Martin-Luther-Universität Halle-Wittenberg
Naturwissenschaftliche Fakultät III
Studiengang Informatik

Keyqueries zur Erweiterung von Suchanfragen mit explizitem Relevanz-Feedback

Bachelorarbeit

Johannes Huck
geb. am: 27.08.1996 in Saalfeld/Saale

Matrikelnummer 215226002

1. Gutachter: Prof. Dr. Matthias Hagen
2. Gutachter: M. Sc. Maik Fröbe

Datum der Abgabe: 3. Mai 2021

Zusammenfassung

Kurze Suchanfragen haben das Problem, dass sie das eigentliche Informationsbedürfnis des Nutzers nur ungenügend charakterisieren. Aus diesem Grund werden schon seit den 1960er-Jahren Methoden erprobt, die solch kurze Suchanfragen mit geeigneten Termen erweitern können. Eine spezielle Form greift dabei auf die Unterstützung durch explizites Relevanz-Feedback zurück. Bereits bestehende Methoden zur Suchanfrageerweiterung mit expliziten Relevanz-Feedback achten beim Erzeugen der erweiterten Suchanfrage nicht auf das durch diese neue Anfrage generierte Ranking und sie garantieren nicht, dass die erzeugte Anfrage minimal ist. Diese Probleme werden durch spezielle Suchanfragen – den sogenannten Keyqueries – behoben. Die Keyqueries achten auf die Position der Dokumente aus dem Relevanz-Feedback und sie achten darauf, dass die erzeugten Anfragen minimal sind. In dieser Arbeit wird untersucht, wie die durch Hagen et al. [9] eingeführten Keyqueries in den Aspekten der Generierung und Auswahl von Anfragekandidaten und der Bildung einer Suchanfrage aus diesen Kandidaten verbessert werden können. Die Evaluation auf dem Robust04-Corpus zeigt, dass eine relative Verbesserung im Bezug auf das Evaluationsmaß NDCG von bis zu 36% gegenüber dem Rankingmodell BM25 erreicht werden kann.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Suchmaschinen verwenden Suchanfrageerweiterungen, da bei Information Retrieval basierend auf Schlüsselwörtern, die Herausforderung besteht, dass Anfragen zu wenige Terme enthalten können und somit der Kontext der Anfrage unzureichend charakterisiert sein kann [2]. Eine Möglichkeit die Performance von Suchanfrageerweiterungen zu verbessern, ist die Verwendung von explizitem oder implizitem Relevanz-Feedback, dieses zusätzliche Wissen führt zu kompetitiven Baselines, die selbst von aufwändig trainierten Deep-Learning-Modellen kaum übertroffen werden, unter der Voraussetzung, dass es keinen Satz guter Trainingsdaten gibt [12]. Herkömmliche Suchanfrageerweiterungen beachten jedoch nicht die durch sie erzeugten Rankings, insbesondere bei Einsatz von explizitem Relevanz-Feedback ist das Nichtbeachten der Rankings eine nicht optimale Verfahrensweise. Denn so werden die Anfragen nicht auf das zugrundeliegende Retrieval-Modell optimiert. In dieser Arbeit werden sogenannte Keyqueries vorgestellt, welche von Hagen et al. erdacht wurden. Keyqueries sind eine neuartige Form von Suchanfragen auf Basis von explizitem Relevanz-Feedback und Suchanfrageerweiterung, bei der die erzeugten Rankings beachtet werden und welche unabhängig vom gewählten Retrieval-Modell funktioniert.

Für die Erweiterung von Suchanfragen mit explizitem Relevanz-Feedback sieht der Ablauf des Algorithmus wie folgt aus: Nach der Eingabe einer Suchanfrage gibt der Nutzer eine Rückmeldung an die Suchmaschine, welche der ihm gerade gezeigten Dokumente er für relevant hält, im Folgenden werden diese Dokumente als Ziel-Dokumente bezeichnet. Auf Grundlage dieses expliziten Relevanz-Feedbacks werden eine Reihe von Keyqueries gebildet, aus denen die Keyqueries ausgewählt werden, die die Ziel-Dokumente an die höchste Position im Ranking bringen. Ebenso werden Methoden des Natural Language Processings eingesetzt, um über Nounphrasen weitere Suchanfragen zu generieren, die im Verbund mit den zuvor ermittelten Keyqueries als eine Suchanfrage

an das Retrieval-Modell gestellt werden. Ziel dieser Arbeit ist es, das soeben beschriebene Verfahren systematisch auf dem Robust04-Corpus und den zugehörigen Fragestellungen zu evaluieren und in den Aspekten der Generierung und Auswahl der möglichen Anfrage-Kandidaten und der Bildung einer Suchanfrage aus diesen Kandidaten zu verbessern. Keyquery-Kandidaten sind dabei alle Suchanfragen, die bei der Keyquery-Berechnung entstehen und die die Keyquery-Eigenschaft erfüllen (eine genaue Definition dieser Eigenschaft findet sich in Kapitel 2.3). Des Weiteren wird betrachtet in welcher Art und Weise sich ein Keyquery-basierter Ansatz realisieren lässt, dem nur implizites Relevanz-Feedback zur Verfügung steht. Zusätzlich wird untersucht, wie sich eine Kombination verschiedener Modelle von Suchanfrageerweiterungen als zusätzliches Relevanz-Feedback auf die Performance eines Keyquery-basierten Ansatzes auswirkt.

Für die Evaluierung werden als Baseline die Werte des NDCG und Recalls für das Rankingmodell BM25 und die Suchanfrageerweiterungsmodelle RM3, BM25PRF und Axiomatic auf dem Robust04-Corpus verwendet. Die Evaluation zeigt, dass durch geeignete Auswahl an Methoden zur Generierung und Auswahl der Suchanfragen eine relative Verbesserung um bis zu 36 % im Vergleich zu BM25 im Bezug auf den NDCG auf dem Robust04-Corpus erreicht wird.

Kapitel 2

Related Work

Die vorliegende Arbeit baut auf drei Gruppen von Verfahren auf, die eine initiale Suchanfrage mit verwandten Termen aus endogenen (aus der Dokumentensammlung) sowie exogenen (Ontologien, Thesauri) Wissen erweitern [6]. Zum einen die Erweiterung von Suchanfragen, das Verwenden von Relevanz-Feedback und zum anderen die Berechnung von Keyqueries. Im folgenden Kapitel wird ein Überblick über die Funktionsweise dieser Verfahren gegeben und verwandte Arbeiten auf diesen drei Gebieten werden vorgestellt.

2.1 Erweiterung von Suchanfragen

Der Begriff Suchanfrageerweiterungen beschreibt eine Gruppe von Verfahren, die eine initiale Suchanfrage um Terme erweitern und somit die Uneindeutigkeit der ursprünglichen Anfrage reduzieren wollen. Somit wird das eigentliche Informationsbedürfnis einer Anfrage besser charakterisiert [6]. Einsatz finden diese Verfahren bei den meisten Suchmaschinen, denn Suchanfragen bestehen oftmals nur aus 2 bis 3 Termen [10] und charakterisieren somit eine Suchanfrage nur ungenügend. Erste Verfahren nutzten manuell erstellte Thesauri, so dass Anfragen mit Synonymen erweitert werden konnten. Beispielhaft sei hier Medical Subject Headings (MeSH) erwähnt, welches in der Suchanwendung der National Library of Medicine verwendet wird. Später wurden diese manuell erstellten Thesauri durch automatisch generierte Thesauri ersetzt, die auf Basis von statistischen Analysen der Dokumente der zugrundeliegenden Dokumentensammlung basieren [7]. Eine weitere Möglichkeit solche automatischen Thesauri zu generieren besteht darin, die grammatikalischen Abhängigkeiten und Beziehungen der Texte der Dokumentensammlung zu analysieren [13].

Diese automatisch generierten Thesauri beachten jedoch nicht den Kontext der Anfrage und führen somit zu ungewollten Topic Drifts. Denn die Uneindeutigkeit von Termen führt sehr leicht zu für den Kontext nicht relevanten,

aber statistisch korrelierenden Termen [13]. Betrachten wir folgendes Beispiel von Croft et al. [7]: Gegeben ist die Suchanfrage `tropical fish tanks` , eine gute Wahl für einen Erweiterungsterm für `tank` ist `aquarium` . Während für die Anfrage `armor for tanks` der Term `aquarium` kein geeigneter Erweiterungsterm ist. Automatisch generierte Thesauri stellt diese geeignete Wahl von Erweiterungstermen vor ein großes Problem. Denn sie haben eine Liste von vielen verwandten Termen aus vielen verschiedenen Kontexten, aber wissen nicht, welcher spezifischer Kontext vorliegt. Dies erschwert die automatische Auswahl von Erweiterungstermen. Eine Möglichkeit diese Problematik zu umgehen, ist die Verwendung von allen Wörtern aus der Anfrage, um verwandte Wörter zu finden, anstelle dass jedes Wort der Anfrage einzeln expandiert wird. Eine weitere Möglichkeit besteht in der Verwendung von Relevanz-Feedback, dies wird in 2.2 diskutiert [7].

Die bisher dargelegten Ansätze zur Suchanfrageerweiterung sind daraus motiviert, dass sie die Uneindeutigkeit von natürlicher Sprache durch das Erweitern der originalen Suchanfrage beheben wollen. Dabei basieren diese Verfahren jedoch zumeist auf Basis von Vektoren von gewichteten Wörtern. Diese Verfahren sind effektiv in Bezug auf Genauigkeit und zeitlichem Aufwand. Jedoch ist die Termauswahl zur Erweiterung vergleichsweise simpel gehalten, nämlich als eine Liste von gewichteten Wörtern. Hierbei gehen wertvolle Informationen über die Positionen der Wörter im Text und die Struktur des Textes verloren. Aus diesem Grund wurden komplexere Methoden entworfen, die eine strukturierte Menge an Wörtern zur Erweiterung auswählen können [6].

In ihrer Arbeit stellen Colace et al. [6] eine neue Suchanfrageerweiterungsmethode vor, die eine eben solche strukturierte Repräsentation von Dokumenten und Anfragen verwendet, sie wird Weighted Word Pairs genannt. Diese neue Methode hat den Zweck, den Einfluss der Uneindeutigkeit natürlicher Sprache zu reduzieren und somit bessere Ergebnisse zu erzielen als eine Methode auf Grundlage eins Vektors von gewichteten Wörtern. Die Methode extrahiert aus einer Menge von Dokumenten eine kompakte Repräsentation der am stärksten charakterisierenden Wortpaare. Als Eingabe erhält das System eine Menge an Relevanz-Feedback-Dokumenten und gibt eine Menge g an Weighted Word Pairs zurück, wobei jedes Element von g das Gewicht repräsentiert, das einem Paar von Wörtern zugeordnet ist. Die Struktur der Weighted Word Pairs kann als Graph betrachtet werden, der aus mehreren Clustern besteht, wobei jeder Cluster aus einer Menge v_s von Wörtern besteht. Die Menge v_s steht in Beziehung zu einer Wurzel r_i. Diese Wurzel ist ein besonderes Wort, welches den Schwerpunkt des Clusters bildet. Das Gewicht ρ_{is} gibt an, wie stark ein Wort mit der Wurzel r_i verwandt ist. Hieraus entsteht ein Teilgraph, dessen Wurzel wiederum r_i ist. Wurzeln können miteinander verbunden sein und somit den Teilgraphen des Schwerpunktes bilden.

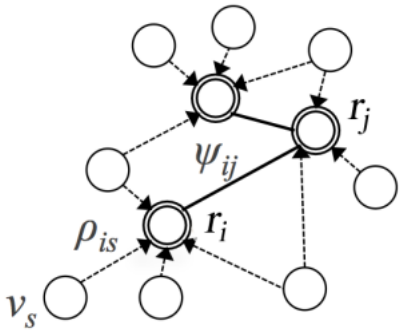


Abbildung 2.1: Weighted Word Pairs Struktur
Quelle: [6]

Das Gewicht ψ_{ij} gibt den Grad der Korrelation zweier Wurzeln r_i und r_j an und kann als $\psi_{ij} = P(r_i, r_j)$ ausgedrückt werden. Aus einer gegebenen Menge an Relevanz-Feedback-Dokumenten können nun die Beziehungen zwischen Wörtern und ihren Wurzeln bestimmt werden. Anschließend kann eine Teilmenge der Wortpaare extrahiert werden. Diese extrahierten Wortpaare werden dann genutzt, um die initiale Suchanfrage zu erweitern. Die Autoren Colace et al. [6] führten eine Evaluation auf dem TREC-8 Corpus durch, wobei sie als Baseline die nicht erweiterten Suchanfragen verwendeten und ihren Ansatz zusätzlich mit Suchanfrageerweiterung über die Termextraktion mit der Kullback-Leibler-Divergenz (KLD) verglichen. Die Experimente zeigten, dass die Verwendung von Weighted Word Pairs die Performance (gemessen wurden die TREC Standardmaße MAP, Precision, R-precision und BPREF) der nicht erweiterten Anfragen und der Termextraktion mit KLD übertrafen [6].

Bei diesen komplexen Verfahren werden aber nur eine Form der Repräsentation der Suchanfrage verwendet, nämlich die durch sie erzeugte Form. Deshalb wurde untersucht wie sich Kombinationen verschiedener Repräsentationen von Suchanfragen verhalten und ob eine erhöhte Komplexität zu einer besseren Retrieval-Performance führt.

Die Autoren Belkin et al. [3] haben sich bereits im Jahre 1993 mit der Fragestellung beschäftigt, wie die Kombination verschiedener Repräsentationen einer Suchanfrage und ihre anschließende Kombination sich auf die Performance der Ergebnismenge der Suchanfrage auswirkt. Dazu wurden 10 erfahrene Experten auf dem Gebiet der Online-Suche gebeten zu gegebenen Informationsbedürfnissen Suchanfragen in Form von booleschen Anfragen, bestehend aus AND, OR und NOT, zu formulieren. Jeder der 10 Experten generierte hierbei für 5 Informationsbedürfnisse je eine Suchanfrage. Insgesamt wurden Suchanfragen für 10 Informationsbedürfnisse gesammelt, alle diese Informationsbedürfnisse stammten aus dem TREC/TIPSTER Projekt. Ausgewählt wurden jeweils 5 In-

formationsbedürfnisse aus den Bereichen International Economics und Science and Technology.

Die von den Experten erstellten Suchanfragen wurden nun in 5 Gruppen aufgeteilt. Die Aufteilung erfolgte so, dass in jeder Gruppe eine Suchanfrage zu jedem Informationsbedürfnis enthalten war und die Verteilung der Experten über die Gruppen gleich war. In jeder Gruppe befand sich dann eine Anfrage von jeden Experten. Anschließend wurde die Retrieval-Performance für jede Gruppe bezüglich der 10 Informationsbedürfnisse untersucht und danach die Performance für die Kombinationen der Gruppen. Dabei wurde aber nur die zweite Hälfte des TREC-Datensatzes als Suchgrundlage verwendet und als Evaluationsmaß wurde Precision mit verschiedenen Recall-Leveln ausgewählt. Im Folgenden soll der Precision-Durchschnitt (P-Avg) über alle Recall-Level betrachtet werden. Als Baseline wurde die Performance gegen das INQUERY-System verglichen, welche auf der Grundlage eines bayesschen Interferenznetzes arbeitet.

Es zeigte sich zum einen, dass eine Schwankung im P-Avg zwischen den einzelnen Gruppen auftrat. Des Weiteren zeigte sich, dass die Kombination der Gruppen zu einer Steigerung des P-Avg führte. Insbesondere zu bemerken ist, dass je mehr Gruppen kombiniert wurden, desto besser wurde der P-Avg. Der beste P-Avg wurde durch Kombination aller Gruppen erreicht, war aber noch nicht besser als der des INQUERY-Systems. Weshalb nun auch noch zusätzlich die Kombination aus allen 5 Gruppen mit der Suchanfrage aus dem INQUERY-System kombiniert wurde. Bei gleicher Gewichtung von booleschen Anfragen und der Anfrage aus dem INQUERY-System wurde sichtbar, dass sich der P-Avg gegenüber der Kombination aus allen 5 Gruppen abermals verbessert hatte, aber immernoch nicht an den P-Avg des INQUERY-System herankam. Durch eine weniger starke Gewichtung (experimentiert wurde mit den Gewichten 0.5 und 0.25) der booleschen Anfragen, konnte die Kombination aus den 5 Gruppen mit dem INQUERY-System den P-Avg des INQUERY-Systems um wenige Prozentpunkte schlagen. Die Autoren schlossen hieraus, dass die Kombination von verschiedenen unabhängig erzeugten booleschen Anfragen im Allgemeinen einen positiven Effekt auf die Retrieval-Performance hat. Ebenso wurde geschlussfolgert, dass auch Kombinationen von Anfragen unterschiedlicher Herkunft und Art zu einer Verbesserung der Performance führen. Die Ergebnisse von Belkin et al. [3] zeigen aber auch, dass der P-Avg der Anfragen von Gruppe 2 fast genauso so gut war, wie der der Kombination aus allen 5 Gruppen. Dies lässt die Annahme zu, dass es eine mögliche beste Kombination aus einer gegebenen Menge an Suchanfragen gibt [3].

Auf Grundlage dieser Erkenntnisse wurde erprobt, ob es hilfreich ist, Suchanfragen aus verschiedenen Erweiterungsmethoden miteinander zu kombinieren, um so verschiedene Repräsentation einer Suchanfrage zu erzeugen und die

Komplexität der Anfrage zu erhöhen. Untersucht wurde diese Fragestellung unter anderem von Pérez-Agüera und Araujo [18] sowie von Abdulla et al. [1].

Pérez-Agüera und Araujo [18] kombinieren eine Kookkurrenzmethode mit einer probabilistischen Methode zur Erweiterung von Suchanfragen. Die Kookkurrenzmethode benutzt Kookkurrenzen von Termen nur aus den höchst gerankten Dokumenten, anstelle von Termen aus allen Dokumenten der Dokumentensammlung. Um die Ähnlichkeit zwischen Termen in den Vektoren des Modells zu bestimmen, werden die Ähnlichkeitsmaße Tanimoto, Dice und Cosinus verwendet. Es werden zusätzlich zwei probabilistische Erweiterungsmethoden eingeführt. Die erste Methode beruht auf der Kullback-Leibler-Divergenz, welche die Divergenz zwischen der Wahrscheinlichkeitsverteilung der Terme der gesamten Dokumentensammlung und der Terme der am höchsten gerankten Dokumente für eine Anfrage berechnet. Die zweite Methode basiert auf dem Divergence from Randomness Termgewichtungsmodell, welches den Informationsgehalt eines Term aus der Divergenz zwischen seiner Verteilung aus den höchst gerankten Dokumenten und einer zufälligen Verteilung folgert. Die Kombination der Kookkurrenzmethode und einer der probabilistischen Methoden erfolgt, indem zunächst jede Methode eine Liste von Erweiterungstermen bestimmt, und anschließend die Listen geschnitten und die verbleibenden Terme neu gewichtet werden. Als Gewichtungsfunktion wird eine Version des Rocchio-Algorithmus verwendet, bei dem nur der Parameter β einen Einfluss auf die Gewichtung hat. Die Autoren konnten in ihrer Evaluation feststellen, dass eine Kombination von Suchanfrageerweiterungsmethoden stets eine bessere Performance im Bezug auf das Maß MAP erzeugte als das Verwenden der einzelnen Methoden. Hieraus schlossen die Autoren, dass die Informationen, die die einzelnen Ansätze nutzen, so unterschiedlich waren, dass sie sich positiv beeinflussten und somit mehr relevante Dokumente gefunden wurden [18].

