Bauhaus-Universität Weimar
Fakultät Medien
Studiengang Mediensysteme

Segmentierung von Anfragen an Suchmaschinen

Bachelorarbeit

Markus Riedel

1. Gutachter: Prof. Dr. Benno Stein

Datum der Abgabe: 28. Juli 2010

Erklärung

Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Die Arbeit wurde weder in dieser oder einer ähnlichen Form noch in Auszügen bereits einer Prüfstelle vorgelegt.

Weimar, den 28. Juli 2010

.....
Markus Riedel

Zusammenfassung

Die Menge an Webseiten und Informationen im World Wide Web ist bereits jetzt unüberschaubar und steigt stetig an. Für Internetnutzer wächst dadurch die Bedeutung der Suchmaschinen. Dabei ist es sowohl für den Nutzer, als auch für die Betreiber wichtig, dass Suchanfragen möglichst präzise gestellt werden, um möglichst präzise Ergebnisse liefern zu können. Ein übliches Vorgehen zur Präzisierung einer Suchanfrage ist das Segmentieren der Anfrage mit Anführungszeichen.

Nur wenige der Internetnutzer wissen von der Möglichkeit ihr Suchanfragen mit Hilfe von Anführungszeichen zu präzisieren. Darum wird in dieser Arbeit das automatische Segmentieren einer Suchanfrage untersucht. Es werden existierende Verfahren analysiert und mit neu entwickelten Verfahren verglichen. Als Grundlage dient das LenPowLen-Verfahren [HPSB10]. Ziel ist es, die Gewichtung des LenPowLen-Verfahrens zu optimieren und durch die Verwendung von Wikipedia die Genauigkeit der Segmentierungen zu steigern.

Anhand von Experimenten wird gezeigt, wie wirkungsvoll die neuen Algorithmen arbeiten und wo noch Verbesserungen vorgenommen werden können.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Seit der Jahrtausendwende hat sich die Zahl der Personen mit Zugriff auf das Internet vervierfacht. Ende 2009 hatten circa 1,8 Milliarden Menschen[1] Zugriff auf über 192 Millionen Top Level Domains[2]. Allein die Vielzahl an Domains mit dem Ländercode .de ist mit 13.706.899 unüberschaubar[3].

Die vielleicht wichtigste Frage, die sowohl Internet-Einsteiger als auch Internet-Firmen beschäftigt, ist: Wie finden Nutzer die Informationen, die sie benötigen?

Die Lösung des Problems bieten Suchmaschinen. Mit ihrer Hilfe werden die zu der eingegebenen Suchanfrage passenden Webseiten schnell und einfach gefunden. Diese Dienstleistung ist für den Nutzer kostenlos und wird hauptsächlich durch Werbeeinblendungen, die den Suchanfragen entsprechen, finanziert.

Die steigende Informationsmenge in Verbindung mit der kostenlosen und einfachen Nutzung und die Möglichkeit gezielt Werbung einzublenden mögen die Ursachen für die stetig wachsende Bedeutung von Suchmaschinen sein. Innerhalb eines Jahres stieg die Anzahl der Suchanfragen an Suchmaschinen um 46% auf circa 4 Milliarden[4] Suchanfragen täglich im Dezember 2009. Damit gehören Suchmaschinen zu den wichtigsten Diensten im World Wide Web und außerdem zu den am stärksten wachsenden Märkten weltweit. Der größte und zugleich bekannteste Anbieter ist der US-amerikanische Konzern Google. Auf der *Global 500*[5], der Liste der größten Unternehmen weltweit anhand ihres

[1] http://www.internetworldstats.com/stats.htm (27.05.2010)

[2] http://www.verisign.com/domain-name-services/domain-information-center/domain-name-resources/domain-name-report-feb10.pdf (27.05.2010)

[3] http://www.denic.de/home.html#c21 (27.05.2010)

[4] http://comscore.com/Press_Events/Press_Releases/2010/1/Global_Search_Market_Grows_46_Percent_in_2009 (24.05.2010)

[5] http://media.ft.com/cms/419e021c-fecd-11de-91d7-00144feab49a.pdf (24.05.2010)

Marktwertes, wird Google mit einem Wert von über 150 Milliarden US-Dollar auf Rang 26 geführt (Stand: 12/2009). Drei Monate zuvor lag Google noch auf dem 36. Platz. Zum Vergleich: Der bayerische Automobilkonzern BMW liegt auf Rang 248 mit etwa einem Fünftel des Marktwertes von Google.

Als populärste Suchmaschine profitiert Google von der stetig steigenden Zahl von Internetnutzern und Webseiten. Oftmals wird „Suchen im Internet“ mit „Googlen“ gleichgesetzt. Folgendes Zitat aus einer TV-Sendung verdeutlicht den Bekanntheitsgrad Googles: „Googlen ist ein Begriff, der sogar im Duden steht. Wenn Sie nicht wissen, was der Duden ist, googlen Sie es einfach.“[6].

Da das Thema dieser Arbeit aber nicht „besser googlen“ lautet, sondern Suchanfragen an alle gängigen Suchmaschinen präzisiert werden sollen, wird hier nur noch von Anfragen an Suchmaschinen gesprochen.

Die einfache Bedienung der Suchmaschinen ermöglicht es, so leicht wie nie zuvor an Informationen zu gelangen, obwohl die Menge an Webseiten und die damit verbundene Datenmenge stets zunimmt. Nichtsdestotrotz belegen Statistiken[7], dass sich viele Internetnutzer von den präsentierten Ergebnissen, die häufig mehrere Millionen Webseiten und Dokumente umfassen, nur jeweils die ersten Ergebnisse anschauen. Wenn diese nicht den Erwartungen des Nutzers entsprechen, werden oft neue Anfragen gestellt oder sogar die Suche abgebrochen. Um so wichtiger ist es, dass schon die ersten Ergebnisse den Vorstellungen des Benutzers entsprechen.

Die bedeutendsten Funktionen von Suchmaschinen zur Präzisierung der Resultate sind die Überprüfungen und gegebenenfalls Korrekturen der Eingabe des Nutzers. Die Rechtschreibehilfe, die unter anderem auf mögliche Tippfehler hinweist, und auch die Eingabevervollständigung sollen in dieser Arbeit keine weitere Beachtung finden. Eine der ältesten und üblichsten Möglichkeiten der Präzisierung von Suchanfragen ist die Segmentierung (Unterteilung) der Anfrage mittels Anführungszeichen.

Anhand eines einfachen Beispiels soll deutlich werden, wie wichtig möglichst präzise Anfragen sind: Angenommen man möchte wissen wie viele Suchanfragen täglich und weltweit an Suchmaschinen gestellt werden. Neben der bereits erwähnten Suchmaschine Google sind Bing und Yahoo! die bekanntesten Wettbewerber.

Die Eingabe von `Suchanfragen pro Tag` bei Yahoo! bringt leider kein gewünschtes Ergebnis unter den ersten drei Resultaten (siehe Abbildung 1.1). Auf der ersten Seite, also unter den ersten zehn Ergebnissen, findet sich als bester Treffer ein Wert für den deutschen Markt, aber kein Ergebnis, dass ei-

[6] http://www.nagelsbaum.de/tag/duden/ (27.05.2010)

[7] http://www.phaydon.de/marktforschung-publikationen-fachartikel-studien_bvdw-google-studie.html (22.07.2010)

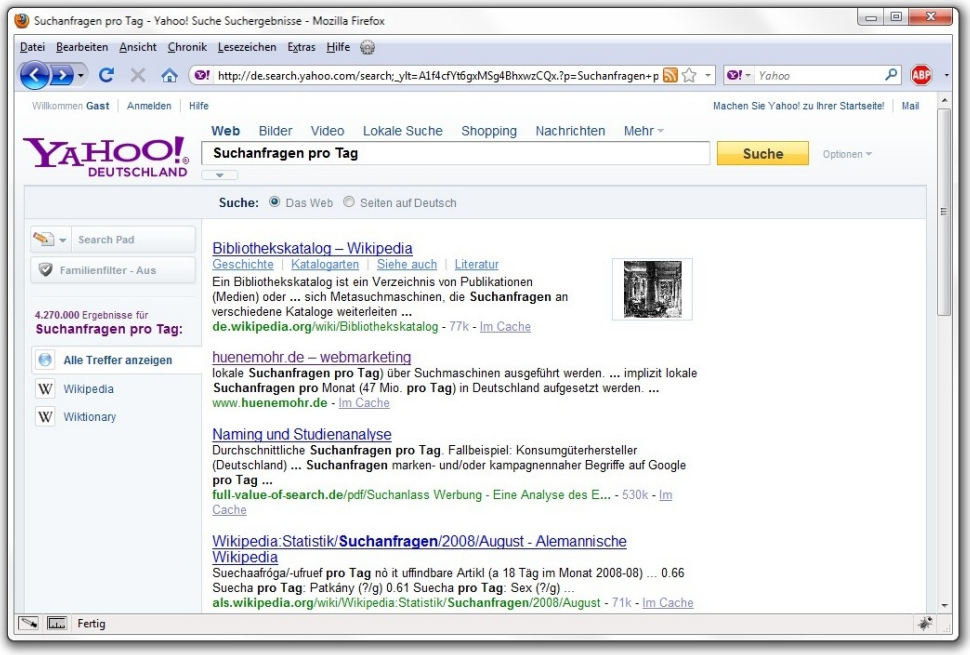


Abbildung 1.1: Suchanfrage ohne Segmentierung.

ne Zahl der weltweiten Suchanfragen liefert. Erweitert man die Eingabe um Anführungszeichen zu "`Suchanfragen pro Tag`", so wird man bereits mit dem ersten Ergebnis fündig (siehe Abbildung 1.2).

Generell werden Worte einer unsegmentierten Anfrage einzeln behandelt. Das heißt, dass zwar alle Worte in der Anfrage im Dokument vorkommen sollen, allerdings wird weder der Reihenfolge noch der Position im Text Beachtung geschenkt. Im Gegensatz dazu bedeutet die Eingabe einer mit Anführungszeichen segmentierten Anfrage, dass die Suchmaschine diese Worte in genau der eingegebenen Reihenfolge und direkt hintereinander im Dokument finden muss. Erst wenn das der Fall ist, wird das Dokument in der Ergebnisliste angezeigt.

Ein ähnliches automatisches Verfahren nutzt zum Beispiel Google. Hier werden nebeneinander stehende Worte stärker gewichtet, sodass Ergebnisse mit der exakten Suchanfrage weiter vorn in der Ergebnisliste aufgeführt werden. Bei der Beispielanfrage `Suchanfragen pro Tag` funktioniert dieses Verfahren so gut, dass Google das gewünschte Ergebnis bereits auf Platz 2 listet. Die Eingabe der Anfrage mit Anführungszeichen ändert daran nichts.

Bei komplexeren oder gar mehrdeutigen Anfragen mit mehreren Wortgruppen reicht allerdings Googles auf Nähe basierendes Gewichtungsverfahren nicht aus. Um die Präzision der Ergebnisse zu steigern wird es nötig die zusammengehörigen Wortgruppen mit Hilfe von Anführungszeichen zu gruppieren.

