Bachelorarbeit durchgeführt bei Springer Nature

SPRINGER NATURE

Heidelberger Platz 3
14197 Berlin

Bachelorarbeit

Suchoptimierung mittels maschinellen Lernens

Zeitraum 04.07.2016 - 11.10.2016

Lukas Abegg

Matrikelnummer 798972
Sommersemester 2016
Fachsemester 6
Studiengang Medieninformatik (B.Sc.)
Beuth Hochschule für Technik

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Beuth Hochschule für Technik
Luxemburger Str. 10
13353 Berlin

Betreuer **Prof. Dr. habil. Alexander Löser**
Fachbereich VI - Informatik und Medien
Beuth Hochschule für Technik

Gutachter **Prof. Dr. Martin Oellrich**
Fachbereich II - Mathematik - Physik - Chemie
Beuth Hochschule für Technik

Inhaltsverzeichnis

1 Einführung

Springer Nature ist ein weltweit führender Verlag für Forschungs-, Bildungs- und Fachliteratur mit einer breiten Palette an angesehenen und bekannten Medienmarken und zudem der weltweit größte Verlag für Wissenschaftsbücher. Für das Unternehmen Springer Nature ist es darum wichtig, auf seinen Web-Applikationen eine Suche anbieten zu können, die Suchintentionen erkennt und möglichst schnell zum gesuchten Content leitet. Die Suche wird vor allem als Hilfsmittel zur Navigation und zum Finden von Literatur und Dienstleistungen genutzt. Durch die vielen von Springer Nature publizierten[1] Zeitschriften und Querverweise in Artikeln, wird sie aber auch oft zur Suche nach Issues[2] und Artikeln verwendet sowie als Hilfestellung um Diagnosen zu Krankheitsbilder stellen zu können.

Springer Nature sammelt viele User-Tracking-Daten und dadurch viel Wissen über das Verhalten der User[3] bei der Nutzung ihrer Suche, lässt dieses Wissen jedoch bisher noch nicht in ihre Suche einfließen. In dieser Arbeit wollen wir untersuchen, ob mithilfe dieses Wissens, die Suche optimiert werden kann.

1.1 Aufbau der Suche bei Springer Nature

White Label Applikation mit Solr-Suche

Damit die verschiedenen Verlage und Zeitschriften der Verlagsgruppe Springer Nature ihre Produkte und Dienstleistungen online anbieten können, nutzt Springer Nature eine eigens entwickelte White Label Applikation[4]. Die White Label Applikation verwendet *Apache Solr* (im Folgenden „Solr“ genannt, siehe [**solr**]) als Suchplattform. Die Solr dient hierbei als eine der Schnittstellen zwischen dem Content-Pool von Springer Nature und der White Label Applikation. Bei den vom Content-Pool gelieferten Inhalten, handelt es sich um von Springer Nature Verlag publizierte Zeitschriften, Artikel, Bücher und redaktionelle Inhalte.

User-Tracking mit Webtrekk

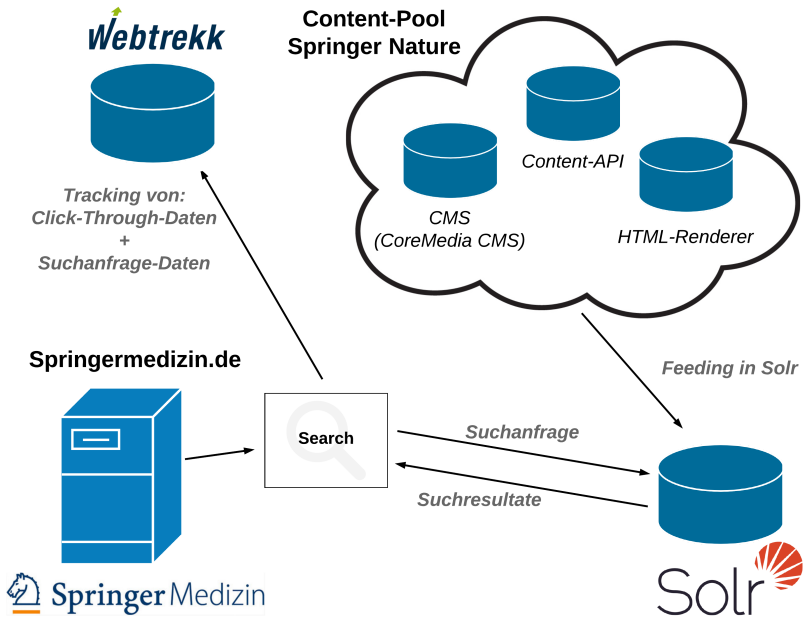
Um das Verhalten der User auf ihren Web-Applikationen zu tracken verwendet Springer Nature das Analysetool *Webtrekk* (siehe [**webtrekk**]). Webtrekk wird als externer Webservice genutzt. Über diesen Webservice können die Tracking-Daten gelesen und ausgewertet werden. Die daraus resultierenden Berichte bieten unter anderem die Möglichkeit, *Suchquery-Logs*[5] und *Click-Through-Rates* (CTR)[6] der User auszuwerten.

[1] Unter publizieren wird in dieser Arbeit die Veröffentlichung auf der Springermedizin-Applikation bezeichnet
[2] Nummer der Zeitschriftenausgabe, in der sich der Artikel befindet.
[3] Als User werden die Nutzer der Springer Nature Suche bezeichnet
[4] Eine White Label Applikation ist eine wiederverwendbare und agil erweiterbare Web-Applikation
[5] Protokoll über alle ausgeführten Suchanfragen auf der Applikation
[6] Kennzahl um die Anzahl der Klicks auf Links im Verhältnis zu den gesamten Impressionen darzustellen

Architektur

In Abb. 12 ist die Suche nochmals grafisch aufbereitet:

Abb. 1: Aufbau der Suche bei Springer Nature



1.2 Problemstellung: Keine Userrelevanz in der Suche

Zu den Stakeholder[7] der in Kapitel 1.1 angesprochenen White Label Applikation gehört *Springermedizin* (siehe [**SMED**]). Springermedizin betreibt ein Fortbildungs- und Informationsportal für Ärzte.

Userrelevante Dokumente werden nicht gefunden

Die User von Springermedizin suchen oft mit einschlägig, fundierten Fachbegriffen nach den neuesten und relevantesten Zeitschriften, Bücher oder Publikationen. Die zeitlich aktuellsten Suchtreffer zu finden ist für Springermedizin kein Problem. Die für den User *relevantesten* jedoch schon.

Der Springer Nature Stakeholder: Springermedizin setzt auf Webtrekk

Mithilfe von Webanalysten und Webtrekk versucht Springermedizin das Marketing seines Webauftrittes zu verbessern und ist sehr interessiert an neuen Ansätzen, um die gesammelten Tracking-Daten besser einzusetzen. In dieser Arbeit wird darum der Fokus auf die Verwendung von Tracking-Daten in der Suche von Springermedizin gesetzt.

Der fast gläserne User

Springermedizin sammelt Tracking-Daten über jegliche Aktivitäten auf deren Applikationen und investiert Zeit und Geld in die Individualisierung[8] der Analysedaten auf Webtrekk. Mittlerweile sind knapp 30 Custom-Parameter[9] auf Webtrekk angelegt, um genau die Daten zu tracken, die zur Analyse des Verhaltens der User auf ihrer Applikationen relevant sind. Dadurch entsteht ein fast „gläsernen User“. Dieses Wissen könnte zum Vorteil des Users eingesetzt werden, indem es in der Suche verwendet wird.

[7] Bezeichnet Springer Nature interne Kunden, die ein Interesse am Ergebnis der White Label Applikation haben
[8] Mit Individualisierung wird die Speicherung eigener Parameter bezeichnet
[9] Individuell erzeugte Parameter für Berichte und Analysen

1.3 Ziel der Arbeit

1.3.1 Suchoptimierung durch Click-Through-Daten

In dieser Arbeit werden wir untersuchen, ob mithilfe der von Springermedizin gesammelten Click-Through-Daten[10] deren Suche verbessert werden kann. Konkret wollen wir dies anhand eines Algorithmus basierend auf dem Klick-Modell[11] *Position-Based Modell* (PBM, siehe [**pbm**]) untersuchen.

Annahmen

Wir gehen dabei von folgenden Annahmen aus. Relevante Dokumente sind wichtiger als nicht relevante Dokumente. Eine Suchergebnis ist dann gut, wenn die relevanten Ergebnisse in der verwendeten Hierarchie vor den nicht relevanten Ergebnisse auftauchen.

Anwendung auf das Springermedizin-Umfeld

Wir werden versuchen, die CTRs der Suchresultate mithilfe des oben angesprochenen Algorithmus zu berechnen und mit diesen ein *Reranking*[12] der Suchresultate auf das Springermedizin-Umfeld abzubilden. Die Herausforderung wird hierbei die Adaptierung des Lösungsansatzes auf das Springermedizin-Umfeld sein. Im Idealfall widerspiegeln die gesammelten Click-Through-Daten die Userrelevanz der einzelnen Dokumente[13].

1.3.2 Abbildung auf das Springermedizin-Umfeld

Potential von Userrelevanzen in der Suchoptimierung analysieren

Die Analyse von User-Tracking-Daten bietet viel Potential bezogen auf Userrelevanzen. Sind anhand des hier umgesetzten Lösungsansatzes Verbesserungen in der Qualität der Suche zu verzeichnen, möchte Springermedizin in Zukunft vermehrt User-Tracking-Daten in die Suche einfließen lassen. Diese Arbeit könnte dann als Fundament für weitere Lösungsansätze dienen.

Bekanntes und wirkungsvolles Information Retrieval Verfahren

Suchoptimierung mittels Userrelevanz ist ein bekanntes und nicht triviales, aber relativ wirkungsvolles Information Retrieval Verfahren (siehe [**IWUSBI**]). Seit Mitte der 2000er Jahre wird mithilfe dieses Verfahrens versucht, Suchmaschinen zu verbessern. Aus dieser Zeit stammen auch die ersten Ansätze, um mithilfe von Click-Through-Daten die Userrelevanz der Suchergebnisse zu berechnen (siehe [**Joachims**]).

Lösungsansatz basierend auf Click-Through-Daten aus Webtrekk

Springermedizin führt ein eigenes Tracking der User durch und verwendet auf Webtrekk selbst definierte Tracking-Parameter. Dadurch hängt die Wahl des in dieser Arbeit zu untersuchenden Lösungsansatzes und dessen Umsetzung stark von den durch Webtrekk gegebenen Analyse-Daten ab.

[10] Mit Click-Through-Daten bezeichnen wir alle Tracking-Daten, welche während der Interaktion zwischen User und Suche protokolliert werden
[11] Als Klick-Modell wird ein Modell zur Berechnung des Userfeedbacks bzw. der Userrelevanz mithilfe von Click-Through-Daten bezeichnet
[12] Mit Reranking bezeichnen wie die Umsortierung einer Liste von Suchresultaten
[13] Als Dokumente werden die einzelnen Suchresultate bezeichnet

Keine Gegenüberstellung mit anderen Lösungsansätzen

Durch den vorgegebenen Zeitraum für die Erstellung dieser Bachelorarbeit bedingt, werden wir den Lösungsansatz so wählen, dass er mit den Gegebenheiten bei Springermedizin sinnvoll und in diesem Zeitrahmen realistisch implementiert werden kann. Wir werden daher in dieser Arbeit keine Gegenüberstellung mit anderen Lösungsansätzen machen.

1.4 Methodik

Wie in Kapitel 1.3.1 angesprochen, wollen wir das Klick-Verhalten der User in der Suche analysieren, um mithilfe der daraus berechenbaren CTRs, die Suchergebnisse zu verbessern. Dieses Klick-Verhalten können wir aus den Click-Through-Daten lesen.

Suchterm semantisch aufschlüsseln

Um mit den Click-Through-Daten arbeiten zu können, müssen wir zunächst die relevanten Click-Through-Daten herausfiltern. Dazu müssen wir die Click-Through-Daten dem *Suchterm*[14] der Anfrage zuordnen können. Zu den Click-Through-Daten wird immer der Suchterm gespeichert, mit dem dabei gesucht wurde. Das heißt wir können eine Relation zwischen dem Suchterm der Click-Through-Daten und dem Suchterm unserer Anfrage herstellen.

Die Click-Through-Daten müssen aber nicht mit dem vollständigen Suchterm in Relation stehen. Sie können auch nur mit einem Wort, einem Teil des Suchterms oder einem Synonym eines dieser Worte in Relation stehen. Wir müssen darum den Suchterm semantisch aufschlüsseln, um alle relevanten Click-Through-Daten filtern zu können.

Aufbereitung Click-Through-Daten

Können wir alle relevanten Click-Through-Daten zu einer Suchanfrage filtern, müssen wir lernen diese richtig aufzubereiten, um die CTR berechnen zu können. Ein wichtiger Punkt bei der Aufbereitung der Click-Through-Daten ist die Interpretation des Relevanzfeedbacks der einzelnen Click-Through-Daten. Nicht jeder Klick ist gleich relevant zu interpretieren. Die Relevanz eines Klicks hängt davon ab, welche Aktionen der User während dem Suchvorgang *vor* und *nach* dem Klick durchgeführt hat. Wir müssen darum zuerst analysieren, welche Informationen wir zu den Click-Through-Daten aus Webtrekk lesen können. Reichen diese Informationen für detailliertere Interpretationen nicht aus, müssen wir alle Click-Through-Daten als gleich relevant lesen.

Result-Reranking mittels PBM basiertem Algorithmus

Wissen wir, wie wir die Click-Through-Daten aufbereiten, können wir diese zur Berechnung der CTR eines Dokumentes und somit dessen Userrelevanz einsetzen. Für die Verwendung dieser Userrelevanz müssen wir festlegen, wann und in welcher Art wir sie einsetzen. Wie bereits in Kapitel 1.3.1 erwähnt, werden wir uns in dieser Arbeit auf die *Aufbereitung* der Suchresultate aus der Solr konzentrieren und dort einen *Reranking-Algorithmus* einbauen.

Der Reranking-Algorithmus basiert auf dem *Klick-Modell PBM*. Ein Klick-Modell verwendet die Click-Through-Daten, um daraus die CTR eines Dokumentes zu berechnen. Die Wahl des Klick-Modells hängt darum stark von den Click-Through-Daten und den darin enthaltenen Informationen ab. Das PBM setzt sich aus zwei Wahrscheinlichkeiten zusammen. Die Wahrscheinlichkeit für einen Klick auf die Position im Suchresultat und die Wahrscheinlichkeit für einen Klick auf das Dokument. In dieser Arbeit

[14] Als Suchterm wird die Sammlung aller, in der Suchanfrage verwendeten Wörter bezeichnet

werden wir veranschaulichen, warum wir den Ansatz des PBMs (siehe [**pbm**]) gewählt haben und wie wir den Algorithmus umgesetzt haben.

Vergessen der alten Daten

Das PBM berechnet Wahrscheinlichkeiten, um die CTR eines Dokumentes zu einer Suchanfrage festzustellen. Dem Algorithmus muss dazu das notwendige Wissen entweder zur Verfügung gestellt oder antrainiert werden. Wird dem Algorithmus das Wissen antrainiert, benötigt der Algorithmus eine Möglichkeit, neues Wissen zu Lernen und altes zu Vergessen.

Mit Webtrekk haben wir eine Wissensbasis, die durch die Springermedizin-Applikation automatisch um neue Click-Through-Daten ergänzt wird. Das heißt wir müssen uns nicht um eine Möglichkeit zum Lernen neuer Daten kümmern. Wir müssen uns aber überlegen, wie wir altes Wissen vergessen und wie wir dem User neues Wissen präsentieren, damit dieser sich durch die CTR-Berechnung nicht auf alten Dokumenten festfährt.

1.5 Gliederung und Aufbau

Der Lösungsansatz und deren Grundlagen

In diesem Kapitel wurde der zu untersuchende Lösungsansatz vorgestellt. Wir haben Problemstellungen und deren Teilprobleme identifiziert. Dabei sind wir auf die Hintergründe dieser Arbeit und die Vorgehensweise eingegangen. Im zweiten Kapitel (Grundlagen) folgt die Theorie des beschriebenen Lösungsansatzes. Hier werden wir uns auf die fachlichen Grundlagen konzentrieren, Problemstellungen analysieren und Lösungsansätze vorstellen.

Umsetzung des Lösungsansatzes

In Kapitel 3 (Reranking mittels CTR) werden wir die in Kapitel 1.4 angesprochene Methodik verfeinern und auf Basis der Grundlagen aus Kapitel 2.1, über die detaillierte Vorgehensweise bei der Umsetzung diskutieren. Die Umsetzung selbst, folgt dann in Kapitel 4 (Implementierung).

Erkenntnisse verarbeiten

Um zu prüfen ob der umgesetzte Lösungsansatz die erhofften Verbesserungen erzielt, werden wir diesen in Kapitel 5 (Evaluation und Auswertung) in einer Evaluation mit der bisherigen Springermedizin-Suche vergleichen. Aufgrund der resultierenden Erkenntnisse, werden wir in Kapitel 6 (Zusammenfassung und Ausblick) ein Fazit ziehen können und einen Ausblick auf mögliche zukünftige Arbeiten geben.

2 Grundlagen

In diesem Kapitel werden wir die fachlichen Grundlagen zu unserem in Kapitel 1.4 vorgestellten Lösungsansatz aufarbeiten. Dabei werden wir den Stand der Technik erklären, auf dem wir aufsetzen und diskutieren wie weit wir diesen nutzen können und wo wir diesen erweitern müssen. Wichtig hierfür ist das Verständnis, für die Problemstellungen in der Interaktion zwischen den Nutzern der Suche und der Suche selbst. Wir werden mögliche Lösungsansätze anschauen und darüber diskutieren. Um mit CTRs die Userrelevanz von Suchergebnissen bestimmen zu können, müssen wir lernen, Click-Through-Daten als Relevanz-Feedback zu deuten. Wie wir anschließend mithilfe dieses Relevanz-Feedbacks ein Reranking des Suchresultats durchführen, lernen wir in den Grundlagen zu unserem Reranking-Algorithmus. Diese Grundlagen sind notwendig für die Umsetzung des Lösungsansatzes in den nachfolgenden Kapiteln 3 und 4.