Abdulla et al. [1] verwenden in ihrer Arbeit vier verschiedene Erweiterungsmethoden. Zum einen eine einfache Methode, die die Erweiterungsterme basiert auf den am häufigsten auftretenden Termen auswählt, die keine Stoppwörter sind. Zum anderen wird Lavrenkos Relevanzmodell verwendet, dessen Parameter für die Dokumentensammlung optimal gewählt wurden. Die dritte Methode nutzt Erweiterungsterme aus dem MetaMap Thesaurus und die vierte Methode verwendet das PubMed-Wörterbuch von der National Library of Medicine, um die Suchanfrage zu erweitern. Kombiniert werden dann immer zwei Verfahren, wobei zunächst die Linearkombination der Scores zweier Erweiterungsmethoden berechnet wird. Anschließend wird nur die Kombination mit dem höchsten Wert der entsprechenden Linearkombination ausgewählt. Die Evaluation auf dem TREC 2006 und 2007 Genomics Datensatz zeigte, dass die beste Kombination daraus bestand, Lavrenkos Relevanzmodell und das PubMed-Wörterbuch zu verwenden. Hiermit konnte eine Verbesserung um

21% im Maß MAP gegenüber der Baseline erreicht werden. Die Baseline bestand aus einem Standardlauf des Indri Toolkits, welches auf der InQuery Query Language basiert [1].

Diese nun komplexeren Verfahren zur Suchanfrageerweiterung haben aber eine entscheidende Schwachstelle, sie achten nicht auf das durch sie erzeugte Ranking. Keyqueries hingegen achten bei der Auswahl der Erweiterungsterme darauf, dass die entstandenen Anfragen alle Ziel-Dokumente in den Top-k Positionen haben. Des Weiteren garantieren Keyqueries auch noch, dass die entstandenen Anfragen minimal sind. Solch starke Bedingungen stellen die vorgestellten Ansätze nicht. Diese theoretische Vorteile der Keyqueries gegenüber von anderen Erweiterungsmethoden soll in dieser Arbeit auf ihre tatsächliche Ausprägung untersucht werden. Die vorgestellten Verfahren zur Kombination von Suchanfragen zeigen auf, dass die Kombination von unterschiedlichen Suchanfrageerweiterungsmethoden einen positiven Einfluss auf die Retrieval-Performance haben. Untersucht wurden aber immer nur Kombinationen aus bis zu zwei Verfahren. In der vorliegenden Arbeit werden nun auch Kombinationen von bis zu drei Suchanfrageerweiterungsverfahren untersucht. Außerdem wird betrachtet, wie sich die etablierten Verfahren RM3, BM25PRF und Axiomatic in Kombination mit Keyqueries verhalten. Des Weiteren wird untersucht wie diese etablierten Verfahren genutzt werden können, um Keyqueries zu berechnen, wenn kein explizites Relevanz-Feedback vorliegt.

2.2 Relevanz-Feedback

Relevanz-Feedback ist ein Verfahren, bei dem durch explizite wie auch implizite Angabe von relevanten Dokumenten die initiale Suchanfrage neu formuliert wird. Dabei wird eine bessere Repräsentation der Suchanfrage in Abhängigkeit des verwendeten Retrieval-Modells berechnet. Anwendung findet dies vor allem für das Probabilistische Modell oder auch für das Vektorraummodell. Vorteilhaft ist dieses Verfahren vor allem dann, wenn der Nutzer Probleme hat, sein Informationsbedürfnis in eine Suchanfrage umzuwandeln. Ebenfalls gut funktioniert dieses Verfahren, wenn der Nutzer möglichst viele Dokumente zu einem Thema betrachten möchte. Ein klassischer Algorithmus für dieses Verfahren ist der Rocchio-Algorithmus aus dem Jahre 1971, welcher mit dem Vektorraummodell arbeitet. Ziel ist es, einen Vektor zu finden, der die Ähnlichkeit mit relevanten Dokumenten maximiert und gleichzeitig die Ähnlichkeit mit nicht relevanten Dokumenten minimiert [13]. Relevanz-Feedback kann auch wie die eben beschriebenen Thesauri automatisch generiert werden, dies wird dann als Pseudo-Relevanz-Feedback bezeichnet. Dabei werden die ersten k Dokumente der initialen Anfrage als relevant angenommen und die Erweiterungsterme

werden auf Grundlage dieser Dokumente ermittelt. Da dies automatisch geschieht muss weder ein Nutzer eine Relevanzbewertung vornehmen, noch muss im Vorfeld eine Relevanzbewertung für die konkrete Anfrage ermittelt worden sein. Aus der Annahme, dass die ersten Top-k Dokumente relevant seien, ergibt sich aber ein nicht vorhersehbares Verhalten des Algorithmus. Denn es können Terme hinzukommen, die nicht dem Informationsbedürfnis der Anfrage entsprechen, da nicht relevante Dokumente fälschlicherweise als relevant eingestuft wurden [7].

Um also möglichst wenig relevante Dokumente vorgeben zu müssen, muss herausgefunden werden, welche Eigenschaften relevante Dokumente besonders effektiv für Relevanz-Feedback-Algorithmen machen. Des Weiteren muss ermittelt werden wie Relevanz-Feedback-Algorithmen aufgebaut sein müssen, damit sie die vorgegebenen Dokumente bestmöglich nutzen.

Im Jahre 2008 wurde im Relevance Feedback Track untersucht, wie sich verschiedene Relevanz-Feedback-Algorithmen verhalten, wenn sie das gleiche Relevanz-Feedback bekommen. Hierfür wurden 5 verschiedene Mengen an Relevanz-Feedback bereitgestellt:(A) kein Relevanz-Feedback, (B) 1 Relevanz-Feedback-Dokument, (C) 3 relevante und 3 nicht relevante Dokumente, (D) 10 gejudgte Dokumente als Obermenge von (C) und (E) 40 bis 800 gejudgte Dokumente. Evaluiert wurden die Runs der Teilnehmenden auf der Terabyte doc Collection mit 50 Fragestellungen aus dem Terabyte Track von 2004 bis 2006 und 214 Fragestellungen aus dem 2007 Million Query Track. Die Evaluation der 118 eingereichten Runs zeigte, dass gerade einmal ein Drittel der Systeme eine monotone Steigerung im MAP bzw. Precision@10 erreichten, wenn die Menge der relevanten Dokumente erhöht wurde. Viele Systeme erreichen bei einer gewissen Anzahl an Relevanz-Feedback-Dokumenten ihr Retrieval-Maximum und fallen mit zunehmender Anzahl an relevanten Dokumenten im MAP und Precision@10 ab. Die Autoren Buckley und Robertson [4] bemerkten weiterhin, dass die eingereichten Runs im MAP und Precision@10 sehr stark streuten. So gab es Runs, deren Performance mit mehreren Relevanz-Feedback-Dokumenten nicht an die Performance der Top-Runs ohne Relevanz-Feedback herankamen. Die Autoren schlussfolgerten hieraus, dass es schwierig sei, im Vorfeld Aussagen über den Einfluss einer ausgewählten Menge an Relevanz-Feedback-Dokumenten auf die Retrieval-Performance zu treffen. Des Weiteren schlossen sie aus den Experimenten, dass das Erhöhen der Menge an Relevanz-Feedback-Dokumenten zu einer besseren Retrieval-Performance führt, wenngleich der Einfluss sehr abhängig von gewählten System ist [4].

Die Arbeit von Ye et al. [20] für den Relevance Feedback Track aus dem Jahre 2009 beschäftigte sich mit den Fragen, wie gut etablierte Retrieval-Modelle relevante Feedback-Dokumente finden und wie sich verschiedene Relevanz-Feedback-Modelle bei Verwendung des Rocchio-Relevanz-Feedback-Framewo-

rks verhalten. Die untersuchten etablierten Termgewichtungsmodelle sind hierbei BM25 und Divergence from Randomness, wobei die Relevanz-Dokumente über die Wahrscheinlichkeit ihrer Relevanz ausgesucht werden. Es werden zusätzlich zwei Gewichtungsmodelle unter Verwendung des Rocchio-Frameworks vorgestellt. Zum einen ein Gewichtungsmodell basiert auf Divergence from Randomness, dessen grundlegende Idee es ist, die Divergenz der Verteilung eines Terms aus pseudo-relevanten Dokumenten aus der Verteilung der gesamten Dokumentensammlung zu messen. Je größer die Divergenz ist, desto höher ist die Wahrscheinlichkeit, dass der Term wichtig für die Anfrage ist. Als Divergenz wird hierbei die Kullback-Leibler-Divergenz verwendet. Als zweites Gewichtungsmodell wird ein kontextsensitives Gewichtungsmodell vorgestellt, dass nach der folgenden Gleichung die Terme rankt.

$$P(t|C) \propto P(t)P(C|t) = P(t)P(c_1, ..., c_m|t) = P(t)\prod_{i=1}^{m} P(c_i|t) \tag{2.1}$$

Wobei C den Kontext der Anfrage beschreibt, der aus den Kontext-Features c_i besteht. $P(t)$ wird interpretiert als die A-Priori-Wahrscheinlichkeit und beschreibt wie wahrscheinlich es ist, dass ein Term t, ohne den Kontext zu kennen, als Erweiterungsterm ausgewählt wird. Die Wahrscheinlichkeit $P(c|t)$ beschreibt, wie wahrscheinlich es ist, dass der Kontext-Feature c_i zum Term t auftritt [20].

Im Relevance Feedback Track aus dem Jahre 2010 [5] wurde die Frage untersucht wie sich unterschiedliche Arten von Relevanz-Feedback auf die Retrieval-Performance auswirken. Dabei wurden für die ersten 100 Fragestellungen des TREC 2009 Million Query Track jeweils 5 Dokumente als Relevanz-Feedback bereitgestellt. Jedes dieser Dokumente sollte dann individuell als Relevanz-Feedback genutzt werden. Die Dokumente wurden dabei wie folgt ausgewählt: ein zufälliges Dokument aus den relevanten Dokumenten, das am häufigsten zurückgegebene relevante Dokument auf Basis früherer Runs, das am seltensten zurückgegebene relevante Dokument, das längste relevante Dokument und das kürzeste relevante Dokument.

Ein Problem welches Information Retrieval Systeme mit händisch gewählten Features haben, ist, dass sich eine Lücke zwischen dem Vokabular der Anfrage und der Dokumentensammlung ergibt. Dieses Problem kann durch neurale Retrieval-Modelle gelöst werden, die die Repräsentation der Sprache durch große Textmengen lernen [14]. In den letzten Jahren wurde gesehen, dass neurale Retrieval-Modelle gute Resultate im Bereich des Ad-hoc-Retrieval liefern können. Jedoch sind diese Modelle mit den bereits existierenden Pseudo-Relevanz-Feedback-Algorithmen nicht kompatibel [11].

Li et al. [11] zeigen in ihrer Arbeit eine Möglichkeit auf, wie neurale Retrieval-Modelle Pseudo-Relevanz-Feedback nutzen können. Dafür wird ein Ende-

zu-Ende neurales Pseudo-Relevanz-Feedback-Framework vorgestellt, dass bereits existierende neurale Retrieval-Modelle einbindet und sie für Pseudo-Relevanz-Feedback nutzbar machen kann. Das Framework bestimmt die Relevanz eines Dokumentes d zur Anfrage q, indem zunächst eine Rankingfunktion $rel_q(q, d)$ auf jedes Dokument angewandt wird und somit die Top-m Dokumente für die Anfrage q bestimmt werden. Diese Dokumente werden als D_q bezeichnet. Um die Relevanz eines Dokumentes zu bestimmen, wird jedes $d_q \in D_q$ verwendet, um die Anfrage q zu erweitern. Anschließend wird das Dokument d mit dem Dokument d_q mit Hilfe der Rankingfunktion $rel_d(d_q, d)$ verglichen. Die Relevanzbewertung $rel_d(d_q, d)$ für jedes $d_q \in D_q$ wird noch weiter über die Rankingfunktion $rel_q(q, d_q)$ gewichtet. Die Rankingfunktion $rel_q(q, d_q)$ dient als Abschätzung für die Konfidenz des Beitrages von d_q im Verhältnis zu q. Über die Rankingfunktion $rel_D(q, D_q, d)$ wird die Relevanzbewertung von d errechnet. Für rel_q wird BM25 vorgeschlagen und für rel_d die beiden State-of-the-Art neuralen Retrieval-Modelle DRMM und K-NRM. In Abbildung 2.2 ist die Architektur des soeben vorgestellten Frameworks als schematische Darstellung zusehen [11].

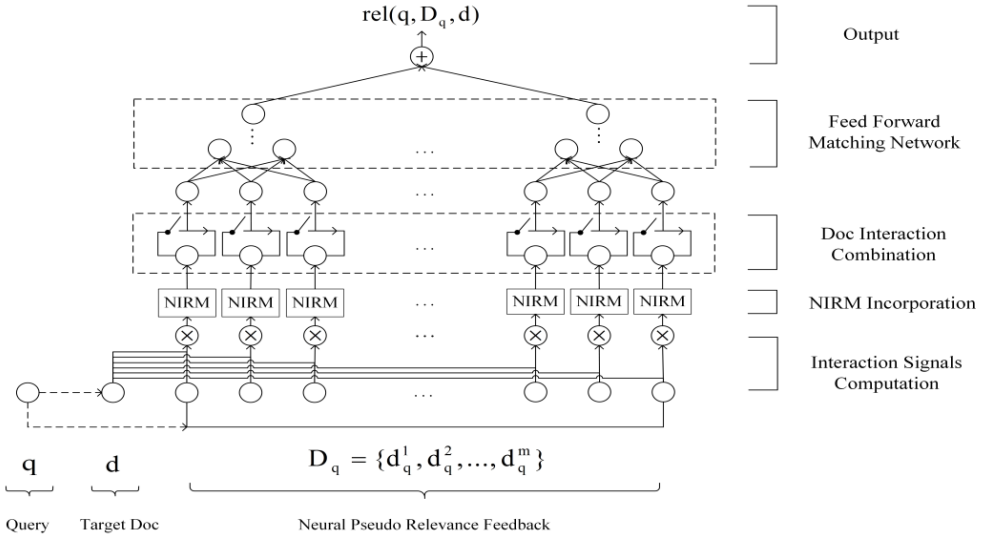


Abbildung 2.2: Architektur des Frameworks von Li et al.
Quelle: [11]

Wir haben gesehen, dass gerade im Bereich des expliziten Relevanz-Feedbacks der Einfluss der Menge und Art des Relevanz-Feedbacks stark zwischen

verschiedenen Ansätzen variiert. In der vorliegenden Arbeit soll deshalb untersucht werden wie sich die Änderung der Menge des Relevanz-Feedbacks und verschiedene Relevanz-Feedback-Arten auf Keyqueries auswirken.

2.3 Keyqueries als Hilfsmittel in der Related Work Suche

Hagen et al. [9] stellen in ihrem Paper einen Ansatz vor, wie mit Hilfe eines Keyquery-basierten Ansatzes mit wenigen vorgegebenen relevanten Veröffentlichungen alle zu dieser Veröffentlichung relevanten Dokumente in einem Corpus gefunden werden können. Hierbei wird für alle vorgegebenen Dokumente eine Menge an Suchanfragen – das sogenannte Keyquery-Cover – aus der Potenzmenge $\mathcal{Q} = 2^W$ gebildet, wobei W hier die Menge der extrahierten Keyphrasen ist. Die Suchanfragen dieser Menge erfüllen sowohl die Bedingungen für Keyqueries als auch die für das Query-Cover. Eine Suchanfrage ist genau dann ein Keyquery, wenn (1) jedes Eingabedokument in den Top-k Treffern der Anfrage ist, (2) die Anfrage mindestens l Treffer hat, und (3) es keine Untermenge der Suchanfrage gibt, sodass die Untermenge alle Eingabedokumente in den Top-k Treffern hat. Werden über alle vorgegebenen Dokumente nicht genügend Dokumente mittels der berechneten Anfragen ermittelt, so wird ein vorgegebenes Dokument aus dem Pool entfernt und es wird erneut ein Keyquery-Cover gebildet. Die Evaluation des Keyquery-basierten Ansatzes wurde auf einem Corpus vorgenommen, der 200 000 Veröffentlichungen aus dem Bereich Informatik enthielt. Die Veröffentlichungen stammten aus den besten 20 Informatik-Konferenzen wie SIGIR, CHI, CIKM, ACL oder STOC [9]. Als Baselines wurde ein Verfahren zur Related Work Suche von Nascimento et al. [15] verwendet, sowie Sofia Search [8] und die vorgeschlagenen Veröffentlichungen von Google Scholar. Es wurde dann eine Nutzerstudie durchgeführt, bei der Studierende der Informatik und Forschende aus dem Gebiet der Informatik zu ihren Forschungsgebieten eine geringe Menge an Veröffentlichungen aus dem Corpus auswählten. Anschließend wurden die Teilnehmenden gefragt, welche weiteren Veröffentlichungen sie als Rückgabe der Algorithmen erwarten würden und wie sie die Veröffentlichungen bestimmt hatten. Als nächstes wurden die Top-10 Ergebnisse der Algorithmen den Teilnehmern vorgelegt und die Teilnehmer bewerteten die Veröffentlichungen nach Relevanz und Vertrautheit. Die Evaluation über alle vorgegebenen Veröffentlichungen zeigte, dass das Keyquery-Cover die Baselines in Bezug auf den NDCG@10 und Precision@10 schlagen konnte, wobei der Unterschied nur zur Nascimento et al. [15] Baseline statistisch signifikant war [9]. Die beste Retrieval-Performance konnte durch die Kombination des Keyquery-Cover, Sofia Search und Google Scholar

erreicht werden, wobei diese Kombination ihre einzelnen Komponenten statistisch signifikant schlagen konnte [9].

Die Autoren Hagen et al. [9] haben die neuartigen Keyqueries vorgestellt und auf einem von ihnen ausgewählten Corpus für die Related Work Suche erprobt. Motiviert durch den Conclusion Paragraph der Autoren Hagen et al. [9] werden diese ersten vielversprechenden Ergebnisse durch eine Auswertung auf dem Robust04-Corpus in dieser Arbeit weiter untersucht. Allerdings nicht im Kontext der Related Work Suche, sondern für die Suchanfrageerweiterung mit explizitem Relevanz-Feedback. Ebenso werden wie von den Autoren vorgeschlagen die Generierung und Auswahl von Keyquery-Kandidaten und Methoden zur Gewichtung von Keyqueries untersucht [9].

Kapitel 3

Suchanfrageerweiterungen mit Keyqueries

In diesem Kapitel soll die Frage beantwortet werden, wie Keyqueries zur Erweiterung von Suchanfragen benutzt werden können und es wird gezeigt, dass das in Kapitel 2.3 vorgestellte Verfahren zur Bestimmung von Keyqueries in den Aspekten der Bereitstellung des Relevanz-Feedbacks, der Kandidatengenerierung und der Kandidatenauswahl verbessert wurde.