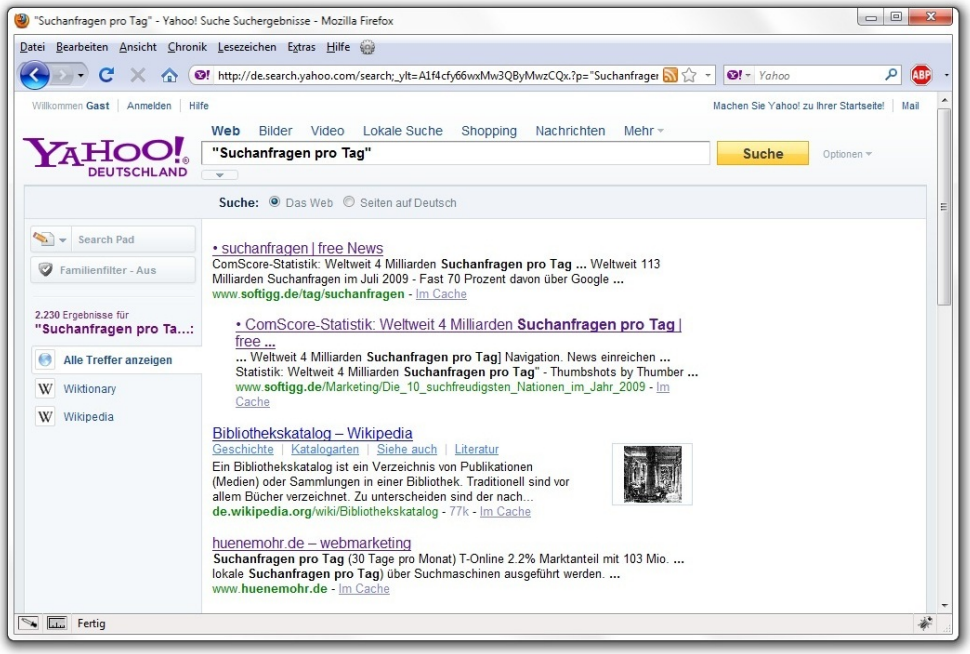


Abbildung 1.2: Suchanfrage mit Segmentierung.

Jedoch muss der Nutzer diese Segmentierung bis jetzt noch selbstständig tätigen. Dazu ist es notwendig, von der Möglichkeit der Segmentierung zu wissen und diese effektiv anwenden zu können. Vor allem Internet-Einsteiger wissen aber oft nichts von der Chance, ihre Anfrage zu präzisieren. Darum sollten noch nicht segmentierte Anfragen durch die Suchmaschine segmentiert werden, bevor nach Resultaten gesucht wird. Der Nutzer muss so keine Kenntnis von dieser Möglichkeit der Anfragenpräzisierung haben, um von dem Vorteil segmentierter Anfragen zu profitieren.

Durch segmentierte Anfragen können Suchmaschinen präzisere Ergebnisse liefern, die mehr den Erwartungen des Nutzers entsprechen [RMB03]. In Folge dessen könnten weitere Suchanfragen zur Präzisierung der Ergebnisse unnötig werden und somit sowohl Traffic als auch die Zeit von Nutzern und Betreibern eingespart werden. Vor allem für Letztere ist somit ein effizientes Verfahren zur Anfragenpräzisierung ein finanzieller Vorteil, da nicht nur durch Einsparung von Traffic Kosten reduziert werden können, sondern auch durch gezieltere Werbung Einnahmen gesteigert werden können. Somit ist die Optimierung und auch speziell die Segmentierung von Anfragen an Suchmaschinen ein wichtiger Zweig in der Forschung.

In dieser Arbeit wird das LenPowLen-Verfahren [HPSB10] genutzt und optimiert. Dabei werden Verbesserungen der Präzision durch die Nutzung von

Wikipedia und einem veränderten Gewichtungsverfahren namens Median untersucht.

In Kapitel 2 werden die grundlegenden Begriffe, die in dieser Arbeit verwendet werden, und bereits existierende Verfahren aufgezeigt. Die von uns entwickelten Verfahren mit entsprechenden Implementierungsdetails werden in Kapitel 3 erläutert. Eine ausführliche Analyse der Ergebnisse unserer Verfahren kann in Kapitel 4 gefunden werden. Kapitel 5 schließt diese Arbeit mit der Zusammenfassung und einem Ausblick ab.

Kapitel 2

Grundlagen und bekannte Verfahren

In der Einleitung wurde bereits über das Problem der Ungenauigkeit von Suchanfragen berichtet. Auch andere Forscher suchen nach Verbesserungen für Web-Suchmaschinen. In diesem Kapitel werden Verfahren vorgestellt, die sich mit dem Thema der Optimierung von Suchanfragen beschäftigen.

Eine Anfrage Q (Query) ist die Sequenz aller Worte, die der Benutzer im Suchfeld einer Suchmaschine eingibt. Das Beispiel `Suchanfragen pro Tag`, aus der Einleitung dieser Arbeit, ist eine Anfrage mit 3 Worten. Mehrere Worte, die eine Wortgruppe bilden heißen Phrase. Für `Suchanfragen pro Tag` existieren 3 Phrasen: `Suchanfragen pro Tag`, `Suchanfragen pro` und `pro Tag`. Phrasen sind also Wortgruppen der Anfrage, ohne Worte zu überspringen. Das heißt, dass `Suchanfragen Tag` keine Phrase ist, da das Wort `pro` ausgelassen wurde. In der Einleitung wurde die Anfrage durch `"Suchanfragen pro Tag"` segmentiert, also mit Anführungszeichen versehen. Diese Anfrage beinhaltet genau eine Phrase, beziehungsweise ein Segment. Ein Segment ist eine segmentierte Phrase. Phrasen beinhalten, laut der hier eingeführten Definition, mindestens zwei Worte. Allerdings wird sich in Kapitel 3 zeigen, dass Segmente auch nur ein Wort enthalten können. Ein Segment S ist also eine segmentierte Phrase oder ein segmentiertes Wort. Eine Anfrage heißt segmentierte Anfrage, wenn alle enthaltenen Worte oder Phrasen segmentiert sind. Ein Beispiel hierfür ist `"Suchanfragen" "pro Tag"`. Es existieren mehrere Möglichkeiten zur Segmentierung einer Anfrage. In Kapitel 3 wird detailliert erläutert welche Segmentierungen von Anfragen existieren und wie diese berechnet werden können.

Einer der ersten und einfachsten Ansätze zur Segmentierung von Anfragen ist der von **Risvik et al.** [RMB03]. Bereits im Jahr 2003 entwickelten sie ein Verfahren, dass benachbarte Worte der Anfrage auf Zusammengehörigkeit mittels deren Häufigkeit und der daraus bestimmten *connexity scores* testet. Die *connexity scores* gewichten dabei die Zusammenhänge der Worte und Phrasen der Anfrage. Die Gewichte repräsentieren die Häufigkeiten der Vorkommen der Phrasen in einem Anfragen-Log. Risvik et al. testeten ihre Algorithmen auf einer eigenen Beispielanfragenmenge aus einem Suchmaschinen-Log, sodass keine mit aktuellen Forschungen vergleichbaren Ergebnisse vorliegen. Sie zeigten jedoch, dass die Segmentierung von Anfragen mit relativ einfachen Methoden Verbesserungen in der Ergebnisgenauigkeit bei hoher Verarbeitungsgeschwindigkeit möglich macht.

Jones et al. [JRMG06] arbeiteten ein Verfahren aus, dass Suchanfragen verändert. Sie kategorisierten den Grad der Veränderung in vier Stufen. Die geringste Änderung ist *precise rewriting*. Ein Beispiel hierfür ist das Ändern der Wortreihenfolge. Dabei wird die Intention der Anfrage nicht verändert und damit die Transparenz, welche Änderungen vorgenommen worden, für den Benutzer komplett erhalten. Mittels *approximate rewriting* (Stufe 2) wird eine Anfrage oder Phrase zu einem nah verwandten Thema geändert, die Intention aber beibehalten (zum Beispiel: `Apple MP3 Player` → `ipod shuffle`). Stufe 3 (*possible rewriting*) ändert die Intention des Nutzers, aber nicht die Kategorie. So wird beispielsweise aus `Brille` → `Kontaktlinsen`. Schlussendlich ist der höchste Grad der Änderung der, der die Intention des Nutzers komplett verfehlt, ein *clear mismatch*.

Mit Hilfe der ersten beiden Stufen kann die Präzision der Suche verbessert werden. Um die Suche auf angrenzende Themengebiete zu erweitern, zieht man *possible rewriting* hinzu. Als Grundlage für die statistischen Untersuchungen werden Quellen wie WordNet[1], eine lexikonähnliche Datenbank der englischen Sprache, und indizierte Web-Dokumente genutzt. Außerdem gebrauchen Jones et al. Ankertexte und benutzerspezifische Anfragen-Logs um Synonyme und mögliche Änderungen für Worte und Phrasen in der Suchanfrage zu finden. Jones et al. nutzen ein oder mehrere Features zur Manipulation von Anfragen. Die insgesamt 37 Features können in drei Typen eingeteilt werden: Charakteristische Unterschiede, syntaktische und statistische Alternativen.

Das Verfahren von **Bergsma und Wang** [BW07] nutzt statistische Merkmale, wie die Häufigkeit eines Wortes beziehungsweise der möglichen Segmente im Web, und sonstige Merkmale, beispielsweise Artikel oder Positionen von

[1] http://wordnet.princeton.edu/ (12.05.2010)

Worten, um die Anfragen zu segmentieren. Untersucht wurde auch, dass die Beachtung von Groß- und Kleinschreibung und die Schreibweise mit beziehungsweise ohne Bindestrich der Worte in der Anfrage keine Verbesserung hervorbrachte. Bergsma und Wang diagnostizierten, dass Segmente die häufiger in anderen Anfragen vorkommen, durch höhere Gewichtung zu einer Verbesserung der Segmentierung beitragen können. Außerdem eruierten sie die Kombination von Worten, die nicht direkt hintereinander in der Anfrage vorkommen. Zu bemerken ist, dass sie für jedes Segment eine neue Anfrage an eine Suchmaschine stellen und somit zur Segmentierung einer Anfrage mehrmals Suchmaschinen anfragen. Das bedeutet, dass der Rechenaufwand und der Traffic für die Berechnung der Segmentierung unnötig steigt.

Ihr Verfahren evaluierten Bergsma und Wang gegen den von ihnen veröffentlichten Goldstandard. Der Goldstandard beinhaltet 500 Suchanfragen aus dem 2006 veröffentlichten AOL-Anfragen-Log[2]. Jede Anfrage umfasst zwischen 4 und 11 Worte. Jeweils drei unabhängige Annotatoren segmentierten die Anfragen. Die Goldstandard Intersection beinhaltet 220 Anfragen, die alle drei Annotatoren gleich segmentiert haben. Einige Forscher im Bereich der Segmentierung von Anfragen nutzen den von Bergsma und Wang [BW07] entwickelten Goldstandard für die Evaluationen ihrer Forschungsergebnisse. Der Goldstandard bietet sich als Testdatenmenge für eigene Verfahren an und bietet zudem eine gute Vergleichbarkeit diverser Verfahren, die ihre Goldstandard-Ergebnisse publizieren. Auch wir verwenden den Goldstandard und stellen unsere Resultate den besten Ergebnissen aller auf dem Goldstandard untersuchten Verfahren in Kapitel 4 gegenüber.

Gou et al. [GXLC08] untersuchten für Suchanfragen mehrere Verbesserungen, die vorrangig einzeln jeweils mit Schreibfehlererkennung, Wortverbindung und -trennung sowie der Segmentierung von Phrasen angewandt wurden. Die Kombination mehrerer Methoden führte durch deren Abhängigkeit voneinander zu Problemen. So kann zum Beispiel ein Fehler durch die erste Korrekturmethode erkannt und verbessert werden, der durch das anschließende Verfahren geeigneter korrigiert worden wäre. Eine weitere Korrektur ist allerdings nicht möglich, da der Fehler nach dem ersten Verfahren nicht mehr existiert. Das automatische Abwägen der geeignetsten Fehlerkorrektur, die die Intention des Nutzers beibehält, wurde von Gou et al. nicht untersucht.