2.1 Grundbegriffe

In Kapitel 2 und dem darauf folgenden Kapitel 3, werden Formeln mit selbst definierten Symbolen eingeführt. Folgend eine Legende der wichtigsten Symbole:

Tab. 2.1: Legende der wichtigsten Formel-Symbolen für den Algorithmus

Symbol	Bedeutung	Symbol	Bedeutung
u	ein Dokument im Suchergebnis	C	das Dokument wird vom User im Suchergebnis angeklickt
q	ein Suchterm	E_{r_u}	ein Dokument wird aufgrund seiner Position r im Suchergebnis angeschaut
r	die Position des Dokumentes im Suchergebnis	A_u	das Dokument wird aufgrund seines Suchsnippets im Suchergebnis, zum Suchterm q angeschaut
c	ein Klick auf ein Dokument im Suchergebnis	X_u	ein Zufallswert, der einem Dokument des Suchergebnisses zugewiesen wird
s	der Suchvorgang zu einer Suchanfrage	r_{X_u}	r in der nach Zufallswert sortierten Liste der Suchergebnisse
S	eine Sammlung von Suchvorgängen s	$r_{P(C_u)}$	r in der nach Klick-Wahrscheinlichkeit sortierten Liste der Suchergebnisse zum Suchterm q
P	die Wahrscheinlichkeit dass ein Fall eintritt	R_u	durch den Reranking-Algorithmus berechneter Relevanz-Wert des Dokumentes u

2.1.1 Semantik von User-Interaktionen

Problemstellungen der Click-Through-Daten: Was analysieren wir?

Einzelne Wörter oder Teile des Suchterms können in weiteren Suchanfragen vorkommen Ein Suchterm kann aus einem oder mehreren Wörtern bestehen. Jeder User formuliert eine Suchanfrage anders. Sei es die Wortwahl, die Zeitform oder die Verwendung von Bindewörtern. Daraus lässt sich vermuten, dass einzelne Wörter oder Teile des Suchterms in weiteren Suchanfragen vorkommen können. Folglich muss der Suchterm semantisch aufgeschlüsselt werden, um Relationen zwischen den Click-Through-Daten und der Suchanfrage herstellen zu können. Nur so können wir alle relevanten Click-Through-Daten filtern.

Suchanfragen mit Synonymen und verwandten Begriffen beachten Nehmen wir als Beispiel die Suchanfrage „chronische Dyspnoe“. Würden wir stattdessen den sinnverwandten Suchterm „konstante Atemnot“ verwenden, würden wir für beide Fälle ähnliche Suchresultate erwarten. Folglich würden wir auch ähnliche Click-Through-Daten vermuten. Wir sollten daher die Synonyme und verwandten Begriffe zu unserem Suchterm ebenfalls beachten und deren Click-Through-Daten, in den Reranking-Algorithmus einfließen lassen. Dazu benötigen wir eine Wissensbasis, welche die Synonyme und verwandte Begriffe zu unserem Suchterm gespeichert hat. Eine solche Basis bieten Wörterbücher und Thesauri[1]. Beim Content von Springermedizin handelt es sich um *medizinische Inhalte* in *deutscher Sprache*. Es ergibt daher Sinn, dies in der Wahl der richtigen Wissensbasis zu berücksichtigen.

Problemstellungen des Lösungsansatzes: Warum kann es schief gehen?

Die folgenden Faktoren leiten sich aus dem verfolgten Lösungsansatz des Reranking-Algorithmus ab bzw. werden in diesem nicht beachtet. Wir müssen davon ausgehen, dass diese das Untersuchungsergebnis des Lösungsansatzes negativ beeinflussen könnten.

Die Relation des Suchterms zu den Click-Through-Daten wird nicht gewichtet Die Click-Through-Daten sind dann relevant, wenn mindestens ein Wort des aufgeschlüsselten Suchterms, in Relation zu diesen Daten steht. Dadurch können falsche Relationen entstehen und nicht relevante Click-Through-Daten, die Klick-Wahrscheinlichkeiten der Dokumente im Reranking-Algorithmus negativ beeinflussen.

Intentionen und die Mehrdeutigkeit von Suchbegriffen werden nicht beachtet Die genaue semantische Analyse eines Suchterms beinhaltet unter anderem die Erkennung von Begriffen und deren Mehrdeutigkeiten. Suchte jemand z.B. nach dem Begriff „Brücke“, hat dieser im medizinischen Kontext mehrere Bedeutungen. Es könnte „ein Teil des zentralen Nervensystems“ gemeint sein oder „eine Form des Zahnersatzes“. Wie bereits im vorherigen Kapitel 2.1.1 erwähnt, können wir mithilfe eines Thesaurus, die verschiedenen Bedeutungen erkennen. Der Thesaurus könnte uns zu dem Begriff „Brücke“ beispielsweise, folgende Informationen zurückgeben:

Abb. 2: Im Thesaurus gespeicherte Werte zum Suchterm „Brücke“

Eintrag 1: Zahnmedizin

Oberbegriff: (Broader Term)
Zahnbrücke

Unterbegriff (Narrower Term):
Brücke

Verwendungszweck (RT):
Zahnmedizin, Zahnheilkunde, Prothetik, Festsitzender Zahnersatz

Eintrag 2: Neurologie

Oberbegriff: (Broader Term)
Hirnstammbrücke

Unterbegriff (Narrower Term):
Brücke, Brücke des Hirnstamms

Verwendungszweck (RT):
Zentrales Nervensystem, Truncus encephali, Stammhirn, Neurologie

Unser Reranking-Algorithmus ignoriert diese Mehrdeutigkeit. Er würde in diesem Fall alle Click-Through-Daten zu beiden Begriffsbedeutungen suchen. Hier können wir zufallsbedingt drei Ausgangslagen haben. Die Click-Through-Daten entsprechen der Suchintention (1) - das wäre der zufallsbedingte Optimalfall. Das Suchresultat wird von der Mehrfachbedeutung nicht beeinflusst. Das Gegenteil tritt ein (2) - das Suchresultat wird in diesem Fall durch eine falsche Relevanz negativ beeinflusst. Es sind keine Click-Through-Daten vorhanden (3) - die Solr und die Klick-Wahrscheinlichkeit der Position im Suchresultat, definieren das Suchergebnis. In diesem Fall kann die Wertigkeit des Algorithmus nicht vorhergesehen werden.

Keine Aktualität in der Suche Der von uns verfolgte Reranking-Algorithmus nimmt keine Rücksicht auf die „Aktualität“ eines Beitrages sondern nur auf die Klick-Wahrscheinlichkeit und kann dadurch, aktuellere Dokumente trotz Relevanz, schlecht positionieren im Suchresultat. Die Klick-Wahrscheinlichkeit kann durch zwei Faktoren beeinflusst werden. Hohe Relevanz in der Solr-Suche (1) - wird in der Volltextsuche die Aktualität des Dokumentes in die Berechnung der Relevanz einbezogen, werden aktuelle Beiträge im Suchergebnis der Solr weit vorne eingestuft. Wie wir aus Abb. 4 erkennen können, haben die ersten

[1] Thesauri sind strukturierte Verzeichnisse von Begriffen, die allesamt in irgendeiner Beziehung zueinander stehen bezeichnet

Positionen des Suchergebnisses eine höhere Klick-Wahrscheinlichkeit. Das könnte die Berechnung des Reranking-Algorithmus positiv beeinflussen. Reranking-Algorithmus um Zufallsfaktor erweitern (2) - mithilfe eines Zufallsfaktor ist die Reihenfolge der Suchresultate weniger vom Algorithmus abhängig und die Wahrscheinlichkeit, aktuelle Dokumente ohne Click-Through-Daten weit vorne im Suchergebnis zu finden, wird erhöht.

Interessante Dokumente werden nie gesehen Wie bereits oben erwähnt, beachtet der Reranking-Algorithmus Dokumente ohne Click-Through-Daten nur, wenn die Position im Suchresultat eine Klick-Wahrscheinlichkeit aufweist. Dadurch kann es sein, dass interessante Dokumente nie gesehen werden. Dem entgegenwirken können wir ebenfalls mit dem oben erwähnten Zufallsfaktor. Dadurch wird die Klick-Wahrscheinlichkeit für interessante Dokumente zufallsbedingt erhöht.

Nicht beeinflussbare Faktoren: Fehlerhafter Content verfälscht die Suchergebnisse

Die folgenden Faktoren beeinflussen das Suchergebnis negativ, sind aber vom Content so vorgegeben. Der von uns verfolgte Lösungsansatz des Reranking-Algorithmus kann diese nicht beeinflussen. Wir beachten diese Faktoren in unserer Arbeit darum nicht.

Mehrfachverwertung des Contents Auf der Springermedizin Suche wird teilweise im Suchergebnis auf denselben Artikel mehrfach verwiesen. Das liegt an der bei Springermedizin praktizierten Mehrfachverwertung des Contents. Es gibt *Journal-Artikel*, das sind aus Journalen, Zeitschriften oder Magazinen stammende Artikel, die auf Springermedizin direkt online[2] gelesen werden können. Bei Neuerscheinung des Artikels, werden dazu oft *redaktionelle Artikel* publiziert, welche auf den Journal-Artikel verweisen sollen. Diese können im CMS (siehe Abb. 12) von der Suche exkludiert werden. Werden diese nicht exkludiert, können beide Artikel im Suchergebnis erscheinen.

Ausspielung von Teaser Springermedizin verwendet Teaser[3] auf der Startseite und auf Übersichtsseiten zu Rubriken, als Einstieg in den nachfolgenden, ausführlichen Beitrag. Diese werden auch in der Suche ausgespielt. Teaser sagen nichts über die Wertigkeit des Beitrages aus. Es wird nicht bekanntgegeben, auf welche Art von Beitrag (z.B. wissenschaftliche Publikation oder ein Artikel aus einem Journal) verwiesen wird und von welchem Autor der Beitrag stammt. Teaser können darum nicht nach Relevanz eingestuft werden und sollten nicht im Suchergebnis erscheinen.

Fehlerhafte Importe der Daten Viele Beiträge sind falschen Rubriken zugeordnet. Beispielsweise werden Beiträge fälschlicherweise als wissenschaftliche Publikationen publiziert, obwohl sie aus einem Journal oder einer Fachzeitschrift stammen. Diese Fehler sind auf fehlerhafte Importe der Daten zurückzuführen und verfälschen die Wertigkeit des Suchergebnisses.

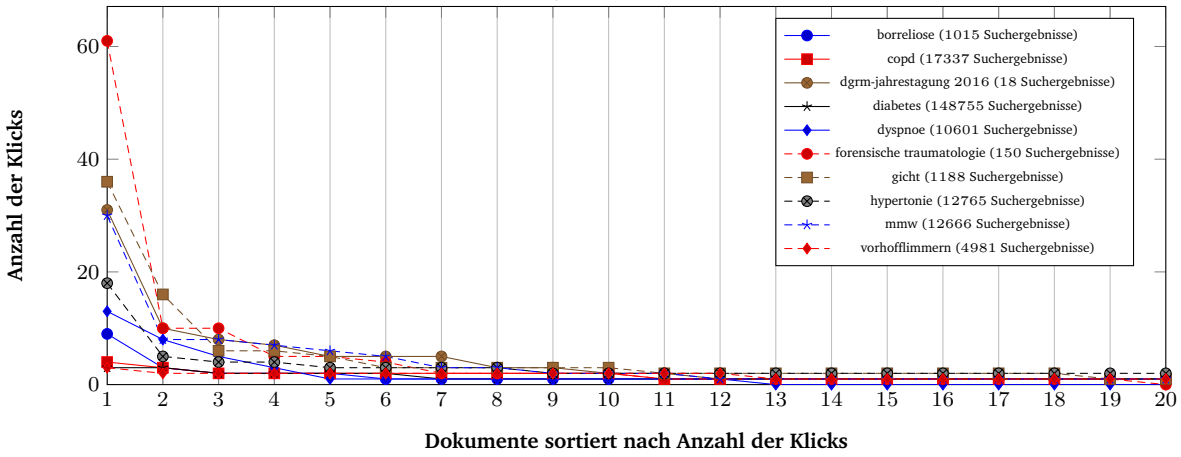
Schlechte Suchsnippets beeinflussen die Klick-Wahrscheinlichkeit eines Dokumentes negativ Das PBM berechnet die Klick-Wahrscheinlichkeit, also die *Attraktivität* eines Dokumentes auf Basis dessen Klick-Häufigkeit zum Suchterm und dessen Position im Suchresultat. Das heißt, der User analysiert das Suchresultat und sobald er auf den Link zu einem Dokument klickt, fließt dieser Klick in die Berechnung ein. Folglich bestimmt nicht der Inhalt des Dokumentes über dessen Attraktivität, sondern dessen Suchsnippet[4]. Wirkt dieses wenig relevant, kann dadurch die Klick-Wahrscheinlichtkeit negativ beeinflusst werden.

[2] Der Begriff „online“ wird hier als Verweis auf die Springermedizin.de-Webseite verwendet
[3] Als Teaser wird ein kurzer Texte bezeichnet, der das Interesse für den nachfolgenden Beitrag wecken soll
[4] Unter einem Suchsnippet wird eine Zusammenfassung des Inhalts des verlinkten Dokumentes als kurzen Teasertext verstanden

Analyse der Click-Through-Daten: Wenige Dokumente erhalten viele Klicks

Um das Klick-Verhalten der User auf der Springermedizin-Suche zu verstehen, ist es wichtig, anhand oft gesuchter Suchphrasen dieses Verhalten zu analysieren. Dazu wurde eine Analyse über einen Zeitraum von 30 Tagen erstellt und die zehn am häufigsten gesuchten Suchphrasen verwendet. Die Analyse vergleicht für jede Suchphrase die 20 Dokumente mit den meisten Klicks. Die Dokumente wurden hierbei nicht nach Position im Suchergebnis sondern nach Klick-Häufigkeit selektiert. Jeder Graph der folgenden Abb. 3 stellt eine Suchphrase dar. Wie wir sehen, zeigen die meisten Graphen ein exponentiell stark abnehmendes Verhalten der Klick-Häufigkeiten. Dieses exponentielle Verhalten zeigt, dass einzelne Dokumente häufig und viele Dokumente selten bis nie angeklickt werden. Dieser Effekt kann wie in vielen natürlichen Phänomenen mit exponentiellem Verhalten, durch das Potenzgesetz (Power Law, siehe [**PowerLaw**]) beschrieben werden.

Abb. 3: Analyse der 20 am häufigsten angeklickten Dokumente der zehn meistgesuchten Suchphrasen.
Zeitraum der Analyse: 19.08.16 - 19.09.16

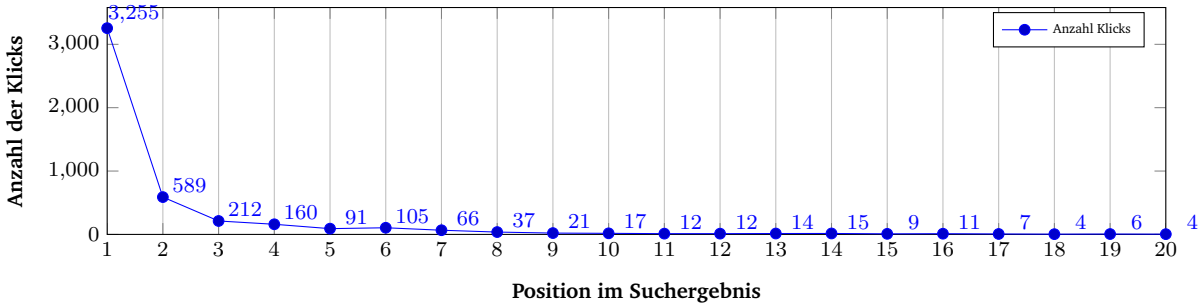


Betrachten wir die Graphen, können wir vor allem für die ersten fünf analysierten Dokumente, verglichen mit den restlichen analysierten Dokumenten, hohe Klick-Häufigkeiten feststellen. Daraus lässt sich die Vermutung ableiten, dass einzelne Dokumente eine sehr hohe Relevanz für die entsprechende Suchanfrage aufweisen und nur wenige Dokumente auf die User als relevant wirken. Ein weitere Vermutung ist, dass der zu durchsuchende Content wenig relevante Dokumente hat. Die Suchphrasen lassen auf sehr diverse Suchintentionen deuten. Es handelt sich hierbei unter anderem um Krankheiten, Zeitschriften und Behandlungen mit mehreren tausend Suchergebnissen. Die Wahrscheinlichkeit, dass wenig relevanter Content für die meisten der analysierten Suchphrasen zutrifft, sollte aufgrund der hohen Anzahl an gefundenen Suchergebnissen zu diesen Suchphrasen, relativ gering sein. Wir müssen darum eher davon ausgehen, dass sich die User auf einzelne, im Suchresultat weit oben stehende Dokumente festfahren. Das könnte an schlechten Suchergebnissen und somit an einer schlechten Suchqualität liegen. Um jedoch ein genaueres Bild über das Verhalten erstellen zu können, müssen wir einen Vergleich mit der nachfolgenden Analyse in Abb. 4 ziehen.

Analyse der angeklickten Positionen: Die ersten Positionen werden häufiger angeklickt

In der unten folgenden Analyse sehen wir das positionsbezogene Klick-Verhalten der User auf der Springermedizin-Suche. Dazu wurden über den Zeitraum von einem Monat, die letzten 1000 Suchanfragen ausgewertet. Dargestellt sehen wir die Häufigkeitsverteilung der Klicks als Graph. Wir beschränken uns hierbei auf die ersten 20 Positionen der Suchresultate. Wie wir sehen, nimmt die Anzahl der Klicks mit zunehmender Position exponentiell ab. Dieser Effekt kann ebenfalls, wie in Abb. 3, durch das Potenzgesetz (Power Law, siehe [**PowerLaw**]) beschrieben werden.