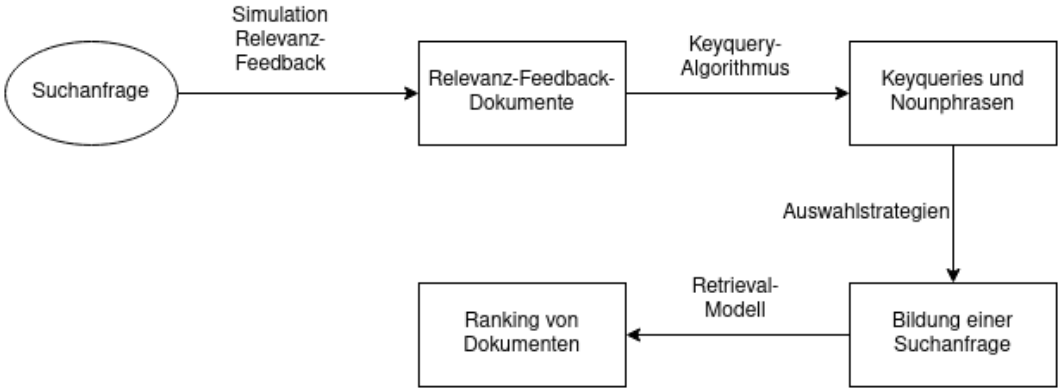


Abbildung 3.1: Architektur des vorgestellten Ansatzes

3.1 Retrieval-Pipeline

Zunächst soll ein Blick auf die Retrieval-Pipeline geworfen werden, um zu verstehen wie Keyqueries zur Erweiterung von Suchanfragen genutzt werden können, einen schematischen Überblick zeigt die Abbildung 3.1.

Im ersten Schritt der Retrieval-Pipeline wird das explizite Relevanz-Feedback simuliert, hierbei wird auf Grundlage einer Suchanfrage eine Menge von sortierten Dokumenten bereitgestellt. Die Sortierung erfolgt aufgrund ausgewählter Dokumenteneigenschaften (siehe hierzu Kapitel 3.2). Diese unterschiedlichen Sortierungen haben den Hintergrund herauszufinden, welche Dokumenteneigenschaften förderlich für den vorgestellten Ansatz sind. Aus einer dieser Sortierungen werden dann die ersten $r \in \mathbb{N}$ relevanten Dokumente als explizites Relevanz-Feedback genutzt, wobei die Entscheidung über die Relevanz eines Dokumentes bezüglich einer Suchanfrage einer annotierten Datei entnommen wird.

Aus den Relevanz-Feedback-Dokumenten werden nun Terme und Nounphrasen extrahiert. Dabei können die Terme aus unterschiedlichen Quellen ausgewählt werden. Zum einen aus dem Index auf Basis ihres TfIdf-Wertes, zum anderen aus den Erweiterungsmodellen RM3, BM25PRF oder Axiomatic auf Basis der Termgewichte. In jedem Fall werden die $t \in \mathbb{N}$ Terme ausgewählt, die den höchsten Wert bezüglich des TfIdf-Wertes bzw. der Termgewichte besitzen und anschließend werden diese Terme als Grundlage für die Berechnung der Keyqueries verwendet. Nounphrasen werden durch einen Part-of-Speech-Tagger (POS-Tagger) klassifiziert und anschließend extrahiert. Die Extraktion von Nounphrasen hat den Hintergrund, dass die Relevanz-Feedback-Dokumente im abschließenden Ranking auf den ersten Plätzen erscheinen sollen, wobei dies aber durch eine Suchanfrage geschehen soll und nicht durch künstliches Nach-Oben-Schieben. Denn dann werden auch weitere relevante Dokumente im Ranking nach oben kommen, während ein künstliches Schieben diesen Vorteil verwehrt.

Aus den gebildeten Keyqueries und Nounphrasen werden nun jeweils die besten Anfragen bezüglich des NDCG ausgewählt. Dabei werden die Anfragen an das Retrieval-Modell gestellt und die von ihnen zurückgelieferten Dokumente werden auf Basis der Relevanz-Feedback-Dokumente bezüglich des NDCG evaluiert. Mit diesen ausgewählten Keyqueries und Nounphrasen wird nun eine Anfrage gebildet und an das Retrieval-Modell gestellt. Der Vorteil einer Suchanfrage ist, dass sich ein kohärentes Ranking bildet und die Anfrage eine beliebige Komplexität erhalten kann. Um eine Suchanfrage zu bilden stehen drei verschiedene Verfahren zur Auswahl. Zum einen können die ausgewählten Keyqueries und Nounphrasen mit einen logischen ODER verknüpft werden, sie können über einen Greedy-Algorithmus bis zu einer festgelegten Komplexität konkateniert werden oder die Häufigkeit der Terme in den Anfragen kann gezählt werden und anschließend können aus den am häufigsten auftretenden Termen erneut Keyqueries berechnet werden.

3.2 Simulation des expliziten Relevanz-Feedbacks

Für die Simulation des expliziten Relevanz-Feedbacks wurden sechs Verfahren eingeführt, sie basieren auf der Idee, dass relevante Dokumente aufgrund ausgewählter Dokumenteneigenschaften sortiert werden und anschließend die ersten relevanten Dokumente als Relevanz-Feedback entnommen werden. Dabei können die Dokumente aufgrund ihrer Länge, der Schwierigkeit gefunden zu werden, zufällig oder über eine Rankingfunktion diese Sortierung erhalten.

Die erste Möglichkeit das explizite Relevanz-Feedback zu simulieren, erfolgt über ein initiales Ranking, welches aus einer Rankingfunktion resultiert, im Folgenden wird dieses Verfahren als SRF (Sequential Relevance Feedback) bezeichnet. Im weiteren Verlauf der Arbeit wird diese Rankingfunktion BM25 sein. Hierbei wird das Ranking von Position 1 bis zum Ende des Rankings abgelaufen und es werden alle Dokumente, die auf diesem Weg liegen, in das explizite Relevanz-Feedback aufgenommen. Dies geschieht solange bis eine zuvor spezifizierte Anzahl $r \in \mathbb{N}$ an relevanten Dokumenten erreicht wird.

Inspiriert durch den TREC Relevance Feedback Track aus dem Jahre 2010 [5] wurden fünf weitere Möglichkeiten der Simulation des expliziten Relevanz-Feedbacks eingeführt. So gibt es einmal eine Variante, die die Dokumente aufsteigend nach ihrer Länge sortiert, im Folgenden als Shortest bezeichnet. Analog dazu gibt es eine Variante, die die Dokumente absteigend ihrer Länge nach sortiert, im Folgenden als Longest bezeichnet. Zum anderen können die Dokumente nach einer zufälligen Reihenfolge sortiert werden, dies wird als Random bezeichnet. Um eine Reproduzierbarkeit der Ergebnisse zu gewährleisten, wurde die zufällige Sortierung einmalig mit einem festen Seed durchgeführt. Desweiteren gibt es noch die Möglichkeit die Dokumente nach der Schwierigkeit sie zu finden zu sortieren. Um die Schwierigkeit die Dokumente zu finden zu ermitteln, wurden 188 Runs auf dem Robust04-Corpus betrachtet, die alle dieselben 250 Suchanfragen bearbeitet hatten wie der in dieser Arbeit vorgestellte Ansatz. Auf Grundlage dieser Runs wurde nun gezählt, wie oft ein Dokument für jede Fragestellung gefunden wurde. Nun konnten die Dokumente aufsteigend nach ihrem Aufkommen, im Folgenden als Most Difficult bezeichnet, bzw. absteigend nach ihrem Aufkommen sortiert werden, im Folgenden als Least Difficult bezeichnet. Für alle diese fünf soeben vorgestellten Verfahren lief die Simulation des expliziten Relevanz-Feedback so ab, dass die ersten $r \in \mathbb{N}$ relevanten Dokumente aus der neuen Sortierung in die Menge des Relevanz-Feedbacks aufgenommen wurden.

3.3 Berechnung von Keyqueries

Um Keyqueries zu berechnen werden zunächst eine Reihe von Termen benötigt, aus denen eine Potenzmenge gebildet werden kann. Die Terme für den in dieser Arbeit vorgestellten Ansatz stammen aus den relevanten Dokumenten, die mit einem der Verfahren zur Simulation des expliziten Relevanz-Feedbacks ermittelt wurden. Dabei gibt es zwei grundlegende Möglichkeiten wie diese Terme extrahiert werden können. Zum einen können sie über ihren TfIdf-Wert aus dem Index extrahiert werden, zum anderen können sie über die Suchanfragenerweiterungsmodelle RM3, BM25PRF und Axiomatic mit Hilfe ihrer zugeordneten Termgewichte ermittelt werden. Dabei werden stets die $t \in \mathbb{N}$ Terme ausgewählt, die den höchsten Wert bezüglich des TfIdf-Wertes bzw. des Termgewichts aufweisen.

Sind die besten $t \in \mathbb{N}$ Terme ausgewählt, so werden diese Terme gefiltert. Alle Terme, die aus nur einem Zeichen oder aus mehr als 20 Zeichen bestehen, werden nicht weiter betrachtet. Ebenso Terme bestehend aus Sonderzeichen, wobei Sonderzeichen alle Zeichen sind, die weder aus Buchstaben noch aus Zahlen bestehen. Als letzte Eigenschaft wird überprüft, ob der zu betrachtende Term in mehr als 10% der Dokumenten der Sammlung auftritt. Sollte dies der Fall sein, so ist der Term ein Stoppwort. Diese Heuristik hat den Vorteil, dass keine spezifische Stoppwortliste für eine Dokumentensammlung angelegt werden muss, jedoch wird hierdurch ein neuer Parameter eingeführt, dessen Optimum ermittelt werden muss. Implementiert wird diese Heuristik von dem Information Retrieval Toolkit Anserini, auf dessen Grundlage die in dieser Arbeit vorgestellten Verfahren basieren [19]. Die so ermittelten Terme werden nun als Eingabe für die Keyquery-Berechnung genutzt, dabei wird die Potenzmenge der Terme gebildet und anschließend wird für jedes Element dieser Potenzmenge geprüft, ob die Keyquery-Eigenschaften erfüllt sind (vgl. Kapitel 2.3).

Aufgrund der restriktiven Eigenschaften der Keyqueries kann es vorkommen, dass gar keine Keyqueries für eine Anfrage berechnet werden können. Für diesen Fall musste eine Ausweichstrategie entwickelt werden. Eine Möglichkeit ist das sogenannte Relaxed Keyquery. Die Idee dahinter basiert auf dem Keyquery-Cover [9]. Kann für eine Anfrage eine festgelegte Anzahl an Keyqueries nicht berechnet werden, dann wird ein Ziel-Dokument entfernt und es werden erneut Keyqueries berechnet, jetzt aber mit $r - 1$ Dokumenten. Die gefundenen Keyqueries werden mit den Keyqueries aus dem ersten Schritt vereint, sollte die festgelegte Anzahl immernoch nicht erreicht sein, so wird ein weiteres Ziel-Dokument entfernt. Dies wird solange wiederholt bis die festgelegte Anzahl an Keyqueries gefunden wurde oder keine Dokumente sich mehr in der Dokumentenmenge befinden. Die in den einzelnen Schritten zu betrachtenden

Dokumente ergeben sich aus der Potenzmenge der Ziel-Dokumente. Diese Methode hat den Nachteil, dass der Keyquery-Algorithmus im schlimmsten Fall $2^{|r|} - 1$ Mal durchlaufen werden muss, wenn r die Anzahl der Ziel-Dokumente ist. Auch garantiert dieses Verfahren nicht, dass hierdurch Keyqueries gefunden werden. Denn es kann passieren, dass alle Anfragen aus der Potenzmenge weniger als l Treffer haben, oder dass alle Anfragen kein einziges Ziel-Dokument in den Top-k Positionen besitzen. Dementsprechend ist eine geeignete Wahl der Parameter k und l entscheidend für eine gute Performance des vorgestellten Ansatzes.

Für eine tiefergehende Analyse der vorgestellten Verfahren wurde eine Experimentierumgebung entwickelt, die es erlaubt einzelne Suchanfragen mit ausgewählten Algorithmen zu bearbeiteten und eine Vielzahl an wichtigen Statistiken bereithält. Hierzu zählen auf welchen Platz die Ziel-Dokumente gelandet sind oder welche Terme zur Bestimmung von Keyqueries genutzt wurden. Auch lassen sich alle berechneten Keyqueries und verwendeten Terme in ihrer ungestemmten Form anzeigen, denn die extrahierten Terme aus dem Index sind bereits gestemmt. Das Zurück-Stemmen dient einer besseren Lesbarkeit für den menschlichen Nutzer und es wird deutlicher welche Terme im Vergleich zu anderen Erweiterungsmodellen entfallen oder hinzugekommen sind. Betrachten wir nun die berechneten Keyqueries für die Suchanfrage `international organized crime` auf Grundlage dieser Experimentierumgebung mit einem expliziten Relevanz-Feedback von 5 Dokumenten und den Keyquery-Parametern $k = 100$ und $l = 10$.

```
crime criminal
organized smuggling criminal
organized crime
organized ministry criminal
smuggling crime ministry
smuggling ministry criminal
```

Zu sehen ist, dass die berechneten Keyqueries aus wenigen Termen (maximal drei in diesem Beispiel) bestehen. Des Weiteren ist auffällig, dass die Keyqueries nur aus insgesamt fünf verschiedenen Termen bestehen, obwohl in diesem Fall 16 verschiedene Terme als Grundlage für die Berechnung der Keyqueries verwendet wurden. Dementsprechend treten Terme wiederholt in den Keyqueries auf. Dies ist ein Indikator dafür, dass diese Terme für eine gute Anfrage besonders wichtig sind, denn sonst würden diese Anfragen die Ziel-Dokumente nicht in die Top-100 der erzeugten Rankings bringen. Interessant ist weiterhin, dass der Term `international` in den berechneten Keyqueries nicht mehr zu finden ist, obwohl er Teil der originalen Anfrage ist. Die Terme `organized`

Term	Gewichtung
vaculik	0.0149
western	0.0149
international	0.1666
crime	0.3015
italian	0.0149
republic	0.0149
police	0.0149
germany	0.0248
east	0.0198
mafia	0.0259
russian	0.0199
organized	0.2551
criminal	0.0553
czech	0.0198
percent	0.0168
corrupt	0.0194

Tabelle 3.1: Erweiterungsterme von RM3 für `international organized crime`

und `crime` , ebenfalls Teil der originalen Anfrage, sind aber in den berechneten Keyqueries zu finden. Dieser Umstand lässt darauf schließen, dass für diese Anfrage der Term `international` keine wichtige Rolle spielt und demnach nicht Bestandteil einer Keyquery ist. Dafür wird aber der Term `ministry` in die Keyqueries aufgenommen, der in keinem unmittelbaren Zusammenhang mit der originalen Anfrage steht. Der Keyquery-Ansatz findet also neben für einen Menschen offensichtlichen wichtigen Termen auch solche Terme, die auf den ersten Blick unwichtig erscheinen, aber aufgrund der Beschaffenheit der Dokumentensammlung Bestandteil einer guten Anfrage sein sollten. Betrachten wir nun die Erweiterungsterme, die das Suchanfragenerweiterungsmodell RM3 für diese Anfrage berechnet hat. Der Tabelle 3.1 kann entnommen werden, dass die am stärksten gewichteten Terme `international` , `crime` und `organized` sind, es tauchen also alle Terme aus der originalen Anfrage wieder in der neu berechneten Anfrage auf. Anhand der Gewichte ist zu sehen, dass RM3 dem Term `international` im Vergleich zu `crime` und `organized` ein sehr geringes Gewicht zuordnet. Interessanterweise ist dies genau der Term aus der originalen Anfrage, den der Keyquery-Ansatz nicht mit in die Keyqueries aufgenommen hat. Es erhärtet sich also der Verdacht, dass `international` die Anfrage schlecht charakterisiert und somit von RM3 und dem Keyquery-Ansatz bestraft wird. Des Weiteren ist zu sehen, dass RM3 Terme findet, die

der Keyquery-Ansatz nicht benutzt und andersherum genauso. Zum Beispiel tauchen die Terme `ministry` oder `smuggling` überhaupt nicht in der RM3-Anfrage auf, während beispielsweise der Term `corrupt` nicht in den Keyqueries zu finden ist. Auffällig ist, dass RM3 eine ganze Reihe von Ländern bzw. ihre zugehörigen Adjektive als Terme extrahiert, während dies in den Keyqueries überhaupt keine Rolle spielt. Ein Vorteil der Keyqueries lässt sich somit feststellen. Sind Terme für eine gute Anfrage nicht notwendig, so lassen die Keyqueries diesen Term weg und die Anfrage wird hierdurch vereinfacht. RM3 hingegen benutzt so viele Erweiterungsterme wie zuvor definiert und Terme aus der originalen Anfrage, die eigentlich nicht wichtig für die Anfrage sind, werden immer aufgenommen. Die Gewichtverteilung leidet darunter und die Komplexität der Anfrage kann nicht verringert werden.

3.4 Bildung einer Suchanfrage

Wie im vorangegangenen Kapitel gesehen, werden für jede Fragestellung mehrere Keyqueries berechnet. Der nächste Pipeline-Schritt besteht also darin aus diesen einzelnen Anfragen eine Anfrage zu bilden. Dies hat den Hintergrund, dass eine Anfrage ein zusammenhängendes Ranking liefert und durch eine Anfrage eine festgelegte Komplexität nicht überschritten werden kann. Hierzu wurden drei verschiedene Verfahren eingeführt, die aus einzelnen Anfragen eine Anfrage bilden.

Die einfachste Möglichkeit besteht darin, dass die besten Keyqueries bezüglich des NDCG ausgewählt werden und mit einem logischen ODER verbunden werden. Anschließend wird die so gebildete Anfrage mit den extrahierten Nounphrasen ebenfalls per ODER-Verknüpfung verbunden und gegen das Retrieval-Modell gestellt und das Ranking wird zurückgegeben. Dieses Verfahren erzeugt eine sehr komplexe Anfrage, die aber im Algemeinen gut performt, dies deckt sich mit den Ergebnissen von Belkin et al. [3].

Eine weitere Möglichkeit besteht darin, dass die Terme aus allen Keyqueries gezählt werden und anschließend die Terme ausgewählt werden, die am häufigsten auftraten. Diese ausgewählten Termen werden dann wieder in die Keyquery-Berechnung gegeben und es entstehen so neue Keyqueries. Diese neuen Keyqueries werden mit einem logischen ODER verbunden und an das Retrieval-Modell gestellt. Dieses Verfahren hat den Vorteil, dass die Häufigkeit der Terme aus den Keyqueries beachtet werden und dass durch die erneute Berechnung von Keyqueries die neue Anfrage nur aus Keyqueries besteht und somit die Vorteile der Keyqueries zum Tragen kommen. Der Nachteil dieser Methode ist aber, dass wieder mehrere Keyqueries berechnet werden können, die dann wieder verbunden werden und somit eine komplexe Anfrage entste-

hen kann. Diese Anfrage wird jedoch nicht so komplex, als wenn die originalen Keyqueries miteinander verbunden worden wären, da weniger Terme als Grundlage zur Berechnung der Keyqueries genutzt werden.

Um das Problem von mehreren berechneten Keyqueries zu umgehen, wurde eine weitere Methode eingeführt. Hierbei werden die berechneten Keyqueries bezüglich des NDCG auf Grundlage der Ziel-Dokumente evaluiert und absteigend nach ihrer Performance sortiert. Nun wird mittels eines Greedy-Algorithmus die neue Anfrage wie folgt zusammengesetzt. Begonnen wird mit einer leeren Anfrage Q und es wird überprüft, ob die Anfrage Q eine geringere Komplexität als die vorgegebene Komplexität besitzt. Dabei wird die Komplexität einer Anfrage in der Anzahl der Terme der Anfrage gemessen. Sollte die Bedingung erfüllt sein, so wird die beste Anfrage aus der sortierten Liste ausgewählt. Die Anfrage wird aus der Liste entfernt und mit der Anfrage Q so konkateniert, dass die Anfrage Q bezüglich ihrer Terme redundanzfrei bleibt. Dies wird solange wiederholt, bis die Komplexität der Anfrage Q größer der vorgegebenen Komplexität ist. Anschließend wird die Anfrage Q an das Retrieval-Modell gestellt. Der Vorteil dieses Verfahren ist, dass genau eine Anfrage berechnet wird, die eine vorgegebene Komplexität besitzt und damit beliebig einfach gemacht werden kann. Der Nachteil der sich durch die Konkatenation der Terme ergibt, ist, dass die finale Anfrage Q nicht zwingend ein Keyquery sein muss, da die Keyquery-Eigenschaften beim Zusammenfügen der Anfrage Q nicht berücksichtigt werden. Hierdurch gehen die Vorteile der Keyqueries verloren.