Ihr Model basiert auf CRF (*conditional random fields*), ein ungerichtetes grafisches Modell, bei dem jeder Knoten eine Zufallsvariable repräsentiert und jede Kante die Beziehung der entsprechenden Knoten darstellt. Für die Anwendung zur Anfragensegmentierung gilt: ausgewählte Worte der Anfragen bilden

[2] http://www.gregsadetsky.com/aol-data/ (22.07.2010)

die Knoten des Graphen, Kanten die Abhängigkeiten der Worte. Gou et al. erweitern das CRF-Modell um bedingte Knoten, die eingegebene Worte der Anfrage repräsentieren, zu CRF-QR (*conditional random fields - query refinement*).

Als Ergebnis zeigten Gou et al., dass die größte Nutzensteigerung die Rechtschreibkorrektur brachte, gefolgt von Worttrennung, Wortvereinigung und schlussendlich die Segmentierung der Anfrage. Die Rechtschreibkorrektur ist eine der komplexesten Methoden bei der Verarbeitung von Suchanfragen. Gleichzeitig impliziert ein Gewinn durch die Rechtschreibkorrektur natürlich das Vorhandensein von vielen Rechtschreibfehlern. Da der Goldstandard, den wir als Testdatensatz verwenden, nur sehr wenig Rechtschreibfehler enthält, würde eine Korrektur sich nur marginal auf die Ergebnisse auswirken.

Ebenfalls ein CRF-Modell nutzen **Yu und Shi** [YS09] und auch **Kiseleva et al.** [KGA+10]. Beide nutzen Label zur Gruppierung von Worten. Bei Kiseleva et al. werden die Eingaben des Nutzers im Suchfeld eines Onlineshops gelabelt, um gezielt nach Produktinformation in der Datenbank suchen zu können. Worte mit gleichem Label werden zusammengefasst und als Segment in einer relationalen Datenbank gesucht. Durch das Nutzen der Shop-Datenbank sind die möglichen Label auf die Spalten der Tabellen der Datenbank begrenzt. Dadurch wird dieser Ansatz bei der Web-Suche leider unmöglich, da dort quasi unendlich viele Gruppierungen von Worten möglich sind. Die Verfahren von Yu und Shi und Kiseleva et al. können somit nicht auf die Untersuchungen für Websuchmaschinen übertragen werden.

Bendersky et al. [BCS09] entwickelten ein Verfahren namens *Two-Stage Query Segmentation*. Es besteht aus zwei Phasen: Analyse der syntaktischen Struktur der Anfrage und Nutzung von externen Informationen, wie Anfragen-Logs. Sie vergleichen Ihr Verfahren hauptsächlich mit dem *sequential dependence model*. Das *Sequential dependence model* ist ein grafisches Modell, dass nur die Abhängigkeiten benachbarter Phrasen beschreibt. Bendersky et al. zeigten experimentell, dass *Two-Stage Query Segmentation* bei ähnlicher Präzision etwa doppelt so schnell arbeitet wie ein auf dem *sequential dependence model* basierendes Verfahren.

Tan und Peng [TP08] segmentierten Anfragen unter Verwendung eines generativen Sprachmodells, dass auf dem EM-Algorithmus (*expectation-maximization*) und Wikipedia-Daten basiert. Letztere bestehen aus 2,3 Millionen Titeln von Wikipedia-Artikeln und den dazugehörigen Ankertexten. Beim Finden einer Phrase in den Wikipedia-Daten wird diese durch einen Bonus von 100.000 Punkten zusätzlich zur ihrer Häufigkeit aufgewertet. Tan und

Peng greifen vorrangig auf vorhandene statistische Daten in Form von 1% aller von Yahoo! erfassten Webseiten zurück. *Expectation-maximization* nutzt die *Maximum-Likelihood* Methode, um unbekannte Daten abzuschätzen.

Laut Tan und Peng haben andere Verfahren zur Anfragensegmentierung große Nachteile. So werden bei dem von ihnen als Baseline genutzten *mutual information* Algorithmus nur die statistischen Verbindungen von zwei Worten untersucht und häufig vorkommende Pattern bevorzugt, auch wenn kein inhaltlicher Zusammenhang besteht. Bei *supervised learning* Verfahren (wie etwa [BW07]) müssen hingegen eine große Menge von gelabelten Daten und gut durchdachte Features vorhanden sein. Das steigert allerdings die Komplexität und erschwert die Portierung auf andere Sprachen.

Bei den Untersuchungen von Tan und Peng ist das *mutual information* Modell das ungenaueste bei der Segmentierung von Suchanfragen. Die Genauigkeit der Segmentierung steigerten sie durch Nutzung des *expectation-maximization* Algorithmus. Die besten Ergebnisse lieferte ihr auf dem EM-Algorithmus basierendes generatives Sprachmodell, dessen Genauigkeit durch die Verwendung der Wikipedia-Daten nochmals gesteigert werden konnte.

Tan und Peng nutzten den Goldstandard von [BW07] um ihre Forschungsergebnisse zu evaluieren. Auch ihre Forschungsergebnisse werden in Kapitel 4 analysiert.

Zhang et al. [ZSH+09] wählen einen ganz anderen Ansatz zur Segmentierung von Anfragen. Sie nutzten Ähnlichkeiten im Vektorraummodell um Suchanfragen zu kategorisieren. Dabei werden sowohl die vorhandenen statischen Daten, beziehungsweise die indizierten Dokumente, als auch die Anfrage als Vektor dargestellt. Ein Wort entspricht dabei einer Dimension des Vektors. Die Vektoren werden anschließend entsprechend des Verfahrens verglichen. Ähnliche Vektoren lassen auf ähnliche Inhalte schließen. Als Resultat der Suche werden also Dokumente ausgewählt, deren Vektor ähnlich zu dem der Anfrage ist. Auch dieses Verfahren wurde auf den Goldstandard evaluiert, sodass die Ergebnisse in Kapitel 4 mit unserem Verfahren verglichen werden.

Hagen et al. [HPSB10] entwickelten das LenPowLen-Verfahren. Sie nutzen die Google-N-Gramm-Kollektion [BF06] als Quelle für statistische Daten, die für die Gewichtung der Phrasen der Anfrage verwendet werden. Da das LenPowLen-Verfahren die Grundlage für diese Arbeit bildet, wird es in Abschnitt 3.2.3 detailliert erläutert. In Kapitel 4 werden die Ergebnisse der Berechnungen von Hagen et al. untersucht und mit den Verfahren von Tan und Peng, Bergsma und Wang, sowie Zhang et al. und den in dieser Arbeit entwickelten Optimierungen des LenPowLen-Verfahrens verglichen.

Kapitel 3

Unser Verfahren

In diesem Kapitel werden die von uns untersuchten Verfahren zur Segmentierung von Suchanfragen an Suchmaschinen beschrieben. Diese lassen sich in zwei Kategorien unterteilen. Die Grenzwertverfahren werden in Abschnitt 3.1 und die Gewichtungsverfahren in Abschnitten 3.2 ausführlich beschrieben. Alle hier untersuchten Verfahren sind statistischer Algorithmus zur Anfragensegmentierung. Das Hauptaugenmerk ist die Weiterentwicklung des bekannten LenPowLen-Verfahrenes [HPSB10].

Für alle untersuchten Verfahren gelten folgende Grundlagen: Zur Gewichtung der Segmente werden zwei verschiedene statistische Merkmale genutzt. Das wichtigste Merkmal, die Häufigkeit eines Segments S in der Google-N-Gram-Kollektion [BF06], wird durch den Score s angegeben. Das Gewicht g eines Segments ist durch das jeweilige Gewichtungsverfahren bestimmt. Die Summe aller Gewichte g der in der Segmentierung enthaltenen Segmente bildet das Gesamtgewicht G einer Segmentierung.

In Abschnitt 3.3 untersuchen wir das zweite statistische Merkmal: Das Vorkommen von Segmenten bei Wikipedia.

Es wird jeweils beispielhaft auf die Anfrage `san jose yellow pages` Bezug genommen. Im Goldstandard ist diese Beispielanfrage durch `"san jose"` `"yellow pages"` segmentiert. Ein optimales Verfahren sollte also diese Segmentierung für die Anfrage `san jose yellow pages` zurückgeben. Detailliertere Ergebnisse für den Goldstandard sind in Kapitel 4 zu finden.

3.1 Grenzwertverfahren

Die „primitivsten“ von uns untersuchten Verfahren sind die Grenzwertverfahren. Während die in Abschnitt 3.2 erläuterten Gewichtungsverfahren Scores in Abhängigkeit von der Länge eines Segments gewichten, nutzen die Grenzwertverfahren Front-to-Back (siehe Abschnitt 3.1.1) und Back-to-Front (siehe

Abschnitt 3.1.2) die Scores direkt.

Die Grenzwertverfahren unterscheiden nur zwischen zwei Stadien: Gültig und ungültig. Über die Gültigkeit eines Segments entscheidet dessen Score und der eingegebene Grenzwert. Der Grenzwert T (Threshold) limitiert die Gültigkeit oder die Gewichtung (bei den Gewichtungsverfahren) der Segmente. Ist der Score s kleiner als der Grenzwert, so ist das Segment ungültig. Ein Segment ist gültig, wenn $s \geq T$. Eine gültige Segmentierung besitzt nur gültige Segmente, hingegen ist eine Segmentierung ungültig, wenn sie mindestens ein ungültiges Segment enthält.

Als Grenzwerte wurden verschiedene Werte, die auf unterschiedlichen Überlegungen basieren, evaluiert. Ein Grenzwert, der alle Segmente die existieren als gültig festlegt hat den Wert 1. Hingegen beschneidet der Grenzwert 1000 alle Segmente, die zu selten vorkommen. So werden Segmente mit einer Häufigkeit von unter 1000 als ungültig definiert. Die Beispiele in den folgenden Abschnitten beziehen sich jeweils auf den Grenzwert $T = 1$.

3.1.1 Front-to-Back

Der Front-to-Back-Algorithmus durchläuft eine Anfrage folgendermaßen: Der Score der ersten beiden Worte wird aus der Google-N-Gramm-Kollektion ausgelesen. Ist dieser Score s größer als der definierte Grenzwert T, so werden beide Worte als ein Segment S in einem Zwischenspeicher gespeichert. In unserem Beispiel besitzen die ersten 2 Worte `san jose` einen Score von circa 14,5 Millionen und liegen damit deutlich über dem Grenzwert 1. Zusammen mit dem Zwischenspeicher wird das nächste Wort als ein Segment gemerkt und deren Häufigkeit bestimmt. Dabei erzielt `san jose yellow` einen Score von circa 8800. Da der Score dieses Segments wiederum größer ist als der Grenzwert, wird das Segment im Zwischenspeicher gespeichert und mit dem nächsten Wort fortgefahren. Im Beispiel wird somit nun noch `san jose yellow pages` mit einem Score von circa 8700 festgestellt. Damit ist die Anfrage durchlaufen und die Berechnung mit dem Ergebniss `"san jose yellow pages"` abgeschlossen.

Sollte der Score kleiner als der Grenzwert sein, dann wird der Zwischenspeicher, der das letzte gültige Segment enthält, als festgelegtes Segment für diese Anfrage gespeichert. Das neue Wort, dass in Kombination mit dem vorhandenen Segment zur Ungültigkeit geführt hat, wird im Zwischenspeicher als Anfang eines neuen Segments abgelegt. Dies wird so lange durchgeführt bis alle Worte in der Anfrage durchlaufen wurden.

Allerdings existiert die Beispielanfrage so häufig in der Google-N-Gramm-Kollektion, dass alle Worte zu einem Segment zusammengefasst werden. Das optimale Ergebnis laut Goldstandard ist jedoch `"san jose" "yellow pages"`.

Segmente	Score
`san`	151.400.000
`san jose`	14.495.804
`san jose yellow`	8.822
`san jose yellow pages`	8.739
`jose`	20.400.000
`jose yellow`	8.831
`jose yellow pages`	8.745
`yellow`	82.000.000
`yellow pages`	41.380.676
`pages`	234.000.000

Tabelle 3.1: Mögliche Segmente und Häufigkeiten für die Anfrage: `san jose yellow pages`.

