 URLs. Die zweithäufigste Domain ist `wikitravel.org` mit 8.716 URLs.

Bei den häufigsten Top-Level-Domains (siehe Tabelle 4.1) erkennen wir keine große Abweichung zu der Verteilung im ClueWeb09. Überraschend ist lediglich, dass der Anteil der Domainendung `.org` steigt, obwohl so wenige Wikipedia-URLs gefunden wurden. Es ist aber zu sehen, dass sich der Anteil der häufigsten Domains insgesamt erhöht hat. Auch der Anteil von `.gov` ist gestiegen. Es fällt auf, dass der Anteil von Top-Level-Domains aus nichtenglischsprachigen Ländern kleiner ist als im ClueWeb09. Liegt der Anteil von `.de`, `.pl` und `.fr` im ClueWeb09 bei 1,2 Prozent, 0,5 Prozent bzw. 0,4 Prozent, so fällt dieser Anteil bei den gemeinsamen URLs mit dem ClueWeb12 auf 0,8 Prozent, 0,2 Prozent und 0,18 Prozent. Dies könnte an besseren Sprachfiltern beim Crawl des ClueWeb12 liegen.

Für die Korpora .GOV und WT10G wurden keine gemeinsamen URLs mit dem ClueWeb12 gefunden. Dies ist vor allem für das Korpus .GOV überraschend, weil dieses ausschließlich aus Dokumenten der `.gov` Top-Level-Domain besteht und URLs dieser Domains sich statistisch am seltensten ändern.

Top-Level-Domain	absolute Häufigkeit	relative Häufigkeit
ClueWeb09		
.com	320.099.661	63,5%
.org	51.324.393	10,2%
.net	27.588.841	5,5%
.uk	17.718.970	3,5%
.edu	16.302.128	3,2%
⋮		
.gov	4.087.177	0,8%
Relevanzurteile		
.com	86.986	56,3%
.org	33.323	21,4%
.net	7.009	4,5%
.edu	4.799	3,1%
.gov	4.024	2,6%
Relevanzurteile mit URL im ClueWeb12		
.com	22.418	65,3%
.org	4.174	12,2%
.net	1.684	4,9%
.edu	1.457	4,2%
.gov	1.315	3,8%
Gemeinsame URLs ClueWeb09 und ClueWeb12		
.com	39.833.465	61,7%
.org	8.024.172	12,4%
.net	3.709.254	5,7%
.edu	2.954.424	4,6%
⋮		
.gov	697.835	1,1%

Tabelle 4.1: Die häufigsten Top-Level-Domains

Kapitel 5

Near-Duplicates

In diesem Kapitel nutzen wir den Inhalt der Dokumente unabhängig von der URL, um eine Abschätzung darüber abzugeben, wie viele Dokumente der Ausgangskorpora im Vergleichskorpus zu finden sind. Als Vergleichskorpus dient ebenfalls das ClueWeb12. Die Ausgangskorpora für diese Experimente sind die Relevanzurteile der TREC-Konferenz, mit Dokumenten des ClueWeb09, sowie eine Kopie der Wikipedia aus dem Jahr 2012. Für den Vergleich zwischen Korpora, vor allem der ClueWeb-Korpora, fehlen aktuell die nötigen Rechenkapazitäten. Es werden Untersuchungen zur Performance verschiedener Einstellungen durchgeführt. Die in diesen Versuchen erzielten Ergebnisse werden quantitativ und qualitativ analysiert.

Zum Einsatz kommt, in allen Versuchsreihen, der SimHash-Algorithmus (siehe Kapitel 3). Der SimHash wird mit einer Länge von 64 bit erzeugt. Der Schwellwert für Near-Duplicates wird als $k = 3$ festgelegt. Bei der Erstellung der Lookup-Tabellen wird für beide untersuchten Ausgangskorpora die gleiche Konfiguration genutzt:

Die Hashes werden in 4 Blöcke zu je 16 bit aufgeteilt ($l' = 16$). Mit dem gewählten Schwellwert von $k = 3$ muss nur ein Block übereinstimmen, um eine Vorauswahl zu treffen (Anzahl der Blöcke $- k = 4 - 3 = 1$). Die Anzahl der nötigen Permutationen ist vier: $\binom{\text{Anzahl der Blöcke}}{\text{Anzahl der Blöcke}-k} = \binom{4}{1} = 4$. Diese Konfiguration benötigt nur einen minimalen Speicheraufwand und liefert, aufgrund der geringen Größe der Ausgangskorpora, überschaubare Ergebnismengen.

Bei den Relevanzurteilen handelt es sich um 155.419 Dokumente. Damit besitzt dieser Korpus einen Informationsgehalt von 18 bit ($l = log_2(155419) = 17.26 \leq 18$). Pro Tabellenlookup müssen wir mit durchschnittlich $2^{l-l'} = 2^{18-16} = 4$ Ergebnissen rechnen, für die ein XOR durchgeführt werden muss.

Wie sich während der Erprobungsphase zeigte, spielt die Länge der N-Gramme, mit denen der SimHash erzeugt wird eine sehr wichtige Rolle. Zunächst werden alle Versuche mit 1-Grammen durchgeführt, dies führt jedoch