3.5 Hybride Suchanfrageerweiterung

In der vorgestellten Retrieval-Pipeline wurden bereits Erweiterungsmodelle genutzt, um damit Terme für die Keyquery-Berechnung zu extrahieren. Als nächster Schritt hieraus wurde nun untersucht, wie sich der vorgestellte Ansatz verhält, wenn zusätzlich zum expliziten Relevanz-Feedback noch Dokumente aus den Erweiterungsmodellen RM3, BM25PRF und Axiomatic als Relevanz-Feedback für die Keyquery-Berechnung verwendet werden. Hierfür wurde das sogenannte hybride Modell entwickelt.

Der Ablauf der Retrieval-Pipeline ändert sich nun wie folgt. Zu Beginn der Pipeline werden über die in Kapitel 3.2 vorgestellten Verfahren relevante Dokumente bereitgestellt. Nun wird dieses Relevanz-Feedback an die Suchanfragenerweiterungsmodelle RM3, BM25PRF und Axiomatic gegeben und jedes dieser Modelle berechnet zunächst individuell seine Ergebnismenge. Aus diesen Ergebnismengen werden jeweils die $n \in \mathbb{N}$ ersten Dokumente entnommen und zusammen mit dem originalen Relevanz-Feedback an die Retrieval-Pipeline

übergeben. Auf Grundlage dieses erweiterten Relevanz-Feedbacks werden nun Keyqueries und Nounphrasen berechnet. Es wird wie in Kapitel 3.4 beschrieben eine Suchanfrage gebildet und an das Retrieval-Modell gestellt. Hierbei ergibt sich die Problematik, dass sich mehrfach das gleiche Dokument im neuen Relevanz-Feedback befindet und dass eigentlich nicht relevante Dokumente in das Relevanz-Feedback kommen können, da bei der Auswahl der neuen vermeintlichen relevanten Dokumente keine Überprüfung der tatsächlichen Relevanz vorgenommen wird. Dies hat den Hintergrund, dass dieser Ansatz sonst mehr Wissen hätte als die ursprüngliche Retrieval-Pipeline und somit nicht mehr vergleichbar wäre. Dies bringt jedoch eine Unsicherheit in das Vertrauen des Relevanz-Feedbacks auf das sich die Retrieval-Pipeline stützt. Inwiefern sich diese Unsicherheit auf die Performance des in diesem Kapitel vorgestellten Verfahrens auswirkt, wird in Kapitel 4.2.2 untersucht.

3.6 Gewichtete Keyqueries

In Kapitel 3.3 wurde bereits beobachtet, dass z.B. RM3 Terme aus der originalen Anfrage nicht entfernt, obwohl sie für eine Anfrage nicht förderlich sind und damit das Gewicht nicht optimal ausnutzt. Des Weiteren können die Anfrageterme von RM3 redundant sein, die Anfrage muss nicht minimal sein und sie garantiert nicht, dass Ziel-Dokumente in den Top-Positionen liegen. All diese Probleme können mit Keyqueries behoben werden. Dementsprechend wurde untersucht, wie die Term-Gewicht-Paare von Erweiterungsmodellen der Keyquery-Berechnung zugänglich gemacht werden können.

Hierzu bildet zunächst ein Erweiterungsmodell auf Grundlage von Relevanz-Feedback-Dokumenten seine expandierte Anfrage und es werden die $t \in \mathbb{N}$ Term-Gewicht-Paare ausgewählt, die das höchste Gewicht besitzen. Anschließend wird die Potenzmenge aller ausgewählter Terme berechnet und es werden all die Teilmengen betrachtet, die eine vorgegebene Kardinalität $k \in \mathbb{N}$ besitzen. Für jede dieser Teilmengen werden neue Termgewichte über den RM3-Algorithmus berechnet. Hierbei werden die Terme jeder Teilmenge und die originalen Anfrageterme an RM3 gestellt und von RM3 neu gewichtet. Die hieraus entstandenen neuen Term-Gewicht-Paare werden als eine Anfrage gegen das Retrieval-Modell gestellt und auf Grundlage der Ziel-Dokumente mit dem Evaluationsmaß NDCG evaluiert. Nun werden die besten $n \in \mathbb{N}$ Anfragen nach dem NDCG ausgewählt. Diese ausgewählten Anfragen werden mit einem logischen ODER verknüpft und abschließend gegen das Retrieval-Modell gestellt. Das entstandene Ranking wird zurückgegeben.

3.7 Suchanfrageerweiterung mit implizitem Relevanz-Feedback

Ein weiterer interessanter Untersuchungsgegenstand ist die Frage, ob die Retrieval-Pipeline auch mit implizitem Relevanz-Feedback funktionieren kann. Die in Kapitel 3.5 betrachteten Suchanfragenerweiterungsmodelle RM3, BM25-PRF und Axiomatic können alle anstelle des explizitem Relevanz-Feedbacks Pseudo-Relevanz-Feedback nutzen und somit Rankings erzeugen.

Die Idee ist also nun zunächst die drei Suchanfragenerweiterungsmodelle auf Basis von Pseudo-Relevanz-Feedback Rankings produzieren zu lassen und aus diesen Rankings dann wieder jeweils die ersten $n \in \mathbb{N}$ Dokumente auszuwählen und als Eingabe für die Retrieval-Pipeline zu nutzen. Hierbei ergibt sich die Problematik, dass ohne explizites Relevanz-Feedback Dokumente ohne Wissen über ihre Relevanz als relevant eingestuft werden müssen. Dadurch wird eine Unsicherheit in die Auswahl der relevanten Dokumente gebracht. Diese Beeinflussung der Rankings, die als Eingabe für die Retrieval-Pipeline genutzt werden, schlägt sich entsprechend auf den vorgestellten Ansatz nieder. Möglich wäre es das Klickverhalten des Nutzers als Unterstützung für das Pseudo-Relevanz-Feedback zu nutzen. Die Betrachtung von Klickverhalten oder anderer Möglichkeiten das Pseudo-Relevanz-Feedback zu unterstützen, sind nicht Bestandteil dieser Arbeit. Ein Vorteil der Methode mit implizitem Relevanz-Feedback ist dennoch, dass ohne Wissen über die Relevanz von Dokumenten die Retrieval-Pipeline verwendbar ist. Auf einer Dokumentensammlung ohne solch ein Wissen für gegebene Suchanfragen, wäre es sonst nicht möglich den vorgestellten Ansatz anzuwenden.

Kapitel 4

Evaluation

Im folgenden Kapitel sollen die bisher theoretisch vorgestellten Algorithmen zur Suchanfrageerweiterung mit Keyqueries systematisch auf dem Robust04-Corpus evaluiert und mit etablierten Verfahren zur Suchanfrageerweiterung verglichen werden.

4.1 Pilotexperiment auf Robust04

Die nun folgenden Evaluationen wurden auf dem Robust04-Corpus ausgeführt, der aus 528 155 Dokumenten besteht. Die dort enthaltenen Dokumente kommen aus den TREC Disks 4 und 5, und enthalten Dokumente aus der Financial Times, dem Federal Register 94, der LA Times, und dem FBIS (Foreign Broadcast Information Service). Als Fragestellungen wurden 250 Suchanfragen bereitgestellt, die sich wie folgt zusammensetzen: Suchanfragen 301–450 stammen aus TREC 6–8, Suchanfragen 601–650 stammen aus dem Robust Track von 2003 und Suchanfragen 651–700 sind neu für den Robust 2004 Track ausgewählt wurden [16]. Ebenfalls wurde eine Qrel-Datei bereitgestellt, die für alle der 250 Fragestellungen Dokumente aus dem Corpus enthält, die relevant bzw. nicht relevant für die jeweilige Fragestellung sind. Der Robust04-Corpus wurde für die folgenden Experimente ausgewählt, da er im Allgemeinen als ein qualitativ hochwertiger Corpus mit genügend vielen Fragestellungen angesehen wird, auf dem Rückschlüsse gezogen werden können, die auch für andere Corpora Gültigkeit behalten [12].

Alle Experimente wurden mit Hilfe des Information Retrieval Toolkits Anserini durchgeführt, welches es sich zur Aufgabe gemacht hat, reproduzierbare Ergebnisse aus Ad-hoc-Retrieval Experimenten zu liefern und diese Experimente zu vereinfachen [19]. Als Evaluationsmaße wurden die etablierten Maße NDCG@1000 und Recall@1000 ausgewählt, die Evaluation der Durchläufe erfolgte auf Basis der Python-Bibliothek trectools [17]. Bei der Evaluation wur-

den nicht auf Relevanz geprüfte Dokumente aus den Rankings entfernt, da bei einem Corpus dieser Größe nicht alle Dokumente auf Relevanz geprüft werden können und es somit vorkommen kann, dass eigentlich relevante Dokumente keine Relevanzbewertung erhalten haben und somit fälschlicherweise als nicht relevant eingestuft würden. Dabei werden die Ergebnisse der Experimente gegen verschiedene Baselines verglichen, zum einen gegen das etablierte Rankingmodell BM25, aber auch gegen die Suchanfrageerweiterungsmodelle RM3, BM25PRF und Axiomatic, welche alle bereits von Anserini implementiert werden. Alle in Kapitel 3 beschriebenen Verfahren wurden von mir in Anserini als Java-Code implementiert.

4.2 Explizites Relevanz-Feedback

Für die Evaluation der Verfahren mit explizitem Relevanz-Feedback wurden je 2 Tabellen für jede Feedback Variante (vgl. Kapitel 3.2) erstellt. Eine Tabelle zeigt die Evaluation für die gewichteten Keyqueries (vgl. Kapitel 3.6) mit Query-Weights (QW) im Bereich von 0 bis 1, einem Relevanz-Feedback von 5 Dokumenten, einer Anfragenlänge von 10 Termen und 11 ausgewählte Terme aus RM3 für die gewichteten Keyqueries. Zusätzlich kann die Baseline RM3 betrachtet werden. Für jeden dieser Ansätze können die drei verschiedenen Rankingarten original, toTop und removed in der Tabelle gefunden werden. Bei toTop wurden die Dokumente aus dem initialen Relevanz-Feedback auf die obersten Positionen geschoben und bei removed wurden diese Relevanz-Feedback-Dokumente komplett aus dem Ranking entfernt. Original gibt an, dass es sich um die Evaluation eines nicht manipulierten Rankings handelt.

Die zweite Tabelle zeigt weitere Keyquery-Ansätze aus den Kapiteln 3.3, 3.4 und 3.5 mit verschiedenen Parametern. Die Abkürzung NP steht dabei für Nounphrasen und die Abkürzung KQ für Keyqueries. Zusätzlich wird die Baseline BM25 mit den Parametern $k1 = 0.9$ und $b = 0.4$ angegeben. Als zusätzliche Baseline ist der beste Ansatz aus der Tabelle für die gewichteten Keyqueries für die zu betrachtende Feedback-Variante aufgetragen. Es gibt zudem noch je eine Tabelle für die besten Ansätze aus diesen 6 Tabellen für die Evaluationsmaße NDCG und Recall.

Des Weiteren sind für die Feedback-Variante SRF 5 Plots zu finden, die den NDCG der gewichteten Keyqueries und des Verfahrens RM3 in Abhängigkeit der Menge des Relevanz-Feedback zeigen. Zu sehen sind für die Query-Weights (QW) 0 bis 1 die Änderung des NDCG bei einer Anfragenlänge von 8 Termen.

Ansatz	Parameter	NDCG@1000	Recall@1000
Keyquery ohne NP	1 KQ	0.539	0.699
	5 KQ	0.599	0.724
	10 KQ	0.602	0.728
Keyquery mit NP	1 NP	0.652	0.733
	2 NP	0.669	0.745
	5 NP	0.686	0.760
Relaxed Keyquery	1 KQ	0.657	0.709
	5 KQ	0.692	0.766
	10 KQ	0.697	0.770
	15 KQ	0.700	0.775
Hybrides Modell	RM3	0.658	0.772
	BM25PRF	0.677	0.789
	Axiom	0.675	0.784
	RM3+BM25PRF	0.661	0.783
	RM3+BM25PRF+Axiom	0.647	0.783
Greedy-Query Zählen	5 Terme	0.383	0.435
	10 Terme	0.495	0.545
	5 Terme	0.598	0.725
	10 Terme	0.604	0.730
BM25 RM3	k1=0.9, b=0.4 QW=0.25	0.534 **0.711**	0.699 **0.802**

Tabelle 4.1: Keyquery-Ansätze mit Feedback-Variante SRF

Ansatz	Parameter	NDCG@1000	Recall@1000
Keyquery ohne NP	1 KQ	0.561	0.710
	5 KQ	0.577	0.723
	10 KQ	0.577	0.724
Keyquery mit NP	1 NP	0.604	0.648
	2 NP	0.643	0.684
	5 NP	0.669	0.718
Relaxed Keyquery	1 KQ	0.653	0.683
	5 KQ	0.699	0.746
	10 KQ	0.700	0.753
	15 KQ	0.711	0.769
Hybrides Modell	RM3	0.664	0.805
	BM25PRF	0.670	0.814
	Axiom	0.662	0.780
	RM3+BM25PRF	0.662	**0.821**
	RM3+BM25PRF+Axiom	0.653	0.816
Greedy-Query Zählen	5 Terme	0.275	0.279
	10 Terme	0.340	0.324
	5 Terme	0.577	0.724
	10 Terme	0.577	0.724
BM25 RM3	k1=0.9, b=0.4 QW=0.5	0.534 **0.726**	0.699 0.802

Tabelle 4.2: Keyquery-Ansätze mit Feedback-Variante Shortest

Ansatz	Parameter	NDCG@1000	Recall@1000
Keyquery ohne NP	1 KQ	0.559	0.690
	5 KQ	0.575	0.709
	10 KQ	0.574	0.709
Keyquery mit NP	1 NP	0.522	0.558
	2 NP	0.542	0.568
	5 NP	0.558	0.585
Relaxed Keyquery	1 KQ	0.573	0.598
	5 KQ	0.595	0.633
	10 KQ	0.589	0.631
	15 KQ	0.587	0.629
Hybrides Modell	RM3	0.617	0.729
	BM25PRF	0.619	0.732
	Axiom	0.578	0.682
	RM3+BM25PRF	0.618	0.761
	RM3+BM25PRF+Axiom	0.609	0.763
Greedy-Query Zählen	5 Terme	0.254	0.276
	10 Terme	0.323	0.322
	5 Terme	0.573	0.707
	10 Terme	0.575	0.709
BM25 RM3	k1=0.9, b=0.4 QW=0.5	0.534 **0.700**	0.699 **0.785**

Tabelle 4.3: Keyquery-Ansätze mit Feedback-Variante Longest

Ansatz	Parameter	NDCG@1000	Recall@1000
Keyquery ohne NP	1 KQ	0.571	0.715
	5 KQ	0.588	0.728
	10 KQ	0.588	0.728
Keyquery mit NP	1 NP	0.600	0.637
	2 NP	0.635	0.673
	5 NP	0.669	0.710
Relaxed Keyquery	1 KQ	0.680	0.720
	5 KQ	0.724	0.782
	10 KQ	0.727	0.790
	15 KQ	0.723	0.785
Hybrides Modell	RM3	0.674	0.794
	BM25PRF	0.684	0.804
	Axiom	0.666	0.780
	RM3+BM25PRF	0.673	**0.817**
	RM3+BM25PRF+Axiom	0.658	0.813
Greedy-Query Zählen	5 Terme	0.291	0.307
	10 Terme	0.360	0.367
	5 Terme	0.587	0.729
	10 Terme	0.588	0.729
BM25 RM3	k1=0.9, b=0.4 QW=0.5	0.534 **0.732**	0.699 0.810

Tabelle 4.4: Keyquery-Ansätze mit Feedback-Variante Random

Ansatz	Parameter	NDCG@1000	Recall@1000
Keyquery ohne NP	1 KQ	0.554	0.673
	5 KQ	0.613	0.741
	10 KQ	0.614	0.743
Keyquery mit NP	1 NP	0.673	0.754
	2 NP	0.687	0.761
	5 NP	0.700	0.770
Relaxed Keyquery	1 KQ	0.673	0.723
	5 KQ	0.710	0.781
	10 KQ	0.715	0.790
	15 KQ	0.716	0.792
Hybrides Modell	RM3	0.649	0.781
	BM25PRF	0.663	0.791
	Axiom	0.654	0.774
	RM3+BM25PRF	0.651	0.800
	RM3+BM25PRF+Axiom	0.644	0.803
Greedy-Query Zählen	5 Terme	0.375	0.426
	10 Terme	0.487	0.526
	5 Terme	0.612	0.741
	10 Terme	0.616	0.745
BM25 RM3	k1=0.9, b=0.4 QW=0.25	0.534	0.699
		0.726	**0.812**

Tabelle 4.5: Keyquery-Ansätze mit Feedback-Variante Least Difficult

Ansatz	Parameter	NDCG@1000	Recall@1000
Keyquery ohne NP	1 KQ	0.553	0.702
	5 KQ	0.560	0.705
	10 KQ	0.560	0.705
Keyquery mit NP	1 NP	0.470	0.484
	2 NP	0.501	0.507
	5 NP	0.533	0.543
Relaxed Keyquery	1 KQ	0.563	0.567
	5 KQ	0.589	0.611
	10 KQ	0.594	0.620
	15 KQ	0.596	0.626
Hybrides Modell	RM3	0.638	0.777
	BM25PRF	0.647	0.784
	Axiom	0.561	0.624
	RM3+BM25PRF	0.628	**0.802**
	RM3+BM25PRF+Axiom	0.621	0.801
Greedy-Query Zählen	5 Terme	0.217	0.211
	10 Terme	0.285	0.259
	5 Terme	0.560	0.705
	10 Terme	0.560	0.705
BM25 RM3KQ	k1=0.9, b=0.4 QW=0.25	0.534	0.699
		0.699	0.790

Tabelle 4.6: Keyquery-Ansätze mit Feedback-Variante Most Difficult

4.2.1 Anzahl der Keyqueries und der Nounphrasen

Betrachten wir zunächst den Unterschied von Keyqueries ohne Nounphrasen im Vergleich zu Keyqueries mit Nounphrasen, wenn beide Verfahren nur durch eine ODER-Verknüpfung ihre Anfrage bauen. Es ist aus den Zeilen Keyquery ohne NP und Keyquery mit NP der Tabellen 4.1 bis 4.6 abzulesen, dass das Verwenden von Nounphrasen einen starken positiven Einfluss auf den NDCG haben kann, im Fall der Feedback-Variante SRF ergibt sich ein Anstieg um 13.9%. Allerdings zeigen die Feedback-Varianten Longest und Most Difficult, dass das Verwenden der Nounphrasen nicht immer zu einem Anstieg des NDCG führt, sondern abhängig von der gewählten Feedback-Variante ist. Der Anstieg des NDCG lässt sich so erklären, dass die Verwendung von Nounphrasen zusätzlich zu den Keyqueries dafür sorgen, dass die Dokumente weit nach oben gezogen werden, aus denen die Nounphrasen stammen. Denn die extrahierten Nounphrasen werden als Phrase gestellt und somit werden nur die Dokumente gefunden, die diese Nounphrase exakt so enthalten. Es werden somit die Dokumente nach oben gezogen, die als initiales Relevanz-Feedback verwendet wurden. Somit steigt der NDCG@1000 an. Der Abfall des NDCG bei den Feedback-Varianten Longest und Most Difficult hängt damit zusammen, dass in der Implementierung der Nounphrasen-Extraktion nur die ersten 1000 Sätze betrachtet wurden, um die Laufzeit der Durchläufe gering zu halten. Hierbei tritt nun der Fall ein, dass gerade die längsten Dokumente und die am schwierigsten zu findenden Dokumente oftmals mehr als 1000 Sätze haben und somit nicht alle Nounphrasen betrachtet werden konnten. Dies führte zu einer Auswahl von schlechteren Nounphrasen im Vergleich zu Dokumenten bei denen alle Sätze betrachtet wurden.