Abb. 4: Analyse der Klicks auf die ersten 20 Positionen der Suchergebnisse aller Suchanfragen.
Zeitraum der Analyse: 19.08.16 - 19.09.16



Betrachten wir den Graphen, sehen wir, dass besonders die erste Position, auffällig oft angeklickt wird. Daraus könnten wir die Vermutungen ableiten, dass die Suche eine sehr gute Qualität besitzt, weil die zu oberst angezeigten Dokumente, sehr relevant sind und die meisten User der Suchmaschine vertrauen. Wie wir aus den Analysen von [**Joachims**] lesen können, müssen wir davon ausgehen, dass die Häufigkeit des Klicks auf die ersten Positionen des Suchresultates eher dem Vertrauen der User der Suchmaschine, als der Qualität der Suche geschuldet ist. Vergleichen wir die Analyse aus Abb. 3 mit dieser Analyse, sehen wir ein sehr ähnliches Muster in der Häufigkeitsverteilung der Klicks. Wir können anhand der Klick-Zahlen ebenfalls vermuten, dass die am häufigsten angeklickten Dokumente, sich dabei auf den ersten Positionen des Suchergebnisses befunden haben.

2.1.2 Userrelevanz mittels CTR

Um mit Click-Through-Daten arbeiten zu können, müssen wir zuerst verstehen, was Click-Through-Daten sind und wie sie entstehen.

Was sind Click-Through-Daten und wie entstehen diese?

Click-Through-Daten sind Tracking-Daten. Tracking-Daten entstehen durch die Interaktion zwischen dem User der Applikation und der Applikation selbst. Sie verfolgen das Verhalten der User auf der Applikation und speichern dieses in einer Datenbank, in unserem Fall in Webtrekk ab. Die für uns interessanten Tracking-Daten entstehen, wenn der User auf der Suche von Springermedizin ein Anfrage stellt und darauf folgend, ein Element aus dem Suchresultat anklickt.

Wie werden die Click-Through-Daten in Webtrekk gespeichert?

Die Speicherung der Daten auf Webtrekk übernimmt die Springermedizin-Applikation. Führt ein User eine Suche durch und klickt dabei ein Resultat an, sendet die Springermedizin-Applikation die Tracking-Informationen an Webtrekk. Die Tracking-Daten für diese Aktion, setzen sich zusammen aus der Suchanfrage, dem Zeitpunkt der Suche, den Userdaten, der angeklickten Position im Suchresultat und den Dokumentinformationen zum angeklickten Dokument.

Wie können wir Click-Through-Daten aus Webtrekk lesen?

Webtrekk ist ein Analysetool. Das heißt für uns, wir können nicht direkt auf die Datenbank mit den Tracking-Daten zugreifen. Um die Tracking-Daten lesen zu können, müssen wir eine Analyse auf Webtrekk ausführen. Mithilfe dieser Analyse können wir uns die Click-Through-Daten so zusammenstellen lassen, wie wir sie für die Berechnung der CTR benötigen.

Klick-Häufigkeiten zu Suchterm selektieren und mittels Filtern einschränken Die Click-Through-Daten bestehen aus einzelnen *Klick-Häufigkeiten*. Eine Klick-Häufigkeit beschreibt die Anzahl der Klicks, die zu einer bestimmten Suchanfrage auf ein bestimmtes Dokument gemacht wurden und auf welcher Position

im Suchresultat, sich dieses Dokument dabei befunden hat. Die Webtrekk-Analysen geben uns eine Sammlung von Klick-Häufigkeiten zurück. Wir können bei diesen Analysen die Klick-Häufigkeiten nach Suchbegriffen filtern und den Zeitraum mitgeben, in welchen die Suchanfragen durchgeführt wurden. Des weiteren gibt es die Möglichkeit, weitere Filter wie die Anzahl zurückzugebender Klick-Häufigkeiten oder auch den „Login-Status[5] des Users“ zu setzen.

Wie sehen die Click-Through-Daten aus?

Eine Beispiel einer Klick-Häufigkeit, wie sie von einer Webtrekk-Analyse ausgespielt wird, sieht wie folgt aus:

Tab. 2.2: Beispiel Click-Through-Daten

Click-Through-Daten	Klick-Häufigkeit
searchresult-1.Course.chronische Dyspnoe bei Erwachsenen.10621768.chronische Dyspnoe	5

Hier die Aufschlüsselung der Click-Through-Daten:

Tab. 2.3: Beispielhafte Aufschlüsselung der Click-Through-Daten

Position	Dokumenttyp	Titel	ID	Suchterm
searchresult-1	Course	chronische Dyspnoe bei Erwachsenen	10621768	chronische Dyspnoe

Die Click-Through-Daten lassen sich wie folgt lesen. In diesem Beispiel haben die User eine Suchanfrage zum Suchterm „chronische Dyspnoe[6]“ gestellt. Dabei haben sie das Dokument mit der ID 10621768 angeklickt. Dieses hat sich dabei auf der Position eins der Suchresultate befunden. Es wurde insgesamt fünfmal angeklickt in der gesuchten Periode.

Aus Merkmalen und Eigenschaften des Userverhaltens ein implizites Feedback bilden

Mit dem Tracking der User auf einer Suchmaschine verfolgen wir die Idee, ein implizites Feedback aus deren Verhalten interpretieren zu können. Das machen wir, indem wir Merkmale und Eigenschaften des Verhaltens lesen und daraus ein Feature-Set[7], wie in [**IWUSBI**] beschrieben erzeugen. Dieses Feature-Set setzt sich zusammen aus den Informationen des *Klick-Verhaltens* der User (Click-Through Features), deren *Browsing-Verhalten*[8] (Browsing Features) während einer Suchanfrage und den *semantischen Relationen* zwischen der Suchanfrage und den dazu ausgespielten Suchresultaten (Query-Text Features). Mithilfe des Feature-Set lassen sich dann Schlussfolgerungen zum Relevanz-Feedback ziehen. Auf diesem Feature-Set werden wir bei der Auswertungen unserer Click-Through-Daten aufbauen, um damit die CTR zu berechnen.

2.1.3 Result-Reranking mittels PBM basierten Algorithmus

Alternative Ansätze um Click-Through-Daten in den Suchprozess einzubinden

Kurzanalyse der möglichen Ansätze um Click-Through-Daten in Suchprozess einzubinden Wir untersuchen in dieser Arbeit die Verwendung der Click-Through-Daten in der Aufbereitung der Suchresultate der Springermedizin-Applikation. Es gibt aber auch andere mögliche Eingriffspunkte während des Suchprozesses, um die Click-Through-Daten zu verwenden. Eine Alternative wäre die Verwendung der CTR in der Aufbereitung der Suchanfrage auf der Springermedizin-Applikation (1). Denkbar wäre auch, die Berechnung der CTR in den Suchindex der Solr einzubauen (2). Wir werden die verschiedenen Ansätze kurz durchgehen und am Ende erläutern, weshalb wir uns für den gewählten Ansatz mit dem PBM basierten Algorithmus entschieden haben.

[5] Mit Login-Status wird zwischen einem zum Zeitpunkt der Suche auf der Springermedizin-Applikation angemeldeten und nicht angemeldeten Usern unterschieden
[6] Als Dyspnoe wird eine unangenehm erschwerte Atemtätigkeit bezeichnet
[7] Mit Feature-Set bezeichnen wir eine Sammlung von Merkmalen und Eigenschaften zum Userverhalten auf der Suchmaschine
[8] Mit Browsing wird hier das Verhalten des Users bei der Navigation durch die Suche beschrieben

Ansatz (1): Aufbereitung der Suchanfrage Die Solr-Suche bietet eine Boost-Funktion namens *DisMax Query Parser* (siehe [**DisMax**]). Mit dieser können basierend auf Feldwerten, einzelne Dokumente besser im Suchergebnis positioniert werden. Die Boost-Funktion müssten wir in der Springermedizin-Applikation einbauen und zwar im Aufbau der Suchanfrage für die Solr-Suche. Dieser Ansatz beinhaltet einige Gefahren, die wir beachten müssten.

Dazu zählen beispielsweise die Abhängigkeiten von anderen *Boost-Faktoren*[9]. Alle Boost-Faktoren hängen voneinander ab und müssten bei jeder Ergänzung um neue Faktoren normalisiert werden, um kein „über-Boosting“[10] einzelner Faktoren zu riskieren. Zudem besteht die Gefahr des „blinden Boosting“ von Dokumenten. Die Solr-Relevanzberechnung ist komplex und der Einfluss des *Boosting* in die Solr-Relevanzberechnung schwer erkennbar. Auch hat Springermedizin bereits sehr schlechte Erfahrungen mit Boosting gemacht und bevorzugt einen Lösungsansatz ohne Boosting.

Ansatz (2): Suchindex-Erweiterung in der Solr-Suche Um die CTR direkt in die Solr einzubeziehen gibt es zwei Varianten. Wir können das *Schema des Suchindexes* über die Schema API (siehe [**SchemaAPISolr**]) erweitern (1) und alle Einträge neu indexieren, oder wir ergänzen den Index um ein *externes Feld* (ExternalFileField, siehe [**ExtFieldSolr**]) (2).

Beide Lösungsansätze ergeben nur bei der Speicherung einer einfachen *Click-Count Popularität*[11] Sinn. Diese genügen allerdings den hier gegebenen Anforderungen nicht, da die CTR abhängig vom Suchterm ist. Der erste Lösungsansatz ist zudem besonders heikel, weil bei jeder Änderung des Click-Count-Wertes, das Dokument in der Solr neu indexiert werden muss.

Der in dieser Arbeit verfolgte Ansatz: Aufbereitung der Suchresultate anhand eines Klick-Modell basierten Algorithmus

Wir verfolgen in dieser Arbeit den Ansatz der Aufbereitung der Suchresultate aus der Solr-Suche mithilfe des PBM basierten Algorithmus. Dieser soll die Suchergebnisliste analysieren, die CTR der Dokumente berechnen und die Liste neu sortieren.

Warum verwenden wir einen Reranking-Algorithmus? Mit dem Reranking-Algorithmus greifen wir zu einem Zeitpunkt in die Suche ein, wo der bisherige Suchprozess der Springermedizin-Suche bereits abgeschlossen ist und die Aufbereitung des Suchresultats noch nicht begonnen hat. Folglich können wir den Algorithmus als ein in sich geschlossenes, unabhängiges Modul zwischen dem Suchprozess und der Aufbereitung des Suchresultats einbinden. Das erleichtert uns die Implementierung und wir sind unabhängig von der Springermedizin-Suchlogik. Ändert sich die Logik der Suche, beeinflusst das zwar die Suchergebnisliste, aber nicht die Logik von unserem Algorithmus und umgekehrt. Dadurch können wir Änderungen in unserer Logik schnell und einfach implementieren. Zudem haben wir zu dem Zeitpunkt, wo unser Algorithmus eingreift, alle wichtigen Informationen zum Suchergebnis. Wir wissen nicht nur die Suchanfrage, sondern auch das Suchergebnis der Solr und können diese Informationen im Algorithmus verwenden.

Die Grundlagen des Algorithmus

Worauf basiert unser Algorithmus? Den PBM basierten Algorithmus werden wir auf Basis des in der Studie [**pbm**] vorgestellten Position-Based Klick-Modells aufbauen. Als Basis für die Definition der Formel-Symbole verwenden wir das für eine Konferenz angefertigte Formel-Tutorial zu der gleichen Studie (siehe [**pbmTutorial**]). Die hier erwähnte Studie gibt uns einen schönen Überblick über die wichtigsten Klick-Modelle und vergleicht diese, in einigen aufschlussreichen Tests. Aus den Ergebnissen dieser Tests lässt sich ein Profil der Stärken und Schwächen des zu untersuchenden PBM erstellen.

[9] Die Solr besitzt eine Boosting-Funktion, um bestimmte Wertübereinstimmungen in der Suche höher gewichtet zu können
[10] Bezeichnet die über-priorisierte Bewertung einzelner Faktoren
[11] Kennzahl für alle Klicks auf ein Dokument unabhängig des Suchterms

Anhand dieses Profils und der Ergebnisse unserer Evaluation in Kapitel 5, können wir am eine Ende ein Fazit ziehen, wie gut der Algorithmus im Springermedizin-Umfeld funktioniert und ob ein anderes Klick-Modell vielleicht besser geeignet gewesen wäre.

Die Eckpunkte der Ergebnisse der Studie zum PBM In der angesprochene Studie [**pbm**] konnte das PBM vor allem mit seiner *Vorhersagegenauigkeit der Relevanz-Bewertung* eines Dokumentes im Suchergebnis (AUC-Messung) überzeugen. Dabei wurde verglichen wie nahe die Relevanz-Bewertung des Modells, der des Bewerters kommt. Ebenfalls sehr gut abgeschlossen hat das Modell bei der Überprüfung, wie gut ein Modell *Klicks einer aktuellen User-Sitzung* in der Klick-Wahrscheinlichkeitsberechnung berücksichtigt (Log-likelihood) und wie hoch die Wahrscheinlichkeit ist, dass das Modell mit einem Klick auf eine beliebige Position im Suchresultat gerechnet hat (Perplexity). Die Studie stellte zudem anhand der Ergebnisse des Log-likelihood fest, das komplexe Klick-Modelle das Userverhalten besser interpretieren können als einfache Klick-Modelle, die nur die Klick-Häufigkeiten zusammenzählen. Zu den komplexen Klick-Modellen zählt auch das PBM. Weniger gut zeigte sich das Modell bei der Prognose der aktuellen CTR anhand von Trainings-Daten. In den Tests wurden die Dokumente an anderen Positionen dargestellt, als in den Trainingsdaten antrainiert. In der Studie wird darum vermutet, dass das PBM sich zu sehr auf die Position des Dokumentes in den Trainingsdaten festgefahren hat.

Schlussfolgerungen der Ergebnisse der Studie zum PBM Das PBM zeigt gute Werte bei der Einschätzung der effektiven Userrelevanz eines Dokumentes zur Suchanfrage. Das sind gute Voraussetzungen für den von uns verfolgten Ansatz, die Suche mittels Userrelevanzen zu optimieren. Anhand der Prognosegüte müssen wir jedoch auch feststellen, dass die Position eines Dokumentes starken Einfluss auf das PBM haben kann. Springermedizin hat einen ständig wachsenden und sich verändernden Content-Pool. Wir müssen darum davon ausgehen, dass eine zu starke Positionsabhängigkeit der Dokumente, das Suchresultat negativ beeinflussen wird. Wir sollten daher im Reranking-Algorithmus die Klick-Wahrscheinlichkeiten der Position weniger stark gewichten als die Dokument-Wahrscheinlichkeit. Ein weitere interessante Analyse in der Studie ist die nDCG-Auswertung[12]. Wir werden in unserer Evaluation ebenfalls mit dieser Metrik arbeiten. In der Studie hat das PBM mittelmäßig abgeschlossen. In der Auswertung der Evaluation werden wir vergleichen können, wie gut der nDCG-Wert unseres PBM basierten Algorithmus im Springermedizin-Umfeld im Vergleich zu dem, der Studie sein wird.

Aufbau der PBM-Formel Das PBM setzt sich aus zwei einfachen Klick-Modellen zusammen, dem *Rank-Based CTR* (RCTR) und dem *Document-Based CTR* (DCTR). Einfache Klick-Modells berechnen eine Klick-Wahrscheinlichkeit direkt anhand der CTR. Diese kann aus dem Feature-Set der Click-Through-Daten gelesen werden und stellt das Verhältnis zwischen *Klicks auf ein bestimmtes Objekt* und der *Anzahl Klicks gesamt* dar.

Die Wahrscheinlichkeit $P(E_{r_u})$ aus dem RCTR-Modell berechnen Das RCTR berechnet die Wahrscheinlichkeit $P(E_{r_u})$, dass ein Dokument aufgrund seiner Position im Suchresultat angeklickt wird. Das Anklicken eines Dokumentes aufgrund seiner Position, wird hierbei als E_{r_u} bezeichnet. Die Formel wird wie folgt definiert:

$$P(E_{r_u}) = \frac{1}{|S|} \cdot \sum_{s \in S} c_r \tag{2.1}$$

Aus den Click-Through-Daten von Springermedizin können wir nicht alle Suchvorgänge (Impressionen) zu einem Suchterm lesen. Wir können nur die Suchvorgänge s lesen, bei welchen effektiv auf ein Objekt geklickt wurde. Mit einem Suchvorgang bezeichnen wir hierbei einen *kompletten Ablauf einer Suche* zu einem Suchterm q, inklusive der Präsentation des Suchresultates und dem möglichen Klick auf ein Dokument u im Suchresultat. S beschreibt bei uns deshalb eine Sammlung aller Suchvorgänge s, bei denen ein Dokument im Suchresultat angeklickt wurde. c_r beschreibt die Anzahl aller Klicks

[12] Der nDCG ist eine Metrik zur Messung der Qualität des Rankings einer Suche anhand der Reihenfolge der Suchergebnisse, verglichen mit den Relevanz-Werten derselben Suchergebnisse

auf die Position r im Suchergebnis aller Suchvorgänge s aus der Menge S. S ist hierbei unabhängig des Suchterms. Das heißt, die Click-Through-Daten beschränken sich *nicht* auf die für den Suchterm relevanten Click-Through-Daten.

Die Wahrscheinlichkeit $P(A_u)$ aus dem DCTR-Modell berechnen Das DCTR berechnet die Wahrscheinlichkeit $P(A_u)$, dass ein bestimmtes Dokument u aufgrund seines Suchsnippets angeklickt wird. Das Anklicken eines Dokumentes aufgrund seines Suchsnippets, wird hierbei als A_u bezeichnet. Die Formel wird wie folgt definiert:

$$P(A_u) = \frac{1}{|S_{uq}|} \cdot \sum_{s_q \in S_{uq}} c_u, \text{ where, } S_{uq} = \{s_q : u \in s_q\} \tag{2.2}$$

Bei dieser Wahrscheinlichkeitsberechnung konzentrieren wir uns nur auf die zum Suchterm q relevanten Click-Through-Daten. S_{uq} beschreibt hierbei eine Sammlung aller Suchvorgänge s_q, bei denen ein Dokument im Suchresultat zum Suchterm q angeklickt wurde. c_u beschreibt die Anzahl aller Klicks auf das Dokument u im Suchergebnis aller Suchvorgänge s_q aus der Menge S_{uq}. Mit der Definition $S_{uq} = \{s_q : u \in s_q\}$ wird beschrieben, dass für S_{uq} die Bedingung gilt, dass das Dokument u, bei jedem Suchvorgang s_q, im Suchresultat präsentiert wurde (ein Element des Suchergebnisses in s_q war). Anhand unserer Click-Trough-Daten können wir diese Annahme leider nicht prüfen. Deshalb müssen wir einfach davon ausgehen, dass bei allen Click-Trough-Daten zu den Suchvorgängen s_q, das Dokument u im Suchresultat erschienen ist.