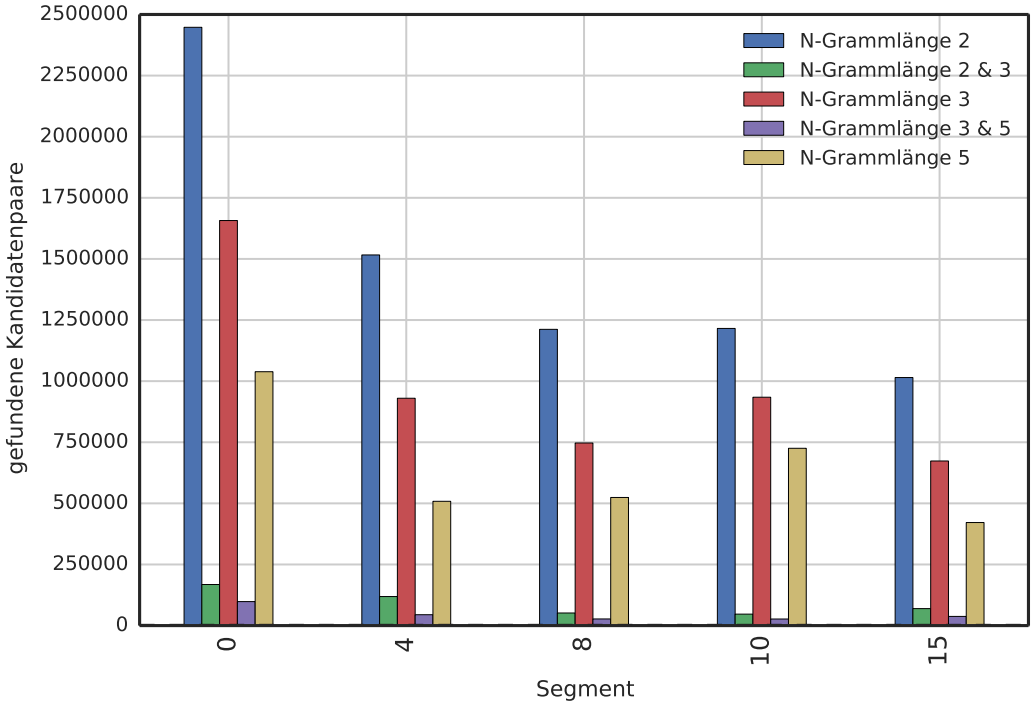


Abbildung 5.1: Gefundene Kandidatenpaare pro Segment und N-Gramm.

zu einem sehr großem Rauschen der Daten. Auch die Entfernung von Stopworten führt nicht zu einem zufriedenstellenden Ergebnis. Deshalb werden die beschriebenen Versuche ausschließlich mit 2, 3 und 5-Grammen durchgeführt. Zu interessanten Ergebnissen führt außerdem die Kombination verschiedener N-Grammlängen. Dafür wird der Vektor, aus dem der SimHash gebaut wird, mit den N-Grammen unterschiedlicher Längen bestückt.

5.1 Relevanzurteile

Um möglichst aussagekräftige Ergebnisse zu erzeugen, wurden alle Versuche auf jeweils fünf Segmenten des ClueWeb12 durchgeführt. Dies war zunächst das erste Segment (die Nummerierung der Segmente beginnt bei 0). Die Segmente 4, 8, 10 und 15 wurden zufällig ausgewählt. Für alle diese Segmente wurden Versuche mit 2-, 3-, 5-, sowie Kombinationen aus 2- und 3-Grammen und 3- und 5-Grammen durchgeführt. Es galt jeweils der Schwellwert von maximal drei unterschiedlichen Bits für den Vergleich zweier SimHashes.

Als Ergebnis liefern die Suchen nach hoch-ähnlichen Dokumenten Paare von Dokument-IDs. Diese Paare enthalten die Dokument-ID eines Relevanz-

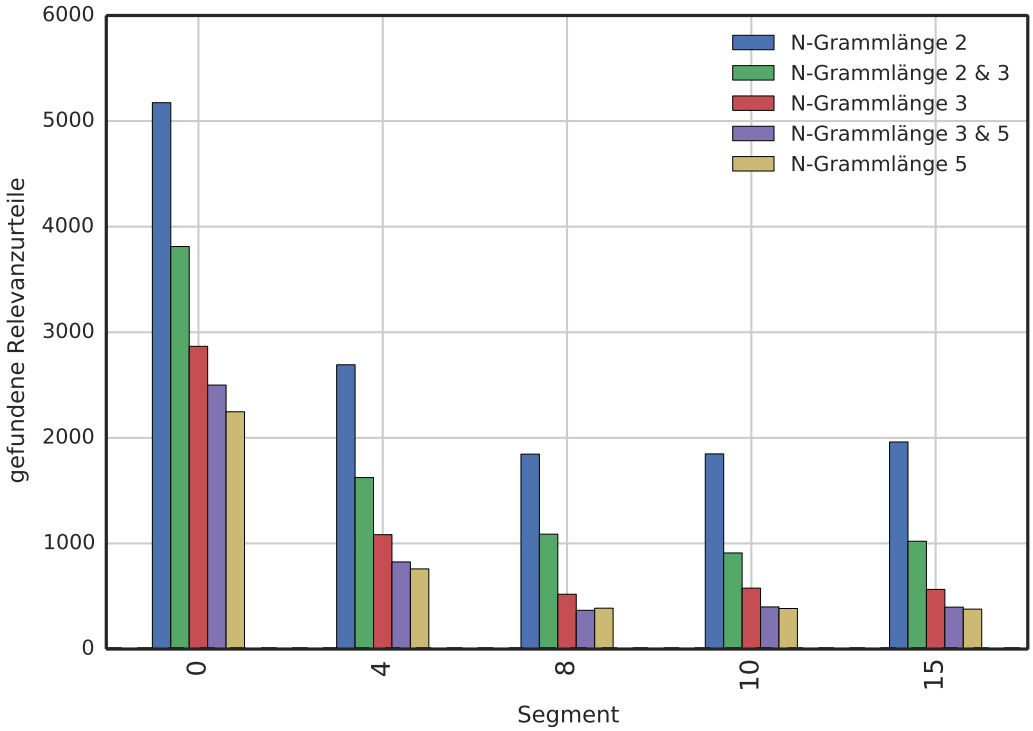


Abbildung 5.2: Gefundene Relevanzurteile pro Segment und N-Gramm.

urteils des ClueWeb09 und die ID des möglichen Near-Duplicates im ClueWeb12. Dieses Dokument nennen wir im folgenden Kandidat, weil es sich nicht zwangsläufig um ein Near-Duplicate handelt. Es kommt sehr häufig vor, dass für ein Relevanzurteil mehrere Kandidaten gefunden werden. In Abbildung 5.1 sehen wir, dass in Segment 0 mit 2-Grammen fast 2,5 Mio. Kandidatenpaare gefunden werden. Die Anzahl der Kandidatenpaare nimmt ab, je länger das gewählte N-Gramm ist und in den hinteren Segmenten werden weniger Kandidatenpaare gefunden als in den vorderen. Besonders auffällig ist die geringe Anzahl an Kandidatenpaaren, die mit Kombinationen von N-Grammen unterschiedlicher Länge gefunden werden. Dies ist oft nur ein Zehntel der Kandidatenpaare, die mit den anderen Konfigurationen gefunden werden. Umso interessanter sind diese Ergebnisse, wenn man ihnen die Anzahl der tatsächlich gefundenen Relevanzurteile gegenüberstellt.