Es ist weiterhin zu sehen, dass der Recall für alle Feedback-Varianten, außer für Least Difficult, bei Verwendung von Nounphrasen im Vergleich zu dem Verfahren ohne Nounphrasen fiel. Least Difficult hat dabei einen Anstieg von 3.6% im Recall bei Verwendung von Nounphrasen gegenüber dem Verfahren ohne Nounphrasen. Wenn der NDCG steigt, aber der Recall fällt, bedeutet dies, dass zwar mehr relevante Dokumente in den obersten Positionen sind, aber insgesamt die Anzahl der relevanten Dokumente abgenommen hat. Der Abfall des Recalls ist damit zu begründen, dass die Nounphrasen zusätzlich zu den relevanten Dokumenten, aus denen sie extrahiert wurden auch noch Dokumente finden, die zwar die Phrase exakt matchen, aber eigentlich nicht relevant sind. Diese Dokumente haben aber einen hohen Score wegen ihres exakten Matches und verdrängen somit relevante Dokumente am Ende des Rankings, welche einen vergleichsweise niedrigen Score hatten. Bei der Feedback-Variante Least Difficult scheinen die Nounphrasen genügend weitere relevante Dokumente zu finden, damit dieser Effekt nicht eintritt und der Recall sogar steigt. Es ist

weiterhin zu sehen, dass das Erhöhen der Anzahl der Nounphrasen für beide Evaluationsmaße durchgehend förderlich ist. Somit lässt sich hieraus schließen, dass die Verwendung von Nounphrasen dann sinnvoll ist, wenn besonders viel Wert auf die Dokumente der obersten Positionen gelegt wird und Relevanz-Feedback-Dokumente gewählt werden, die nicht zu viele Sätze beinhalten.

Schauen wir nun auf den Einfluss der Anzahl der Keyqueries und betrachten die Zeilen 1 KQ bis 10 KQ für die Tabellen 4.1 bis 4.6. Es ist zu sehen, dass je mehr Keyqueries verwendet werden können, desto besser ist der NDCG für die meisten Feedback-Varianten. Nur die Feedback-Variante Longest zeigt einen leichten Abfall des NDCG um 0.18% bei Erhöhung von 5 auf 10 Keyqueries. Der Recall zeigt ebenfalls eine Verbesserung bei Erhöhen der Anzahl der Keyqueries, für SRF liegt dieser Anstieg bei 4.15% bei Erhöhung von 1 KQ auf 10 KQ. Dies liegt daran, dass die entstehende Anfrage durch mehr Keyqueries komplexer wird und hierdurch eine bessere Retrieval-Performance erreicht wird [3]. Weiterhin ist zu sehen, dass der Anstieg von 1 Keyquery auf 5 Keyqueries einen stärkeren Einfluss auf den NDCG und Recall hat, als die Verwendung von 5 Keyqueries im Vergleich zu 10 Keyqueries. Dies liegt vor allem daran, dass nicht für alle Anfragen auch tatsächlich 10 Keyqueries berechnet werden können und die Komplexität nicht so stark ansteigt. Für einige Feedback-Varianten können wir beobachten, dass sogar der NDCG und der Recall stagniert, wenn mehr Keyqueries verwendet werden.

Es muss also erreicht werden, dass möglichst viele Keyqueries zu einer Anfrage berechnet werden können, wenn der NDCG bzw. der Recall maximiert werden soll. Da es bei einigen Anfragen dazu kam, dass nur wenige Keyqueries bestimmt wurden konnten, untersuchen wir nun die Relaxed Keyqueries aus Kapitel 3.3. Es zeigt sich, dass die Relaxed Keyqueries bei gleichbleibender Schranke der Anzahl der Keyqueries einen besseren NDCG und Recall aufweisen als der soeben diskutierte Keyquery-Ansatz mit Nounphrasen. Die Feedback-Variante Random zeigt einen Anstieg von 8% im NDCG. Es zeigt sich auch hier wieder, dass das Erhöhen der Anzahl der Keyqueries zu einer besseren Retrieval-Performance sowohl für den NDCG als auch für den Recall führt. Hierbei ist zu sehen, dass mehr Keyqueries zwar zu einer höheren Performance im NDCG und Recall führen, aber insbesondere der Performance-Unterschied zwischen einem verwendeten Keyquery und 5 Keyqueries höher ist als bei 5 und 10 Keyqueries oder noch mehr. Dies liegt daran, dass je mehr relaxierte Keyqueries verwendet werden, desto höher ist die Wahrscheinlichkeit dafür, dass diese Keyqueries nicht mehr alle Ziel-Dokumente in die obersten k Positionen bringen. Somit ist ihre Retrieval-Performance schwächer als die von Keyqueries, die tatsächlich die Keyquery-Eigenschaften für alle Ziel-Dokumente erfüllen müssen. Zu sehen ist jedoch, dass die strikten Keyquery-Eigenschaften dazu führen, dass sehr wenige Keyqueries berechnet

werden können und es für die Retrieval-Performance sinnvoll ist, diese Eigenschaften abzuschwächen, sollte eine gewählte Anzahl an Keyqueries noch nicht erreicht sein.

4.2.2 Hybride Suchanfrageerweiterung

Betrachten wir die Zeilen hybrides Modell aus den Tabellen 4.1 bis 4.6 so stellen wir fest, dass das Hybride Modell über alle Parameterkonfigurationen einen hohen Recall-Wert aufweist. Weiterhin zeigt sich, dass der Recall-Wert für das hybride Modell mit Relevanz-Feedback aus BM25PRF und Kombinationen mit BM25PRF den höchsten Recall-Wert über alle Feedback-Varianten aufweist. Für die Feedback-Varianten Longest und Most Difficult zeigt sich, dass der NDCG bei Verwendung des hybriden Modells am höchsten ist, während für die restlichen Feedback-Varianten andere Ansätze einen höheren NDCG erbringen. Der NDCG bleibt aber über alle Feedback-Varianten konstant hoch und kann die Baseline BM25 für die Feedback-Variante Random um 28% schlagen, nicht jedoch die Baseline RM3 mit der besten Parameterkonfiguration. Für den Recall können die Feedback-Varianten Shortest und Random sogar die Baseline RM3 mit der besten Parameterkonfiguration um 2.3% bzw. 0.86% schlagen. Der Vorsprung im Recall gegenüber den anderen Verfahren hat den Hintergrund, dass das hybride Modell mehr Relevanz-Feedback-Dokumente zur Verfügung hat und damit mehr relevante Dokumente im Ranking sind. Zusätzlich zu dem Relevanz-Feedback, welches alle Keyquery-Verfahren bekommen, kommt noch Pseudo-Relevanz-Feedback aus den angegeben Erweiterungsmethoden hinzu. Da hierbei aber auch Dokumente in das Relevanz-Feedback aufgenommen werden, die nicht relevant sind, hängt die Performance dieses Verfahrens stark davon ab, wie viele relevante Dokumente sich tatsächlich in den ersten Positionen der einfließenden Rankings befinden. Wie zu sehen ist, führt das Erhöhen der Menge des Relevanz-Feedbacks nicht zwangsläufig zu einer besseren Performance. So sind die Verfahren, die Relevanz-Feedback aus nur einem Erweiterungsmodell bekommen, in der Regel besser als Verfahren mit mehreren Einflüssen, sowohl im NDCG als auch im Recall. Mögliche Ursache für diese Beobachtung ist, dass bei Verwendung von mehr Pseudo-Relevanz-Feedback die Wahrscheinlichkeit steigt, dass nicht relevante Dokumente in die Menge des Relevanz-Feedbacks aufgenommen werden. Somit werden weniger und qualitativ schlechtere Keyqueries berechnet, da Terme aus nicht relevanten Dokumenten extrahiert wurden. Dies führt dazu, dass relevante Dokumente stärker im Ranking verteilt sind und somit sinkt der NDCG gegenüber den Keyqueries mit 5 Nounphrasen. Es sind also insgesamt im Ranking mehr relevante Dokumente enthalten, diese befinden sich jedoch nicht alle zusammenhängend auf den oberen Positionen.

4.2.3 Bildung einer Suchanfrage

Betrachten wir nun die Zeilen Greedy-Query und Zählen aus den Tabellen 4.1 bis 4.6, die die entsprechenden Verfahren aus Kapitel 3.4 evaluieren. Die Verfahren zur Bildung einer Suchanfrage zeigen, dass der einfache Ansatz über das Greedy-Query schlechter abschneidet als die Baseline BM25 und das sowohl im Bezug auf den Recall als auch auf den NDCG. Eine Erklärung hierfür ist, dass das Anfügen von Termen bis zu einer festgelegten Komplexitätsschranke zwar zu einer leichten Anfrage führt, jedoch die Keyquery-Eigenschaften nicht beachtet. In dessen Folge werden Anfragen produziert, die keine Garantie haben, dass die Ziel-Dokumente in den ersten Top-k Treffern liegen. Weiterhin hat das Anfügen das Problem, dass nicht darauf geachtet wird, dass die Terme die Anfrage gut abdecken. So kann es passieren, dass Terme das Informationsbedürfnis der Anfrage nur teilweise charakterisieren und somit nicht genügend relevante Dokumente finden. Die Ergebnisse zeigen deutlich, dass zu wenig relevante Dokumente gefunden wurden im Vergleich zur Baseline BM25. Eine Erhöhung der Anfrageterme führte zu einer Verbesserung der Evaluationsmaße, bei einigen Ansätzen auch zu Stagnation. Stagnierten die Werte der Evaluationsmaße, so konnten keine Anfragen berechnet werden, die die Länge von 5 Termen überschritten. Da aber voneinander verschiedene Terme nur in endlicher Anzahl verfügbar sind, wird die Erhöhung der Anfrageterme als Möglichkeit der Erhöhung der Retrieval-Performance schnell an ihre Grenzen stoßen.

Das Verfahren Zählen welches die Anzahl der Terme der Keyqueries zählt und anschließend aus diesen Termen neue Keyqueries berechnet, kann die Baseline BM25 um 15.4% im NDCG bei Verwendung der Feedback-Variante Least Difficult schlagen, aber nicht die Baseline RM3. Es zeigt sich, dass das Erhöhen der Terme, die verwendet werden dürfen, zu einer Performance-Steigerung im NDCG und Recall führt. Die Performance dieses Ansatzes liegt im Bereich des Keyquery-Ansatzes ohne Verwendung von Nounphrasen. Erhöht man die Anzahl der zu verwendenden Terme, so wird irgendwann der Punkt erreicht sein, bei dem so viele Terme verwendet werden wie es unterschiedliche Terme gibt und die Performance wird gleich mit dem Keyquery-Ansatz ohne Nounphrasen sein. Es hat sich also gezeigt, dass die Reduktion der Anzahl der Keyqueries über die Verwendung der Top-n am häufigsten auftretenden Terme zu keiner Verbesserung der Retrieval-Performance sowohl im NDCG als auch im Recall gegenüber der einfachen ODER-Verknüpfung von Keyqueries geführt hat. Hintergrund ist, dass die Komplexität der Anfrage reduziert wurde, da weniger Keyqueries verwendet wurden.

Wir haben gesehen, dass es möglich ist, weniger komplexere Anfragen zu generieren als alle Keyqueries mit dem logischen ODER zu verbinden. Dabei

wird jedoch der NDCG und Recall geschwächt. Wird eine leichtere für den menschlichen Nutzer verständliche Anfrage gewünscht, so muss dies zu Lasten der Retrieval-Performance erfolgen.

4.2.4 Gewichtete Keyqueries

Die Tabellen 4.7 bis 4.12 zeigen den Vergleich der Performance von gewichteten Keyqueries mit dem Suchanfrageerweiterungsmodell RM3. Es ist zu sehen, dass für die Feedback-Varianten SRF und Least Difficult sowohl im NDCG als auch im Recall die gewichteten Keyqueries schlechter sind als die Baseline RM3. Wobei hingegen für die Feedback-Varianten Most Difficult und Longest bei einem QW von 0 die gewichteten Keyqueries 64.7% bzw. 24.8% besser als die Baseline RM3 im Bezug auf den NDCG sind. Auch im QW 0.25 sind diese Feedback-Varianten besser im NDCG und Recall als die Baseline RM3. Bei Shortest und Random sind die gewichteten Keyqueries nur für das QW 0 um 7.3% bzw. 4.3% besser als die Baseline RM3 im Bezug auf den NDCG. Dies ist interessant, da gerade bei den Feedback-Varianten, die die niedrigsten Werte im NDCG und Recall für die Keyquery-Ansätze aus den Tabellen 4.1 bis 4.6 hatten, der Vorsprung gegenüber der Baseline RM3 am höchsten ist. Während hingegen für die Feedback-Varianten bei denen höhere Werte für diese Maße berechnet wurden, die Baseline nie übertroffen wird. Scheinbar ist es so, dass für die Feedback-Varianten Longest und Most Difficult von RM3 besonders schlechte Termgewichte vergeben werden, die durch den Ansatz der gewichteten Keyqueries deutlich besser verteilt werden. Interessant ist auch der Fall QW=1, hier ist zu sehen, dass die Werte für eine Feedback-Variante immer gleich sind. Dies ist darauf zurückzuführen, dass der RM3-Algorithmus bei einem QW von 1 die originalen Anfrageterme stellt. Weiterhin ist zu beobachten, dass die besten Werte für beide Ansätze mit einem QW von 0.25 bzw. 0.5 erreicht wurden und bei anderen QW die Performance im NDCG und Recall wieder abflacht. Somit lässt sich schließen, dass offenbar eine Gewichtung der originalen Anfrage von einem Viertel bis zu einer Hälfte die besten Ergebnisse erzielt. Es werden also durch beide Verfahren Erweiterungsterme gefunden, die in Kombination mit den originalen Anfragetermen eine bessere Retrieval-Performance im NDCG und Recall erreichen als die originalen Anfrageterme selbst.

Betrachten wir jetzt die Rankings original, toTop und removed, so stellen wir fest, dass das Nach-Oben-Schieben der relevanten Dokumente einen positiven Effekt auf den NDCG und Recall hat. Das bedeutet also, dass noch nicht alle relevanten Dokumente auf den obersten Positionen im Ranking stehen, denn sonst würde sich der NDCG nicht ändern. Die Änderung im Recall fällt eher gering aus bzw. stagniert und liegt daran, dass die Dokumente zum

großen Teil nur umsortiert werden. Sollten nicht relevante Dokumente zuvor nicht im Ranking gewesen sein, so erscheinen sie jetzt und der Recall steigt leicht. Das removed Ranking zeigt, dass das Entfernen der Ziel-Dokumente zunächst zu einem Einbruch der Retrieval-Performance im NDCG und Recall führt. Dies ist dem geschuldet, dass aktiv relevante Dokumente entfernt wurden. Die Stärke des Einbruches des Recalls zeigt aber, dass der Ansatz der gewichteten Keyqueries nach dem Entfernen der Ziel-Dokumente immer noch relevante Dokumente im Ranking enthält. Zudem befinden sich immer noch recht viele relevante Dokumente auf oberen Positionen im Ranking, da der NDCG zwar etwas eingebrochen ist, aber nicht viel stärker als der Recall. Wir können also sehen, dass der Ansatz zusätzliche relevante Dokumente neben den Ziel-Dokumenten findet und ein gutes aber noch nicht perfektes Ranking erzeugt.

Abbildung 4.1 zeigt wie sich der NDCG für die gewichteten Keyqueries mit 8 bis 11 ausgewählten Termen aus RM3 und RM3 in Abhängigkeit der Menge des Relevanz-Feedbacks ändert für die QW 0 bis 1 und einer Anfragenlänge von 8 Termen für die Feedback-Variante SRF. Exemplarisch ist für jedes QW nur eine Anfragelänge abgebildet, da sich zwischen den verschiedenen Anfragelängen für ein QW keine großen Unterschiede ergeben. Alle erstellten Abbildungen können im Anhang A betrachtet werden. Zu sehen ist, dass die Erhöhung der Menge des Relevanz-Feedbacks für fast alle aufgetragenen Verfahren zu einer Steigerung des NDCG führt. Das orange eingetragene Verfahren mit 8 ausgewählten Termen aus RM3 zeigt den stärksten negativen Einfluss, wenn das Relevanz-Feedback von 4 auf 5 Dokumente erhöht wird. Bei allen anderen Erhöhungen ist der Einfluss aber positiv. Insbesondere bei einem QW von 0 ist der negative Einfluss auf den NDCG sehr stark und flacht mit zunehmenden QW immer weiter ab. Betrachten wir die Plots zu den Anfragelängen 5 bis 7 (im Anhang A), so ist zu erkennen, dass der vermeintliche negative Einfluss auf den NDCG bei Erhöhung des Relevanz-Feedbacks hier nicht auftritt. Die Plots für die Anfragelängen 9 bis 10 zeigen ähnliche negative Einflüsse für die Verfahren mit 9 und 10 ausgewählten Termen aus RM3. Dies lässt sich damit begründen, dass wenn Anfragelänge und ausgewählte Termanzahl gleich sind, nur eine Teilmenge im Algorithmus die vorgegebene Kardinalität besitzt und es somit keine konkurrierenden Anfragen gibt, die über den NDCG ausgewählt werden.