Die effektive Klick-Wahrscheinlichkeit $P(C_u)$ aus dem PBM-Modell berechnen Das PBM geht davon aus, dass ein Dokument u nur angeklickt wird, wenn es untersucht wurde E_{r_u} und dabei vom Suchenden als relevant bewertet wurde A_u. Die Formel zu C_u wird wie folgt definiert:

$$C_u = (E_{r_u} \cdot A_u) \tag{2.3}$$

Daraus und aus der Tatsache, dass E_{r_u} und A_u immer ≤ 1 sind, lässt sich folgende Equivalenz definieren:

$$C_u = 1 \Leftrightarrow (E_{r_u} = 1 \text{ und } A_u = 1) \tag{2.4}$$

2.2 Zusammenfassung

Nicht beeinflussbare Faktoren in der Auswertung beachten Bei der Analyse der Problemstellungen und Teilprobleme mussten wir feststellen, dass wir bedingt durch den Springermedizin-Content, nicht beeinflussbare Faktoren in der Suche haben. Diese können das Qualitätsmaß unseres Algorithmus negativ beeinflussen. Diese Faktoren müssen wir in der Auswertung der Evaluation berücksichtigen und deswegen auf langfristige Sicht beurteilen, ob der Ansatz funktionieren kann und die gewünschten Optimierungen in der Suche bringt.

Stand der Technik reicht in diesem Umfeld nicht Wir haben bewiesen, dass der Stand der Technik für unseren verfolgten Lösungsansatz des Reranking-Algorithmus nicht ausreicht bzw. das Springermedizin-Umfeld die notwendige Datengrundlage nicht bietet, um diesen so umzusetzen. Wir werden darum basierend auf den vorgestellten Lösungsansätzen, einige Modifikationen am Algorithmus vornehmen müssen. Bei der Aufarbeitung der Grundlagen zum verwendeten PBM haben wir die Stärken und Schwächen unseres Algorithmus kennengelernt. Wir haben festgestellt dass das PBM dazu neigt, die Positionsabhängigkeit zu stark zu priorisieren und der Content des Springermedizin-Umfeldes dafür eine schlechte Basis bietet, da sich dieser stetig verändert. Wir haben darum Lösungsansätze entworfen um diesem Problem entgegenzuwirken. Diese werden wir im nächsten Kapitel verarbeiten. Bei der Auswertung werden wir dann sehen, ob diese den gewünschten Effekt gebracht haben.

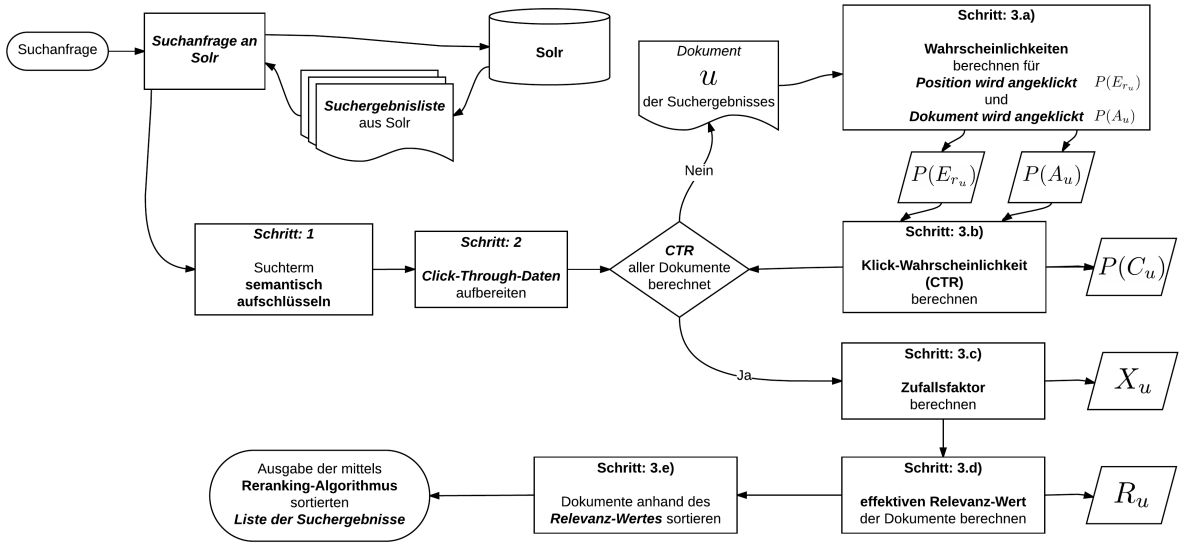
3 Reranking mittels CTR

Aufgrund der Diskussionen zu den Problemstellungen des Lösungsansatzes im vorhergehenden Kapitel 2, kennen wir verschiedene Varianten der Lösungsansätze und deren Eigenschaften. In diesem Kapitel geht es nun darum, aus diesem Wissen unseren detaillierten Lösungsansatz des Reranking-Algorithmus zusammenzubauen. Dazu werden wir uns zuerst den Prozess des neuen Reranking-Algorithmus anschauen. Wie wir zu diesem Prozessbild kommen, lernen wir anschließend in der Methodik kennen. Diese baut auf der Methodik aus Kapitel 1.4 auf. Dabei werden wir ausführlich die Beweggründe zum gewählten Lösungsansatz kennenlernen und die theoretische Umsetzung diskutieren. Im nächsten Kapitel folgt dann die praktische Umsetzung des hier ausgearbeiteten Lösungsansatzes.

3.1 Prozessaufbau des Lösungsansatzes

3.1.1 Prozessaufbau als Bild

Abb. 5: Prozessaufbau des Lösungsansatzes



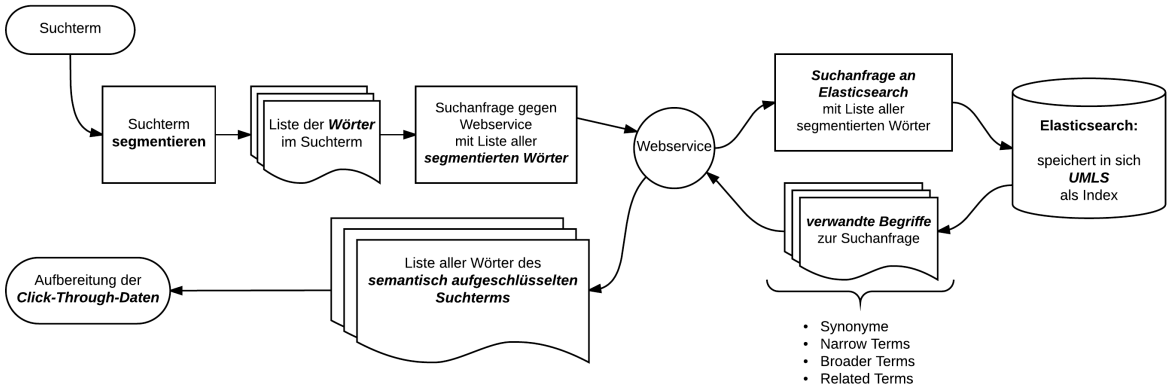
3.2 Methodik

Wie bereits in der Einführung zu diesem Kapitel besprochen, haben wir gelernt wie Click-Through-Daten entstehen, wie sie zu lesen sind und wie wir Aussagen zu ihnen treffen können. Nun werden wir mithilfe dieses Wissens, die CTR der Dokumente berechnen. Mithilfe der berechneten Click-Through-Daten werden wir dann ein *Reranking* der Suchresultate durchführen, bevor diese dem User präsentiert werden. So wollen wir die Userrelevanz mittels CTR in die Suche einbinden. Die Vorgehensweise dazu sieht wie folgt aus.

3.2.1 Suchterm Segmentierung

Um alle relevanten Click-Through-Daten lesen zu können, müssen wir zunächst den Suchterm auftrennen, wie in Kapitel 2.1.1 angesprochen. Der Prozess dazu sieht wie folgt aus:

Abb. 6: Prozess der semantischen Segmentierung des Suchterms



Suchterm semantisch aufschlüsseln mittels Segmentierung

Die Auftrennung des Suchterms in die einzelne Worte können wir mithilfe einer Segmentierung[1] durchführen. Hier könnten wir uns überlegen, zusätzlich mit Stoppwörtern[2], nicht relevante Wörter aus dem Suchterm zu entfernen. Dieses Verfahren macht aber im Springermedizin-Kontext keinen Sinn. Wie in Kapitel 1.2 angesprochen, suchen die User der Springermedizin-Applikation oft mit einschlägig, fundierten Fachbegriffen. Wir gehen darum davon aus, dass alle Wörter des verwendeten Suchterms für das Suchergebnis relevant sind. Diese Erkenntnis basiert auf Aussagen der Redakteure von Springermedizin und Webtrekk-Analysen der meist gesuchtesten Suchtermen der letzten Monate. Auch sind Stoppwörter veraltet und werden in modernen Information Retrieval Verfahren nicht mehr eingesetzt. Wir verzichten darum auf den Einstatz von Stoppwörtern.

Suchterm semantisch erweitern mittels Thesaurus

Wie ebenfalls in Kapitel 2.1.1 thematisiert, wollen wir die Click-Through-Daten zu unserem Suchterm um Click-Through-Daten zu verwandten Begriffen erweitern. Für diese semantische Erweiterung eines Suchwortes werden wir einen Thesaurus verwenden. Die Erweiterung umfasst zum Suchterm gleichbedeutende Begriffe (*Synonyme*), sehr ähnliche Begriffe (*Narrow Terms*), ähnliche Begriffe im weiteren Sinne (*Broader Terms*) und verwandte Begriffe (*Related Terms*).

Mittels Webservice einen UMLS-Thesaurus nach relevanten Begriffen durchsuchen Springer Nature besitzt einen Webservice mit welchem auf den Thesaurus *Unified Medical Language System* (UMLS, siehe [**UMLS**]) zugegriffen werden kann. Der Webservice nimmt einzelne Wörter und Wörter-Listen entgegen. Zu jedem dieser Wörter durchsucht der Webservice den Thesaurus nach den oben erwähnten Arten von verwandten Begriffen. Der Webservice verwendet für diese Suche eine Elasticsearch[3] (siehe [**elasticsearch**]). Die dabei gefundenen Begriffe, liefert der Webservice als Antwort zurück. Um alle relevanten Click-Through-Daten zu finden, werden wir mit dem segmentierten Suchterm eine Anfrage gegen diesen Webservice stellen und anschließend den segmentierten Suchterm um die gefundenen Begriffe erweitern. Mithilfe des erweiterten Suchterms können wir dann anschließend eine Analyse in Webtrekk starten und alle relevanten Click-Through-Daten lesen.

[1] Bezeichnet die Aufteilung in Abschnitte, in diesem Fall in einzelne Worte
[2] Stoppwörter sind Wörter, die sehr häufig auftreten und für gewöhnlich keine Relevanz für den Dokumentinhalt besitzen
[3] Eine Elasticsearch ist eine Volltextsuchmaschine

3.2.2 Aufbereitung Click-Through-Daten

Jeder Klick auf ein Dokument ist relevant

Wie in Kapitel 2.1.2 beschrieben, reichen Webtrekk-Analysen für komplexe Auswertungen der Click-Through-Daten nicht aus. Wir können darum in dieser Arbeit *Feedback-Strategien* für die CTR Auswertung, wie in [**Joachims**] beschrieben, nicht verwenden. Stattdessen greifen wir wie ebenfalls in Kapitel 2.1.2 beschrieben auf die Click-Through Features zu, die uns Webtrekk zur Verfügung stellt. Daraus entsteht das folgende, interpretierbare Feature-Set:

Tab. 3.1: Interpretierbares Feature-Set aus den Webtrekk Click-Through-Daten

Click-Through Features	
Feature	**Beschreibung**
Position	Position dieses Dokumentes im Suchergebnis (angeklickte Position)
ClickFrequency	Anzahl Klicks für dieses Dokument zum angefragten Suchterm (Klick-Häufigkeit)
ClickProbability	Klick-Wahrscheinlichkeit zum Suchterm (*ClickFrequency* geteilt durch *Gesamtanzahl der Klicks zum angefragten Suchterm*)

Wie wir sehen, ist unser Feature-Set im Vergleich zu dem in [**IWUSBI**] beschriebenen, sehr stark eingeschränkt und enthält keine Relevanz-Informationen um ein Relevanz-Feedback zu den Klick-Häufigkeiten daraus ableiten zu können. Wir müssen wir darum davon ausgehen, dass jeder Klick auf ein Dokument relevant ist.

Gewichtung der Click-Through-Daten

Durch die semantische Aufschlüsselung des Suchterms haben wir verschieden starke Relationen zwischen Click-Through-Daten und dem Suchterm. Die Gewichtung der Stärke dieser Relation ist aber nicht Kern dieser Arbeit. Wir gehen darum davon aus, dass unabhängig der stärke der Relation zum Suchterm, alle Click-Through-Daten eine gleiche Relevanz besitzen.

Berechnung der CTR

Einfache CTR ignoriert viele Problemstellungen der Interaktion zwischen dem User und der Suche Wie wir bereits wissen, stellt die CTR die Anzahl der Klicks auf ein Dokument im Verhältnis zu den gesamten Impressionen dar. Bezogen auf das in Kapitel 2.1.2 angesprochene Feature-Set, würden wir die *ClickProbability* direkt als CTR verwenden. Dazu müssten wir nur die Click-Through-Daten eines Dokuments ins Verhältnis zu allen Click-Through-Daten für einer Suchanfrage stellen. Wie wir aber bereits in Kapitel 2.1.1 gelernt haben, würden wir damit viele Problemstellungen der Interaktion des Users mit der Suche ignorieren. Deswegen haben wir uns für eine Lösung basierend auf einem Klick-Modell entschieden.

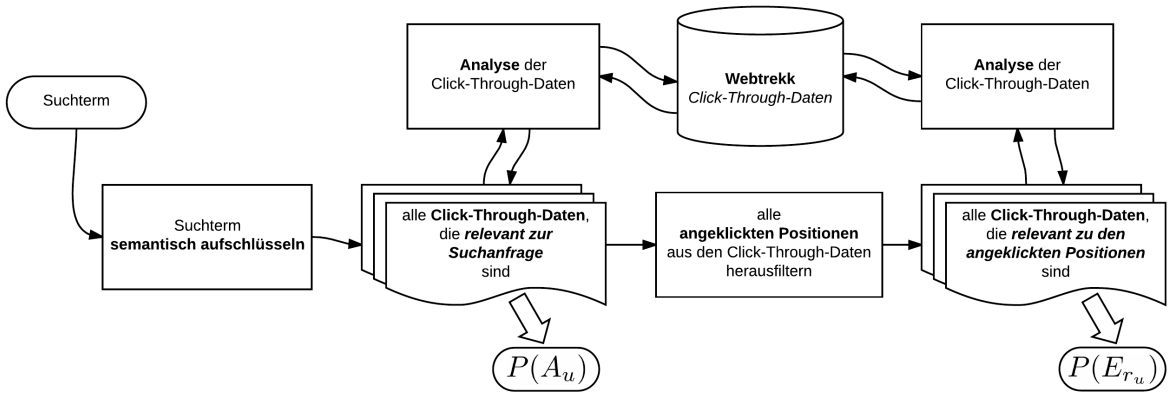
Klick-Wahrscheinlichkeit als CTR für Reranking verwenden Klick-Modelle versuchen die Click-Through-Daten zu interpretieren und aus ihnen ein Relevanz-Feedback zu schlussfolgern. Das machen sie mithilfe des bereits angesprochenen Feature-Sets der Click-Through-Daten. Aus dem Feature-Set von Springermedizin können wir zwei wichtige Informationen zu jeder Suchanfrage ermitteln. Wir wissen welches Dokument auf welcher Position im Suchresultat angeklickt wurde und wir kennen die Reihenfolge der Dokumente im Suchresultat der Solr. Der *PBM basierte Algorithmus* baut genau auf diesen Click-Through-Informationen auf und berechnet daraus eine Klick-Wahrscheinlichkeit, welche wir anstelle der einfachen CTR, für das Reranking der Suchergebnisliste verwenden können.

3.2.3 Result-Reranking mittels PBM basiertem Algorithmus

Click-Through-Daten für Positions- und Dokument-Wahrscheinlichkeit abfragen

In Kapitel 2.1.3 haben wir gelernt, dass sich das PBM aus der Dokument-Wahrscheinlichkeit $P(A_u)$ und der Positions-Wahrscheinlichkeit $P(E_{r_u})$ zusammensetzt. Die dabei vorgestellten Formeln zur Berechnung der beiden Wahrscheinlichkeiten verwenden unterschiedliche Click-Through-Daten. Der Prozess für das Lesen der relevanten Click-Through-Daten sieht wie folgt aus:

Abb. 7: Click-Through-Daten für Wahrscheinlichkeitswerte lesen



Click-Through-Daten der Dokument-Wahrscheinlichkeit $P(A_u)$ filtern Für die Berechnung der Dokument-Wahrscheinlichkeit verwenden wir das weiter oben in Kapitel 3.2.2 angesprochene Verfahren zur Suchterm-Segmentierung, um *alle für die Suchanfrage relevanten Click-Through-Daten* zu lesen. Aus diesen Click-Through-Daten können wir mithilfe der vorgestellten Formel für $P(A_u)$ die Wahrscheinlichkeit berechnen, dass ein Dokument aufgrund seines Suchsnippets im Suchergebnis angeschaut wird.

Click-Through-Daten der Positions-Wahrscheinlichkeit $P(E_u)$ filtern Für die Berechnung der Positions-Wahrscheinlichkeit filtern wir aus den für die Dokument-Wahrscheinlichkeit gelesenen Click-Through-Daten *alle angeklickten Positionen* heraus. Zu diesen Positionen lesen wir dann unabhängig des Suchterms, alle Click-Through-Daten, in denen auf diese Positionen geklickt wurde. Aus den resultierenden Click-Through-Daten können wir mithilfe der vorgestellten Formel für $P(E_{r_u})$ die Wahrscheinlichkeit berechnen, dass ein Dokument aufgrund aufgrund seiner Position im Suchergebnis angeschaut wird.