3.1.2 Back-to-Front

Anhand des Beispiels sieht man einen entscheidenden Nachteil, den Front-to-Back mit sich bringt. Ist die Anfrage auch als ein Segment in der Google-N-Gramm-Kollektion zu finden, dann segmentiert Front-to-Back die komplette Anfrage als ein Segment. Das ist aber nicht zwangsweise die beste Variante der Segmentierung.

Der Back-to-Front-Algorithmus ist dem Front-to-Back-Algorithmus sehr ähnlich. Seinem Namen entsprechend wird beim Back-to-Front-Algorithmus die Anfrage beim letzten Wort beginnend bis zum ersten Wort durchlaufen. Dies soll den Nachteil der Front-to-Back-Routine beseitigen. Die Idee dabei ist: Der Nutzer gibt die Anfrage vom ersten zum letzten Wort schlüssig ein. Beim Parsen der Anfrage von vorn nach hinten werden von Front-to-Back oft „zu lange“ Segmente gefunden. Das Parsen von hinten nach vorn könnte eher abbrechen, wenn die Worte dann nicht mehr so lange Segmente ergeben.

Anhand unserer Beispielanfrage ist ersichtlich, dass auch dieses Vorgehen das Problem nicht beseitigt. In Tabelle 3.1 wird deutlich, dass alle für diese Anfrage möglichen Segmente einen Score haben, der größer als die Grenzwerte 1 oder 1000 ist. So segmentiert auch Back-to-Front diese Anfrage mit "`san jose yellow pages`". Eine andere Anfrage aus dem Goldstandard beweist die Vorteile von Back-to-Front. Die Anfrage `college football draft prospects` wird von Front-to-Back mit dem Grenzwert 1 mit "`college football draft`" "`prospects`" segmentiert. Denn die Phrase `college football draft` besitzt in der Google-N-Gramm-Kollektion immer noch eine Häufigkeit von 215. Eine geeignetere Segmentierung, die auch der Goldstandard vorschlägt, ist hingegen "`college football`" "`draft prospects`". So segmentiert auch Back-to-

"san"	"jose"	"yellow"	"pages"
"san"	"jose"	"yellow	pages"
"san"	"jose	yellow"	"pages"
"san"	"jose	yellow	pages"
"san	jose"	"yellow"	"pages"
"san	jose"	"yellow	pages"
"san	jose	yellow"	"pages"
"san	jose	yellow	pages"

Tabelle 3.2: Mögliche Segmentierungen der Anfrage: `san jose yellow pages`.

Front diese Anfrage. Da das Segment `football draft prospects` keinen Score besitzt bricht der Algorithmus die Segmentbildung nach `draft prospects` ab und bildet ein neues Segment `college football`. Aber auch hier gilt: Der Vorteil von Back-to-Front ist gering. Denn wählt man $T = 1$, so segmentiert auch Front-to-Back diese Anfrage so, wie es der Goldstandard vorschreibt. Die Wahl des besten Grenzwertes der Grenzwertverfahren soll in dieser Arbeit keine weitere Beachtung finden. Stattdessen werden die genaueren Gewichtungsverfahren untersucht.

3.2 Gewichtungsverfahren

Ziel eines Gewichtungsverfahren ist es, die statistischen Merkmale der Segmente so zu gewichten, dass die geeignetste der möglichen Segmentierungen ausgewählt wird.

Die grundlegende Herangehensweise zur Segmentierung einer Anfrage bei den Algorithmen Summe, LenPowLen und Median ist immer gleich:

Um eine Anfrage zu segmentieren werden alle möglichen Segmente (siehe Abschnitt 3.2.1) errechnet und deren Häufigkeiten in der Google-N-Gramm-Kollektion bestimmt. Diese Scores werden nun anhand der Gewichtungsverfahren bewertet. Die dem Verfahren nach am besten gewichtete Segmentierung der Anfrage wird gewählt.

3.2.1 Mögliche Segmentierungen einer Anfrage

Bei der Analyse der Anfragen gehen wir davon aus, dass der Benutzer die Worte bewusst gewählt und in einer sinnvollen Reihenfolge eingegeben hat. Darum bleibt die Wortreihenfolge bei der Erstellung der möglichen Segmente pro Anfrage erhalten. Außerdem wollen wir keine Worte überspringen, da jedes Wort in der Anfrage einen Informationsgehalt hat, der beim Auslassen

Segmentierungen mit S(Gewicht g)			G
"w_1"(15)	"w_2"(30)	"w_3"(5)	0
"w_1"(15)	"w_2	w_3"(0)	-1
"w_1	w_2"(10)	"w_3"(5)	10
"w_1	w_2	w_3"(0)	-1

Tabelle 3.3: Allgemeine Beispielsegmentierung mit Gesamtgewicht G des Summe Algorithmus

des Wortes verloren gehen würde. Allgemein gilt: für n Worte in der Anfrage ist die Anzahl der Segmentierungen 2^{n-1}. Tabelle 3.2 zeigt die möglichen Segmentierungen für die Beispielanfrage `san jose yellow pages`. Aus der dieser Anfrage der Länge 4 ergeben sich $2^{4-1} = 8$ mögliche Segmentierungen.

Unser Algorithmus zur Berechnung der möglichen Segmentierungen arbeitet nicht direkt mit Worten und deren Kombinationsmöglichkeiten zu Segmenten, sondern mit Trennstellen innerhalb der Anfrage. Eine Trennstelle ist die Position zwischen zwei Worten. Es ergeben sich also bei n Worten $n - 1$ Trennstellen. Wird an einer Trennstelle nicht getrennt, so bedeutet das, dass die Worte vor und nach der Trennstelle zu einem Segment gehören (Implementierung siehe Abschnitt 3.4).

3.2.2 Summe

Das Summe-Verfahren bildet in Hinblick auf die Arbeitsweise und Komplexität eine Brücke zwischen den Grenzwertverfahren und den umfassenderen Gewichtungsverfahren LenPowLen und Median. Letztere basieren jedoch auf der Herangehensweise des Summe-Algorithmus.

$$G = \sum g = \sum s \text{ für } s > T \text{ und } l > 1$$

Der Summe-Algorithmus iteriert über alle möglichen Segmentierungen der Anfrage. Dabei wird für jede Segmentierung das Gesamtgewicht G errechnet. Für jedes Segment S einer Segmentierung wird der Score s angefragt. Dieser ist das Gewicht des Segments. Es gilt also $g = s$. Durch das Summieren der Gewichte aller Segmente erhält die Segmentierung ihr Gesamtgewicht. Existiert ein Segment nicht in der Google-N-Gramm-Kollektion und ist damit ungültig, dann ist der Score dieses Segments $s = 0$ und damit kleiner als der definierte Grenzwert $T = 1$. Der Grenzwert kann höher gewählt werden, wenn Segmentierungen mit seltenen Segmenten aus den Segmentierungsmöglichkeiten aussortiert werden sollen. Wenn das Gewicht eines Segments kleiner als

Mögliche Segmentierungen				Gewicht A	Gewicht B
"san"	"jose"	"yellow"	"pages"	487.800.000	0
"san"	"jose"	"yellow	pages"	213.180.676	41.380.676
"san"	"jose	yellow"	"pages"	385.408.831	8.831
"san"	"jose	yellow	pages"	151.408.745	8.745
"san	jose"	"yellow"	"pages"	330.495.804	14.495.804
"san	jose"	"yellow	pages"	55.876.480	55.876.480
"san	jose	yellow"	"pages"	234.008.822	8.822
"san	jose	yellow	pages"	8.739	8.739

Tabelle 3.4: Segmentierungen von `san jose yellow pages` durch Summe.

der Grenzwert ist, erhält die komplette Segmentierung das Gesamtgewicht $G = -1$. Das hat zur Folge, dass das Gesamtgewicht dieser ungültigen Segmentierung kleiner ist, als das Gesamtgewicht jeder gültigen Segmentierung. Dadurch wird garantiert, dass immer mindestens eine gültige Segmentierung existiert: die Segmentierung, bei der alle Worte einzeln segmentiert sind.

Tabelle 3.3 zeigt ein allgemeines Beispiel der Gewichtung. Es ist zu sehen, dass die Worte w_2 und w_3 als Segment keinen Score haben, genau wie das Segment $w_1w_2w_3$. Das führt dazu, dass das Gesamtgewicht dieser Segmentierungen -1 ist. Segmente der Länge $l = 1$ werden nicht mit summiert, sodass die erste Segmentierung das Gesamtgewicht $G = 0$ hat. Die dritte Möglichkeit der Segmentierung wird vom Summe-Algorithmus ausgewählt, da ihr Gesamtgewicht am größten ist. Wäre das Segment w_1w_2 ebenfalls ungültig, dann hätte auch die dritte Segmentierung ein Gesamtgewicht von -1, sodass die erste Segmentierung mit $G = 0$ ausgewählt werden würde. Nach Durchlaufen aller möglichen Segmentierungen wählt der Algorithmus die Segmentierung mit dem höchsten Gesamtgewicht aus.

Für die Beispielanfrage `san jose yellow pages` wird die Segmentierung `"san jose" "yellow pages"` ausgewählt. In Tabelle 3.4 sind in Spalte „Gewicht B“ die Gesamtgewichte der Segmentierungen abzulesen. In Spalte „Gewicht A“ sind die Gesamtgewichte inklusive der Gewichte der Segmente der Länge 1 aufgelistet. Es ist ersichtlich, dass die erste Segmentierung das höchste Gewicht A besitzt. Da einzelne Worte fast immer deutlich häufiger auftreten als Segmente mit mehr als einem Wort, werden Segmente mit $l = 1$ nicht mit gewichtet.

Mögliche Segmentierungen				Gewicht A	Gewicht B
"san"	"jose"	"yellow"	"pages"	487.800.000	0
"san"	"jose"	"yellow	pages"	337.322.704	165.522.704
"san"	"jose	yellow"	"pages"	385.435.324	35.324
"san"	"jose	yellow	pages"	151.636.115	236.115
"san	jose"	"yellow"	"pages"	373.983.216	57.983.216
"san	jose"	"yellow	pages"	223.505.920	223.505.920
"san	jose	yellow"	"pages"	234.238.194	238.194
"san	jose	yellow	pages"	8.948.736	8.948.736

Tabelle 3.5: Segmentierungen von `san jose yellow pages` durch LenPowLen.

3.2.3 LenPowLen

Dieses Gewichtungsverfahren basiert auf der Überlegung, alle möglichen Segmentierungen einer Anfrage zu berechnen und bei der Gewichtung längere Segmente höher zu gewichten als kürzere Segmente. Der Grund für diese Überlegung ist die Abnahme der Scores bei zunehmender Segmentlänge l. Diese Abnahme des Scores soll nun durch eine Bevorzugung längerer Segmente so ausgeglichen werden, dass längere Segmente das gleiche oder ein größeres Gewicht besitzen wie kurze Segmente mit kongruentem Inhalt. In Tabelle 3.1 sind alle möglichen Segmente für die Anfrage `san jose yellow pages` aufgelistet. Es ist zu erkennen, dass der Score der längeren Segmente wie `san jose yellow` deutlich niedriger ist, als der Score der Segmente, die nur ein Teil der Worte dieses Segments enthalten. So besitzt zum Beispiel das Segment `san jose` einen 1643 mal so hohen Score. Darum wurde eine Gewichtung gewählt, die in Abhängigkeit von der Segmentlänge arbeitet.

$$G = \sum g = \sum l^l \cdot s \text{ für } s > T \text{ und } l > 1$$

Der LenPowLen-Algorithmus potenziert die Segmentlänge l wiederum mit l, was sich in zahlreichen Testdurchläufen (Experimente siehe Abschnitt 4.3) als der beste Ausgleich für die Abnahme der Scores bewährt hat. Die Potenz wird schließlich mit dem Score des Segments multipliziert. Das Gewicht eines Segments ist also $l^l \cdot s$.