Somit tritt in diesem Randfall tatsächlich ein negativer Einfluss auf, wenn das Wissen in Form der Relevanz-Feedback-Dokumente erhöht wird. Ebenso folgt daraus, dass es im Allgemeinen förderlich für den NDCG ist, die Menge des Relevanz-Feedbacks zu erhöhen, da alle anderen Verfahren eine Steigerung im NDCG erfahren haben. Mit zunehmenden QW ist zu erkennen, dass die Streuung der gewichteten Keyqueries untereinander immer weiter abnimmt,

		QW=0		QW=0.25		QW=0.5		QW=0.75		QW=1	
Ansatz	Ranking	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall
RM3	original	0.652	0.764	0.680	0.802	0.665	0.800	0.613	0.760	0.534	0.699
	toTop	**0.695**	**0.768**	**0.711**	**0.802**	**0.701**	**0.800**	**0.668**	**0.760**	**0.623**	**0.699**
	removed	0.449	0.577	0.466	0.612	0.445	0.609	0.399	0.569	0.343	0.508
RM3KQ	original	0.649	0.760	0.680	0.800	0.653	0.784	0.602	0.743	0.534	0.699
	toTop	0.686	0.760	0.708	0.800	0.694	0.788	0.660	0.746	**0.623**	**0.699**
	removed	0.437	0.569	0.459	0.609	0.436	0.597	0.389	0.555	0.343	0.508

Tabelle 4.7: Gewichtete Keyqueries mit Feedback-Variante SRF

		QW=0		QW=0.25		QW=0.5		QW=0.75		QW=1	
Ansatz	Ranking	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall
RM3	original	0.560	0.643	0.649	0.762	0.642	0.770	0.597	0.746	0.534	0.699
	toTop	0.644	0.678	**0.724**	**0.792**	**0.726**	**0.802**	**0.708**	**0.786**	**0.677**	**0.748**
	removed	0.418	0.479	0.510	0.593	0.511	0.603	0.485	0.587	0.442	0.550
RM3KQ	original	0.586	0.727	0.582	0.724	0.558	0.707	0.542	0.699	0.534	0.699
	toTop	**0.691**	**0.764**	0.693	0.766	0.683	0.754	0.678	0.749	**0.677**	**0.748**
	removed	0.463	0.565	0.466	0.567	0.454	0.554	0.446	0.550	0.442	0.550

Tabelle 4.8: Gewichtete Keyqueries mit Feedback-Variante Shortest

bis sie beim QW von 1 für alle Verfahren auf 0 gesunken ist und der NDCG gleich ist. Dies wurde bereits aus den Tabellen 4.7 bis 4.12 ersichtlich. Anscheinend nähern sich die berechneten Termgewichte aus RM3 für die verschiedenen Verfahren immer weiter an. Wir sehen zudem, dass gerade für das QW 0 der Einfluss der Menge des Relevanz-Feedbacks besonders stark ist und es einen großen Unterschied macht, ob 1 oder 2 Dokumente verwendet werden. Für zunehmende Query-Weights wird dieser Einfluss immer marginaler. Dies lässt sich damit begründen, dass der Einfluss der Terme aus dem Relevanz-Feedback bei steigendem QW immer geringer wird, denn die Gewichtung verschiebt sich dann immer mehr in Richtung originaler Anfrageterme und weg von den Termen aus dem Relevanz-Feedback. Weiterhin können wir sehen, dass für die QW 0.25 und 0.5 die höchsten Werte für den NDCG erreicht werden und dass nur für das QW 0 und 0.25 RM3 durch die gewichteten Keyqueries geschlagen werden kann. Für die gewichteten Keyqueries ist bei der Feedback-Variante SRF der Einfluss der Terme aus dem Relevanz-Feedback stärker als für RM3, welches erst bei einem QW von 0.5 an die Werte der gewichteten Keyqueries für die QW 0 bzw. 0.25 herankommt. Die gewichteten Keyqueries nutzen also das bereitgestellte Relevanz-Feedback viel besser aus als RM3.

		QW=0		QW=0.25		QW=0.5		QW=0.75		QW=1	
Ansatz	Ranking	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall
RM3	original	0.474	0.545	0.597	0.706	0.629	0.761	0.595	0.743	0.534	0.699
	toTop	0.551	0.575	0.661	0.730	**0.700**	**0.785**	**0.693**	**0.777**	**0.667**	**0.741**
	removed	0.298	0.376	0.426	0.531	0.475	0.586	0.468	0.578	0.434	0.542
RM3KQ	original	0.576	0.708	0.581	0.719	0.562	0.710	0.543	0.702	0.534	0.699
	toTop	**0.668**	**0.741**	**0.678**	**0.754**	0.675	0.751	0.670	0.745	**0.667**	**0.741**
	removed	0.436	0.542	0.448	0.555	0.446	0.552	0.438	0.545	0.434	0.542

Tabelle 4.9: Gewichtete Keyqueries mit Feedback-Variante Longest

		QW=0		QW=0.25		QW=0.5		QW=0.75		QW=1	
Ansatz	Ranking	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall
RM3	original	0.583	0.669	0.659	0.768	0.658	0.788	0.609	0.755	0.534	0.699
	toTop	0.656	0.696	**0.721**	**0.789**	**0.732**	**0.810**	**0.709**	**0.787**	**0.670**	**0.740**
	removed	0.424	0.497	0.504	0.590	0.517	0.611	0.489	0.588	0.437	0.541
RM3KQ	original	0.587	0.721	0.584	0.723	0.567	0.714	0.542	0.702	0.534	0.699
	toTop	**0.684**	**0.750**	0.687	0.757	0.682	0.751	0.673	0.742	**0.670**	**0.740**
	removed	0.455	0.551	0.460	0.558	0.454	0.552	0.441	0.543	0.437	0.541

Tabelle 4.10: Gewichtete Keyqueries mit Feedback-Variante Random

		QW=0		QW=0.25		QW=0.5		QW=0.75		QW=1	
Ansatz	Ranking	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall
RM3	original	0.662	0.772	0.689	0.810	0.672	0.808	0.617	0.766	0.534	0.699
	toTop	**0.707**	**0.777**	**0.726**	**0.812**	**0.718**	**0.810**	**0.686**	**0.772**	**0.640**	**0.708**
	removed	0.457	0.578	0.478	0.613	0.464	0.611	0.422	0.573	0.366	0.509
RM3KQ	original	0.661	0.772	0.671	0.793	0.642	0.775	0.593	0.740	0.534	0.699
	toTop	0.706	0.776	0.716	0.798	0.701	0.782	0.672	0.749	**0.640**	**0.708**
	removed	0.458	0.577	0.466	0.599	0.444	0.584	0.406	0.550	0.366	0.509

Tabelle 4.11: Gewichtete Keyqueries mit Feedback-Variante Least Difficult

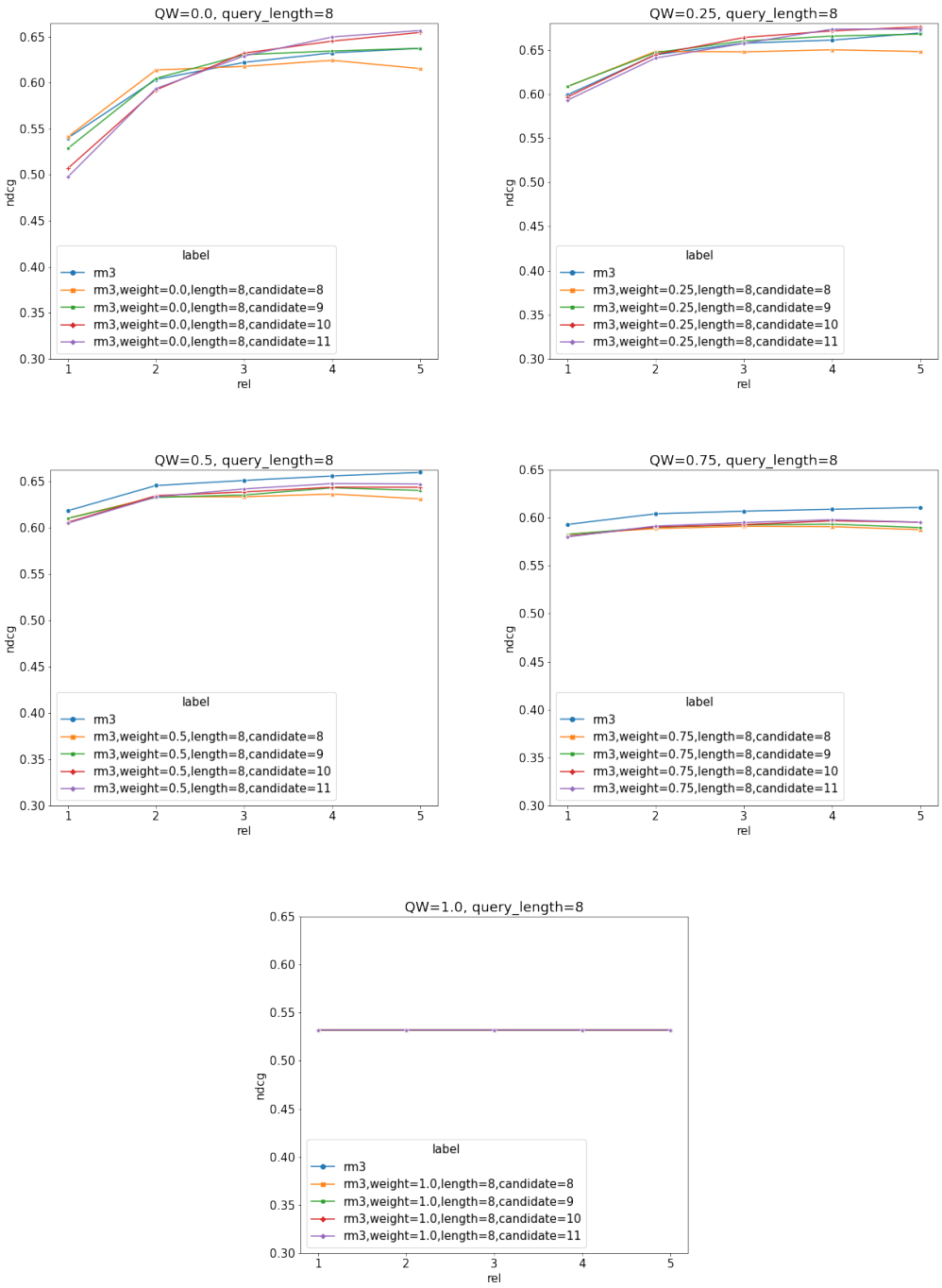


Abbildung 4.1: Änderung des NDCG für die gewichteten Keyqueries und RM3 mit Feedback-Variante SRF in Abhängigkeit der Menge des Relevanz-Feedbacks für verschiedene Query-Weights (QW) und einer Anfragenlänge von 8 Termen

		QW=0		QW=0.25		QW=0.5		QW=0.75		QW=1	
Ansatz	Ranking	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall
RM3	original	0.267	0.314	0.543	0.668	0.571	0.710	0.557	0.708	0.534	0.699
	toTop	0.420	0.392	0.665	0.740	**0.697**	0.783	**0.699**	**0.787**	**0.692**	**0.785**
	removed	0.159	0.194	0.447	0.542	0.489	0.584	0.492	0.589	0.479	0.587
RM3KQ	original	0.569	0.706	0.568	0.712	0.549	0.703	0.539	0.699	0.534	0.699
	toTop	**0.692**	**0.782**	**0.699**	**0.790**	0.694	**0.786**	0.692	0.785	**0.692**	**0.785**
	removed	0.480	0.574	0.492	0.592	0.484	0.587	0.481	0.587	0.479	0.587

Tabelle 4.12: Gewichtete Keyqueries mit Feedback-Variante Most Difficult

4.2.5 Einfluss der Art des Relevanz-Feedbacks

Betrachten wir nun wie sich die Art des Relevanz-Feedbacks auf den NDCG auswirkt. Aus Tabelle 4.13 können wir entnehmen, dass der bestmögliche NDCG-Wert für die Feedback-Varianten SRF, Shortest, Random und Least Difficult aus dem Relaxed Keyquery resultiert und sich im Bereich von 0.700 bis 0.727 bewegt. Wobei Random am besten abgeschnitten hat und SRF am schlechtesten. Für die Feedback-Varianten Longest und Most Difficult können wir sehen, dass die besten Ergebnisse aus dem hybriden Modell resultieren und um 10% schlechter im NDCG sind, als die Varianten mit den besten Ergebnissen aus dem Relaxed Keyquery. Es zeigt sich also, dass die Feedback-Varianten Longest und Most Difficult bessere Ergebnisse erzielt haben, wenn sie die zusätzlichen Dokumente aus den Erweiterungsmethoden bekommen hatten. Für die Feedback-Varianten SRF, Shortest, Random und Least Difficult war dieser Einfluss nicht so stark und diese Verfahren haben stärker davon profitiert, dass sie die Keyquery-Eigenschaften relaxieren konnten. Vermutlich können bei den Feedback-Varianten Longest und Most Difficult weniger gute Keyqueries gebildet werden, wenn nur die initialen Feedback-Dokumente aus der zugehörigen Simulation verwendet wurden und sie benötigten noch zusätzliche Dokumente, um gute Keyqueries zu bilden. Die anderen Feedback-Varianten konnten auf den initialen Feedback-Dokumenten bessere Keyqueries im Bezug auf den NDCG errechnen. Diese Verfahren konnten durch die nicht relevanten Dokumente, die durch das hybride Modell mit in die Menge des Relevanz-Feedbacks kommen, keine besseren Keyqueries errechnen. Offensichtlich ist der vorgestellte Keyquery-Ansatz empfindlich gegenüber nicht relevanten Dokumenten. Wir können außerdem sehen, dass für keinen Ansatz der tabellierten Feedback-Varianten der NDCG die zugehörige Baseline aus den gewichteten Keyqueries schlagen konnte, aber die Baseline BM25 wurde für jeden Ansatz um bis zu 36.1% bei Verwendung der Feedback-Variante Random geschlagen.

In Tabelle 4.14 können wir erkennen, dass die besten Ansätze im Bezug

Feedback-Variante	Ansatz	Parameter	NDCG
SRF	Relaxed Keyquery	15 KQ	0.700
	BM25	k1=0.9, b=0.4	0.534
	RM3	QW=0.25	**0.711**
Shortest	Relaxed Keyquery	15 KQ	0.711
	BM25	k1=0.9, b=0.4	0.534
	RM3	QW=0.5	**0.726**
Longest	Hybrides Modell	RM3+BM25PRF	0.618
	BM25	k1=0.9, b=0.4	0.534
	RM3	QW=0.5	**0.700**
Random	Relaxed Keyquery	10 KQ	0.727
	BM25	k1=0.9, b=0.4	0.534
	RM3	QW=0.5	**0.732**
Least Difficult	Relaxed Keyquery	15 KQ	0.716
	BM25	k1=0.9, b=0.4	0.534
	RM3	QW=0.25	**0.726**
Most Difficult	Hybrides Modell	RM3+BM25PRF	0.628
	BM25	k1=0.9, b=0.4	0.534
	RM3KQ	QW=0.25	**0.699**

Tabelle 4.13: Beste Ansätze für den NDCG aus den Tabellen 4.1 bis 4.6

auf den Recall für alle Feedback-Varianten aus dem hybriden Modell resultieren. Weiterhin ist zu bemerken, dass diese Werte immer mit der Erweiterungsmethode BM25PRF oder in Kombination mit diesem Modell entstanden sind. Der hohe Recall für das hybride Verfahren wurde bereits in 4.2.2 besprochen und ist vor allem auf die erhöhte Menge an Relevanz-Feedback als Eingabe für den Keyquery-Ansatz zurückzuführen. Die Recall-Werte für die Feedback-Varianten Shortest und Random sind dabei besonders hoch und schaffen es sogar die zugehörige Baseline RM3 mit einem Query-Weight von 0.5 um bis zu 2.3% zu übertreffen. Die Feedback-Variante SRF ist noch einmal ein kleines Stück schlechter als die beiden eben betrachteten Feedback-Varianten und die Feedback-Varianten Most Difficult und Longest bilden das Schlusslicht im Bezug auf den Recall. Wir können also abermals erkennen, dass die Feedback-Varianten Most Difficult und Longest deutlich schlechter abschneiden als die anderen Feedback-Varianten, dies hat den Hintergrund, dass zum einen bei der Nounphrasen-Extraktion bei diesen Verfahren besonders häufig der Grenzwert für die betrachteten Sätze (1000 Sätze) erreicht wird. Zum anderen scheint es schwierig zu sein aus den zugehörigen Relevanz-Feedback-Dokumenten gute Terme zu extrahieren, denn das etablierte Verfahren RM3 hat hierbei ebenso niedrigere Evaluationswerte geliefert (vgl. Kapitel 4.2.4) als für andere Feedback-Varianten. Aus weniger guten Termen ist es oftmals schwer gute und viele Keyqueries zu berechnen.

Feedback-Variante	Ansatz	Parameter	Recall
SRF	Hybrides Modell	BM25PRF	0.789
	BM25	k1=0.9, b=0.4	0.699
	RM3	QW=0.25	**0.802**
Shortest	Hybrides Modell	RM3+BM25PRF	**0.821**
	BM25	k1=0.9, b=0.4	0.699
	RM3	QW=0.5	0.802
Longest	Hybrides Modell	RM3+BM25PRF+Axiom	0.629
	BM25	k1=0.9, b=0.4	0.699
	RM3	QW=0.5	**0.785**
Random	Hybrides Modell	RM3+BM25PRF	**0.817**
	BM25	k1=0.9, b=0.4	0.699
	RM3	QW=0.5	0.810
Least Difficult	Hybrides Modell	RM3+BM25PRF+Axiom	0.803
	BM25	k1=0.9, b=0.4	0.699
	RM3	QW=0.25	**0.812**
Most Difficult	Hybrides Modell	RM3+BM25PRF	0.628
	BM25	k1=0.9, b=0.4	0.699
	RM3KQ	QW=0.25	0.790

Tabelle 4.14: Beste Ansätze für den Recall aus den Tabellen 4.1 bis 4.6

4.3 Implizites Relevanz-Feedback

In Tabelle 4.15 ist die Retrieval-Performance des Keyquery-Ansatzes mit implizitem Relevanz-Feedback, wie in Kapitel 3.7 beschrieben, zu sehen. Dabei werden die Evaluationsmaße NDCG@1000 und Recall@1000 angegeben. Als Baseline wird BM25 mit den Parametern k1=0.9 und b=0.4 betrachtet. Zudem sind die Erweiterungsmethoden RM3, BM25PRF und Axiomatic ohne explizites Relevanz-Feedback aufgetragen. Der Parameter n Doc(s) gibt an, wie viele Dokumente jeweils aus dem Pseudo-Relevanz-Feedback entnommen wurden. Zu sehen ist, dass der Recall größer wird, je mehr Verfahren Pseudo-Relevanz-Feedback beitragen. Dies liegt daran, dass mehr Verfahren, die Pseudo-Relevanz-Feedback beisteuern, bedeuten, dass auch mehr relevante Dokumente mit in das Ranking aufgenommen werden. Im Bezug auf den Recall kann die Baseline BM25 um 2.1% geschlagen werden. Gleichzeitig steigt aber auch die Wahrscheinlichkeit, dass nicht relevante Dokumente aufgenommen werden, ähnlich wie bei dem hybriden Modell. Dies hat zur Folge, dass der NDCG nur sehr wenig steigt bzw. sogar abfällt, wenn die Menge des Pseudo-Relevanz-Feedbacks erhöht wird. Keiner der Parameterkonfigurationen schafft es die Baseline BM25 im Bezug auf den NDCG zu schlagen.

Der Einfluss der Anzahl der verwendeten Dokumente aus dem Pseudo-Relevanz-Feedback zeigt, dass das Erhöhen dieser Menge zu einem besseren Recall führt, aber der NDCG sich kaum verändert. Es befinden sich zwar mehr relevante Dokumente im Ranking, aber ihre Positionierung verbessert

sich nicht. Dies lässt sich damit erklären, dass bei einer höheren Menge an Relevanz-Feedback, welches in den Keyquery-Ansatz gegeben wird, sich automatisch mehr relevante Dokumente im Ranking befinden. Aber die berechneten Keyqueries sorgen nicht dafür, dass dieser Vorteil weiter ausgebaut wird und diese weiteren relevanten Dokumente nach oben gezogen werden.

Die vorgeschlagene Methode Pseudo-Relevanz-Feedback aus den Erweiterungsmethoden RM3, BM25PRF und Axiomatic zu nutzen, hat zu einem Verfahren geführt, dass ohne explizites Relevanz-Feedback mit BM25 im NDCG und Recall mithalten kann. Es wurde also gezeigt, dass es möglich ist, Keyqueries auch ohne explizites Relevanz-Feedback zu berechnen und dabei mit einer etablierten Methode mithalten zu können, die ebenfalls kein explizites Relevanz-Feedback verwendet. Es ist aber eine deutliche Verschlechterung im NDCG und Recall gegenüber dem ähnlich gelagerten hybriden Modell zu sehen. Dies rührt daher, dass die Erweiterungsmethoden im hybriden Modell explizites Relevanz-Feedback erhalten haben, während beim Ansatz mit impliziten Relevanz-Feedback nur Pseudo-Relevanz-Feedback verwendet wurde. Dadurch ist das Ranking aus dem die Relevanz-Feedback-Dokumente für die Keyquery-Berechnung genommen werden mit weniger relevanten Dokumenten in den ersten Positionen versehen. Somit werden weniger relevante Dokumente für den impliziten Ansatz verwendet. Auch ist zu sehen, dass die Verfahren RM3, BM25PRF und Axiomatic nicht geschlagen werden können, obwohl dieser Ansatz auf ihren Rankings basiert.