Klick-Wahrscheinlichkeit mit PBM berechnen

Wie in Kapitel 2.1.3 angesprochen, werden wir unseren Reranking-Algorithmus in die Aufbereitung der Suchresultate aus der Solr-Suche integrieren. Wir werden die Click-Through-Daten wie im vorherigen Teil der Methodik besprochen aufbereiten. Auf diesen werden wir dann unseren Reranking-Algorithmus anwenden, um die CTRs der einzelnen Dokumente des Suchergebnisses zu berechnen.

Reranking-Algorithmus kann nur die Top-N-Ergebnisse betrachten Wir müssen bei unserem Ansatz beachten, dass die Solr durch die Pagination-Funktion (siehe [**Pagination**]) nur die Top-N-Ergebnisse zurückgibt. Dadurch sehen wir nur einen Teil der Suchergebnisse. Um sicherzustellen, dass wir möglichst viele relevante Suchergebnisse berücksichtigen, werden wir die ersten 100 Suchergebnisse der Solr im Reranking-Algorithmus verarbeiten. Aus der daraus resultierenden Liste der Suchergebnisse, filtern wir die ersten 20 Ergebnisse und stellen diese dar. Für die Untersuchung des Reranking-Algorithmus werden wir uns bei der Auswertung jeweils auf die Seite 1 der Suchergebnisse konzentrieren. Bei Springermedizin somit auf die ersten 20 Suchresultate. Die Pagination der Folgeseiten der Suchresultate werden wir

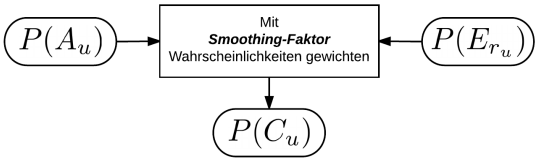
nicht untersuchen. Würden wir diesen Algorithmus in einer Live-Applikation[4] implementieren wollen, müssten wir uns für die Ausspielung der Folgeseiten des Suchresultats einen zusätzlichen Lösungsansatz überlegen. Mit diesem müsste erreicht werden, dass die Solr die durch den Reranking-Algorithmus bereits ausgespielten Suchresultate, nicht mehrfach ausspielt.

Ausarbeitung des effektiven Algorithmus Unseren Reranking-Algorithmus bauen wir auf der in Kapitel 2.1.3 vorgestellten Formel des PBMs auf. Wie wir in der Analyse der Grundlagen in Kapitel 2.1 festgestellt haben, reicht die triviale Umsetzung unseres Klick-Modells für unseren Algorithmus nicht aus. Wir haben darum verschiedene Lösungsansätze für die Problemstellungen vorgestellt. Mithilfe einiger dieser Lösungsansätzen, wollen wir nun den effektiven Algorithmus ausarbeiten.

Smoothing-Faktor in PBM einführen

Wir wissen dass eine Wahrscheinlichkeit einen Wert zwischen 0 und 1 besitzt. Dadurch können Nullwerte entstehen. Das PBM multipliziert die Positions- und Dokument-Wahrscheinlichkeit miteinander, um die Klick-Wahrscheinlichkeit zu berechnen. Wir müssen aber davon ausgehen, dass es Dokumente im Suchresultat geben kann, deren Position nie angeklickt wurde und umgekehrt. Multiplikationen mit Null ergeben immer einen Nullwert. Wir führen darum an dieser Stelle einen *Smoothing-Faktor* ein:

Abb. 8: Berechnung der CTR mittels PBM



Mit Smoothing-Faktor Wahrscheinlichkeitswerte gewichten Der Smoothing-Faktor soll zwei Probleme lösen. Zum Einen wollen wir einen Wahrscheinlichkeitswert trotz der Multiplikation mit Null beachten (1). Zum Anderen wollen wir die im vorherigen Absatz beschriebene Gewichtung abhängig des Relevanz-Feedbacks, in den Algorithmus einbeziehen (2). Wir transformieren dazu das Produkt der beiden Wahrscheinlichkeiten in eine gewichtete Summe, indem wir eine *Smooting-Konstante* einführen. Die Smoothing-Konstante λ soll die Gewichte der beiden Klick-Wahrscheinlichkeiten $P(E_{r_u})$ und $P(A_u)$ zu eins aufsummieren. Wir verwenden dazu das *Exponential Smoothing Modell* (siehe [**ExpSmoothing**]). Bilden wir dieses Modell auf den PBM basierten Algorithmus ab, sieht die Formel für die CTR (= $P(C_u)$) wie folgt aus:

$$P(C_u) = \lambda \cdot P(E_{r_u}) + (1 - \lambda) \cdot P(A_u) \tag{3.1}$$

Smoothing-Faktor abhängig der Position im Suchresultat definieren

Die in Kapitel 2.1.3 angesprochene Studie [**pbm**] hat aufgezeigt, dass das PBM sich stark an der Position eines Dokumentes im Suchresultat orientiert und dies negative Einflüsse auf die Prognose der CTR haben kann. Die Klick-Wahrscheinlichkeit $P(E_{r_u})$ darf darum nicht zu viel Einfluss haben. Aus eigener Erfahrung wissen wir zudem, dass die ersten Dokumente im Suchresultat immer zuerst gesehen werden. Die dahinter gelisteten Dokumente werden fortlaufend analysiert. Dies bestätigt die in Abb. 4 dargestellte Analyse der Klicks auf die ersten 20 Positionen eines Suchergebnisses. Wir sollten darum darauf achten, dass je *schlechter* die Position des angeklickten Dokumentes im Suchresultat der Solr ist, desto *höher* das Relevanz-Feedback zu bewerten ist.

Das machen wir, indem wir für die Berechnung der CTR des Dokumentes, den Smoothing-Faktor abhängig der Position im Suchresultat definieren. Dadurch können wir das Verhältnis zwischen Klick-Wahrscheinlichkeit der Position und Klick-Wahrscheinlichkeit des Dokumentes steuern. Als

[4] Mit Live-Applikation wird hier eine öffentliche, für Kunden zugängliches Applikation beschrieben

Grundlage hierbei dient uns die Position des Suchresultats der Solr. Wie die Aufteilung dazu aussehen wird, sehen wir folgend:

Tab. 3.2: Smoothing-Faktor abghängig der Position im Suchergebnis

Position	Verhältnis Position zu Dokument	Wert des Smoothing-Faktors λ
1 bis 10	1:1	0.50
11 bis 20	1:2	0.34
größer 20	1:3	0.25

Für die Suchresultate mit einer Position über 20, verstärken wir die Gewichtung der Klick-Wahrscheinlichkeit des Dokumentes erheblich. Wir gehen davon aus, dass bei Klicks auf Dokumente mit einer solch hohen Position, die suchende Person die Suchresultate genau analysierte, bevor sie ein Dokument angeklickt hat.

3.2.4 Vergessen der alten Daten

Ein Algorithmus zur Berechnung von Wahrscheinlichkeiten muss sich ein gewisses Grundwissen aneignen. Dies geschieht üblicherweise durch Trainingsdaten. Genauso muss er alte Daten wieder vergessen können, um Overfitting[5] zu vermeiden.

Durch Webtrekk ist kein komplexer Lern-Algorithmus notwendig

Durch Webtrekk haben wir eine Wissensbasis, die sich stetig und zeitnah aktualisiert. So muss der Algorithmus nicht ständig neues Wissen lernen und altes vergessen, sondern er kann direkt auf diese Wissensbasis zugreifen. Dies geschieht, indem zur Laufzeit[6] Analysen gegen Webtrekk über eine frei definierbare Periode gemacht werden. Dadurch kann *Overfitting* durch veraltete Daten vermieden werden. Deshalb verwenden wir keinen komplexen Lern-Algorithmen wie in [**IWUSBI**] vorgestellt.

Die Klick-Wahrscheinlichkeit ist kein absoluter Wert für die Userrelevanz

Nun könnten wir die Klick-Wahrscheinlichkeit als absoluten Wert für die *Userrelevanz* betrachten. Dies wäre jedoch falsch, wie in Kapitel 2.1.1 analysiert, müssen wir davon ausgehen, dass viele User der Qualität der Suchmaschine vertrauen. Diese betrachten die „Top-Suchresultate“ als die relevanten Suchresultate. Denkbar wäre auch, dass User unabsichtlich das falsche Dokument anklicken und dadurch die CTR eines Dokumentes verfälschen. Dadurch kann ein *Overfitting* des Algorithmus durch nicht relevante Click-Through-Daten entstehen.

Overfitting vermeiden

Um das angesprochene Overfitting zu vermeiden, darf der Algorithmus nicht immer anschlagen. Wir müssen sicherstellen, dass vereinzelt *zufällige Dokumente* in den „Top-Suchresultaten“ angezeigt werden. So können auch andere Dokumente in den Fokus des Users gerückt werden. Das System fährt sich dadurch nicht auf falschen Annotationen fest.

Zusätzliche Varianz durch Zufallsfaktor Mithilfe eines Zufallsfaktors kann eine solche Varianz[7] in den Klick-Modell basierten Algorithmus gebracht werden. Konkret wollen wir den Reranking-Algorithmus beeinflussen, indem wir mithilfe eines Zufallsfaktors, Abweichungen von der berechneten Klick-Wahrscheinlichkeit eines Dokumentes provozieren. Den dazu verwendeten Zufallswert X_u, lassen wir uns als positive ganze Zahl generieren. Er soll eine Zufallsposition im Suchergebnis darstellen:

$$X_u \in \{\text{Positionen im Suchresultat}\} \subset \mathbb{N} \tag{3.2}$$

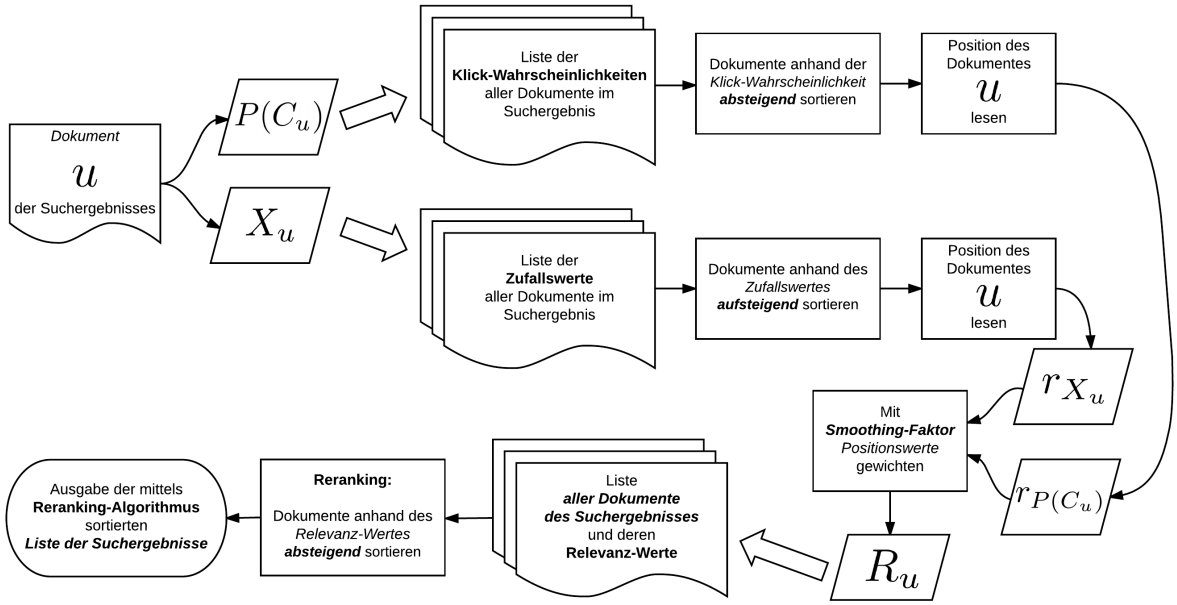
[5] Überanpassung des Algorithmus durch zu viele (falsche oder veraltete) Daten
[6] Unter Laufzeit wird in diesem Fall der Zeitpunkt der direkte Abfrage während der Suchanfrage bezeichnet
[7] Mit Varianz bezeichnen wir hierbei eine Abweichung vom berechneten Erfahrungswert

Zufallsfaktor mittels Smoothing-Faktor in die Sortierung der Suchergebnisse einbauen

Mit unserem berechneten Zufallswert können wir nun die angesprochenen Abweichungen in die Klick-Wahrscheinlichkeiten der Dokumente einbauen. Wie bereits in Kapitel 2.1.1 thematisiert, werden viele Suchresultate nie und deren Position selten bis gar nicht angeklickt. Sie haben darum keine Click-Through-Daten. Deren Klick-Wahrscheinlichkeit ist entweder Null oder sehr klein. Der Zufallswert soll darum nur leichte Einflüsse in die Klick-Wahrscheinlichkeit haben.

Effektive Position im Suchergebnis mittels Smoothing-Faktor berechnen Ein direkter Einbau in die Klick-Wahrscheinlichkeitsberechnung ist jedoch schwierig, da die Wahrscheinlichkeitsverteilungen auf die Dokumente im Suchergebnis stark schwanken und sich der Einfluss des Zufallswertes dadurch stetig verändert. Wir werden darum die Klick-Wahrscheinlichkeit und den Zufallswert der Dokumente separat verarbeiten und zum Schluss die zwei daraus resultierenden Werte eines Dokumentes, mithilfe eines Smoothing-Faktors gewichtet aufsummieren. Diese Summe dient dann als Relevanz-Wert eines Dokumentes. Das Reranking der Dokumente werden wir anhand dieser Relevanz-Werte durchführen. Die Verarbeitung sieht wie folgt aus:

Abb. 9: Berechnung der effektiven Position im Suchergebnis mittels Smoothing-Faktor



Aus den Klick-Wahrscheinlichkeiten und den Zufallswerten der Dokumente des Suchergebnisses erzeugen wir zwei Listen und sortieren diese wie in der Abbildung beschrieben. Aus diesen Listen können wir für jedes Dokument u jeweils eine Position aus der Liste der Klick-Wahrscheinlichkeiten $r_{P(C_u)}$ und eine Position aus der Liste der Zufallswerte r_{X_u} lesen. Aus diesen beiden Positionswerten des Dokumentes berechnen wir den effektiven Relevanz-Wert des Dokumentes R_u. Dazu verwenden wir wieder das *Exponential Smoothing Modell* für den Smoothing-Faktor und berechnen damit die gewichtete Summe der beiden Positionswerte. Wie bereits weiter oben erwähnt, setzen wir den Smoothing-Faktor so an, dass der Zufallswert nur leichte Einflüsse in den effektiven Relevanz-Wert hat. Wir verwenden dazu einen Smoothing-Faktor zwischen 0.05 und 0.1. Den genauen Einfluss-Wert werden wir in der später folgenden Evaluation evaluieren. Folgend die Formel zur Berechnung des effektiven Relevanz-Wertes eines Dokumentes R_u:

$$R_u = \frac{1}{\lambda \cdot r_{X_u} + (1 - \lambda) \cdot r_{P(C_u)}} \tag{3.3}$$

Reranking des Suchresultats nach R_u-Wert der Dokumente Mithilfe dieser Berechnung erzeugen wir eine Liste von Tupeln mit zwei Komponenten (R_u, u), bestehend aus dem Relevanz-Wert R_u und dem Dokument u. Diese Liste sortieren wir nach R_u absteigend und erzeugen daraus eine neue Liste der in den Tupeln enthaltenen Dokumente u. Diese umsortierte Liste ist das Resultat des Rerankings mittels unserem PBM basierten Algorithmus. Diese präsentieren wir zum Schluss dem User, als Suchresultat zur Suchanfrage.

3.3 Zusammenfassung

Stark modifizierter Reranking-Algorithmus Mithilfe des PBM können wir trotz stark begrenzter Click-Trough-Informationen ein relativ gutes Relevanz-Feedback gewinnen, welches wir im Reranking-Algorithmus verwenden können. Das Grundgerüst der ausgearbeiteten Algorithmus basiert immer noch auf dem PBM. Wir mussten jedoch einige Modifikationen vornehmen, um den Algorithmus zu erhalten, mit welchem wir nun im nächsten Kapitel die Implementierung durchführen wollen. Diese Modifikationen sind auf die vielen identifizierten Problemstellungen und den daraus folgenden Einarbeitungen der Lösungsansätze in den Algorithmus zurückzuführen. Einige der Modifikationen sind sicherlich der Implementierung im Springermedizin-Umfeld geschuldet. Wir sollten aber davon ausgehen, dass in jedem anderen realistischen Umfeld, ähnliche Problemstellungen auftreten können. Diese praxisnahe Ausarbeitung des Algorithmus ist daher ein guter Test, um die praktische Einsatzfähigkeit des in diesem Kapitel vorgestellten, theoretischen Ansatz zu evaluieren.

Mittels Smoothing-Faktor Maximum an Informationen verwerten Ein wichtiges Hilfsmittel bei der Einarbeitung der Lösungsansätze in der Algorithmus, war der in diesem Kapitel eingeführte Smoothing-Faktor. Mithilfe des Smoothing-Faktors konnten wir die Nullwert-Problematik der Wahrscheinlichkeitsberechnung lösen und die Einflüsse der verschiedenen Faktoren in den Algorithmus besser kontrollieren. Dadurch konnten wir das Maximum an Informationen aus unserem PBM basierten Algorithmus herausholen.

In der Implementierung und der anschließenden Evaluation wird sich nun zeigen, ob der Lösungsansatz wie angedacht, umgesetzt werden kann und die angestrebten Verbesserungen des Qualitätsmaßes der Suche, erreicht werden können.