In Abbildung 5.2 sehen wir dagegen das Verhalten, das wir eigentlich schon bei den Kandidatenpaaren erwartet hätten: Kürzere N-Grammlängen liefern mehr Ergebnisse; kombinierte N-Grammlängen liegen zwischen den genutzten Längen. Betrachten wir das Verhältnis zwischen gefunden Kandidatenpaaren und den darin enthaltenen Relevanzurteilen, sehen wir, dass die Menge der

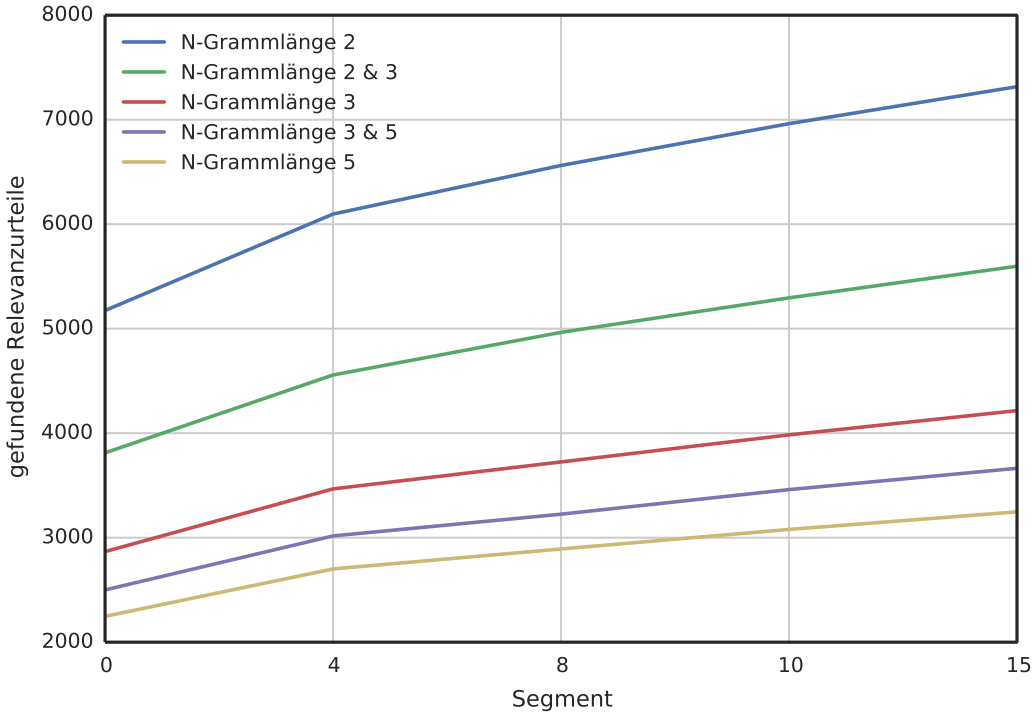


Abbildung 5.3: Akkumulierte Relevanzurteile pro Segment und verwendeten N-Grammlängen.

Kandidatenpaare um ein vielfaches größer ist. Für die Suche mit 2-Grammen im Segment 0 kommen durchschnittlich 473 Kandidatenpaare auf ein Relevanzurteil. Das Maximum sind 29.932 Kandidaten für ein Relevanzurteil. Der Median liegt jedoch bei 1 und selbst das 75% Perzentil beträgt lediglich 6. Es gibt also Relevanzurteile mit extrem vielen Ergebnissen. Die meisten gefunden Relevanzurteile sind aber eindeutig. Der Durchschnitt ist bei den Ergebnissen für 3-Gramme mit 578 Kandidaten pro Relevanzurteil sogar noch größer. Ausreißer sind die kombinierten N-Grammlängen mit einem Durchschnitt von 44 Kandidaten pro Relevanzurteil für die Kombination aus 2- und 3-Grammen und 39 Kandidaten pro Relevanzurteil für die Kombination aus 3- und 5-Grammen. Die Maxima sind mit 9.834 und 9.193 ebenfalls deutlich geringer.

Im Kontext einer Pilotstudie mag uns die Anzahl der gefundenen Near-Duplicates pro Segment durchaus interessieren, aber in der Praxis würden wir uns eher auf das Korpus als ganzes konzentrieren. Denn die pro Segment gefunden Relevanzurteile werden auch in anderen Segmenten gefunden. In Abbildung 5.3 sehen wir die akkumulierten Ergebnisse über alle untersuchten Segmente. Für die meisten Relevanzurteile werden schon mit dem ersten un-

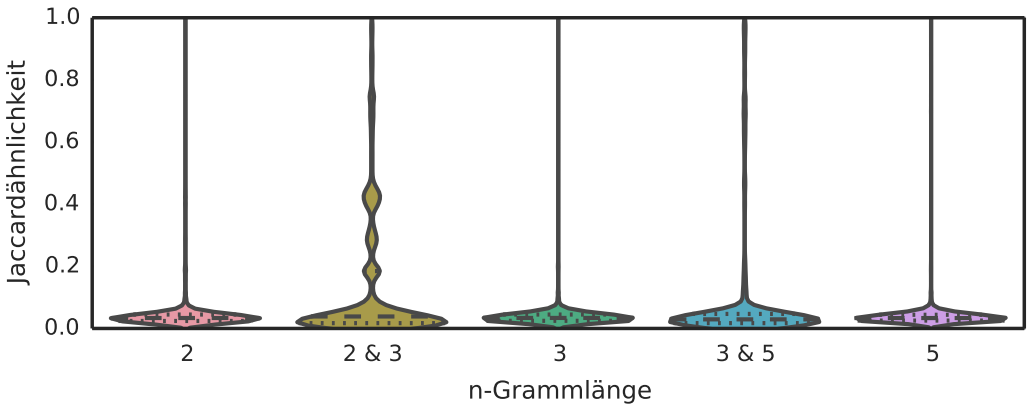


Abbildung 5.4: Verteilung der Jaccard-Ähnlichkeiten für die gefundenen Kandidatenpaare.

tersuchten Segment Kandidaten gefunden. Insgesamt finden wir 7.316 Relevanzurteile mit 2-Grammen als SimHash-Basis, 5.598 für die Kombination von 2- und 3-Grammen, 4.216 für 3-Gramme, 3.664 für die Kombination aus 3- und 5-Grammen und 3.274 Relevanzurteile für 5-Gramme. Nach Analyse von einem Viertel des ClueWeb12 haben wir weniger als 5 Prozent der 155.419 Relevanzurteile wiedergefunden.

Wir nutzen das Mittel der Linearen Regression, um eine Vorhersage über die Anzahl der Ergebnisse im ClueWeb12 zu treffen. Es ist uns bewusst, dass sich die Daten nicht linear verhalten, somit ist es sehr wahrscheinlich, dass unsere Vorhersagen die Ergebnismenge überschätzen. Für 20 untersuchte Segmente würde der SimHash mit 2-Grammen demnach 15.175 Relevanzurteile finden, mit der Kombination aus 2- und 3-Grammen 12.172, mit 3-Grammen 9.117, mit der Kombination aus 3- und 5-Grammen 7.884 und mit 5-Grammen maximal 6.876.