Relevanz-Feedback aus					
RM3	BM25PRF	Axiom	Parameter	NDCG@1000	Recall@1000
✓	x	x	1 Doc	0.421	0.534
x	✓	x	1 Doc	0.410	0.520
x	x	✓	1 Doc	0.420	0.529
✓	✓	x	1 Doc	0.474	0.605
✓	✓	✓	1 Doc	0.506	0.655
✓	x	x	2 Docs	0.472	0.608
x	✓	x	2 Docs	0.462	0.587
x	x	✓	2 Docs	0.467	0.596
✓	✓	x	2 Docs	0.519	0.672
✓	✓	✓	2 Docs	0.527	0.685
✓	x	x	3 Docs	0.495	0.638
x	✓	x	3 Docs	0.486	0.625
x	x	✓	3 Docs	0.498	0.639
✓	✓	x	3 Docs	0.522	0.687
✓	✓	✓	3 Docs	0.533	0.704
✓	x	x	4 Docs	0.516	0.678
x	✓	x	4 Docs	0.515	0.665
x	x	✓	4 Docs	0.517	0.680
✓	✓	x	4 Docs	0.530	0.692
✓	✓	✓	4 Docs	0.530	0.700
✓	x	x	5 Docs	0.523	0.684
x	✓	x	5 Docs	0.524	0.683
x	x	✓	5 Docs	0.525	0.686
✓	✓	x	5 Docs	0.529	0.692
✓	✓	✓	5 Docs	0.533	0.714
	BM25		k1=0.9, b=0.4	0.534	0.699
	RM3		Ohne RF	0.573	0.755
	BM25PRF		Ohne RF	0.576	0.762
	Axiom		Ohne RF	**0.577**	**0.768**

Tabelle 4.15: Performance des Ansatzes mit implizitem Relevanz-Feedback

Kapitel 5

Diskussion und Schlussfolgerungen

In dieser Arbeit wurde betrachtet wie Keyqueries zur Erweiterung von Suchanfragen mit explizitem Relevanz-Feedback genutzt werden können. Dabei wurde gesehen, dass die Keyqueries im Gegensatz zu bisherigen Erweiterungsmethoden auf das Ranking der erzeugten Anfragen achten. Außerdem erzeugen Keyqueries minimale Anfragen und garantieren, dass alle Dokumente des initialen Relevanz-Feedbacks in den Top-k Positionen des neuen Rankings liegen. Solche Garantien versprechen etablierte Erweiterungsmethoden wie RM3, BM25PRF oder Axiomatic nicht.

Des Weiteren wurde gesehen, dass die Art und die Menge der Relevanz-Feedback-Dokumente ein wichtiger Parameter für einen Keyquery-basierten Erweiterungsansatz ist. Gerade die Verdopplung der Menge des Relevanz-Feedbacks von einem aus zwei Dokumente hatte einen starken positiven Einfluss auf den NDCG. Es war zu sehen, dass das Erhöhen der Menge des Relevanz-Feedbacks auf bis zu 5 Dokumente in fast jeder Parameterkonfiguration der gewichteten Keyqueries zu einer Verbesserung des NDCG geführt hat. Weiterhin war zu sehen, dass die relevanten Dokumente, die aus dem Rankingmodell BM25 stammten, von anderen Systemen oft gefunden wurden oder zufällig sortiert wurden, einen besonders positiven Einfluss auf die Performance im NDCG und Recall hatten. Während relevante Dokumente, die nach ihrer Länge sortiert waren oder danach wie selten sie von anderen Systemen gefunden wurden, einen negativen Einfluss auf den Ansatz hatten.

Auch zu sehen war, dass die Auswahl von Keyquery-Kandidaten einen positiven Einfluss auf die Performance im NDCG und Recall des Ansatz hatte. Es wurde festgestellt, dass je weniger Keyqueries verwendet wurden, desto schlechter wurde der NDCG und der Recall. Dies deckt sich mit den Beobachtungen von Belkin et al., die beobachteten, dass eine höhere Komplexität einer Anfrage auch zu einer besseren Performance führte. Gerade die relaxierten Keyqueries waren für fast alle Feedback-Varianten die besten Ansätze im Bezug auf den

NDCG und konnten die Baseline BM25 um bis zu 36% schlagen. Ebenso zeigte sich, dass das Extrahieren von Nounphrasen und eine anschließende Kombination von Nounphrasen mit Keyqueries für die meisten Feedback-Varianten eine Verbesserung im NDCG und Recall erzielte. Nur für die Feedback-Varianten Longest und Most Difficult wurde dabei eine Verschlechterung der Retrieval-Performance festgestellt.

Die Verfahren zur Erzeugung einer Suchanfrage haben gezeigt, dass es am besten ist, so viele Keyqueries und Nounphrasen wie möglich miteinander zu kombinieren, um die Komplexität zu erhöhen. Sowohl der Ansatz für einen Greedy-Algorithmus als auch über das Zählen der Anfrageterme und anschließendes Neuberechnen von Keyqueries hat zu keinem Verfahren geführt, welches in der Lage ist an den NDCG oder Recall einer ODER-Verknüpfung von Keyqueries heranzukommen. Da beim Greedy-Ansatz keine Keyqueries entstehen, weil beim Anfügen der neuen Terme nicht auf die Keyquery-Eigenschaften geachtet wird, kann dieser Ansatz nicht einmal die Baseline BM25 schlagen. Das Neuberechnen von Keyqueries mit den am häufigsten auftretenden Termen lässt Keyqueries entstehen, seine Performance im NDCG und Recall reicht aber nur um die Baseline BM25 um 15.4% zu schlagen. Das Vereinfachen der Anfragen hat über diese Verfahren also nicht gut funktioniert. Hier besteht noch Forschungsbedarf.

Die Einführung von hybriden Suchanfrageerweiterungen hat gezeigt, dass es möglich ist, Erweiterungsmodelle miteinander zu kombinieren und die erzeugten Rankings als erweitertes Relevanz-Feedback für die Berechnung von Keyqueries zu nutzen. Dabei erzeugte dieses Verfahren einen besonders hohen Recall über alle Feedback-Varianten. Hiermit war es auch möglich eine Berechnung von Keyqueries ohne explizites Relevanz-Feedback durchzuführen, wenngleich die Performance im NDCG und Recall dieses Verfahrens deutlich unter dem der Keyquery-Ansätze mit expliziten Relevanz-Feedbacks liegt.

Besonders beachtenswert war die Einführung von gewichteten Keyqueries auf Grundlage der Term-Gewicht-Paare aus dem Erweiterungsmodell RM3. Hierbei gelang es durch die Anwendung von Keyqueries neue Anfragen mit den Termen aus RM3 zu generieren und die Gewichte neu zu normalisieren. Daraus resultierte eine bessere Performance gegenüber RM3 im Bezug auf das Evaluationsmaß NDCG für die Feedback-Varianten Longest und Most Difficult mit ausgewählten QW. Für die restlichen Feedback-Varianten waren die gewichteten Keyqueries leicht schlechter als die Baseline RM3. Es hat sich aber gezeigt, dass es möglich ist, kurze und performante Anfragen auf Grundlage von Keyqueries mit Termgewichten zu erzeugen. Auch zeigte sich, dass die gewichteten Keyqueries die Informationen aus dem Relevanz-Feedback besser ausnutzen können als zum Beispiel RM3.

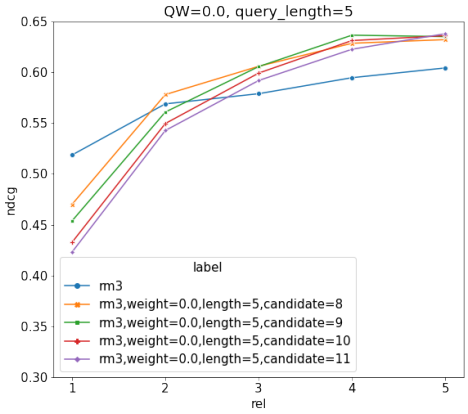
Im Laufe der Arbeit zeigte sich, dass Keyqueries in der Lage waren das

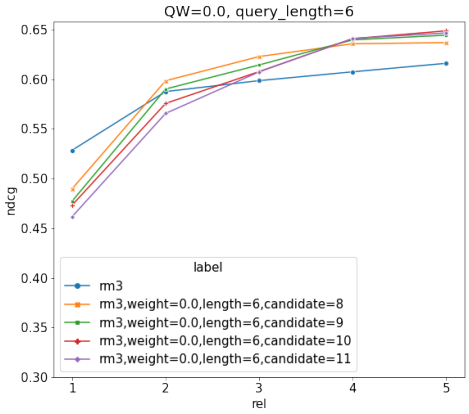
Rankingmodell BM25 zu schlagen und in Kombination mit Nounphrasen oder mit anderen Erweiterungsmethoden war es sogar möglich etablierte Methoden zur Suchanfrageerweiterung mit expliziten Relevanz-Feedback in einigen Parameterkonfigurationen zu übertreffen. Dies zeigt eindrucksvoll die Vorteile von Keyqueries gegenüber herkömmlichen Erweiterungsmodellen. Das Achten auf die Rankings der durch die Keyqueries erzeugten Suchanfragen und das Minimalitätskriterium sorgen für eine performante Anfrage sowohl im NDCG als auch im Recall. Diese Anfragen ist so einfach wie möglich und erzielen über die betrachteten Parameterkonfigurationen etwa gleich gute Ergebnisse im NDCG und Recall wie etablierte Erweiterungsmethoden. Ebenso enthalten die Anfragen keine redundante oder überflüssige Terme. Es zeigt sich also, dass die vorgeschlagenen Ansätze zur Verbesserung von Keyqueries in weiten Teilen zu Erfolgen geführt haben. Auch wurden wertvolle Informationen gewonnen wie Keyqueries auf Dokumente aus unterschiedlichen Feedback-Varianten reagieren und wie sich der Einfluss der Menge des Relevanz-Feedbacks darstellt. Ebenso ist klar geworden, dass eine leichte Anfrage aus Keyqueries mit Termgewichten erfolgsversprechender ist als ohne solche Gewichte. Nicht erfolgreich hingegen war das Erstellen von leichten Anfragen ohne Termgewichte und ein Keyquery-Verfahren, welches ohne explizites Relevanz-Feedback auskommt.

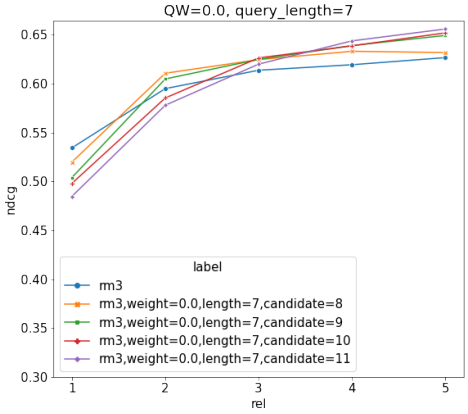
Auf Grundlage dieser Rückschläge lassen sich weitere mögliche Forschungsgegenstände formulieren. Zum einen steht die Frage im Raum wie ein robustes Verfahren entwickelt werden kann, so dass Keyqueries auch ohne explizites Relevanz-Feedback gleich gut oder sogar besser performen als RM3, BM25PRF oder Axiomatic bezüglich des NDCG bzw. Recalls. Des Weiteren sollte genauer untersucht werden wie empfindlich die Berechnung von Keyqueries gegenüber nicht relevanten Dokumenten ist. Auch die Frage warum gerade die Feedback-Varianten, die im NDCG und Recall schlecht abschnitten, die Baseline RM3 im Fall der gewichteten Keyqueries schlagen konnten, ist noch nicht vollständig geklärt. Ebenso wurde nicht untersucht wie gewichtete Keyqueries auf Basis von BM25PRF oder Axiomatic verhalten. Im Bereich der Keyqueries und insbesondere in der Erweiterung von Suchanfragen mit ihnen gibt es also noch viele spannende Fragestellungen zu klären.

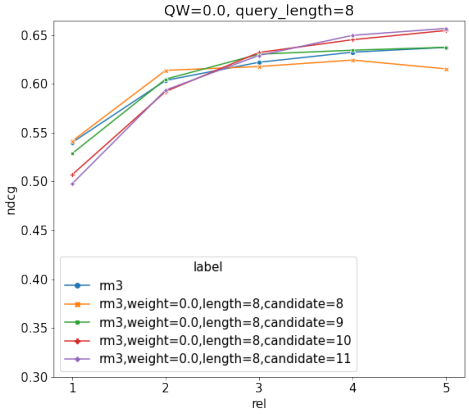
Anhang A

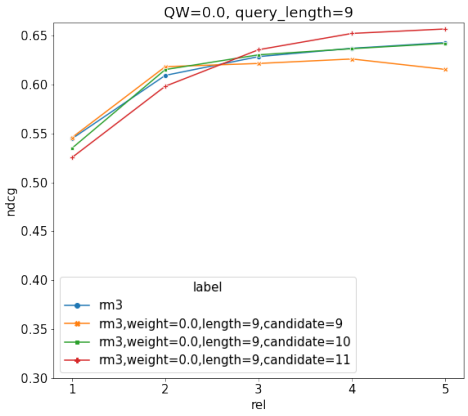
Vollständige Auflistung der Plots

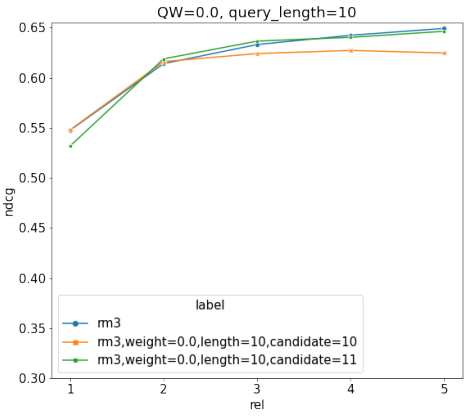


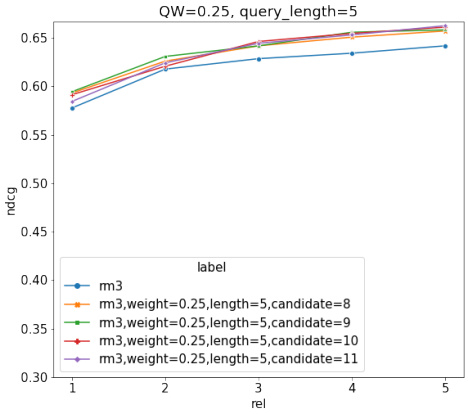


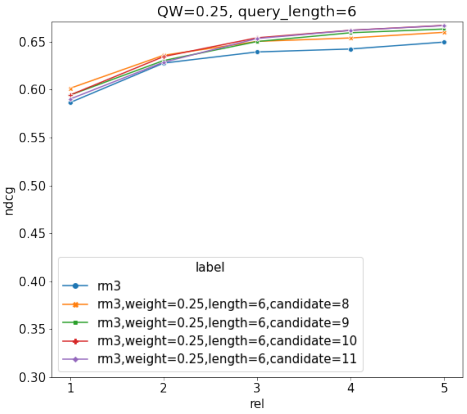


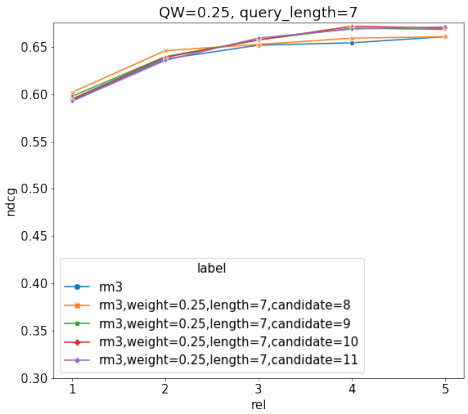


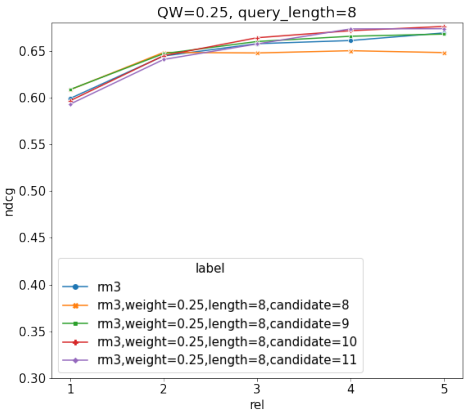

QW=0.0, query_length=9
ndcg
rel
label
rm3
rm3,weight=0.0,length=9,candidate=9
rm3,weight=0.0,length=9,candidate=10
rm3,weight=0.0,length=9,candidate=11

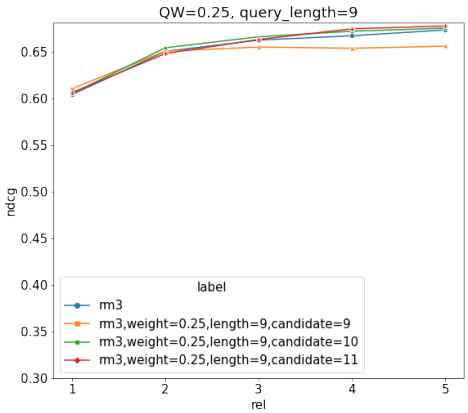

QW=0.0, query_length=10
ndcg
rel
label
rm3
rm3,weight=0.0,length=10,candidate=10
rm3,weight=0.0,length=10,candidate=11

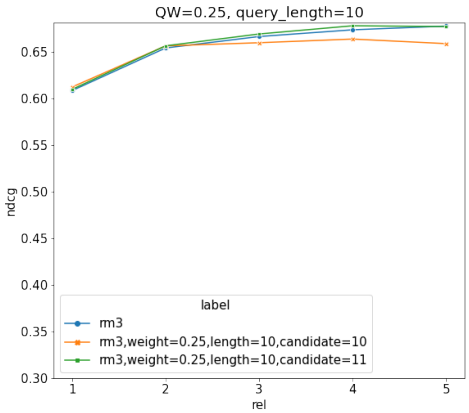

QW=0.25, query_length=5
ndcg
rel
label
rm3
rm3,weight=0.25,length=5,candidate=8
rm3,weight=0.25,length=5,candidate=9
rm3,weight=0.25,length=5,candidate=10
rm3,weight=0.25,length=5,candidate=11

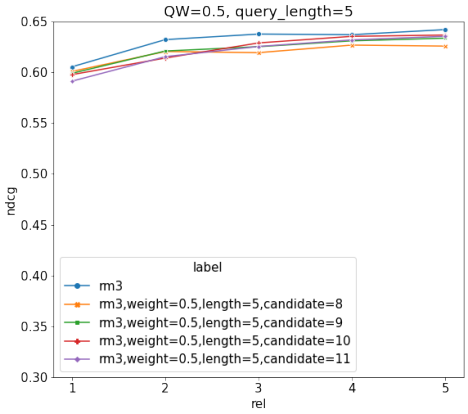

QW=0.25, query_length=6
ndcg
rel
label
rm3
rm3,weight=0.25,length=6,candidate=8
rm3,weight=0.25,length=6,candidate=9
rm3,weight=0.25,length=6,candidate=10
rm3,weight=0.25,length=6,candidate=11