Diese Rechnung wird für alle möglichen Segmentierungen einer Anfrage durchgeführt. Die Segmentierung mit dem höchsten Gesamtgewicht wird als beste Segmentierung ausgewählt.

Wie bereits in Abschnitt 3.2.1 erläutert, existieren für unsere Beispielanfrage `san jose yellow pages` 8 mögliche Segmentierungen. Die Scores der

Segmente S	Score s	Gewicht g
`san jose yellow pages`	8.739	4.107.330
`san jose yellow`	8.822	388.168
`san jose`	14.495.804	14.495.804
`san`	151.400.000	-

Tabelle 3.6: Auszug möglicher Segmente und Gewichte für die Anfrage: `san jose yellow pages`.

Segmente sind in Tabelle 3.1 abgebildet. Das Ergebnis der beschriebenen Berechnung lässt sich in Tabelle 3.5 in der Spalte „Gewicht A“ ablesen. Es ist ersichtlich, dass die Häufigkeiten der alleinstehenden Worte so hoch sind, dass die erste Segmentierung ausgewählt werden würde. Darum werden bei diesem Verfahren nur Segmente mit der Segmentlänge $l > 1$ gewichtet. Wie auch bei dem Summe-Algorithmus wird somit die Möglichkeit gewährleistet auch einzeln segmentierte Anfragen (wie in Tabelle 3.5 Zeile 1) als Ergebnis zu erhalten, wenn die anderen Segmentierungen ungültige Segmente enthalten.

In Tabelle 3.5 in der Spalte „Gewicht B“ sind die tatsächlichen Gewichte der Segmentierungen aufgelistet. Hier ist auch ersichtlich, dass die Segmentierung `"san jose" "yellow pages"`, die mit dem Goldstandard identisch ist, das höchste Gewicht hat und somit von LenPowLen ausgewählt wird.

3.2.4 Median

Das Median Gewichtungsverfahren beschreibt eine weitere Überlegung die Abnahme der Häufigkeiten bei zunehmender Segmentlänge zu kompensieren. Im Gegensatz zu LenPowLen, bei dem die Länge der Segmente mit sich selbst potenziert wird und somit nur eine grobe Schätzung der Kompensation darstellt, soll bei der Gewichtung mit Median ein präziserer Ausgleich geschaffen werden.

Grundlage für eine Berechnung dieses Ausgleichs ist auch hier die Google-N-Gramm-Kollektion [BF06]. Aus den 5-Grammen wurde zufällig 1 Million mal gezogen. Von diesen Phrasen sind 878.142 Datensätze Schlüsselwort-Phrasen. Um die Abnahme der Häufigkeiten dieser Phrasen zu erhalten, wurden die 1- bis 4-Gramme dieser 878.142 5-Gramme kalkuliert. Dazu wurde stets das letzte Wort der Phrase entfernten und der Score der neuen Phrase ermittelt. Tabelle 3.6 zeigt beispielhaft eine der 878.142 Phrasen. Aus der Kalkulation ergeben sich 878.142 Datensätze, jeweils mit den Scores der 1- bis 5-Gramme, die in 5 Listen gespeichert sind. Aus jeder Liste wurden nun der Medianwert der enthaltenen Werte errechnet. Das bedeutet, dass der Median

aller Scores der 5-Gramme gebildet wurde und äquivalent dazu auch der Median der 1- bis 4-Gramme. Der Median der 5-Gramme ist 1.129, der 4-Gramme 7.356, der 3-Gramme 78.733, der 2-Gramme 3.461.030 und der Median der 1-Gramme 488.046.401. Da wir auch bei diesem Gewichtungsverfahren einzeln stehende Worte nicht gewichten wollen, lassen wir den Median der 1-Gramme außer Acht.

Für die Ausgleichsgewichtung ist der Faktor entscheidend, mit dem ein längeres Segment gegenüber einem kürzeren Segment benachteiligt ist. Diesen Faktor erhält man aus den eben errechneten Medianwerten. So kommen beispielsweise 2-Gramme 44, 31-mal so häufig im Web vor wie ein durchschnittlicher 3-Gramm. Daraus ergibt sich ein Faktor von 44 mit dem die Häufigkeit eines Segments der Länge 3 multipliziert wird, um die Häufigkeitsabnahme zu einem Segment der Länge 2 zu kompensieren. Entsprechend multiplizieren wir Segmente mit $l = 4$ mit 470 und $l = 5$ mit 3065. Ein Segment der Länge 2 erhält bei uns den Faktor 1, da es als Grundlage gilt. Segmente, die nur ein Wort enthalten, werden wie bereits erwähnt nicht gewichtet. Die Gewichte einiger Segmente der Beispielanfrage sind in Tabelle 3.6 in der Spalte „Gewicht g“ aufgeführt.

$$G = \sum g = \sum median(l) \cdot s \text{ für } s > T \text{ und } l > 1$$

Der Median-Algorithmus arbeitet folgendermaßen: Für jede mögliche Segmentierungen der Anfrage wird das Gesamtgewicht G, als Summe der Gewichte g der enthaltenen Segmente, ermittelt. Das Gewicht eines Segments ergibt sich aus dem Produkt des errechneten Median-Faktor, der in Abhängigkeit der Segmentlänge l eingesetzt wird, und dem Score s des Segments. Für das Beispiel `"san jose" "yellow pages"` errechnet sich das Gesamtgewicht mit:
$G = (median(2) \cdot score(\text{"}san\ jose\text{"})) + (median(2) \cdot score(\text{"}yellow\ pages\text{"}))$
$G = (1 \cdot 14.495.804) + (1 \cdot 41.380.676)$
$G = 55.876.480$

Aus allen Segmentierungen wählt Median die Segmentierung mit dem höchsten Gesamtgewicht als beste Segmentierung der Anfrage aus. In Tabelle 3.7 sind alle Segmentierungen mit ihren jeweiligen Gesamtgewichten ersichtlich. Es ist zu sehen, dass die Segmentierung, die auch im Goldstandard bevorzugt wird, das höchste Gesamtgewicht besitzt.

Mögliche Segmentierungen				Gesamtgewicht
"san"	"jose"	"yellow"	"pages"	0
"san"	"jose"	"yellow	pages"	41.380.676
"san"	"jose	yellow"	"pages"	388.564
"san"	"jose	yellow	pages"	384.780
"san	jose"	"yellow"	"pages"	14.495.804
"san	jose"	"yellow	pages"	55.876.480
"san	jose	yellow"	"pages"	388.168
"san	jose	yellow	pages"	4.107.330

Tabelle 3.7: Segmentierungen von `san jose yellow pages` durch Median.

3.3 Wikipedia

In Kapitel 2 wurden bereits andere Herangehensweisen zur Segmentierung von Anfragen beschrieben. Einige dieser Verfahren (zum Beispiel [TP08]) nutzen Wikipedia als Quelle für statistische Daten um ihr Segmentierungsverfahren zu optimieren.

Auch wir haben uns für die Nutzung von Wikipedia entschieden um unseren Algorithmus zu optimieren. Wikipedia hat den Vorteil, dass es ständig aktuell gehalten wird und einen großen Umfang an Daten aller Fachrichtungen liefert. Nicht zuletzt bietet Wikipedia den Vorzug der kostenlosen und freien Nutzung.

In Abschnitt 4.3 wird beschrieben, welche Auswirkungen auf die Genauigkeit der Segmentierung das Wissen über das Vorkommen der Segmente bei Wikipedia hat. Segmente, die bei Wikipedia gefunden werden, sollen gegenüber nicht gefundenen Segmenten einen Vorteil erlangen. Dafür ist es notwendig zu wissen, welche Segmente bei Wikipedia vorkommen und welche nicht. Es existieren mehrere Möglichkeiten auf Daten von Wikipedia zuzugreifen. Eine Möglichkeit ist das Durchsuchen eines Titelregisters. In diesem sind alle Wikipedia-Titel und Weiterleitungen (Alias-Titel) gelistet. Eine weitere Option ist das Herunterladen aller Artikel[1]. Das hätte den Vorteil, sowohl in Titeln und Weiterleitungen als auch in den Artikel-Texten nach Segmenten suchen zu können. Aufgrund der zu verarbeitenden Datenmenge mit über 1, 8 GB und vor allem der aufwändigen Implementierung, entschieden wir uns für eine weitere Möglichkeit: Die Abfrage der Wikipedia-Webseite. Da wir ausschließlich englische Anfragen untersuchen wollen und nur wissen wollen ob ein Artikel mit dem entsprechenden Segment existiert oder dieses im Text beinhaltet, nutzen wir die englischsprachige Webseite für Mobiltelefone[2]. Das hat für uns weitere

[1] http://de.wikipedia.org/wiki/Wikipedia:Download (27.05.2010)
[2] http://mobile.wikipedia.org (05.06.2010)

Vorteile: Die Aktualität und der Minimalismus der Webseite. Sie beinhaltet nur ein Suchfeld und liefert als Ergebnis einen kurzen Text beziehungsweise Links auf die Artikel. So ist es möglich jedes Segment auf dieser Webseite als Anfrage zu stellen und das Resultat mit Hilfe von regulären Ausdrücken zu parsen. Dabei wird nur überprüft, ob ein oder mehrere Artikel gefunden wurden. Es wird außer Acht gelassen, ob die Suchanfrage eines Segments nur einen oder mehrere Artikel als Ergebnis liefert und ob das Segment im Titel oder im Text des Artikels steht.

Das Wissen über ein Segment bei Wikipedia wird je nach Verfahren unterschiedlich behandelt. Ob ein Segment bei Wikipedia vorkommt gibt das Zusatzgewicht z an.

Das LenPowLen-Verfahren addiert Zusatzgewichte in Abhängigkeit von der Segmentlänge l zu der in Abschnitt 3.2.3 beschriebenen Gewichtung. Daraus ergibt sich folgende Formel für das Gesamtgewicht G einer Segmentierung:

$$G = \sum g = \sum l^l \cdot s + z(l) \text{ für } s > T \text{ und } l > 1$$

Ein Segment der Länge 3 besitzt ein Zusatzgewicht $z(3)$ von 400.000, wenn es bei Wikipedia vorkommt. Für $l = 4$ ist $z(4) = 1.000.000$ und 2.000.000 für ein Segment, dass 5 Worte enthält. Segmente mit einer Länge von über 5 werden von unserem Verfahren nicht unterstützt. Wie bei der Gewichtung des Median-Verfahrens werden auch hier Segmente mit $l \leq 2$ nicht bevorzugt gewichtet, erhalten also kein Zusatzgewicht.

Das Median-Verfahren addiert einen festen Wert zum bisherigen Gewicht.

$$G = \sum g = \sum median(l) \cdot s + z \text{ für } s > T \text{ und } l > 1$$

Für $z = 100$ Millionen erzeugte das Median-Verfahren das beste Ergebnis. Alle Zusatzgewichte wurden experimentell ermittelt. In Abschnitt 4.3 kann die Ermittlung der Zusatzgewichte nachvollzogen werden.

3.4 Implementierungsdetails

In diesem Abschnitt werden Details der Implementierung der in den letzten Abschnitten vorgestellten Verfahren erläutert. Es wird dabei nicht auf die komplette Implementierung in JAVA eingegangen, sondern auf Besonderheiten und wichtige Code-Abschnitte. Für die Implementierung wurde das AItool-Framework[3] des Fachbereichs Web Technology & Information Systems der Bauhaus-Universität genutzt.