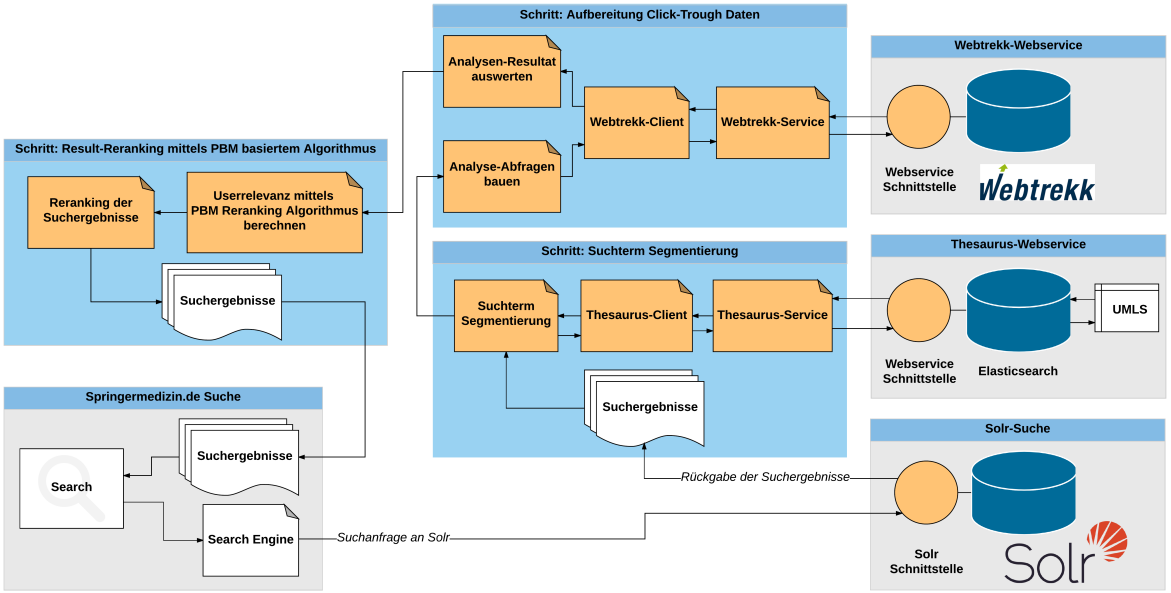
4 Implementierung

Im letzen Kapitel haben wir unseren Reranking-Algorithmus so detailliert ausgearbeitet, dass er nun implementiert werden kann. In diesem Kapitel geht es nun darum, diesen Algorithmus in der Springermedizin-Suche einzubauen. Mit der Implementierung wollen wir herausfinden, ob der theoretische Ansatz praktisch umgesetzt werden kann und die Gedankengänge bei der Ausarbeitung des Lösungsansatzes korrekt waren.Wir werden in diesem Kapitel nicht den Code der Lösung vorstellen. Wir werden aber beschreiben, wo wir in die Suche eingreifen, wie wir eingreifen und was wir genau machen, um dadurch einen Überblick über den implementierten Lösungsansatz schaffen zu können.

4.1 Architektur der Implementierung

Um zu verdeutlichen, an welcher Stelle des Suchprozesses der Springermedizin-Suche wir eingreifen, sehen wir unten folgend das Prozessbild der Implementierung unseres Lösungsansatzes. Warum wir genau an dieser Stelle eingreifen, haben wir bereits in 2.1.3 ausdiskutiert. Der Suchprozess ist im Prozessbild in mehrere Komponenten aufgeteilt. Die grau hinterlegten Komponenten zeigen bereits bestehende, vom Lösungsansatz unabhängige Teile der Architektur. Die blau hinterlegten Komponenten sind die in der Implementierung neu hinzugefügte Komponenten des Lösungsansatzes. Sie sind in die drei Hauptschritte des Reranking-Algorithmus unterteilt.

Abb. 10: Prozessbild der Implementierung



Wie wir in der Abbildung sehen können, wird zuerst der Suchvorgang auf der Solr durchgeführt, bevor wir die Suchergebnisliste entgegennehmen, verarbeiten und mit unserem Reranking-Algorithmus neu sortieren. Die daraus resultierende Ergebnisliste geben dann der Suche als Suchresultat zurück.

Verwendeter Technologiestack Die Springermedizin-Suche ist ein Teil der White Label Applikation von Springer Nature. Diese baut auf dem MVC-Framework *Play* [**Play**] auf und verwendet *Scala* [**Scala**] als Programmiersprache. Unsere Implementierung wird daher in Scala geschrieben und direkt in der White Label Applikation integriert.

4.2 Highlight: PBM basierter Reranking-Algorithmus

Der grafisch dargestellte Prozess in Abb. 10 entspricht hier nicht nicht der Reihenfolge, wie die Komponenten in der Suche aufgerufen werden, sondern der Reihenfolge der Verarbeitungsschritte im Prozess. In der effektiven Implementierung, übernimmt die Reranking-Komponente die Koordination der Verarbeitungsschritte. Sie wird im Suchprozess direkt *vor der Aufbereitung* der Suchergebnisse aufgerufen und nimmt die Liste der Suchergebnisse der Solr entgegen. Sind alle Schritte des Reranking-Algorithmus verarbeitet, gibt die Reranking-Komponente die *neu sortierte Liste* der Suchergebnisse zurück. Diese wird dann wieder von der Springermedizin-Suche für die Ausgabe als Suchergebnisse aufbereitet.

Pseudo-Code der Reranking-Komponente

Um den angesprochenen Vorgang des Rerankings der Suchergebnisse besser zu verstehen, sehen wir hier folgend den Programm-Ablauf der Reranking-Komponente als Pseudo-Code beschrieben:

Abb. 11: Pseudcode Reranking-Algorithmus

Algorithmus 1 : PBM basierter Reranking-Algorithmus

Daten : $searchTerm$ (Suchterm der Suchanfrage), $searchResult$ (zu verarbeitende Suchergebnisliste)
Ergebnis : $rerankedSearchResult \leftarrow$ durch Reranking-Algorithmus sortierte Suchergebnisliste

begin
  $keywords \leftarrow$ Segmentiere und erweitere Suchterm mittels Thesaurus
  $ctrClickDataBySearchTerm \leftarrow$ Lese Click-Through-Daten aus Webtrekk mithilfe von $keywords$
  if $ctrClickDataBySearchTerm$ *ist gefüllt* **then**
    $ranks \leftarrow$ Lese die angeklickten Positionen aus $ctrClickDataBySearchTerm$
    $ctrClickDataByRanks \leftarrow$ Lese Click-Through-Daten aus Webtrekk mithilfe von $ranks$
    if $ctrClickDataByRanks$ *ist gefüllt* **then**
      `/* Berechne Klick-Wahrscheinlichkeit` $P(C_u)$ `aller Dokumente` u `*/`
      for $u \in searchResult$ **do**
        $\lambda \leftarrow$ Definiere λ anhand des vordefinierten Gewichtungsfaktors für die Position r des Dokumentes u
        $P(E_{r_u}) \leftarrow$ Berechne Klick-Wahrscheinlichkeit für Position r des Dokumentes u
        $P(A_u) \leftarrow$ Berechne Klick-Wahrscheinlichkeit für Dokument u zu Suchterm $searchTerm$
        $P(C_u) \leftarrow \lambda \cdot P(E_{r_u}) + (1 - \lambda) \cdot P(A_u)$
      end
      $ranksByClickProbability \leftarrow$ Sortiere Liste $searchResult$ anhand der $P(C_u)$ Werte
      `/* Berechne Zufallswert` X_u `aller Dokumente` u `*/`
      for $u \in searchResult$ **do**
        $X_u \leftarrow$ Berechne Zufallswert zwischen 1 und $maxPosition(searchResult)$
      end
      $ranksByRandomValue \leftarrow$ Sortiere Liste $searchResult$ anhand der X_u Werte
      `/* Berechne effektiven Relevanz-Wert` R_u `aller Dokumente` u `*/`
      for $u \in searchResult$ **do**
        $\lambda \leftarrow$ Definiere λ anhand des vordefinierten Gewichtungsfaktors für den Zufallswert X_u
        $r_{P(C_u)} \leftarrow$ Lese Position des Wahrscheinlichkeits-Wertes $P(C_u)$ aus $ranksByClickProbability$
        $r_{X_u} \leftarrow$ Lese Position des Zufallswertes X_u aus $ranksByRandomValue$
        $R_u \leftarrow 1/\left(\lambda \cdot r_{X_u} + (1 - \lambda) \cdot r_{P(C_u)}\right)$
      end
      $rerankedSearchResult \leftarrow$ Sortiere Liste $searchResult$ anhand der R_u Werte
      `/* Rückgabe der durch den Reranking-Algorithmus umsortierten Suchergebnisse */`
      return *die ersten 20 Elemente von* $rerankedSearchResult$
    else
      `/* Keine Click-Through-Daten für die angeklickten Positionen gefunden ⇒ */`
      `/* Keine Umsortierung der Suchergebnisse mit Reranking-Algorithmus */`
      return *die ersten 20 Elemente von* $searchResult$
    end
  else
    `/* Keine Click-Through-Daten für Suchterm gefunden ⇒ */`
    `/* Keine Umsortierung der Suchergebnisse mit Reranking-Algorithmus */`
    return *die ersten 20 Elemente von* $searchResult$
  end
end

4.3 Highlight: Webtrekk-Analysen

Wie in der Einführung dieser Arbeit in 1.1 angesprochen, speichert Springermedizin seine Tracking-Daten auf Webtrekk. Für die Implementierung des Reranking-Algorithmus müssen wir die Click-Trough-Daten für die Wahrscheinlichkeits-Berechnungen, wie in 3.2.3 besprochen, mithilfe von Analysen aus Webtrekk abfragen und auswerten. Wie wir diese Analysen abfragen, sehen wir folgend.

Abfrage von Analysen

Der Webservice von Webtrekk bietet verschiedene Schnittstellenmethoden zum Export und Herunterladen der Tracking-Daten an. Die für uns relevante Schnittstellenmethode lautet *getAnalysisData*. Mithilfe dieser Methode können die Daten der verschiedenen Analysen per REST-Schnittstelle[1] abgerufen werden. Um zu definieren, welche Analyse und mit welchen Filtern diese Analyse abgefragt werden soll, müssen wir folgende Parameter der Methode mitgeben:

Request: *Springermedizin-Suche* $\rightarrow$ *Webtrekk REST-Webservice*

Tab. 4.1: Beschreibung Analyse-Aufbau Webtrekk-Schnittstelle

Parameter	Beschreibung
analysis	Gewünschte Analyse (alle Analysen der Weboberfläche von Webtrekk sind so abrufbar)
time_start	Startzeit des Analysezeitraumes
time_stop	Stopzeit des Analysezeitraumes
column	Ausgabe bestimmter, in der Standard-Analyse nicht vorhandener Datenspalten
analysis_filter	Komplexe Filter bestehen aus einem oder mehreren Filtern, die logisch miteinander verknüpft werden

Zur Berechnung der Klick-Wahrscheinlichkeiten benötigen wir zwei verschieden Analysen. Beispiel-Anfragen für diese beiden Analysen sehen wie folgt aus:

Suchanfrage: *Krebs* $\rightarrow$ *Springermedizin-Suche*

Tab. 4.2: Beispiel Analyse-Aufbau Webtrekk-Schnittstelle

Parameter	**Wahrscheinlichkeit** $P(A_u)$	**Wahrscheinlichkeit** $P(E_{r_u})$
analysis	Navigation|Events	Navigation|Events
time_start	2016-08-01	2016-08-01
time_stop	2016-08-31	2016-08-31
column	[Page Impressions,% Visits,Ø - Suchtreffer]	[Page Impressions,% Visits,Ø - Suchtreffer]
analysis_filter	[[,Events,LIKE,searchresult-*], [AND,Internal Search Phrases,LIKE,*Krebs*], [OR,Internal Search Phrases,LIKE,*Maligne Tumoren*]]	[[,Events,LIKE,searchresult-1.*], [OR,Events,LIKE,searchresult-4.*], [OR,Events,LIKE,searchresult-29.*]]

In der Antwort für diese Analysen werden die Click-Trough-Daten als JSON-Array[2] zurückgegeben. Wie der ausgewertete JSON-Array aussehen kann, haben wir in Kapitel 2.1.2 angeschaut.

4.4 Zusammenfassung

In diesem Kapitel haben wir eine Übersicht über die Architektur und die wichtigsten Highlights der Implementierung gekriegt. Wir wissen nun *wie* der Lösungsansatz des Reranking-Algorithmus implementiert werden kann. Verknüpfen wir dieses Wissen mit dem aus Kapitel 3, könnten wir den Reranking-Algorithmus nun auch auf eine andere Suche adaptieren. Im nächsten Kapitel wollen wir nun mithilfe der implementierten Suche evaluieren, welche Verbesserung für das Qualitätsmaß der Suche von Springermedizin, der Lösungsansatz bringt.

[1] Representational State Transfer, ist ein Architekturmodell mit dem Webservices mit den Standard-HTTP-Methoden (GET, POST, PUT und DELETE) realisiert werden können

[2] Die JavaScript Object Notation (JSON) ist ein kompaktes Format zum Austausch von Daten

5 Evaluation und Auswertung

5.1 Einführung

In vorherigen Kapitel haben wir unseren Lösungsansatz in der Springermedizin-Suche implementiert. Wir wissen nun wie und warum wir den Reranking-Algorithmus umgesetzt haben. Was wir bisher nicht wissen ist, *wie gut* er funktioniert. Mithilfe einer Evaluation wollen wir darum nun messen, wie gut die Suchergebnis-Qualität der aktuellen Springermedizin-Suche im Vergleich zur im Zuge dieser Arbeit entwickelten Lösung mit dem Reranking-Algorithmus ist.

Ziel der Evaluation

Die Evaluation soll Informationen darüber liefern, wie viel Verbesserung der neue Lösungsansatz bringt. Aus den Ergebnissen wollen wir erkennen, was an dem Lösungsansatz geändert werden muss, damit die Suche wirklich gute Ergebnisse aus Sicht der User liefert und unter welchen Voraussetzungen, sie im Springermedizin-Umfeld eingesetzt werden kann.

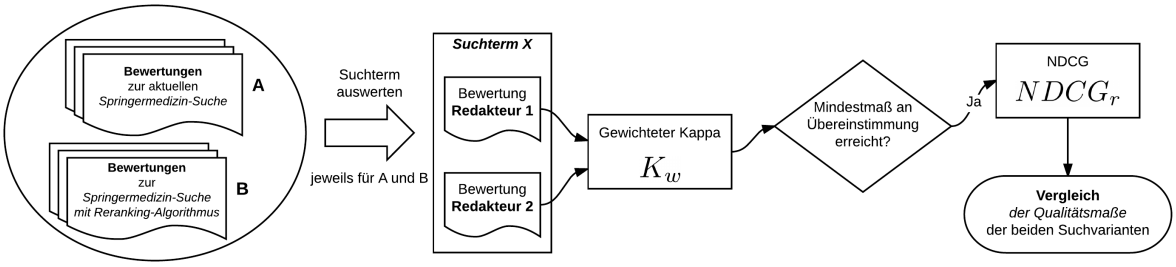
Methodik

Wir werden durch fachliche Experten (Redakteure von Springermedizin) die Suchergebnisse von oft gesuchten Suchanfragen bewerten lassen. Dazu werden wir eine Testumgebung mit einem Evaluations-System und den beiden oben angesprochenen Suchvarianten aufbauen. Die Redakteure sollen bei der Evaluation, zu jeder Suchanfrage die ersten zehn Suchergebnisse analysieren und anhand der Suchsnippets, die Relevanz bewerten. Diese Analyse sollen sie jeweils einmal auf der aktuellen Springermedizin-Suche und einmal auf der Springermedizin-Suche mit dem Reranking-Algorithmus durchführen. Am Ende werden wir die Ergebnisse auswerten und einen Vergleich der beiden Suchvarianten machen.

5.2 Aufbau der Analyse

Mithilfe der Relevanz-Bewertungen wollen wir das Qualitätsmaß der Suchvarianten bestimmen. Dazu müssen wir zuerst die „zuverlässigen“ Bewertungen mittels *Kappa-Koeffizienten* filtern und dann mithilfe der *nDCG-Metrik* auswerten. Der Prozess dazu sieht wie folgt aus:

Abb. 12: Prozess der Datenaufbereitung und Metrik der Auswertung



In diesem Kapitel werden Formeln eingeführt. Folgend eine Legende der wichtigsten Symbole:

Tab. 5.1: Legende der wichtigsten Formel-Symbolen für die Evaluation

Kappa		*nDCG*	
Bedeutung	**Symbol**	**Bedeutung**	**Symbol**
K	Cohens-Kappa-Koeffizient	r	die Anzahl der zu prüfenden Positionen
p_0	Anteil tatsächlich beobachteter Übereinstimmungen	i	die zu prüfende Position
p_e	Anteil zufälliger Übereinstimmungen	rel_i	Relevanz-Wert der Position i
w	Gewichtung anhand der Abweichung zur Hauptdiagonale	REL	die Menge der Relevanz-Werte der Dokumente in r in idealer Reihenfolge
K_w	Gewichteter Cohens-Kappa-Koeffizient	DCG_r	Discounted Cumulative Gain
R_i	Zeilensumme der Zeile i in Übereinstimmungsmatrix	$IDCG_r$	Ideal Discounted Cumulative Gain
C_i	Spaltensumme der Spalte i in Übereinstimmungsmatrix	$nDCG_r$	Normalized Discounted Cumulative Gain

5.2.1 Hypothesen

Springermedizin-Content bietet keine optimale Grundlage für Relevanz-Bewertungen Aufgrund der Analysen der Click-Through-Daten und des Springermedizin-Contents in Kapitel 2.1, wissen wir, dass wir keine optimale Grundlage für unseren Reranking-Algorithmus haben. Das liegt vor allem an den nicht beeinflussbaren Faktoren, wie die Ausspielung von Teasern mit geringer Aussagekraft und könnte sich vor allem in den Relevanz-Bewertungen der Suchergebnisse widerspiegeln. Da dieses Problem aber denselben Einfluss auf die aktuelle, sowie die neu implementierte Suche hat, sollte das Qualitätsmaß der nDCG-Auswertung davon nicht stark beeinflusst werden.

Falsche Aussagen in den Click-Through-Daten verringern Qualitätsmaß des Reranking-Algorithmus Bei der Messung des Qualitätsmaßes mittels nDCG wird der Einfluss der Click-Through-Daten interessanter sein. Grundsätzlich müsste davon ausgegangen werden, dass viele Click-Through-Daten das Ergebnis positiv beeinflussen. Es sei denn, die verwendeten Click-Through-Daten beinhalten falsche Aussagen, da sie durch viele Klicks auf wenig relevante Suchergebnisse entstanden sind und wir die User-Aktivitäten nach einem Klick nicht messen können. Wir gehen darum davon aus, dass diese Daten das Ergebnis negativ beeinflussen werden.