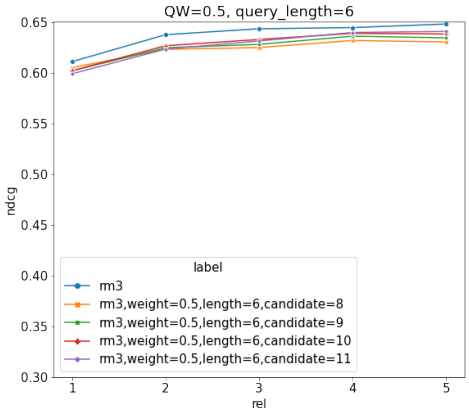

QW=0.25, query_length=7
ndcg
rel
label
rm3
rm3,weight=0.25,length=7,candidate=8
rm3,weight=0.25,length=7,candidate=9
rm3,weight=0.25,length=7,candidate=10
rm3,weight=0.25,length=7,candidate=11

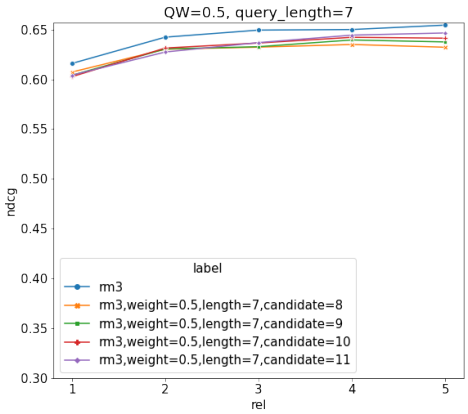

QW=0.25, query_length=8
ndcg
rel
label
rm3
rm3,weight=0.25,length=8,candidate=8
rm3,weight=0.25,length=8,candidate=9
rm3,weight=0.25,length=8,candidate=10
rm3,weight=0.25,length=8,candidate=11

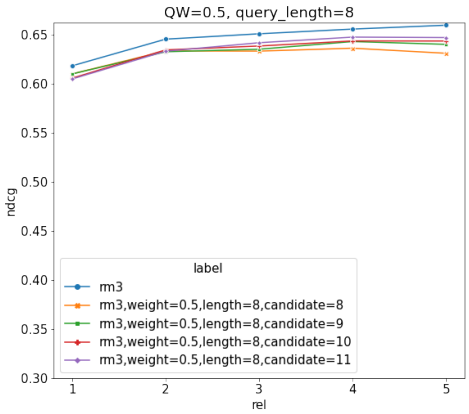

QW=0.25, query_length=9
ndcg
rel
label
rm3
rm3,weight=0.25,length=9,candidate=9
rm3,weight=0.25,length=9,candidate=10
rm3,weight=0.25,length=9,candidate=11

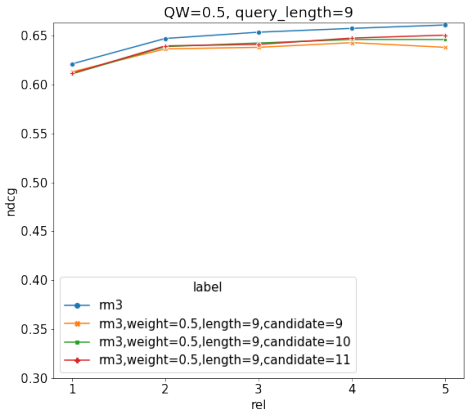

QW=0.25, query_length=10
ndcg
rel
label
rm3
rm3,weight=0.25,length=10,candidate=10
rm3,weight=0.25,length=10,candidate=11

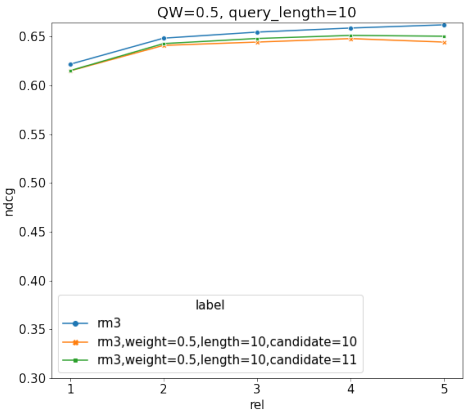

QW=0.5, query_length=5
ndcg
rel
label
rm3
rm3,weight=0.5,length=5,candidate=8
rm3,weight=0.5,length=5,candidate=9
rm3,weight=0.5,length=5,candidate=10
rm3,weight=0.5,length=5,candidate=11

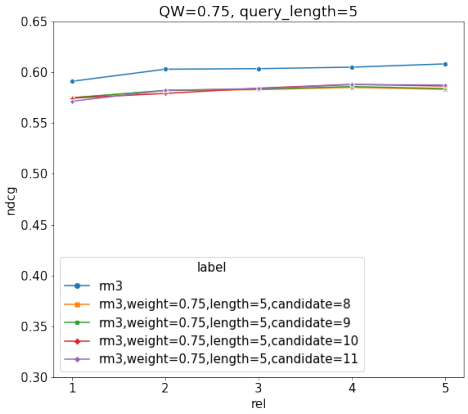

QW=0.5, query_length=6
ndcg
rel
label
rm3
rm3,weight=0.5,length=6,candidate=8
rm3,weight=0.5,length=6,candidate=9
rm3,weight=0.5,length=6,candidate=10
rm3,weight=0.5,length=6,candidate=11

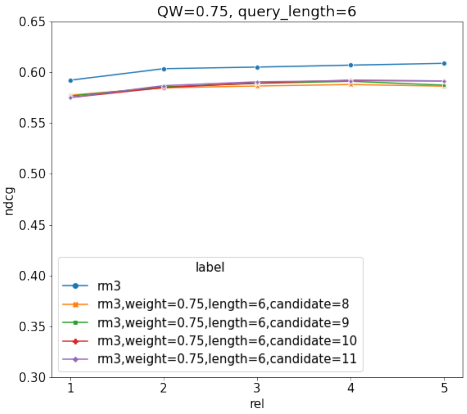

QW=0.5, query_length=7
ndcg
rel
label
rm3
rm3,weight=0.5,length=7,candidate=8
rm3,weight=0.5,length=7,candidate=9
rm3,weight=0.5,length=7,candidate=10
rm3,weight=0.5,length=7,candidate=11

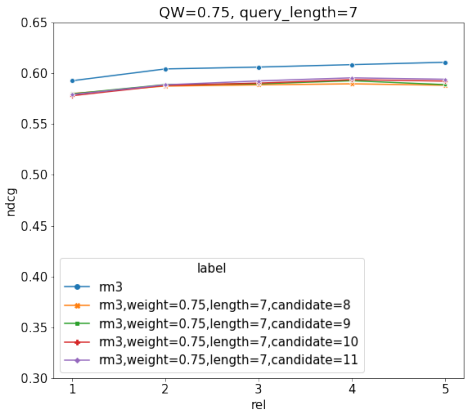

QW=0.5, query_length=8
ndcg
rel
label
rm3
rm3,weight=0.5,length=8,candidate=8
rm3,weight=0.5,length=8,candidate=9
rm3,weight=0.5,length=8,candidate=10
rm3,weight=0.5,length=8,candidate=11

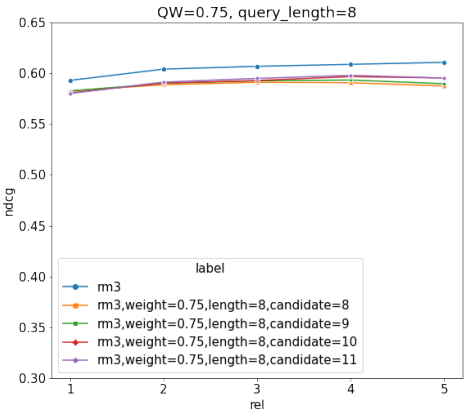

QW=0.5, query_length=9
ndcg
rel
label
rm3
rm3,weight=0.5,length=9,candidate=9
rm3,weight=0.5,length=9,candidate=10
rm3,weight=0.5,length=9,candidate=11

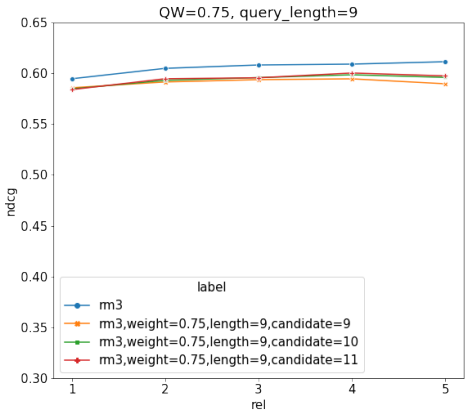

QW=0.5, query_length=10
ndcg
rel
label
rm3
rm3,weight=0.5,length=10,candidate=10
rm3,weight=0.5,length=10,candidate=11

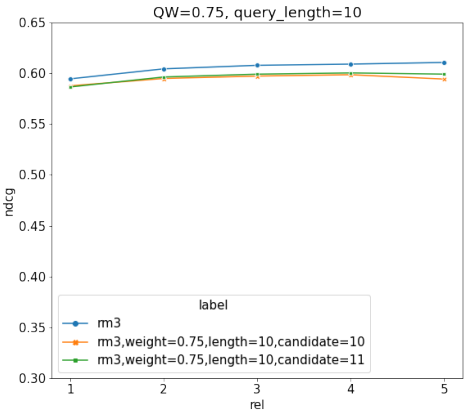

QW=0.75, query_length=5
ndcg
rel
label
rm3
rm3,weight=0.75,length=5,candidate=8
rm3,weight=0.75,length=5,candidate=9
rm3,weight=0.75,length=5,candidate=10
rm3,weight=0.75,length=5,candidate=11

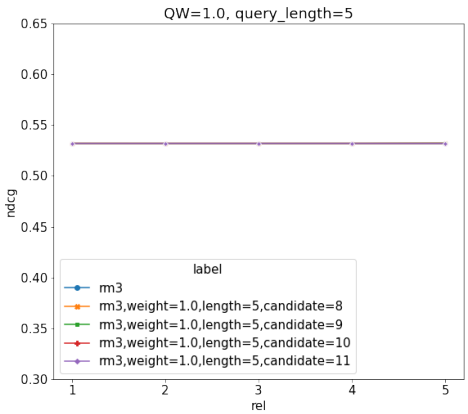

QW=0.75, query_length=6
ndcg
rel
label
rm3
rm3,weight=0.75,length=6,candidate=8
rm3,weight=0.75,length=6,candidate=9
rm3,weight=0.75,length=6,candidate=10
rm3,weight=0.75,length=6,candidate=11

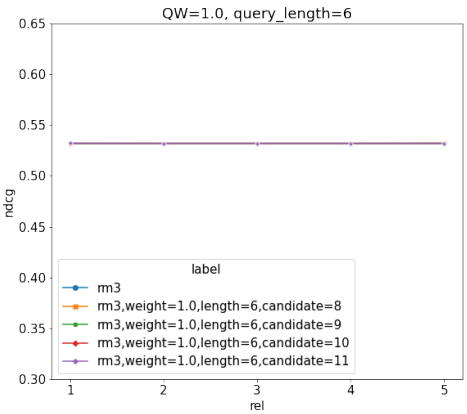

QW=0.75, query_length=7
ndcg
rel
label
rm3
rm3,weight=0.75,length=7,candidate=8
rm3,weight=0.75,length=7,candidate=9
rm3,weight=0.75,length=7,candidate=10
rm3,weight=0.75,length=7,candidate=11

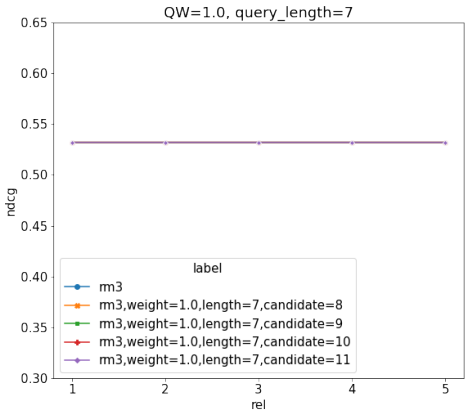

QW=0.75, query_length=8
ndcg
rel
label
rm3
rm3,weight=0.75,length=8,candidate=8
rm3,weight=0.75,length=8,candidate=9
rm3,weight=0.75,length=8,candidate=10
rm3,weight=0.75,length=8,candidate=11

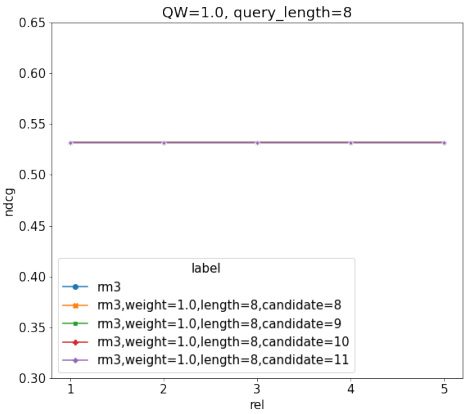

QW=0.75, query_length=9
ndcg
rel
label
rm3
rm3,weight=0.75,length=9,candidate=9
rm3,weight=0.75,length=9,candidate=10
rm3,weight=0.75,length=9,candidate=11

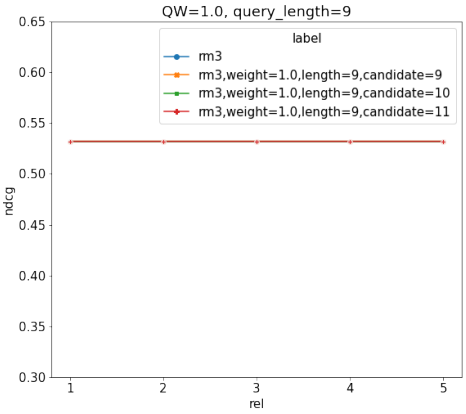

QW=0.75, query_length=10
ndcg
rel
label
rm3
rm3,weight=0.75,length=10,candidate=10
rm3,weight=0.75,length=10,candidate=11

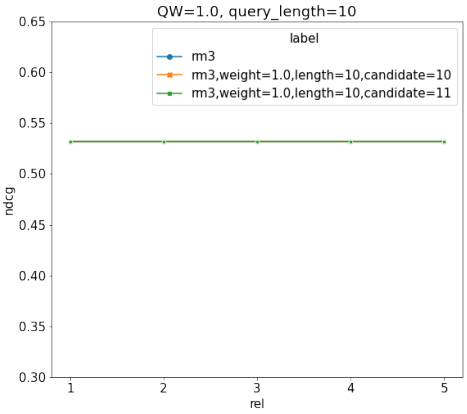

QW=1.0, query_length=5
ndcg
rel
label
rm3
rm3,weight=1.0,length=5,candidate=8
rm3,weight=1.0,length=5,candidate=9
rm3,weight=1.0,length=5,candidate=10
rm3,weight=1.0,length=5,candidate=11


QW=1.0, query_length=6
ndcg
rel
label
rm3
rm3,weight=1.0,length=6,candidate=8
rm3,weight=1.0,length=6,candidate=9
rm3,weight=1.0,length=6,candidate=10
rm3,weight=1.0,length=6,candidate=11


QW=1.0, query_length=7
ndcg
rel
label
rm3
rm3,weight=1.0,length=7,candidate=8
rm3,weight=1.0,length=7,candidate=9
rm3,weight=1.0,length=7,candidate=10
rm3,weight=1.0,length=7,candidate=11


QW=1.0, query_length=8
ndcg
rel
label
rm3
rm3,weight=1.0,length=8,candidate=8
rm3,weight=1.0,length=8,candidate=9
rm3,weight=1.0,length=8,candidate=10
rm3,weight=1.0,length=8,candidate=11


QW=1.0, query_length=9
label
rm3
rm3,weight=1.0,length=9,candidate=9
rm3,weight=1.0,length=9,candidate=10
rm3,weight=1.0,length=9,candidate=11
ndcg
rel
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
1
2
3
4
5


QW=1.0, query_length=10
label
rm3
rm3,weight=1.0,length=10,candidate=10
rm3,weight=1.0,length=10,candidate=11
ndcg
rel
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
1
2
3
4
5

Literaturverzeichnis

[1] Ahmed AbdoAziz Ahmed Abdulla, Hongfei Lin, Bo Xu, und Santosh Kumar Banbhrani. Improving biomedical information retrieval by linear combinations of different query expansion techniques. *BMC Bioinform.*, 17 (S-7):238, 2016. 2.1

[2] Ricardo Baeza-Yates und Berthier Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999. 1

[3] N. Belkin, Colleen Cool, W. Croft, und J. Callan. The effect multiple query representations on information retrieval system performance. In *SIGIR*, 1993. 2.1, 3.4, 4.2.1, 5

[4] C. Buckley und S. Robertson. Relevance feedback track overview: Trec 2008. In *TREC*, 2008. 2.2

[5] Chris Buckley, Matthew Lease, und M. Smucker. Overview of the trec 2010 relevance feedback track (notebook). 2010. 2.2, 3.2

[6] F. Colace, M. D. Santo, L. Greco, und P. Napoletano. Improving relevance feedback-based query expansion by the use of a weighted word pairs approach. *Journal of the Association for Information Science and Technology*, 66, 2015. 2, 2.1, 2.1

[7] W. Bruce Croft, Donald Metzler, und Trevor Strohman. *Search Engines. Information Retrieval in Practice*. Pearson Education, Inc, 2015. 2.1, 2.2

[8] Behzad Golshan, Theodoros Lappas, und E. Terzi. Sofia search: a tool for automating related-work search. *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, 2012. 2.3

[9] Matthias Hagen, Anna Beyer, Tim Gollub, Kristof Komlossy, und Benno Stein. Supporting scholarly search with keyqueries. In *ECIR*, 2016. (document), 1, 2.3, 3.3

[10] B. Jansen, A. Spink, und T. Saracevic. Real life, real users, and real needs: a study and analysis of user queries on the web. *Inf. Process. Manag.*, 36: 207–227, 2000. 2.1

[11] Canjia Li, Yingfei Sun, Ben He, Le Wang, Kai Hui, Andrew Yates, Le Sun, und Jungang Xu. Nprf: A neural pseudo relevance feedback framework for ad-hoc information retrieval. In *EMNLP*, 2018. 2.2, 2.2

[12] Jimmy Lin. The neural hype and comparisons against weak baselines. *SIGIR Forum*, 52:40–51, 2019. 1, 4.1

[13] Christopher D. Manning, Prabhakar Raghavan, und Hinrich Schütze. *An Introduction to Information Retrieval*. Cambridge University Press, 2009. 2.1, 2.2

[14] Bhaskar Mitra, Nick Craswell, et al. *An introduction to neural information retrieval*. Now Foundations and Trends, 2018. 2.2

[15] C. Nascimento, Alberto H. F. Laender, A. Silva, und Marcos André Gonçalves. A source independent framework for research paper recommendation. In *JCDL '11*, 2011. 2.3

[16] NIST. Trec 2004 robust track guidelines, 2005. URL `https://trec.nist.gov/data/robust/04.guidelines.html`. 4.1

[17] Joao Palotti, Harrisen Scells, und Guido Zuccon. Trectools: an open-source python library for information retrieval practitioners involved in trec-like campaigns. SIGIR'19. ACM, 2019. 4.1

[18] José R. Pérez-Agüera und L. Araujo. Comparing and combining methods for automatic query expansion. *ArXiv*, abs/0804.2057, 2008. 2.1

[19] P. Yang, H. Fang, und J. Lin. Anserini: Reproducible ranking baselines using lucene. *Journal of Data and Information Quality (JDIQ)*, 10:1 – 20, 2018. 3.3, 4.1

[20] Zheng Ye, X. Huang, Ben He, und H. Lin. York university at trec 2009: Relevance feedback track. In *TREC*, 2009. 2.2, 2.2

Erklärung

Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Halle, 3. Mai 2021

. . . . . .
Johannes Huck