[3] http://www.uni-weimar.de/cms/index.php?id=10585 (21.07.2010)

Berechnung der möglichen Segmente

Die Klasse *QuerySegmentizer* errechnet die möglichen Segmentierungen einer Anfrage. Dazu werden nicht die Worte selbst, sondern der Zwischenraum der Worte genutzt. Eine Trennstelle ist die Position zwischen zwei Worten. Da wir eine Anfrage als Sequenz von Worten definiert haben, die durch Leerzeichen getrennt sind, ist jedes Leerzeichen eine Trennstellen. Trennstellen können 2 Zustände einnehmen: 0 und 1. Der Zustand 0 bedeutet, dass die Worte vor und nach der Trennstelle nicht getrennt werden, also ein Segment bilden. Im Gegensatz dazu bilden die Worte vor und nach einer Trennstelle des Zustandes 1 verschiedene Segmente.

Als erster Schritt wird die Anfrage zunächst an allen Trennstellen geteilt und in einem *Array* gespeichert. Zählt man die Einträge in diesem *Array*, erhält man die Anzahl der möglichen Trennungen der Anfrage. Bei n Worten in der Anfrage existieren $n-1$ Trennstellen. Dieser Wert wird an die Methode *createBinary* übergeben, die alle möglichen Trennungen einer Anfrage der Länge n berechnet und als *Array bp* (Bit Pattern) mit der Länge 2^{n-1} (Anzahl der möglichen Segmentierungen) zurück gibt. Jedes Element des *Arrays bp* hat die Länge $n-1$. Jedes dieser Zeichen repräsentiert eine Trennstelle und kann damit nur 2 Zustände einnehmen. Deshalb wird es auch Bit Pattern genannt.

Mit Hilfe des *Arrays bp*, dass gewissermaßen die Bildungsvorschrift für alle Möglichkeiten der Segmentierung der Anfrage enthält, wird nun die Anfrage in Phrasen zerlegt. Für jedes Element des *Arrays bp* wird die Methode *splitQuery* aufgerufen und ihr das aktuelle Element und die Anfrage übergeben. Die Anfrage wird hier ebenso bei jedem Leerzeichen zerteilt und als *Array qs* (query segments) gespeichert. Nun wird das Bit Pattern Zeichen für Zeichen geparst und gleichzeitig das *Array qs*, dass alle Worte der Anfrage in ursprünglicher Reihenfolge beinhaltet, mit durchlaufen. Bei einer 0 in *bp* wird das aktuelle Wort aus *qs* zum Zwischenspeicher hinzugefügt, bei einer 1 wird der Zwischenspeicher, der nun ein Segment enthält, zu einem *Objekt* vom Typ *pqs* (possible query segmentation) hinzugefügt. Diese Methode gibt also pro Bit Pattern ein *Objekt* vom Typ *pqs* mit den Segmenten für das jeweilige Pattern zurück. Das Ergebnis ist nun ein mehrdimensionales *Array* mit 2^{n-1} ungleichen Segmentierungen als Elemente. Diese Elemente sind *Objekte* vom Typ *pqs*, die die jeweiligen Segmente dieser Segmentierung enthalten.

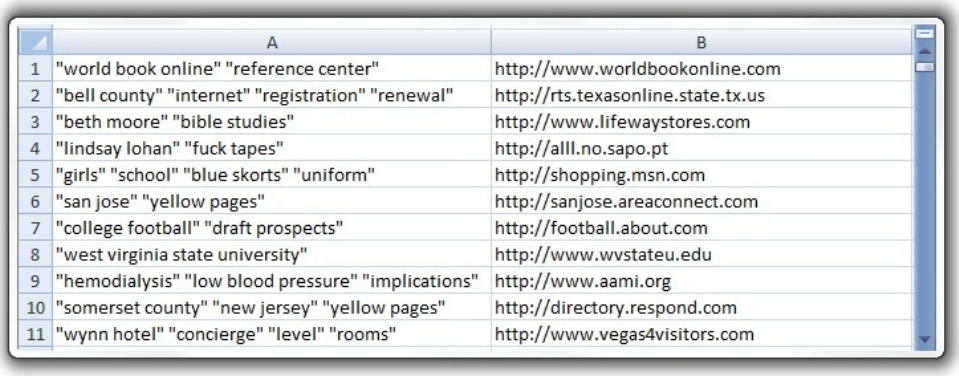
Kapitel 4

Experimente

Mit den Untersuchungen in diesem Kapitel wird die Genauigkeit der Segmentierung der in Kapitel 3 beschriebenen Verfahren untersucht. Die Vergleichbarkeit der Verfahren untereinander und mit anderen Verfahren wird durch die Verwendung des Goldstandards, über den bereits in Kapitel 2 berichtet wurde, gewährleistet. Angegeben wird die Genauigkeit eines Algorithmus. Diese gibt Auskunft wie exakt der Algorithmus die Anfragen im Vergleich zum Goldstandard segmentiert. Dabei wird zwischen fünf Messungen unterschieden: Die Segmentierungsgenauigkeit bezeichnet die Genauigkeit der Segmentierung der ganzen Anfrage. Eine Segmentierungsgenauigkeit Q_a von 0,8 bedeutet, dass 80% aller segmentierten Anfragen mit der Segmentierung des Goldstandards übereinstimmen. Die Trenngenauigkeit gibt an wie viele Trennungen B_a genauso gesetzt wurden, wie im Goldstandard. Die Genauigkeit der Segmente wird sogar durch drei Messungen verdeutlicht: Die Segmentgenauigkeit S_p gibt an, wie viele Segmente genauso segmentiert wurden wie im Goldstandard. Segment-recall S_r zeigt an, wie viele der in der Goldstandard-Datei segmentierten Segmente in der Vergleichsdatei vorkommen. Segment-F S_F wird aus Segmentgenauigkeit und Segment-recall folgendermaßen errechnet:
$S_F = (2 \cdot S_p \cdot S_r)/(S_p + S_r)$.

4.1 Ziele

Die Experimente sollen vor allem zeigen, wie gut unsere Algorithmen im Vergleich zu den in Kapitel 2 vorgestellten Verfahren sind. Damit unsere Algorithmen möglichst präzise arbeiten, ist es notwendig die Grenzwerte und die Gewichte der verschiedenen Grenzwert- und Gewichtungsverfahren zu justieren. Anhand der existierenden Scores wurden mittels theoretischer Vorüberlegungen Grenzwerte, Gewichtungen und Zusatzgewichte bestimmt. Mit Hilfe des Versuchsaufbaus erwies sich, dass diese Werte nicht immer die beste Ge-



	A	B
1	"world book online" "reference center"	http://www.worldbookonline.com
2	"bell county" "internet" "registration" "renewal"	http://rts.texasonline.state.tx.us
3	"beth moore" "bible studies"	http://www.lifewaystores.com
4	"lindsay lohan" "fuck tapes"	http://alll.no.sapo.pt
5	"girls" "school" "blue skorts" "uniform"	http://shopping.msn.com
6	"san jose" "yellow pages"	http://sanjose.areaconnect.com
7	"college football" "draft prospects"	http://football.about.com
8	"west virginia state university"	http://www.wvstateu.edu
9	"hemodialysis" "low blood pressure" "implications"	http://www.aami.org
10	"somerset county" "new jersey" "yellow pages"	http://directory.respond.com
11	"wynn hotel" "concierge" "level" "rooms"	http://www.vegas4visitors.com

Abbildung 4.1: Ausschnitt aus dem Goldstandard. Segmentierung durch Annotator A.

nauigkeit bei der Segmentierung lieferten. Darum ist es unumgänglich durch Experimente die möglichst besten Werte zu ermitteln. In der Betrachtung der Ergebnisse wird sich zeigen, dass durch die Optimierung der Grenzwerte und Gewichte eine Steigerung der Präzision der Segmentierung von einigen Prozent möglich ist.

4.2 Experimentbeschreibung

In diesem Abschnitt wird die Durchführung der Experimente beschrieben. Beginnend mit den Vorbereitungen der Datensätze zur Optimierung der Versuchsdurchführung im folgenden Abschnitt, über die Berechnung der Segmentierung in Abschnitt 4.2.2, bis hin zur Beschreibung der statistischen Auswertung der berechneten Segmentierung in Abschnitt 4.2.3.

4.2.1 Vorbereitung der Datensätze

Unsere Untersuchungen werden auf dem von Bergsma und Wang [BW07] entwickelten Goldstandard durchgeführt. Dieser beinhaltet drei Dateien, je eine pro Annotator. Jede Datei beinhaltet die gleichen 500 Anfragen, die allerdings teilweise verschieden segmentiert wurden. Die 220 identisch segmentierten Anfragen finden sich in einer separaten, vierten Datei der Goldstandard Intersection wieder. Hinter jeder Anfrage ist der von dem Annotator geklickte, zur Anfrage passende Hyperlink protokolliert (siehe Abbildung 4.1).

Nach dem Entfernen der Anführungszeichen und URLs stehen die unsegmentierten Anfragen zur Verfügung, die nun durch unser Verfahren so segmen-

tiert werden sollen, dass das Ergebnis möglichst identisch zu dem der Annotatoren von Bergsma und Wang [BW07] ist.

Dadurch, dass wir viele Versuche mit den gleichen Datensatz durchführen wollen, ist es sinnvoll allgemeine Operationen auf diesen vorab zu tätigen und die Zwischenergebnisse zu speichern, sodass Rechenzeit während der Experimente eingespart wird. Diese Operationen sind die Kalkulation von möglichen Segmenten und die Bestimmung der Häufigkeiten dieser.

Tabelle 3.2 zeigt die möglichen Segmentierungen für die Beispielanfrage `san jose yellow pages`. Für diese existieren 8 mögliche Segmentierungen und 10 mögliche Segmente.

Für die 500 Goldstandard Anfragen mit durchschnittlich 4,3 Worten pro Anfrage sind insgesamt 3478 verschiedene Segmente möglich. Von all diesen wird der Score mit Hilfe von Netspeak[1] [PTS10] ermittelt. Die 3478 Segmente werden nun zusammen mit ihren Häufigkeiten in einer Textdatei gespeichert. Das hat den Vorteil, dass beim Ermitteln der Häufigkeiten eines Segments keine Netspeak-Anfrage gestellt werden muss, sondern nur in der Datei gesucht wird. Das verbessert die Geschwindigkeit der Segmentierung deutlich. Eine Netspeak-Anfrage benötigt circa 120ms. Für alle nötigen 3478 Anfragen werden allein für die Abfrage der Scores fast 7 Minuten benötigt. Im Unterschied dazu dauert das Auslesen und Parsen einer Textdatei mit 85kB auf aktuellen PCs weniger als 1 Sekunde.

Das gleiche Verfahren wird außerdem für die Überprüfung der Inhalte bei Wikipedia vorgenommen. Wie in Abschnitt 3.3 beschrieben, werden Segmente direkt bei Wikipedia angefragt. Wir erhalten dadurch eine Auflistung für alle möglichen Segmente der Goldstandard-Anfragen. Diese Auflistung enthält die Segmente, gefolgt von einem Booleschen Wert, der angibt ob die jeweilige Phrase bei Wikipedia gefunden wurde. Diese Liste speichern wir wiederum in einer Textdatei, um diese Werte, wie auch die Scores, direkt auslesen zu können, ohne ein weiteres mal eine Suchmaschine anfragen zu müssen.