Bisher haben wir noch nicht die Qualität der gefundenen Dokumente betrachtet. Nicht bei all diesen Dokumenten muss es sich um Near-Duplicates handeln. Ein geläufiges Maß zur Bestimmung von Near-Duplicates ist die Jaccardähnlichkeit. Man spricht von einem Near-Duplicate, wenn die Jaccardähnlichkeit größer oder gleich 0,9 ist. In Abbildung 5.4 sehen wir die Verteilung der Jaccardähnlichkeit für alle gefunden Kandidatenpaare. Es fällt auf, dass die meisten Kandidatenpaare keine sehr hohe Jaccardähnlichkeit besitzen. Die Qualität der Ergebnisse des SimHash-Algorithmus wirkt zunächst nicht überzeugend. Es gibt auch kaum Unterschiede zwischen den verwendeten N-Grammlängen. Entscheidend ist aber, ob für jedes gefundene Relevanzurteil ein Near-Duplicate gefunden wurde. Deshalb betrachten wir für jedes Rele-

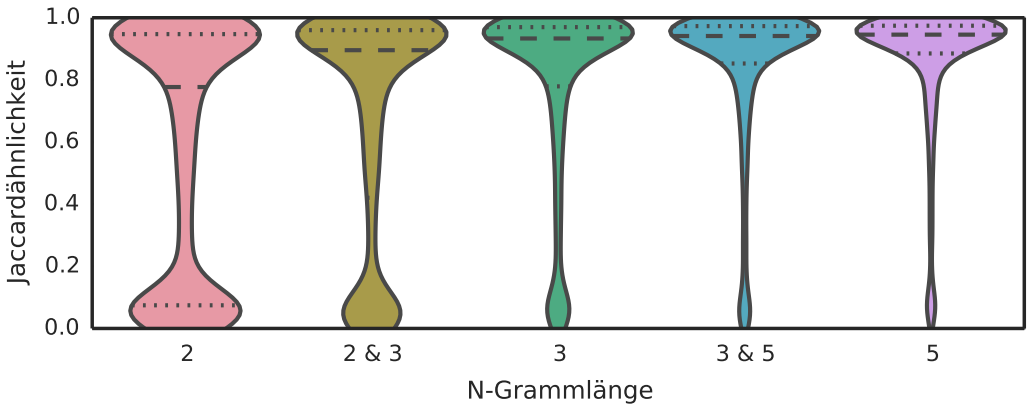


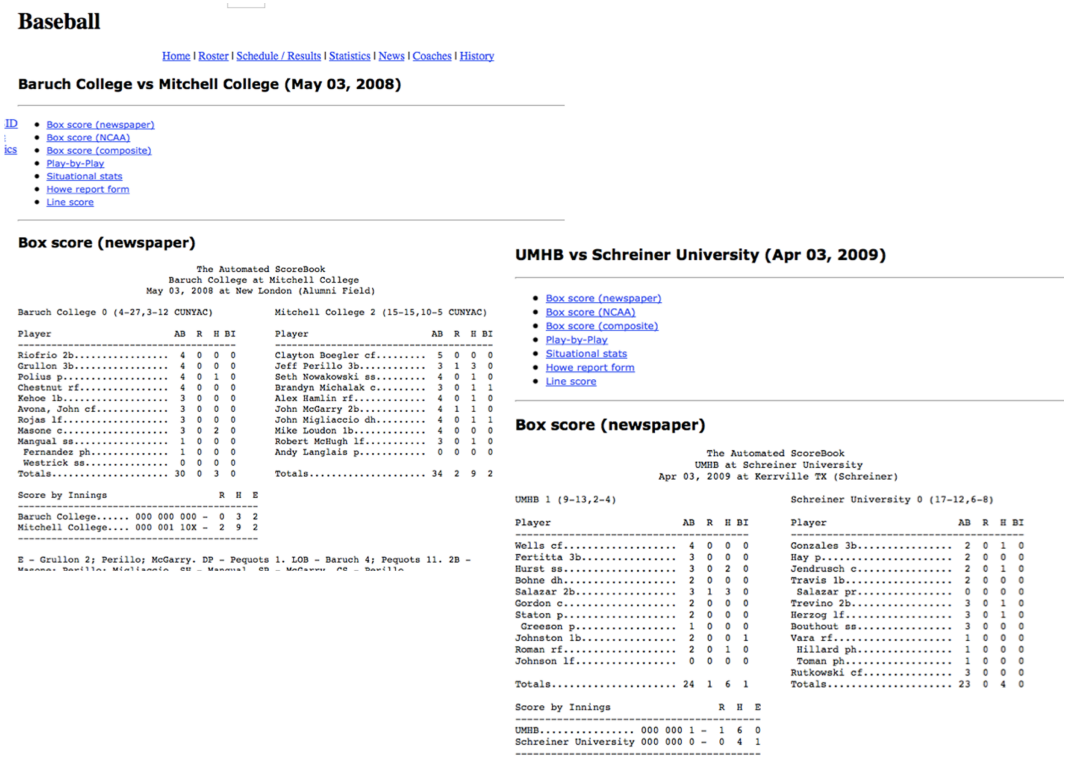
Abbildung 5.5: Verteilung der Jaccard-Ähnlichkeiten für das beste Kandidatenpaar pro Relevanzurteil.

vanzurteil nur das Kandidatenpaar mit der höchsten Jaccardähnlichkeit (siehe Abbildung 5.5). Die Ähnlichkeitsverteilung kehrt sich nahezu um. Für den Großteil der Relevanzurteile wird ein Dokument mit einer hohen Ähnlichkeit gefunden. Je länger die verwendeten N-Gramme sind, desto ausgeprägter ist die Verteilung im Bereich der hohen Ähnlichkeiten.

Als weiteres Qualitätskriterium für einen Retrieval-Algorithmus sind Precision und Recall. Die Precision sagt etwas darüber aus, wie viele der gefunden Dokumente relevant sind. Der Recall misst wie viele der relevanten Dokumente gefunden wurden. In diesem Zusammenhang betrachten wir die Menge der Relevanzurteile. Alle Relevanzurteile, für die ein Kandidatenpaar mit einer Jaccardähnlichkeit größer oder gleich 0,9 gefunden wurde, gelten als relevant. So ermitteln wir eine Precision von 0,39 für 2-Gramme als SimHash-Basis, 0,49 für die Kombination aus 2- und 3-Grammen, 0,61 für 3-Gramme, 0,67 für die Kombination aus 3- und 5-Grammen und 0,71 für 5-Gramme. Der Recall lässt sich für die verwendete Datenkollektion nicht ermitteln, weil uns nicht alle relevanten Dokumente bekannt sind. Jedoch ist anzunehmen, dass der Recall mit steigender Precision abnimmt. Dies können wir bei unseren Ergebnismengen beobachten. Zwar steigt der Anteil an Near-Duplicates entsprechend der Precision an, aber absolut betrachtet nimmt die Anzahl der gefundenen Near-Duplicates ab. Wurden mit einem auf 2-Grammen basierendem SimHash noch 2.828 Near-Duplicates gefunden, sind es bei 3-Grammen 2.443 und mit 5-Grammen finden wir nur noch 2.194.