Stärkere Userrelevanz im Suchergebnis verbessert nDCG-Ergebnis Grundsätzlich sollte der Reranking-Algorithmus in der nDCG-Auswertung dennoch Verbesserungen im Vergleich zur aktuellen Suche zeigen, aufgrund der stärkeren Userrelevanz im Suchergebnis. Wir gehen darum von fünf bis zehn Prozent Verbesserung in der nDCG-Auswertung aus. Da bereits kleine Verbesserungen in der nDCG-Auswertung psychologisch wertvoll sind für den Endnutzer, könnten wir bei einem Wert in diesem Bereich bereits von einem Erfolg reden.

Gute Kappa-Werte durch fachgebietsspezifische Zuteilungen der Redakteure zu den Suchtermen In der Aufteilung der Analysen zu den Suchtermen haben wir darauf geachtet, dass die zugeteilten Redakteure entweder aus dem spezifischen Fachgebiet kommen oder fachgebietsnahes Wissen aufweisen. Das ähnlich gute Fachwissen der Beurteiler, sollte sich positiv auf die Kappa-Werte der Bewertungen auswirken. Wir gehen darum davon aus, dass der Kappa-Wert in den meisten Fällen (mindestens 80 Prozent) gut sein wird.

5.2.2 Datengrundlage

Filterung der nutzbaren Daten mittels Cohens Kappa

Um die Zuverlässigkeit der Relevanz-Bewertungen zu messen, werden wir die gleichen Suchterme von jeweils zwei Redakteuren von Springermedizin bewerten lassen. Haben die Relevanz-Bewertungen ein zu geringes Maß an Übereinstimmung, sind sie für die anschliessenden Auswertungen zu wenig zuverlässig und werden darum in dieser nicht verwendet. Das meist verwendete Maß zur Bewertung der Übereinstimmungsgüte ist der *Cohens-Kappa-Koeffizient* K (siehe [**Kappa**]). Dieser misst den Anteil *übereinstimmender Bewertungen* p_0 und berechnet daraus die Zuverlässigkeit der Bewertung. Hierbei

müssen wir berücksichtigen, dass die Beurteiler mit einer gewissen Wahrscheinlichkeit auch zufällig zur gleichen Einschätzung gelangen können. Der Cohens-Kappa-Koeffizient korrigiert das Maß an Übereinstimmung um diesen *Zufallsfaktor* p_e. Die Berechnungsformel zur Bestimmung des Cohens-Kappa-Koeffizienten sieht wie folgt aus:

$$K = \frac{p_0 - p_e}{1 - p_e} \tag{5.1}$$

Die Übereinstimmungen der beiden Redakteure aus einer Übereinstimmungsmatrix lesen Um die Stärke der Übereinstimmung der beiden Redakteuren bei der Relevanz-Bewertung eines Suchterms zu messen, erstellen wir zu jedem Suchterm eine Übereinstimmungsmatrix. Diese enthält die vier Relevanzstufen aus der Bewertung. Die Bewertungen der Suchergebnisse ordnen wir diesen Relevanzstufen zu:

Tab. 5.2: Übereinstimmungsmatrix von zwei Redakteuren bei der Klassifikation einer Suchanfrage

		Redakteur 2				Gesamt
		0	**1**	**2**	**3**	
Redakteur 1	**0**	a	b	c	d	$\underbrace{(a+b+c+d)/n}_{:= R_0}$
	1	e	f	g	h	$\underbrace{(e+f+g+h)/n}_{:= R_1}$
	2	i	j	k	l	$\underbrace{(i+j+k+l)/n}_{:= R_2}$
	3	m	n	o	p	$\underbrace{(m+n+o+p)/n}_{:= R_3}$
Gesamt		$\underbrace{(a+e+i+m)/n}_{:= C_0}$	$\underbrace{(b+f+j+n)/n}_{:= C_1}$	$\underbrace{(c+g+k+o)/n}_{:= C_2}$	$\underbrace{(d+h+l+p)/n}_{:= C_3}$	$\underbrace{\sum \text{ aller Matrixelemente } (a, b...o, p)}_{:= \mathbf{n}}$

Den Anteil tatsächlich beobachteter Übereinstimmungen p_0 können wir direkt aus den Werten der *Hauptdiagonalen* der Matrix berechnen. Die Berechnungsformel dazu sieht wie folgt aus:

$$p_0 = \frac{\sum \text{ der Übereinstimmungen } (a + f + k + p)}{\sum \text{ aller Übereinstimmungen } n} \tag{5.2}$$

Den daraus resultierenden Wert, müssen wir um den Anteil zufälliger Übereinstimmungen pe korrigieren. Der Wert von pe wird mithilfe der Randsummen der Matrix (Spalten- bzw. Zeilensummen) berechnet. Dazu muss jede Randsumme zuerst durch n dividiert werden. Danach wird zu jeder Kategorie das *Produkt der Spalten und Zeilensumme* gebildet, mit welchem anschließend die Summe der Kategorien berechnet wird. Die komplette Berechnungsformel von pe sieht wie folgt aus:

$$p_e = \left(\left(\frac{R_0}{n} \cdot \frac{C_0}{n}\right) + \left(\frac{R_1}{n} \cdot \frac{C_1}{n}\right) + \left(\frac{R_2}{n} \cdot \frac{C_2}{n}\right) + \left(\frac{R_3}{n} \cdot \frac{C_3}{n}\right)\right) \tag{5.3}$$

Kappa-Koeffizienten gewichten Relevanz-Bewertungen werden meist sehr subjektiv gefällt. Wir müssen darum davon ausgehen, dass die Redakteure häufig kleinere Abweichungen in den Bewertungen haben werden. Weichen sie mehrere Kategorien voneinander ab, sollten diese Abweichung schwerer wiegen als die, bei benachbarten Kategorien. Cohens-Kappa-Koeffizienten misst für p_0 *nur die genauen Übereinstimmungen* und stuft alle Abweichungen gleich ein. Mithilfe der Erweiterung des Kappa-Koeffizienten um eine Gewichtung der Abweichungsstärke zwischen 0 und 1, können wir die Berechnungsformel zur Bestimmung des Kappa-Koeffizienten in den gewichteten Kappa K_w (siehe [**KappaWerte**]) transformieren. Dazu müssen wir bei der Berechnung der Spaltensummen C_{i_w} und Zeilensummen R_{i_w} mit $i \in \{0, 1, 2, 3\}$ *die Anzahl der Kategorien*, die das Matrixelement zur Hauptdiagonalen abweicht, berücksichtigen und diese Abweichung gewichten. Die transformierte Berechnungsformel K_w sieht dann wie folgt aus:

$$K_w = \frac{p_{0_w} - p_{e_w}}{1 - p_{e_w}} \tag{5.4}$$

Zur Gewichtung der Abweichung benutzen wir einen Gewichtungsfaktor $w \in \mathbb{R}$. Die im folgenden auftretenden Notationen w beschreiben die Berücksichtigung des gewichteten Faktors w. Die Berechnung der Gewichte und die Beschreibung der Gewichtung der Spalten- und Zeilensummen, folgt unten. Um

den Anteil tatsächlich beobachteter Übereinstimmungen p_0 in p_{0_w} zu transformieren, werden wir *anstatt nur* den Werte der Hauptdiagonalen, *die Summe der gewichteten Zeilenelemente* R_{i_w} verwenden, wobei der Wert der Hauptdiagonale die *höchste Gewichtung* haben soll:

$$p_{0_w} = \frac{\sum \text{ der Übereinstimmungen } (R_{0_w} + R_{1_w} + R_{2_w} + R_{3_w})}{\sum \text{ aller Übereinstimmungen } n} \tag{5.5}$$

Um den Anteil zufälliger Übereinstimmungen p_e in p_{e_w} zu transformieren, überführen wir die Spaltensummen C_i und Zeilensummen R_i in die gewichteten Spaltensummen C_{i_w} und Zeilensummen R_{i_w}, wobei die Werte der Hauptdiagonalen die *kleinste Gewichtung* haben sollen:

$$p_{e_w} = \left(\left(\frac{R_{0_w}}{n} \cdot \frac{C_{0_w}}{n}\right) + \left(\frac{R_{1_w}}{n} \cdot \frac{C_{1_w}}{n}\right) + \left(\frac{R_{2_w}}{n} \cdot \frac{C_{2_w}}{n}\right) + \left(\frac{R_{3_w}}{n} \cdot \frac{C_{3_w}}{n}\right)\right) \tag{5.6}$$

Wir beachten hierbei, dass wir zwar die Spalten- und Zeilensummen gewichten, in der Berechnung von n, diese Gewichtung jedoch weiterhin nicht beachten. Das gilt für die Berechnung von n in p_{0_w} und p_{e_w}. Für die Berechnung von p_{0_w} und p_{e_w} definieren wir die Gewichte wie folgt:

Tab. 5.3: Gewichtung der Abweichungsstärke einer Kategorie zur Hauptdiagonalen

Gewichtung w für p_{0_w}	**Gewichtung w für p_{e_w}**	**Anzahl Kategorien Abweichung**
1	*0.25*	0 *(auf Hauptdiagonal)*
0.75	*0.5*	1 *(benachbarte Kategorie)*
0.5	*0.75*	2
0.25	*1*	3

Um besser zu verstehen, wie diese Gewichtung eingebaut wird, sehen wir folgend den Zeilenausschnitt R_0 überführt in R_{0_w} der angepassten Übereinstimmungsmatrix für den gewichteten Kappa K_w in der Berechnung von p_{0_w}:

Tab. 5.4: Beispiel Gewichtung in Übereinstimmungsmatrix für p_{0_w}

	0	**1**	**2**	**3**	
0	$a_w \leftarrow 1 \cdot a$	$b_w \leftarrow 0.75 \cdot b$	$c_w \leftarrow 0.5 \cdot c$	$d_w \leftarrow 0.25 \cdot d$	$\underbrace{(a_w + b_w + c_w + d_w)/n}_{:=R_{0_w}}$

Kappa interpretieren Wie in [**KappaWerte**] beschrieben, müssen die Kappa-Werte individuell interpretiert werden. Es gibt jedoch Richtlinien. In [**Kappa**] wird bei einem Kappa-Wert ab 0.60 von einer guten Übereinstimmung ausgegangen. Wir werden versuchen dieses Mindestmaß ebenfalls auf 0.60 anzusetzen. Die unter dem Mindestmaß liegenden Bewertung, werden wir in der nDCG-Auswertung (siehe unten folgende Metrik) ignorieren.

5.2.3 Metrik

Qualitätsmaß einer Suchvariante mittels nDCG bestimmen In der Evaluation werden zu jedem Suchterm zwei Bewertungen für die aktuelle Springermedizin-Suche und zwei Bewertungen für die Springermedizin-Suche mit dem Reranking-Algorithmus abgegeben. Um das Qualitätsmaß einer Suche zu bestimmen, werden wir den *Normalized Discounted Cumulative Gain* (nDCG, siehe [**nDCG**]) verwenden. Der nDCG misst die Qualität des Rankings der Suche und wird in der Information Retrieval[1] oft eingesetzt um die Effektivität eines Such-Algorithmus zu messen. Um das Qualitätsmaß der beiden in der Evaluation verwendeten Suchvarianten zum Suchterm zu bestimmen, berechnen wir zu jeder Suchvariante den nDCG der beiden Bewertungen des Suchterms. Nehmen wir den Mittelwert der beiden resultierenden nDCG-Werte, erhalten wir den effektiven nDCG-Wert. Die nDCG-Werte der beiden Suchvarianten können wir dann miteinander vergleichen.

[1] Mit Information Retrieval werden Methoden und Verfahren, die der Aufbereitung und Speicherung von Wissen und der Gewinnung von Informationen dienen bezeichnet

Evaluationsdaten mittels nDCG auswerten

Der nDCG verfolgt die Grundidee, Suchergebnislisten dahingehend zu untersuchen, ob Dokumente mit hoher Relevanz zum Suchterm, vor denen mit weniger Relevanz stehen. Der nDCG vergleicht dazu die Relevanz der Dokumente mit ihrer Reihenfolge im Suchresultat. Diese Metrik macht für unseren Reranking-Algorithmus insofern Sinn, dass sie nicht von bestimmten Relevanz-Werten für die Suchresultate ausgeht, sondern wie unser Algorithmus, sich *nur* auf die Reihenfolge der Relevanz-Werte konzentriert.

Berechnung des Qualitätsmaß einer Suchvariante mittels nDCG

Der nDCG baut auf dem DCG auf Um den nDCG-Wert zu bestimmen, müssen wir zuerst den *Discount Cumulative Gain* (DCG) jeder Position des untersuchten Suchresultats berechnen. Der DCG zielt darauf ab, das Qualitätsmaß einer Suchergebnisliste herunterzustufen, wenn relevante Dokumente schlechter, als weniger relevante positioniert sind. Die Formel des DCG wird wie folgt definiert:

$$DCG_r = \sum_{i=1}^{r} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{5.7}$$

Wie wir sehen, werden Dokumente umso geringer bewertet, je weiter hinten sie im Suchresultat erscheinen. Dafür sorgt $\log_2(i+1)$ als Divisor in der Summenfunktion, dessen Wert mit steigendem Wert der Dokumentposition größer wird.

Der DCG muss normalisiert werden, um das Qualitätsmaß einer Suche über unterschiedliche Anfragen zu bewerten Der DCG ist auf den Vergleich von Suchanfragen mit gleicher Resultatslänge ausgelegt. Haben die zu untersuchenden Suchanfragen, eine unterschiedliche Anzahl der zu untersuchenden Positionen, variiert die maximal erreichbare Punktzahl. Der nDCG normalisiert diese Punktewerte, indem er die Reihenfolge der Relevanz-Werte des Suchergebnisses mit der *idealen Reihenfolge* derselben Relevanz-Werte vergleicht. Die Formel des nDCG wird wie folgt definiert:

$$nDCG_r = \frac{DCG_r}{IDCG_r} \tag{5.8}$$

wobei:

$$IDCG_r = \sum_{i=1}^{|REL|} \frac{2^{rel_i} - 1}{\log_2(i+1)} \tag{5.9}$$

Der resultierende Ergebnis-Wert des $nDCG_r$ bewegt sich zwischen 0 und 1. Entspricht die Reihenfolge der Suchergebnisse, der Relevanz der Suchergebnisse, so gilt $DCG_r = IDCG_r$. Dies entspricht dem Idealfall, die Suche besitzt in diesem Fall den $nDCG_r$-Wert 1 und somit das maximal mögliche Qualitätsmaß für den getesteten Suchterm. Um die nDCG-Metrik anhand eines Beispiels etwas besser zu verstehen, sehen wir hier folgend die nDCG-Auswertung zweier Ranking-Funktionen für vier Dokumente $d1, d2, d3, d4$ mit einer Relevanz-Skala von 0-3. Der $MaxDCG$ beschreibt hier den IDCG:

Abb. 13: Beispiel: nDCG-Auswertung für vier Dokumente

i	Ground Truth		Ranking Function$_1$		Ranking Function$_2$	
	Document Order	r_i	Document Order	r_i	Document Order	r_i
1	d4	2	d3	2	d3	2
2	d3	2	d4	2	d2	1
3	d2	1	d2	1	d4	2
4	d1	0	d1	0	d1	0
	$NDCG_{GT}$=1.00		$NDCG_{RF1}$=1.00		$NDCG_{RF2}$=0.9203	

$$DCG_{GT} = 2 + \left(\frac{2}{\log_2 2} + \frac{1}{\log_2 3} + \frac{0}{\log_2 4}\right) = 4.6309$$

$$DCG_{RF1} = 2 + \left(\frac{2}{\log_2 2} + \frac{1}{\log_2 3} + \frac{0}{\log_2 4}\right) = 4.6309$$

$$DCG_{RF2} = 2 + \left(\frac{1}{\log_2 2} + \frac{2}{\log_2 3} + \frac{0}{\log_2 4}\right) = 4.2619$$

$$MaxDCG = DCG_{GT} = 4.6309$$

5.2.4 Vorgehen

Testumgebung aufbauen

Um eine Evaluation durchführen zu können, müssen wir eine passende Testumgebung aufbauen. Diese besteht aus einem Evaluations-System, einer Instanz der aktuellen Springermedizin-Applikation und einer Instanz des neu implementierten Lösungsansatzes. Auf dem Evaluations-System sollen fachliche Experten (Redakteure von Springermedizin) die Relevanz der Suchergebnisse der beiden Suchmaschinen vergleichen. Dazu sollen die jeweils ersten zehn Suchergebnisse nach Relevanz zum Suchterm bewertet werden. Der Ergebnisse werden in einer Datenbank gespeichert, um sie später auszuwerten.

Aufbau des Evaluations-Systems Für die Analysen der Beurteiler implementieren wir selber ein kleines Evaluations-System als Web-Applikation. Das hat den Vorteil, dass wir selber definieren können, wie der Analyse-Prozess und wie die Datenstruktur der Analyse-Ergebnisse aussehen sollen. Das System umfasst eine Administrationsoberfläche zur Verwaltung der Analysedaten und eine Anwenderoberfläche zur Analyse der zugeteilten Suchterme. Der Beurteiler soll nur seine, ihm zugeteilten Suchterm-Analysen sehen. Um Analysen durchführen zu können, muss sich der Beurteiler darum an der Applikation anmelden. So können wir jedem Beurteiler ein eigenes Profil anlegen und ihm seine zu analysierenden Suchterme zuweisen.

Verwendeter Technologiestack Unser Evaluations-System werden wir wie die Springermedizin-Applikation (White Label Applikation) auf dem MVC-Framework *Play* [**Play**] aufbauen und *Scala* [**Scala**] als Programmiersprache verwenden. Um die Evaluationsdaten zu speichern verwenden wir eine PostgreSQL-Datenbank[2] [**Postgresql**]. Unsere beiden zu vergleichenden Springermedizin-Suchen bauen auf der Live-Applikation von Springermedizin auf und verwenden den Live-Content von Springermedizin zur Suche. Wir werden die beiden Suchen und das Evaluations-System darum in der Live-Umgebung der Springer Nature Cloud-Plattform Cloud Foundry[3] laufen lassen. So haben wir ein realistisches Umfeld für die Evaluation.