4.2.2 Berechnung der Anfragen

Die Klasse *ExperimentsSingleMethods* verwaltet die Berechnung der Anfragen über alle implementierten Segmentierungsverfahren. Wie bereits im letzten Abschnitt erwähnt, sind alle Anfragen des Goldstandards in einer Datei gespeichert. Diese Datei wird Zeile für Zeile geparst. Jede Zeile beinhaltet genau eine Anfrage. *ExperimentsSingleMethods* leitet jede Anfrage an jedes Segmentierungsverfahren weiter. Dazu erstellt es jeweils ein neues Objekt der jeweiligen Klasse und übergibt dem Konstruktor die Anfrage. Durch Aufrufen der

[1] http://www.netspeak.cc (22.07.2010)

Gewichtung	Segmentierungsgenauigkeit
$l^{1/2 \cdot l}$	$0,6136$
l^{l}	$0,6181$
$l^{2 \cdot l}$	$0,5681$
$l^{l^{l}}$	$0,4181$

Tabelle 4.1: Test verschiedener Exponenten des LenPowLen-Verfahrens

Methode *getSegmentation* wird die beste Segmentierung für das jeweilige Verfahren zurückgegeben und von *ExperimentsSingleMethods* in einer Textdatei pro Segmentierungsverfahren gespeichert.

4.2.3 Auswertung der Daten

Für jedes Segmentierungsverfahren existiert eine Textdatei, die alle Anfragen aus dem Goldstandard in segmentierter Form enthält. Das hat nun den Vorteil, dass mit Hilfe eines einfachen Shell-Scripts die erstellten Dateien mit den Dateien des Goldstandard verglichen werden können. Dabei vergleicht das Script die Dateien zeilenweise, also Anfrage für Anfrage. Protokolliert werden die Anzahl der identisch und verschieden segmentierten Phrasen und Anfragen, sowie die Anzahl der korrekten und inkorrekten Trennstellen. Aus diesen Daten errechnet das Script die statistischen Werte, auf die im Abschnitt 4.3 eingegangen wird.

4.3 Ergebnisse

4.3.1 Optimierung des LenPowLen-Verfahrens

Um die bestmögliche von der Segmentlänge abhängige Gewichtung der Segmente zu finden, wurde durch Veränderung des Exponenten die Segmentierungsgenauigkeit bestimmt. In Tabelle 4.1 ist ersichtlich, dass das Vergrößern oder Verkleinern des Exponenten eine negative Auswirkung auf die Segmentierungsgenauigkeit hat. Daraus folgt, dass der in Abschnitt 3.2.3 beschriebene Algorithmus mit der Gewichtung $l^l \cdot s$ bereits die beste Gewichtung von den untersuchten Variationen bietet.

4.3.2 Optimierung des Median-Verfahrens

In Abschnitt 3.2.4 ist das Median-Verfahren erläutert, dass in Abschnitt 3.3 um die Überprüfung der Segmente bei Wikipedia erweitert wurde. Ebenfalls

Zusatzgewicht	Segmentierungsgenauigkeit
$1 \cdot 10^0$ *bis* $1 \cdot 10^5$	$0,6500$
$1 \cdot 10^6$	$0,6454$
$1 \cdot 10^7$	$0,6727$
$1 \cdot 10^8$ *bis* $1 \cdot 10^{11}$	$0,6818$

Tabelle 4.2: Test von Zusatzgewichten mit dem Median-Verfahren

ist dort beschrieben worden, dass die Zusatzgewichtung der Segmente, die bei Wikipedia gefunden werden, bei dem Median-Verfahren einheitlich ist. Dieses Zusatzgewicht muss experimentell ermittelt werden. In Tabelle 4.2 sind Zusatzgewichte z zwischen 1 und 100 Millionen und die Segmentierungsgenauigkeit des Algorithmus mit dem jeweiligen z aufgelistet. Das Zusatzgewicht wird beim Median-Verfahren zum Produkt aus Score und Median-Faktor addiert ($G = \sum median(l) \cdot s + z$). Es ist ersichtlich, dass durch die Optimierung des Wikipedia-Zusatzgewichtes eine Steigerung von bis zu $3,64\%$ erreicht wurde. Erstaunlicherweise werden die Anfragen mit Zusatzgewichten zwischen 1 und 100.000 und zwischen 100 Millionen und 100 Milliarden absolut identisch segmentiert. Zusatzgewichte unter 1 Million sind zu niedrig für die Höhe der Gewichte der Segmente, sodass sie keine Auswirkung auf die Segmentierung haben. Zusatzgewichte über 100 Millionen sind wiederum so hoch, sodass sie auf jeden Fall die größte Gewichtung produzieren. Der Abstand zu nicht bei Wikipedia gefundenen Segmenten erhöht sich zwar mit zunehmenden Zusatzgewicht, auf die Genauigkeit hat dies aber keine Auswirkung. Aus Tabelle 4.2 ist also abzulesen, dass ein Zusatzgewicht von 100 Millionen mit einer Segmentierungsgenauigkeit von $0,6818$ die beste Segmentierung liefert. Die Trenngenauigkeiten dieser Segmentierung ist mit $0,8437$ identisch zu der Segmentierung mit dem Zusatzgewicht 1.

Eine längenabhängige Zusatzgewichtung, wie für das LenPowLen-Verfahren genutzt, ergab eine Segmentierungsgenauigkeit von $0,6500$ und eine Trenngenauigkeit von $0,8465$.

4.3.3 Vergleich der Verfahren

In Tabelle 4.3 sind die Ergebnisse aller implementierten Verfahren, sowie drei bekannter Verfahren dargestellt. Die Tabelle ist folgendermaßen aufgebaut: In Spalte 1 ist der Annotator ersichtlich. Drei Annotatoren haben den Goldstandard segmentiert, diese sind mit A, B und C bezeichnet. Die 220 Anfragen, die von allen Annotatoren gleich segmentiert wurden und damit die beste Basis für Vergleiche bilden, sind in der Goldstandard Intersection gespeichert,

		[BW07]	[TP08]	[ZSH+09]	$F2B$	$B2F$	S	M	M_W	LPL	LPL_W
	Q_a	0,638	0,526		0,372	0,326	0,522	0,482	0,548	0,536	0,604
	B_a	0,863	0,810		0,735	0,707	0,799	0,776	0,776	0,807	0,831
A	S_p		0,657	0,652	0,565	0,523	0,652	0,630	0,665	0,665	0,723
	S_r		0,657	0,699	0,519	0,474	0,705	0,649	0,632	0,708	0,705
	S_F		0,657	0,675	0,541	0,497	0,678	0,639	0,648	0,686	0,713
	Q_a		0,494		0,398	0,356	0,370	0,372	0,430	0,380	0,450
	B_a		0,802		0,739	0,719	0,744	0,742	0,750	0,751	0,778
B	S_p		0,623	0,632	0,520	0,489	0,511	0,510	0,532	0,519	0,563
	S_r		0,640	0,659	0,541	0,502	0,626	0,595	0,574	0,626	0,623
	S_F		0,631	0,645	0,530	0,495	0,563	0,550	0,552	0,568	0,591
	Q_a		0,494		0,400	0,366	0,430	0,422	0,492	0,454	0,498
	B_a		0,796		0,744	0,722	0,761	0,756	0,767	0,771	0,785
C	S_p		0,634	0,614	0,550	0,521	0,564	0,564	0,599	0,581	0,615
	S_r		0,642	0,649	0,533	0,499	0,645	0,614	0,602	0,653	0,634
	S_F		0,638	0,631	0,541	0,509	0,602	0,588	0,601	0,615	0,624
	Q_a	0,717	0,671		0,500	0,431	0,600	0,595	0,681	0,627	0,718
	B_a	0,892	0,871		0,788	0,755	0,839	0,828	0,842	0,850	0,880
I	S_p		0,767	0,772	0,646	0,592	0,696	0,694	0,751	0,718	0,788
	S_r		0,782	0,826	0,612	0,557	0,768	0,738	0,744	0,778	0,797
	S_F		0,774	0,798	0,628	0,574	0,730	0,715	0,748	0,746	0,792

Tabelle 4.3: Vergleich bekannter und unserer Verfahren

hier mit I gekennzeichnet. Außerdem werden zwischen Segmentierungsgenauigkeit Q_a (query accuracy), Trenngenauigkeit B_a (break accuracy), Segmentgenauigkeit S_p, Segment-recall S_r und Segment-F S_F für die Genauigkeit der Ergebnisse unterschieden. Das bedeutet, dass pro Verfahren und Annotator (beziehungsweise Goldstandard Intersection) fünf Werte aufgelistet sind.

Folgende Verfahren sind in dieser Tabelle zu finden: Die besten Ergebnisse der in Kapitel 2 beschriebenen Verfahren von Bergsma und Wang [BW07], Tan und Peng [TP08] und Zhang et al. [ZSH+09]. Desweiteren sind Front-to-Back ($F2B$), Back-to-Front ($B2F$), Summe (S), Median (M), Median mit Wikipedia-Optimierung (M_W), LenPowLen (LPL) und LenPowLen mit Wikipedia-Optimierung (LPL_W) dargestellt.

Beim Betrachten der Werte ist auffällig, dass die Trenngenauigkeit deutlich höher sind, als die Segmentierungsgenauigkeit. Das liegt daran, dass bei den Annotatoren jeweils 500 beziehungsweise 220 Anfragen existieren, pro Anfrage aber oft mehrere Trennungen. Die Zahl der Trennungen ist also deutlich

höher als die der Anfragen. Das Beispiel `san jose yellow pages` aus dem vorherigen Kapitel weist 3 Trennstellen auf. Ist die Anfrage zum Beispiel mit `"san jose" "yellow" "pages"` segmentiert, so ist die Segmentierungsgenauigkeit Q_a für diese Anfrage $0,000$, da der Goldstandard `"san jose" "yellow pages"` vorsieht. Der Wert der Trenngenauigkeit B_a ist allerdings $0,666$, da an den ersten beiden der drei Trennstellen der richtige Status gewählt wurde: 0 und 1 (nicht trennen und trennen). Der Status der dritten Trennstelle unterscheidet sich in beiden Segmentierungen. Eine nicht übereinstimmende Trennung beeinflusst also das Ergebnis der Trenngenauigkeit nicht so negativ wie den Werte der Segmentierungsgenauigkeit.

Desweiteren fällt auf, dass die Genauigkeiten unserer Verfahren für den Vergleich der Segmentierung des Annotators B oft niedriger ausfallen, als bei den anderen Annotatoren. Das ist der unterschiedlichen Segmentierung der Annotatoren geschuldet. Während Annotator A und C ähnlich segmentierten, wählte Annotator B oft andere Segmentierungsmöglichkeiten. Beispielsweise segmentierten A und C die Anfrage `johnson county community college` mit `"johnson county" "community college"`, hingegen wählte B die Segmentierung `"johnson county community college"`. Darum beziehen wir uns ausschließlich auf die Goldstandard Intersection zum Vergleich der Werte, alle anderen Daten sind zur Vollständigkeit angegeben.

Erwartungsgemäß sind die „einfachen“ Grenzwertverfahren im Vergleich zu den komplexeren Verfahren deutlich unpräziser. Front-to-Back segmentiert exakt die Hälfte der Anfragen so wie im Goldstandard. Der Back-to-Front-Algorithmus ist sogar fast 7% ungenauer. Die Ursache dafür ist genau die Überlegung, die den Back-to-Front-Algorithmus gegenüber dem Front-to-Back-Algorithmus verbessern sollte. Wie in Abschnitt 3.1.2 beschrieben, sollten dadurch überlange Segmente vermieden werden.

Das Summe-Verfahren bildet nicht nur in Hinblick auf die Arbeitsweise und Komplexität eine Brücke zwischen den Grenzwertverfahren und den umfassenderen Gewichtungsverfahren LenPowLen und Median, auch die statistischen Ergebnisse liegen zwischen diesen Verfahren. Mit einer Segmentierungsgenauigkeit von $0,600$ liegt es genau zwischen Front-to-Back und LenPowLen mit Wikipedia-Optimierung. Auffällig ist, dass es damit sogar eine höhere Segmentierungsgenauigkeit besitzt als das Median-Verfahren (ohne Wikipedia-Optimierung).