Wir können aber nicht davon ausgehen, dass alle Near-Duplicates, die mit 5-Grammen gefunden werden, auch mit 2-Grammen gefunden werden. So befinden sich beispielsweise 347 Relevanzurteile, für die ein Near-Duplicate mit 3-

und 5-Grammen in Segment 0 gefunden wurde, nicht unter den Ergebnissen der 3-Gramme und 278 Relevanzurteile nicht in den Ergebnissen der 2-Gramme. Andersherum ist diese Menge natürlich deutlich größer, jedoch zeigt dieses Beispiel, dass die Suche nach der optimalen Einstellung sehr schwierig ist und dass es fast unmöglich ist alle Near-Duplicates zu finden.



Baseball

Home | Roster | Schedule / Results | Statistics | News | Coaches | History

Baruch College vs Mitchell College (May 03, 2008)

- Box score (newspaper)
- Box score (NCAA)
- Box score (composite)
- Play-by-Play
- Situational stats
- Howe report form
- Line score

Box score (newspaper)

The Automated ScoreBook
Baruch College at Mitchell College
May 03, 2008 at New London (Alumni Field)

Baruch College 0 (4-27,3-12 CUNYAC) — Mitchell College 2 (15-15,10-5 CUNYAC)

Player	AB	R	H	BI
Riofrio 2b	4	0	0	0
Grullon 3b	4	0	0	0
Polius p	4	0	1	0
Chestnut rf	4	0	0	0
Kehoe 1b	3	0	0	0
Avona, John cf	3	0	0	0
Rojas lf	3	0	0	0
Masone c	3	0	2	0
Mangual ss	1	0	0	0
Fernandez ph	1	0	0	0
Westrick ss	0	0	0	0
Totals	30	0	3	0

Player	AB	R	H	BI
Clayton Boegler cf	5	0	0	0
Jeff Perillo 3b	3	1	3	0
Seth Nowakowski ss	4	0	1	0
Brandyn Michalak c	3	0	1	1
Alex Hamlin rf	4	0	1	0
John McGarry 2b	4	1	1	0
John Migliaccio dh	4	0	1	1
Mike Loudon 1b	4	0	0	0
Robert McHugh lf	3	0	1	0
Andy Langlais p	0	0	0	0
Totals	34	2	9	2

Score by Innings	R	H	E
Baruch College 000 000 000	0	3	2
Mitchell College 000 001 10X	2	9	2

E - Grullon 2; Perillo; McGarry. DP - Pequots 1. LOB - Baruch 4; Pequots 11. 2B -

UMHB vs Schreiner University (Apr 03, 2009)

- Box score (newspaper)
- Box score (NCAA)
- Box score (composite)
- Play-by-Play
- Situational stats
- Howe report form
- Line score

Box score (newspaper)

The Automated ScoreBook
UMHB at Schreiner University
Apr 03, 2009 at Kerrville TX (Schreiner)

UMHB 1 (9-13,2-4) — Schreiner University 0 (17-12,6-8)

Player	AB	R	H	BI
Wells cf	4	0	0	0
Fertitta 3b	3	0	0	0
Hurst ss	3	0	2	0
Bohne dh	2	0	0	0
Salazar 2b	3	1	3	0
Gordon c	2	0	0	0
Staton p	2	0	0	0
Greeson p	1	0	0	0
Johnston 1b	2	0	0	1
Roman rf	2	0	1	0
Johnson lf	0	0	0	0
Totals	24	1	6	1

Player	AB	R	H	BI
Gonzales 3b	2	0	1	0
Hay p	2	0	0	0
Jendrusch c	2	0	1	0
Travis 1b	2	0	0	0
Salazar pr	0	0	0	0
Trevino 2b	3	0	1	0
Herzog lf	3	0	1	0
Bouthout ss	3	0	0	0
Vara rf	1	0	0	0
Hillard ph	1	0	0	0
Toman ph	1	0	0	0
Rutkowski cf	3	0	0	0
Totals	23	0	4	0

Score by Innings	R	H	E
UMHB 000 000 1	1	6	0
Schreiner University 000 000 0	0	4	1

Abbildung 5.6: Beispiel für zwei Dokumente, bei denen es sich visuell um Near-Duplicates handelt, aber inhaltich keine Near-Duplicates sind.

Die Einschätzung, ob ein Dokument ein Near-Duplicate eines anderen ist, ist oft subjektiv. Die Schranke der Jaccardähnlichkeit von 0,9 gilt zwar für die meisten Dokumente, es kann aber auch möglich sein, dass zwei Dokumente eine niedrigere Jaccardähnlichkeit besitzen und von einem Menschen dennoch sofort als Near-Duplicate eingestuft werden. Deshalb wurden für verschiedene N-Grammlängen 100 Dokumente zufällig ausgewählt und manuell als Near-Duplicates eingestuft. Dabei wurden grundsätzlich mehr Dokumente als Near-Duplicates eingestuft, als die Precision erwarten ließ. Das beste Ergebnis erzielte der SimHash, der eine Kombination aus 3- und 5-Grammen nutzt: Nur fünf der 100 zufällig ausgewählten Dokumente sind, dem subjektiven Eindruck nach, keine Near-Duplicates. Für den SimHash mit 2-Grammen sind es 37 von 100 und für den mit 3-Grammen sind es 16 von 100. In Abbil-