Analyse eines Suchterms Wir wollen zu jedem Suchterm eine Bewertung der aktuellen Springermedizin-Suche und eine, der Suche mit dem implementierten Reranking-Algorithmus. Wir bauen die Analysen darum so, dass der Benutzer beide Bewertungen nacheinander, in derselben Analyse ausführt. Zur Bewertung implementieren wir ein Maske, bestehend aus der zu beurteilende Suchvariante und einem Formular zur Bewertung der ersten zehn Suchresultate des Suchergebnisses. Die Bewertung besteht aus einer Skala von vier Relevanz-Werten, wobei *nicht relevante* Ergebnisse bestraft werden sollen und darum einen negativen Relevanz-Wert erhalten:

Tab. 5.5: Relevanz-Werte für Bewertung der Suchresultate

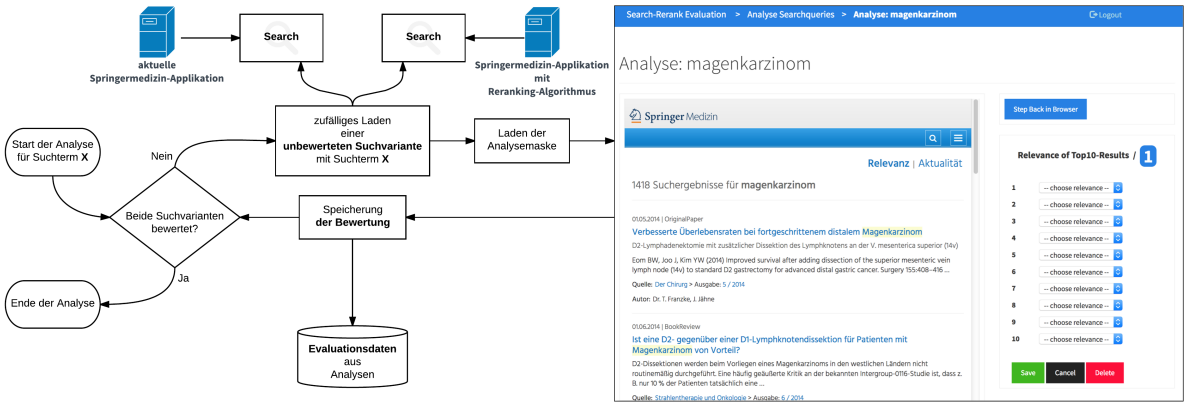
Relevanz	Beschreibung	Relevanz-Wert
not relevant	Ergebnis hat ***gar keine*** Relevanz	0
moderately relevant	Ergebnis ist ***eher irrelevant***	1
relevant	Ergebnis ist ***eher relevant***	2
highly relevant	Ergebnis ist ***sehr relevant***	3

Ist eine Bewertung gespeichert, wird die zweite Variante der Suche geladen. Eine Analyse ist dann abgeschlossen, wenn beide Suchvarianten bewertet sind.

[2] PostgreSQL ist eine Objektrelationale Datenbank (ORDB)
[3] Cloud Foundry bietet Platform-as-a-Service (PaaS) und ist eine Cloud die eine Computer-Plattform für Webanwendungen zur Verfügung stellt

Der Ablauf einer Analyse sieht wie folgt aus:

Abb. 14: Analyseprozess für Bewertung einer Suchvariante



Evaluations-Daten speichern

Zu jeder Analyse werden wir neben den Relevanz-Bewertungen und den Suchterm-Informationen auch wichtige Informationen zu den Click-Through-Daten und den Suchergebnissen speichern. Folgende Informationen liegen uns am Ende einer Analyse vor:

Tab. 5.6: Gespeicherte Evaluations-Daten zur Suchterm-Analyse

Objekt	Beide Suchvarianten	Suche mit Reranking-Algorithmus	Beschreibung
Analyse	*Suchterm*		Analysierter Suchterm
	Beurteiler		User-Informationen des Beurteilers
	Suchvariante		Suche mit / ohne Reranking-Algorithmus
		Wert des Zufallsfaktors	Einfluss-Wert Zufallsfaktor
Bewertungen	*Relevanz-Werte der Positionen*		Relevanz-Bewertungen der ersten zehn Suchresultate
Click-Through-Daten		*User-Filter*	Alle Benutzer / eingeloggte Benutzer der Suche
		Anzahl Klicks Gesamt	Anzahl gelesener Klicks aus Click-Through-Daten
Suchergebnis	*Dokument-ID's der Suchresultate*		Dokument-ID's der ersten zehn Suchergebnisse

Evaluations-System auswerten

Nach Abschluss der Evaluationsphase werden wir die Evaluations-Daten auswerten. Die Auswertung der Daten findet direkt im Evaluations-System statt. Dazu werden wir die Relevanz-Bewertungen aus der Datenbank lesen und wie oben beschrieben mit dem Cohens-Kappa-Koeffizienten und der nDCG-Metrik auswerten. Mithilfe der Evaluations-Daten können wir dann auch weitere Auswertungen zu den Click-Through-Daten und den Suchergebnissen der Suchterme machen.

5.2.5 Durchführung

Zu analysierende Suchterme

Um die Evaluation praxisrelevant und mit genügend Click-Through-Daten ausführen zu können, wurden die Suchanfragen der letzten zwei Monate auf der Springermedizin-Suche analysiert und 80 oft gesuchte Suchterme ausgewählt (siehe Anhang 7.1). Springermedizin hat ein breites Spektrum an medizinischen Fachgebieten. Um fachlich gute Bewertungen zu kriegen, haben wir die Suchterme nach Fachrichtung unterteilt und jeweils zwei Beurteilern zugeteilt, die aus der Fachrichtung kommen:

Tab. 5.7: Fachrichtungen der Suchterme der Evaluation

Fachrichtung	Medizinisches Gebiet / Medizinischer Zweig	Anzahl zugeteilte Suchterme
Onkologie	Tumorerkrankungen	10
Zahnmedizin	Fachgebiet der Zahn-, Mund- und Kieferheilkunde	20
Gynäkologie	Spezifischen Erkrankungen des weiblichen Körpers	10
AINS	Anästhesiologie, Intensivmedizin, Notfallmedizin und Schmerztherapie	10
Neurologie, Psychiatrie	Erkrankungen des Nervensystems und der Psyche	10
Innere Medizin	Erkrankungen der inneren Organe	10
Orthopädie *Urologie* *HNO*	Aufbau der Knochen und Muskeln des Menschen Erkrankungen der Niere, Harnblase, Harnleiter und Harnröhre Hals-Nasen-Ohren-Heilkunde	10
Gesamt:		**80**

Aufgabenstellung der Analyse

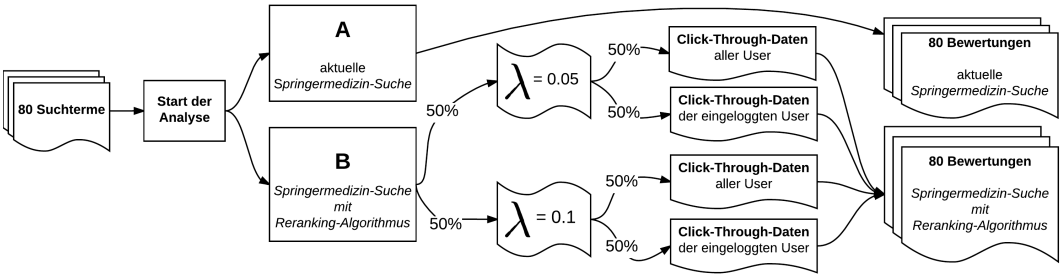
Die Aufgabe der Analyse besteht darin, die jeweils ersten zehn Suchergebnisse nach Relevanz zum Suchterm zu bewerten. Insgesamt, soll jeder Beurteiler die ihm zugeteilten 20 Suchterme analysieren. Eine Analyse beinhaltet jeweils eine Bewertung des Suchterms, auf beiden Suchvarianten. In welcher Reihenfolge die Suchvarianten den Beurteilern während der Analyse präsentiert werden, ist rein *zufällig* und *variiert*. Es soll nicht ersichtlich sein, welche Variante jeweils bewertet wird. Dadurch können wir ausschließen, dass der Beurteiler durch die Bekanntgabe der Suchvariante subjektiv beeinflusst wird.

Verschiedene Varianten des neuen Lösungsansatzes evaluieren

Wie wir wissen, hat der Reranking-Algorithmus einige Faktoren, die variabel definiert werden können. Zu diesen Faktoren gehört der Einfluss des Zufallswertes in das Reranking (λ) und der „Login-Status der User“ während der Suche, für die Selektion der Click-Through-Daten. Wir werden darum verschiedene Konstellationen testen, um die optimale Konstellation finden zu können.

Definition der variablen Faktoren für den Reranking-Algorithmus Für den *Einfluss des Zufallsfaktors* werden wir zwei Werte für λ definieren. Diese sind 0.1 (zehn Prozent) und 0.05 (fünf Prozent). Für die *Selektion der Click-Through-Daten* werden wir zwischen an der Applikation angemeldeten Benutzern und allen Benutzern (*inkl. anonymen Benutzern*) unterscheiden. Aus den beiden Einflusswerten des Zufallsfaktors und der Unterscheidung zwischen angemeldeten und allen Benutzern, ergeben sich vier Konstellationen. Jeder Konstellation werden wir jeweils 25 Prozent der Suchterme zuteilen. Die Zuteilung der Suchterme werden wir mithilfe des Evaluations-Systems zufällig generieren lassen:

Abb. 15: Aufteilung der Analysen für Evaluation



5.3 Auswertung der Suchergebnis-Qualität

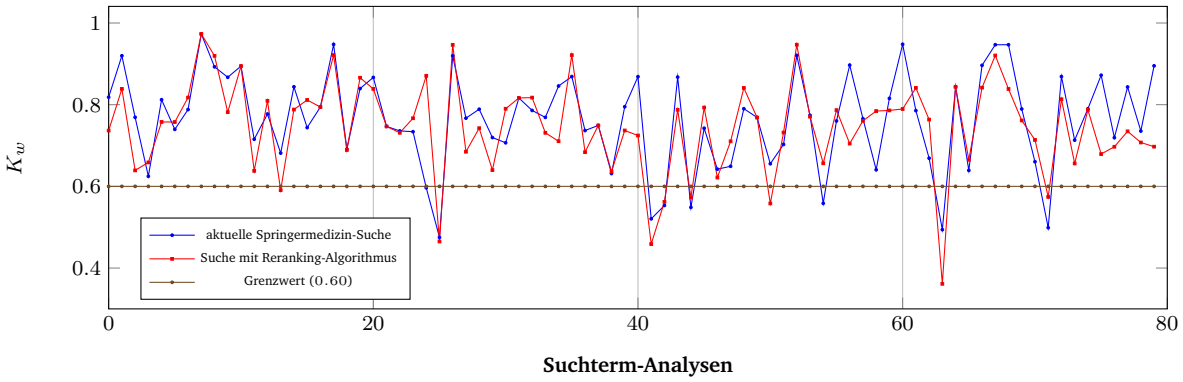
Bei der Auswertung der Evaluationsdaten werden wir uns zuerst auf die Kappa-Werte fokussieren und analysieren, wie gut das Übereinstimmungsmaß der Beurteiler zu den Suchtermen K_w ist. Danach folgt der Kern der Auswertung, die Bestimmung des Qualitätsmaßes der Suchvarianten mithilfe der nDCG-Metrik. Wir wollen herausfinden, welche Verbesserungen, mit welcher Konstellation des Reranking-Algorthmus gemacht werden können. Da der nDCG sich nur auf die Reihenfolge der Relevanz-Bewertungen konzentriert, wollen wir in einer weiteren Auswertung analysieren, welche durchschnittlichen Relevanz-Bewertungen die ersten zehn Suchresultate effektiv hatten. Zum Schluss werden wir die Einflüsse der Click-Through-Daten auf den nDCG analysieren und auswerten.

5.3.1 Quantitative Auswertung

Auswertung des Übereinstimmungsmaßes der Beurteiler mittels Kappa

In dieser Analyse werden wir sehen, wie gut das Übereinstimmungsmaß K_w der Beurteiler ist und wie viele der Suchterm-Bewertungen für die nDCG-Auswertung überhaupt in Frage kommen.

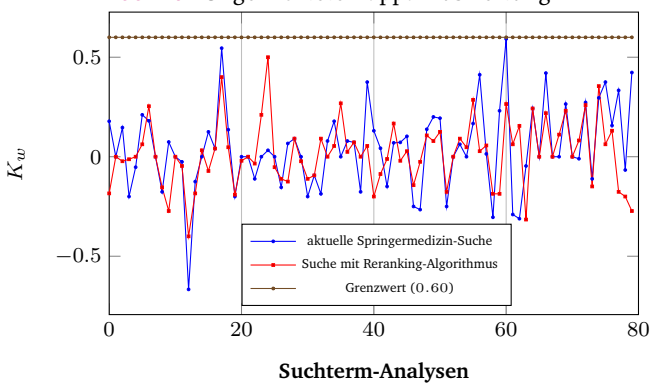
Abb. 16: Kappa-Auswertung der Suchterm-Bewertungen



Zum Vergleich der ungewichtete Kappa Wie bereits in der Datengrundlage im Kapitel 5.2.2 diskutiert, bewerten wir nicht den Cohens Kappa Koeffizienten K sondern verwenden dessen gewichtete Variante K_w. Wir sind davon ausgegangen, dass die Wahrscheinlichkeit der genauen Übereinstimmung der Relevanz-Bewertung relativ klein ist, hingegen die der Bewertungen mit leichter Abweichung des Relevanz-Wertes, relativ groß sein könnte. Mithilfe dieser Analyse wollen wir herausfinden, ob wir mit unserer Vermutung richtig liegen.

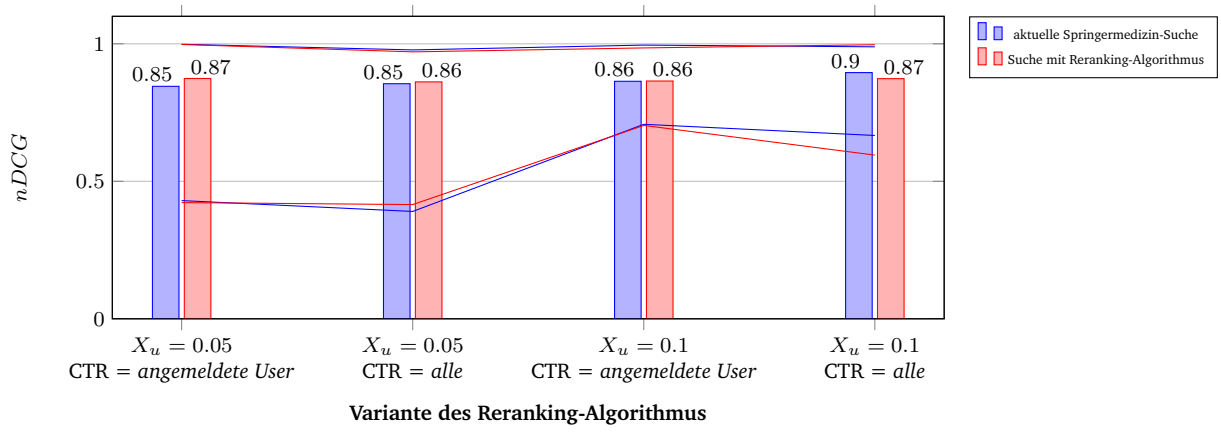
sadsdfsdf

Abb. 17: Ungewichtete Kappa-Auswertung



Bestimmung des Qualitätsmaßes der Suchvarianten durch nDCG-Auswertungen

Abb. 18: nDCG-Auswertung aller Suchvarianten



Auswertung des Einflusses der Click-Through-Daten auf den nDCG

5.3.2 Diskussion

Auswertung des Übereinstimmungsmaßes der Beurteiler mittels Kappa

Bestimmung des Qualitätsmaßes der Suchvarianten durch nDCG-Auswertungen

Auswertung des Einflusses der Click-Through-Daten auf den nDCG

5.4 Zusammenfassung

6 Zusammenfassung und Ausblick

Anhang

7

7.1 Verwendete Suchterme für die Evaluation

Tab. 7.1: Tabelle der verwendeten Suchterme für die Evaluation

Fachgebiet	Suchphrasen	Fachgebiet	Suchphrasen
Onkologie	pankreaskarzinom	***AINS***	Borreliose
	schilddrüsenkarzinom		myokarditis
	ernährung onkologie		herzinsuffizienz
	plasmozytom		adipositas
	thyreoidektomie		gastrointestinale blutung
	magenkarzinom		helicobacter pylori
	rektumkarzinom radiochemotherapie		diabetes mellitus
	operative therapie		stroke
	Prostata-Adenom		darmatonie
	lebermetastasen		herpes zoster
Zahnmedizin	mandibular advancement device	***Neurologie, Psychiatrie***	schlaganfall
	fibrinogen		demenz
	lidocain		schlafstörungen
	therapietreue		delir
	Berufsdermatologie		virusenzephalitis
	trigeminusneuralgie		invagination
	anaphylaxie		adhs
	konservative therapie		demenz therapie
	mukositis		psychiatrische erkrankungen
	kraniofaziale chirurgie		syringomyelie
	versiegelung zähne	***Innere Medizin***	hyponatriämie
	präoperative aufklärung		osteoporose
	antibiotika		copd
	hiv-infektion		sarkoidose
	implantatallergie		vorhofflimmern
	patientenakte		tuberkulose
	blutung postoperativ		migration
	disposition		lungenembolie
	überempfindlichkeit		betablocker
	schweigepflicht		das fiebernde kleinkind
Gynäkologie	reproduktion	***Orthopädie, Urologie, HNO***	talusfrakturen
	konisation		mediale schenkelhalsfraktur
	impfen in der schwangerschaft		winkelstabile osteosynthese
	schwangerschaft melisse		chronische rhinosinusitis
	risikoschwangerschaft		urethralsyndrom
	sectio cesarea		nacken schmerzen
	adenoviren neonatal		implantierbare hörgeräte
	mammakarzinom		schmerzen an der wirbelsäule
	sexuelle gewalt spurensicherung dokumentation		ac gelenk
	frühschwangerschaft		basaliom

Abbildungs-Verzeichnis

Tabellen-Verzeichnis