Für das LenPowLen-Verfahren sind die original Werte aus der Veröffentlichung von Hagen et al. [HPSB10] angegeben. Bei den Untersuchungen für diese Arbeit stellten wir jedoch fest, dass vier Segmente, die über die richti-

ge Segmentierung von vier Anfragen entscheiden könnten nicht von Netspeak gefunden wurden, obwohl sie in der Google-N-Gramm-Kollektion einen Score besitzen. Wir entschlossen uns diese Scores zu ergänzen. Das Resultat ist ein Verbesserung der Genauigkeit des LenPowLen-Verfahrens von Q_a auf $0,645$ und B_a auf $0,55$ beim Test auf der Goldstandard Intersection. Durch das Hinzufügen der Wikipedia-Gewichtung konnten wir eine Steigerung der Segmentierungsgenauigkeit um 9,1% erreichen. Damit segmentiert das LenPowLen-Verfahren (LPL_W) 71,8% aller Anfragen so, wie sie im Goldstandard segmentiert wurden. Das bedeutet, dass unser Verfahren gleich gut arbeitet, wie das Segmentierverfahren von Bergsma und Wang [BW07], obwohl jene nicht nur den Goldstandard veröffentlichten, sondern auch einen höheren Aufwand zur Segmentierung betreiben (siehe Kapitel 2). Der Unterschied von $0,1\%$ lässt sich durch Rundungsfehler bei der Rechnung erklären. Von den 220 Anfragen der Intersection sind vermutlich bei beiden Verfahren 158 richtig segmentiert wurden. Diese entsprechen einen Segmentierungsgenauigkeit von $0,71\overline{81}$. Die in der Veröffentlichung von Bergsma und Wang angegebenen $0,717$ sind also prinzipiell nicht möglich, da dann $157,74$ Anfragen richtig segmentiert wären. Nichtsdestotrotz liegen diese beiden Verfahren bei der Segmentierungsgenauigkeit etwa gleich auf. Die Trenngenauigkeit ist bei Bergsma und Wang 1,2% höher als bei unserem Verfahren, allerdings ist das wohl für den Nutzer nicht entscheidend.

Zhang et al. [ZSH+09] geben keine Werte für Q_a und B_a an. Auf der Goldstandard Intersection ist die Segmentgenauigkeit S_p bei LPL_W etwa 1% höher, bei den Datensätzen A und B sogar biszu 7% höher als bei [ZSH+09]. Im Gegensatz dazu ist der S_r Wert bei Zhang et al. etwa 3% höher, als bei LPL_W. Die Kombination beider Werte spiegelt die Segment-F Messung wieder. Hier liegen beide Verfahren mit etwa 79% gleich auf. Zu beachten ist, dass das Verfahren von Zhang et al. durch die Verwendung des Vektorraummodells deutlich komplexer ist, als unser Verfahren, dass auf einfachen statistischen Daten basiert.

Das Median-Verfahren sollte zwar durch eine gerechtere, längenabhängige Gewichtung genauer Segmentieren als das LenPowLen-Verfahren, doch wie in Tabelle 4.3 zu sehen ist, ist es 3,7% ungenauer als LenPowLen. Mit einer Segmentierungsgenauigkeit von $0,681$ ist es aber genau 1% (rund 2 Anfragen) genauer als das Verfahren von Tan und Peng [TP08]. Tan und Peng nutzen in ihrem Verfahren ebenfalls Wikipedia. Da ihr Gewichtungsverfahren andere statistische Daten nutzt als unseres, können die Zusatzgewichte nicht direkt verglichen werden. Es ist zu vermuten, dass der komplexere EM-Algorithmus von Tan und Peng mehr Rechenzeit benötigt, als unser Median-Verfahren. Genaue Angaben machen Tan und Peng allerding nicht.

4.4 Fehleranalyse

Beim direkten Vergleich unserer Segmentierungen mit denen der Goldstandard Intersection werden die Grenzen unserer Vorgehensweise deutlich. Die Nutzung der Google-N-Gramm-Kollektion erwies sich als eingeschränkt, sodass wir Wikipedia hinzu zogen, um Namen besser zu erkennen. Bei unserem besten Verfahren (LenPowLen mit Wikipedia) werden 62 der 220 Goldstandard Intersection Anfragen nicht so segmentiert, wie es der Goldstandard vorschlägt.

In 28 der 62 Anfragen ist die Differenz der Häufigkeiten von Segmenten so hoch, dass sie von keinem unsere Gewichtungsverfahren ausgeglichen werden. Typische Beispiele sind Städtenamen und oft genutzte Begriffe. So wurde zum Beispiel die Anfrage `animals redwood national park` in der Intersection mit `"animals" "redwood national park"` segmentiert. Da die Häufigkeit von `"national park"` aber fast 500-mal so hoch ist wie die von `"redwood national park"`, segmentierten alle unsere Gewichtungsverfahren diese Anfrage mit `"animals" "redwood" "national park"`.

Von den 62 verschieden segmentierten Anfragen sind 23 Anfragen mit größeren Segmenten segmentiert, als es der Goldstandard vorsieht. Dieses Verhalten ist zwar auf die gleiche Ursache, wie die zuerst erwähnte Fehlerquelle zurück zu führen, hat aber einen anderen Effekt. Während bei der bereits beschriebenen Fehlerquelle Begriffe so oft vorkommen, dass sie ein höheres Gewicht besitzen als längere Segmente, wird hier das längere Segment bevorzugt, also zu hoch gewichtet. Das geschieht zum Beispiel bei der Anfrage `google earth free play`. Während der Goldstandard eine einzelne Segmentierung von `free` und `play` vorsieht, segmentieren unsere Gewichtungsverfahren `"free play"` zusammen. Wie in Kapitel 3 beschrieben, bevorzugen wir immer gültige Segmente mit einer Länge $l \geq 2$ gegenüber einzeln segmentierten Worten.

Neben den zu großen Häufigkeiten bei einigen Segmenten und den zu hoch gewichteten Segmenten, existiert noch eine weitere Fehlerquelle. Einige Segmente, die der Goldstandard vorschlägt, werden weder von Netspeak noch bei Wikipedia gefunden. Zum Beispiel `"queensboro community college"`. Das betrifft 13 der 62 Anfragen. Um dieses Problem zu lösen, müssten die vorhanden statistischen Daten überprüft und vervollständigt werden oder eine weitere statistische Quelle hinzugezogen werden.

Da die Goldstandard-Anfragen aus dem AOL-Anfragen-Log stammen und somit von Nutzern eingegeben wurden, existieren auch Tippfehler. Weil unsere Verfahren keine Algorithmen zum Korrigieren von Rechtschreibfehlern beinhalten, ist es uns nicht möglich Anfragen wie `dallas texas alterntaive school` oder `drexel uiversity law school` korrekt zu segmentieren. Insgesamt beinhaltet der Goldstandard 9 eindeutige Rechtschreibfehler. Davon kommen 5 auch in der Goldstandard Intersection vor, von denen wiederum 3 durch

LenPowLen fehlerhaft segmentiert werden. Wenn ein Wort in der Google-N-Gramm-Kollektion nicht gefunden wird, wird es einzeln segmentieren. Da die falsch geschriebenen Worte auch im Goldstandard teilweise einzeln segmentiert wurden, segmentieren wir somit auch 2 der fehlerhaften Anfragen korrekt.

Kapitel 5

Resümee und Ausblick

Wir konnten zeigen, dass unsere einfachen Gewichtungsverfahren zur Segmentierung von Suchanfragen mit komplexeren Verfahren, in Hinblick auf die Genauigkeit der Segmentierung, gleich auf sind. Über 70% der eingegebenen Anfragen wurden von unserem Verfahren LenPowLen so segmentiert, wie sie auch ein Mensch segmentiert hätte. Obwohl die neuartige Gewichtung des Median-Verfahrens keine Verbesserung gegenüber dem LenPowLen-Verfahren darstellt, konnte mit Hilfe von Wikipedia eine Verbesserung des LenPowLen-Verfahrens von über 9% erreicht werden.

Zukünftige Untersuchungen sollten die Korrektur der in Abschnitt 4.4 beschriebenen Fehler analysieren. Der Ausgleich extrem häufiger Begriffe könnte zum Beispiel durch ein Verfahren geschehen, dass die Gewichte dieser Begriffe ignoriert, sobald sie in einem anderen, längeren und gültigen Segment vorkommen. Desweiteren soll die Abfrage der Wikipedia-Daten auch lokal implementiert werden, sodass die Suche von Segmenten in den Wikipedia-Daten effizienter gehandhabt werden kann. Dabei sollte auch untersucht werden, inwiefern eine Unterscheidung, ob Segmente im Titel, Ankertext oder Artikeltext eines Wikipedia-Artikels gefunden werden, eine Verbesserung der Genauigkeit der Verfahren erlaubt.

In der Einleitung wurde als Ziel das automatische Segmentieren von Suchanfragen vor der Suche mit einer Suchmaschine genannt. Für zukünftige Arbeiten bleibt genau die Implementierung der vorgestellten Verfahren für die Eingabe einer Anfrage an eine Suchmaschine. Bisher wurde untersucht, wie genau Anfragen im Vergleich zur Segmentierung durch einen Menschen, segmentiert werden. Es muss noch ermittelt werden, welches Verfahren beziehungsweise welche Segmentierung den größten Vorteil für den Nutzer bietet.

Literaturverzeichnis

[BCS09] Michael Bendersky, W. Bruce Croft und David A. Smith. Two-stage query segmentation for information retrieval. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Boston, MA, USA, July 19-23, 2009*, Seiten 810–811. ACM, 2009.

[BF06] Thorsten Brants und Alex Franz. Web 1t 5-gram version 1, September 2006.

[BW07] Shane Bergsma und Qin Iris Wang. Learning noun phrase query segmentation. In *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*, Seiten 819–826. ACL, 2007.

[GXLC08] Jiafeng Guo, Gu Xu, Hang Li und Xueqi Cheng. A unified and discriminative model for query refinement. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, Singapore, July 20-24, 2008*, Seiten 379–386. ACM, 2008.

[HPSB10] Matthias Hagen, Martin Potthast, Benno Stein und Christof Bräutigam. The Power of Naïve Query Segmentation. In *33rd Annual International ACM SIGIR Conference (SIGIR 10) (to appear)*. ACM, Juli 2010.

[JRMG06] Rosie Jones, Benjamin Rey, Omid Madani und Wiley Greiner. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006*, Seiten 387–396. ACM, 2006.

[KGA+10] Julia Kiseleva, Qi Guo, Eugene Agichtein, Daniel Billsus und Wei Chai. Unsupervised query segmentation using click data: preliminary results. In *Proceedings of the 19th International Conference*

on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010, Seiten 1131–1132. ACM, 2010.

[PTS10] Martin Potthast, Martin Trenkmann und Benno Stein. Netspeak: Assisting writers in choosing words. In *Advances in Information Retrieval: Proceedings of the 32nd European Conference on Information Retrieval*, Seite 672. ECIR 2010, 2010.

[RMB03] Knut Magne Risvik, Tomasz Mikolajewski und Peter Boros. Query segmentation for web search. In *WWW (Posters)*, 2003.

[TP08] Bin Tan und Fuchun Peng. Unsupervised query segmentation using generative language models and Wikipedia. In *Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008*, Seiten 347–356. ACM, 2008.

[YS09] Xiaohui Yu und Huxia Shi. Query segmentation using conditional random fields. In *Proceedings of the First International Workshop on Keyword Search on Structured Data, KEYS 2009, Providence, Rhode Island, USA, June 28, 2009*, Seiten 21–26. ACM, 2009.

[ZSH+09] Chao Zhang, Nan Sun, Xia Hu, Tingzhu Huang und Tat-Seng Chua. Query segmentation based on eigenspace similarity, 2009.