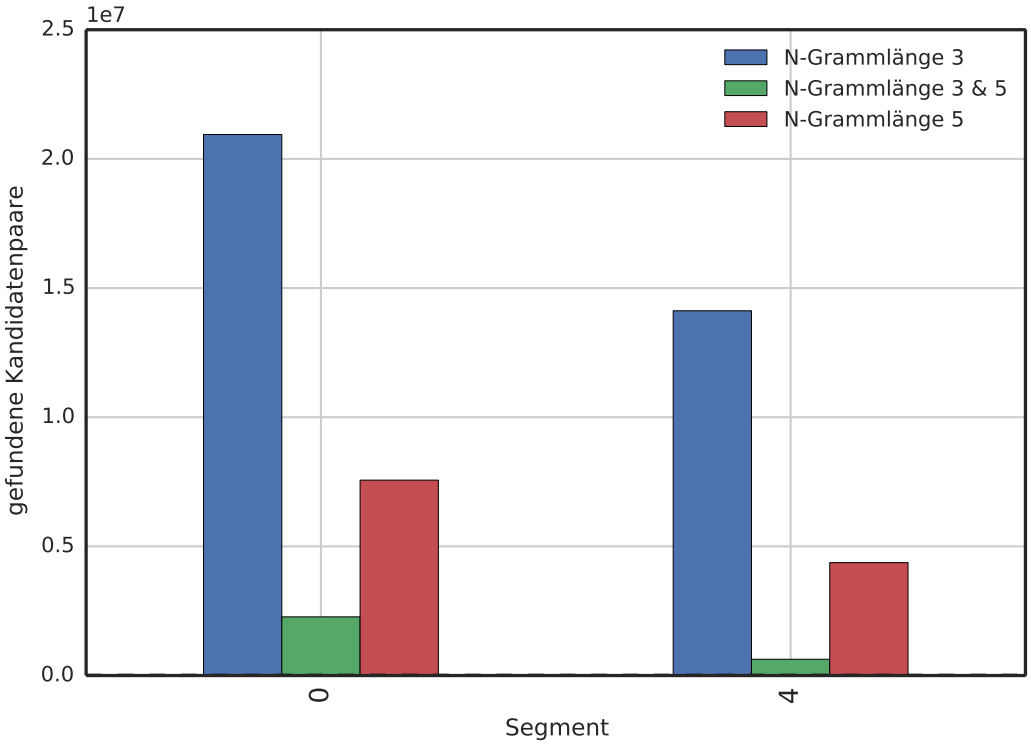


Abbildung 5.7: Gefundene Kandidatenpaare pro Segment und N-Gramm.

dung 5.6 sehen wir ein Beispiel für zwei Dokumente, die als Near-Duplicates einzustufen waren.

Abschließend vergleichen wir noch die Ergebnisse, die bei der Analyse der URLs erzielt wurden (siehe Kapitel 4), mit den Ergebnissen aus diesem Kapitel. Unter den 7.316 mit 2-Gramm SimHash gefundenen Relevanzurteilen finden sich 3.767 mit gemeinsamer URL. Dies ist ein Anteil 51 Prozent. Dieser Anteil steigt mit zunehmender N-Grammlänge an. Für 2- und 3-Gramme sind es 60 Prozent, für 3-Gramme 69 Prozent, für 3- und 5-Gramme 72 Prozent und für 5-Gramme finden wir 72 Prozent mit gemeinsamer URL.

5.2 Wikipedia

Analog zu den Relevanzurteilen wurde auch die Wikipedia mit dem ClueWeb12 verglichen. Alle Versuche wurden lediglich auf zwei Segmenten des ClueWeb12 und mit weniger N-Gramm Konfigurationen durchgeführt. Als Ergebnis lieferte der Algorithmus ebenfalls Kandidatenpaare, die jeweils eine Wikipedia-Seite und eine ID eines Dokuments im ClueWeb12 enthalten.

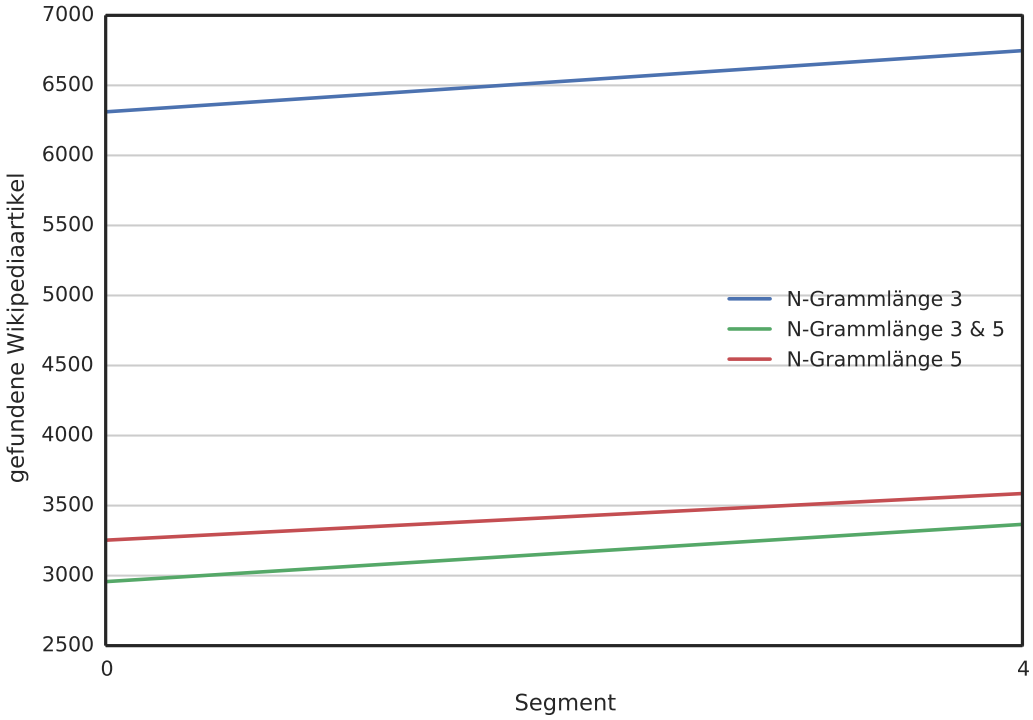


Abbildung 5.8: Akkumulierte Wikipedia-Seiten pro Segment und verwendeten N-Grammlängen.

In Abbildung 5.7 sehen wir die Menge der Kandidatenpaare pro Segment und die zur Konstruktion des SimHash verwendeten N-Grammlängen. Die Anzahl der gefunden Kandidatenpaare ist mehr als zehn mal so groß wie bei den Versuchen zu den Relevanzurteilen. Zu bedenken ist auch, dass es sich bei der Wikipedia um deutlich mehr Dokumente, als bei den Relevanzurteilen handelt. Das Verhalten, das wir bei der Kombination aus 3- und 5-Grammen sehen, fällt analog zu den Relevanzurteilen aus: Für die Kombination von N-Grammlängen finden sich deutlich weniger Kandidatenpaare. Der relative Unterschied ist in Segment 0 jedoch geringer. Für Segment 0 werden bei den Relevanzurteilen 16,88 mal so viele Kandidatenpaare für 3-Gramme gefunden, als für die Kombination aus 3- und 5-Grammen. Bei den Wikipedia-Seiten sind es nur noch 9,22 mal so viele. Für Segment 4 ist dieses Verhalten jedoch nicht mehr erkennbar. Dort stimmt der relative Unterschied zwischen 3-Grammen und 3- und 5-Grammen nahezu überein (21 mal mehr 3-Gramme als 3- und 5-Gramme bei den Relevanzurteilen und 22,39 mal so viele bei der Wikipedia).

In Abbildung 5.8 sehen wir die Anzahl der gefunden Wikipedia-Seiten pro verwendeter N-Grammlänge, akkumuliert über beide Segmente. Dabei fällt

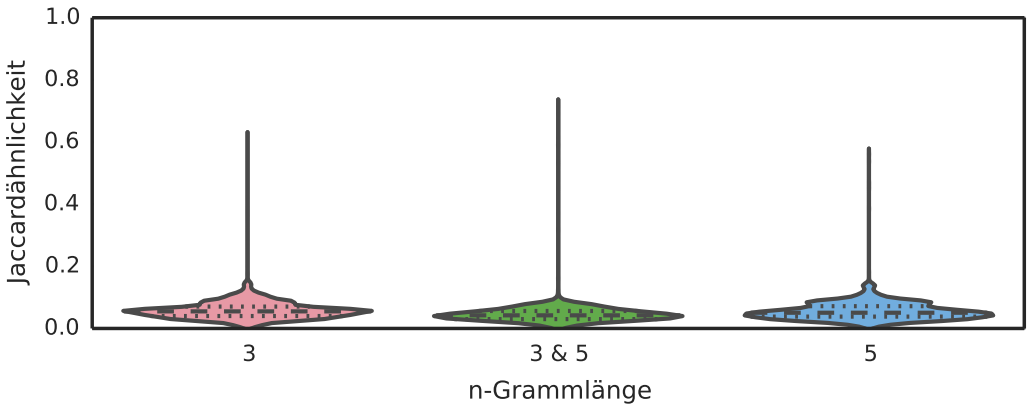


Abbildung 5.9: Verteilung der Jaccard-Ähnlichkeiten für das beste Kandidatenpaar pro Wikipedia-Seiten.

zunächst auf, dass die Kombination aus 3- und 5-Grammen weniger Ergebnisse liefert als Versuche mit einzelnen N-Grammlängen. Weiterhin treten deutlich mehr Kandidatenpaare pro Wikipedia-Seite auf als dies bei den Relevanzurteilen der Fall war. Bei der Suche mit 3-Grammen kommen auf jede gefundene Wikipedia-Seite durchschnittlich 3.318 Kandidatenpaare, für 5-Gramme finden wir durchschnittlich 2.325 und für die Kombination aus 3- und 5-Grammen sind es immer noch 768.

In Abbildung 5.9 sehen wir die Verteilung der Jaccardähnlichkeit für das beste Kandidatenpaar einer Wikipedia-Seite. Im Unterschied zu Abbildung 5.5, die die gleiche Verteilung für die Relevanzurteile anzeigt, wurden fast ausschließlich Dokumente mit einer sehr geringen Jaccardähnlichkeit gefunden. Es wurde kein einziges Dokument gefunden, das eine genügend hohe Jaccardähnlichkeit für ein Near-Duplicate erreicht.

Kapitel 6

Zusammenfassung und Ausblick

Das Ziel dieser Arbeit war es abzuschätzen, ob und in welchem Umfang Forschungsergebnisse, die mithilfe eines Web-Korpus erstellt wurden, auf ein anderes Korpus übertragen werden können.

Wir haben gezeigt, dass 20 Prozent der untersuchten Relevanzurteile und 12,8 Prozent des ClueWeb09 im ClueWeb12 enthalten sind. Von den gemeinsamen Relevanzurteilen sind 23,02 Prozent Near-Duplicates. Des weiteren haben wir gezeigt, dass lediglich 0,43 Prozent der URLs der englischsprachigen Wikipedia im ClueWeb12 enthalten sind.

In einem Vergleich des SimHash Verfahrens mit den Ergebnissen anderer Forschungsgruppen, konnten wir zeigen, dass das SimHash-Verfahren in Verbindung mit Lookup-Tabellen ein geeignetes Verfahren zur Suche nach Near-Duplicates in großen Web-Korpora ist.

Für die geeignete Parametrisierung des SimHash konnten wir noch keine eindeutige Antwort liefern. Wir konnten aber zeigen, dass die Kombination verschiedener N-Grammlängen bei der Konstruktion des SimHash, die Kandidatenmenge um bis zu 95 Prozent reduziert, die Anzahl der gefunden Dokumente aber lediglich um 24 Prozent. Diese deutlich kleinere Kandidatenmenge kann dafür genutzt werden, die weiteren Parameter weniger streng einzustellen und somit den Recall zu steigern.

Insgesamt haben wir mithilfe des SimHash-Verfahrens nur einen kleinen Teil der Relevanzurteile des ClueWeb09 im ClueWeb12 wiedergefunden. Nach der Untersuchung eines Viertels des ClueWeb12 können wir davon ausgehen, dass weniger als 10 Prozent der Relevanzurteile im ClueWeb12 enthalten sind. Und die Ergebnisse der URL-Analysen deuten darauf hin, dass der Anteil des gesamten ClueWeb09 im ClueWeb12 eher geringer ist, als bei den Relevanzurteilen. Eine Übertragbarkeit von Forschungsergebnissen zwischen dem ClueWeb09 und dem ClueWeb12 scheint deshalb nur in sehr geringem Umfang möglich.

Für zukünftige Arbeiten können wir uns eine weitere Evaluation des SimHash-Verfahrens vorstellen. In der Kombination verschiedener N-Grammlängen zur Konstruktion des SimHash sehen wir einen Ansatz, den Recall bei gleichbleibender Precision zu steigern. Vor allem eine Skalierung des Verfahrens, die es uns ermöglichen würde, ganze Korpora miteinander zu vergleichen, interessiert uns. Ein Austausch des verwendeten Hashalgorithmus könnte beispielsweise die Geschwindigkeit bei der Erzeugung des SimHash steigern. Die maximale Größe der Lookup-Tabellen ist derzeit noch sehr stark von der Größe des Arbeitsspeichers abhängig, sodass hier auch noch Optimierungsbedarf besteht.

Weitere Untersuchungen könnten auch die Einschätzung von Near-Duplicates betreffen. Bei der manuellen Betrachtung der Ergebnisse des SimHash konnten wir feststellen, dass Dokumente nicht zwangsläufig eine Jaccardähnlichkeit von 0,9 besitzen müssen, um als Near-Duplicates zu gelten. Hier könnte man Verfahren entwickeln, die eine differenzierte Aussage über die Ähnlichkeit von Dokumenten treffen.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[APC03] Serge Abiteboul, Mihai Preda und Gregory Cobena. Adaptive online page importance computation. In *Proceedings of the Twelfth International World Wide Web Conference, WWW 2003, Budapest, Hungary, May 20-24, 2003*, Seiten 280–290, 2003.

[ATDE09] Eytan Adar, Jaime Teevan, Susan T. Dumais und Jonathan L. Elsas. The web changes everything: understanding the dynamics of web content. In *Proceedings of the Second International Conference on Web Search and Web Data Mining, WSDM 2009, Barcelona, Spain, February 9-11, 2009*, Seiten 282–291, 2009.

[BCFM00] Andrei Z. Broder, Moses Charikar, Alan M. Frieze und Michael Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630–659, 2000.

[BGMZ97] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse und Geoffrey Zweig. Syntactic clustering of the web. *Computer Networks*, 29(8-13):1157–1166, 1997.

[Bro97] Andrei Z. Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences 1997*, SEQUENCES '97, Seiten 21–29, Washington, DC, USA, 1997. IEEE Computer Society.

[Bro00] Andrei Z. Broder. Identifying and filtering near-duplicate documents. In *Combinatorial Pattern Matching, 11th Annual Symposium, CPM'00, Montreal, Canada, June 21-23, 2000, Proceedings*, Seiten 1–10, 2000.

[BZ04] Yaniv Bernstein und Justin Zobel. A scalable system for identifying co-derivative documents. In *String Processing and Information Retrieval, 11th International Conference, SPIRE'04, Padova, Italy, October 5-8, 2004, Proceedings*, Seiten 55–67, 2004.

[CFGM02] Abdur Chowdhury, Ophir Frieder, David A. Grossman und M. Catherine McCabe. Collection statistics for fast duplicate document detection. *Transactions on Information Systems*, 20(2):171–191, 2002.

[CGM00] Junghoo Cho und Hector Garcia-Molina. The evolution of the web and implications for an incremental crawler. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, Seiten 200–209, 2000.

[CGS03] Jack G. Conrad, Xi S. Guo und Cindy P. Schriber. Online duplicate document detection: signature reliability in a dynamic retrieval environment. In *Proceedings of the 2003 ACM CIKM International Conference on Information and Knowledge Management, New Orleans, Louisiana, USA, November 2-8, 2003*, Seiten 443–452, 2003.

[Cha02] Moses Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, Seiten 380–388, 2002.

[DFKM97] Fred Douglis, Anja Feldmann, Balachander Krishnamurthy und Jeffrey C. Mogul. Rate of change and other metrics: a live study of the world wide web. In *1st USENIX Symposium on Internet Technologies and Systems, USITS'97, Monterey, California, USA, December 8-11, 1997*, 1997.

[FMN03] Dennis Fetterly, Mark Manasse und Marc Najork. On the evolution of clusters of near-duplicate web pages. In *1st Latin American Web Congress, LA-WEB'03, Empowering Our Web, 10-12 November, 2003, Sanitago, Chile*, Seiten 37–45, 2003.

[FMNW03] Dennis Fetterly, Mark Manasse, Marc Najork und Janet L. Wiener. A large-scale study of the evolution of web pages. In *Proceedings of the Twelfth International World Wide Web Conference, WWW'03, Budapest, Hungary, May 20-24, 2003*, Seiten 669–678, 2003.

[GW95] Michel X. Goemans und David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Information and Computation*, 42(6):1115–1145, 1995.

[Hei96] Nevin Heintze. Scalable document fingerprinting. In *USENIX workshop on electronic commerce*, Band 3, 1996.

[Hen06] Monika Rauch Henzinger. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *Proceedings of the 29th Annual International ACM Conference on Research and Development in Information Retrieval, SIGIR'06, Seattle, Washington, USA, August 6-11, 2006*, Seiten 284–291, 2006.

[KCA04] Aleksander Kolcz, Abdur Chowdhury und Joshua Alspector. Improved robustness of signature-based near-replica detection via lexicon randomization. In *Proceedings of the Tenth ACM International Conference on Knowledge Discovery and Data Mining, SIGKDD'04, Seattle, Washington, USA, August 22-25, 2004*, Seiten 605–610, 2004.

[KL05] Sung Jin Kim und Sang Ho Lee. An empirical study on the change of web pages. In *Web Technologies Research and Development - APWeb 2005, 7th Asia-Pacific Web Conference, Shanghai, China, March 29 - April 1, 2005, Proceedings*, Seiten 632–642, 2005.

[Koe02] Wallace Koehler. Web page change and persistence - a four-year longitudinal study. *Journal of the American Society for Information Science and Technology*, 53(2):162–171, 2002.

[MJS07] Gurmeet Singh Manku, Arvind Jain und Anish Das Sarma. Detecting near-duplicates for web crawling. In *Proceedings of the 16th International Conference on World Wide Web, WWW'07, Banff, Alberta, Canada, May 8-12, 2007*, Seiten 141–150, 2007.

[NCO04] Alexandros Ntoulas, Junghoo Cho und Christopher Olston. What's new on the web?: the evolution of the web from a search engine perspective. In *Proceedings of the 13th international conference on World Wide Web, WWW'04, New York, NY, USA, May 17-20, 2004*, Seiten 1–12, 2004.

[OP08] Christopher Olston und Sandeep Pandey. Recrawl scheduling based on information longevity. In *Proceedings of the 17th International Conference on World Wide Web, WWW'08, Beijing, China, April 21-25, 2008*, Seiten 437–446, 2008.

[PS08] Martin Potthast und Benno Stein. New issues in near-duplicate detection. In *Data Analysis, Machine Learning and Applications*, Seiten 601–609. Springer, 2008.

[SL11] Sadhan Sood und Dmitri Loguinov. Probabilistic near-duplicate detection using simhash. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM'11, Glasgow, United Kingdom, October 24-28, 2011*, Seiten 1117–1126, 2011.

[Ste05] Benno Stein. Fuzzy-fingerprints for text-based information retrieval. In Klaus Tochtermann und Hermann Maurer (Hrsg.), *5th International Conference on Knowledge Management (I-KNOW 05)*, Journal of Universal Computer Science, Seiten 572–579, Graz, Austria, Juli 2005. Know-Center.