

**Universität Leipzig  
Fakultät für Mathematik und Informatik  
Institut für Informatik**

## **Kurs Information Retrieval**

Praktikumsbericht

**Zusammenfassung:**  
Dieser Bericht ...

Leipzig, März 2018

vorgelegt von  
Maik Fröbe, Danilo Morgado, Sebastian Günther

Betreuer:

Jun.-Prof. Dr. Martin Potthast  
Fakultät für Mathematik und Informatik  
Text Mining und Retrieval

---

# 1 Einführung

Blaaa

---

## 2 Von der Datenbeschaffung zum fertigen Index

Die in einer Dokumentensammlung enthaltenen Informationen bestimmen die Sinnhaftigkeit einer Suchmaschine basierend auf diesen Dokumenten in besonderem Maße [CMS09b]. Dementsprechend stellt eine geeignete, mit vertretbarem Aufwand durchführbare, Erzeugung dieser Dokumentensammlung den Anfang der Wertschöpfungskette dar.

Die im Kontext von Web-Suchmaschinen dafür benötigten Programme werden Crawler genannt [CMS09b]. Diese beschäftigen sich damit, Webseiten zu finden und herunterzuladen, um sie für spätere Verarbeitungsschritte lokal Verfügbar zu haben. Die in der Problembeschreibung ?? dargestellte Beschränkung auf die Website der Universität Leipzig vereinfacht dieses, im Allgemeinen sehr anspruchsvolle, Problem und wird als Site Search bezeichnet [CMS09a].

Im Rahmen dieser Arbeit wurde zur Erzeugung der Dokumentensammlung die Freie Software [wik18a] Apache Nutch<sup>1</sup> verwendet. Nutch übernahm dabei sowohl die Aufgaben eines Crawlers<sup>2</sup> als auch die unmittelbar folgenden Schritte bis einschließlich der Indexerzeugung. Im folgenden werden verschiedene, während der Arbeit angewandten Anpassungen und Vereinfachungen des von Nutch bereitgestellten Workflows vorgestellt.

### 2.1 Domänenspezifische, manuell gepflegte Selectivity

Unter den Features, die ein Crawler umsetzen sollte [MRS09], findet sich auch die Notwendigkeit, die Qualität der zu crawlenden Seiten in die Crawling-Reihenfolge einfließen zu lassen. Am Beispiel von Nutch ist diese Funktionalität durch eine Bewertung der einzelnen Seiten auf Basis der aktuell bekannten Linkstruktur realisiert [nut13]. Dabei werden Seiten, welchen eine höhere Wichtigkeit vorhergesagt wird, zeitiger heruntergeladen.

Aufgrund begrenzter Ressourcen wurde diese Idee weiter verschärft, so dass bestimmte Seiten erst gar nicht gecrawlt wurden. Dies betrifft Seiten, welche nicht direkt zur Universität gehören, sowie Spider-Traps<sup>3</sup>, umfangreiche Webanwendungen<sup>4</sup>, oder auch vollständige Suchmaschinen<sup>5</sup> innerhalb der Universität.

---

<sup>1</sup> Nutch ist [unter Apache Version 2.0 lizenziert](#)

<sup>2</sup> Der finale Crawling Durchgang benötigte 14 Tage um 390 000 Dokumente in 50GB zu sammeln.

<sup>3</sup> TBA, Kalender mit []

<sup>4</sup> Wortschatz, Wörterbücher, Semantic-Web-Anwendungen

<sup>5</sup> Dokumentenserver für Abschlussarbeiten

---

Insbesondere für die letzteren besteht die Herausforderung darin, genug Informationen über Existenz und Einsatzzweck einer solchen Seite zu sammeln, ohne gleichzeitig unnötig vielen dynamisch erzeugten Links zu folgen. Beispiel 2.1 beschreibt dies am Fall eines Dokumentenservers.

### Beispiel 2.1: Dokumentenserver FMI

Der [Dokumentenserver der Fakultät für Mathematik und Informatik](#) stellt unter anderem folgende Funktionalitäten bereit:

#### Den Download von Dokumenten:

Ein Nutzer kann die im Dokumentenserver enthaltenen Dokumente herunterladen. Dazu besitzt jedes Dokument eine Übersichtsseite, welche Metadaten zum Dokument bereitstellt sowie einen Link zum vollständigen Dokument. Beispiele dafür sind:

- Eine Übersichtsseite: <http://lips.informatik.uni-leipzig.de/pub/2017-0>
- Ein Dokument: <http://lips.informatik.uni-leipzig.de/files/thesis.pdf>

Beide enthalten wichtige Informationen und sollten gecrawlt werden.

#### Das Filtern nach Dokumenten:

Ein Nutzer kann über eine Facettensuche [[wik18b](#)] innerhalb der Dokumente filtern. Beispiele dafür sind:

- Filterung nach „Organisation Ifl“: <http://lips.informatik.uni-leipzig.de/browse/results/taxonomy%3A912>
- Filterung nach „Organisation Ifl und Author Erhard Rahm“: [http://lips.informatik.uni-leipzig.de/browse/results/taxonomy%3A912%20field\\_authors%3A%22Rahm%2C%20Erhard%22](http://lips.informatik.uni-leipzig.de/browse/results/taxonomy%3A912%20field_authors%3A%22Rahm%2C%20Erhard%22)

Die enorme Menge an möglichen Filter-Permutationen, sowie [teilweise auftretenden Endlosschleifen](#) lassen den Aufwand für ein Crawling dieser Seiten als unangemessen erscheinen. Dies wird abgerundet durch den Umstand, dass die oben beschriebenen Zielseiten ohne Filterung erreichbar sind. Dazu ist eine Traversierung von folgenden Seiten sinnvoll:

- Die erste Seite einer Liste aller vorhandenen Dokumente: <http://lips.informatik.uni-leipzig.de/browse/results/>
- Für welche nachfolgende Seiten direkt über Links zugänglich sind: <http://lips.informatik.uni-leipzig.de/browse/results?page=4>

Um derartige Problemfälle zu behandeln müssen diese im ersten Schritt identifiziert werden. Diesbezüglich bietet es sich an, während kleinerer Probe-

---

Crawlings eine Auswertung der Logs vorzunehmen. Um dies in vereinfachter Form durchzuführen, wurde ein Statistik-Plugin für Nutch implementiert<sup>6</sup>. Dieses Plugin identifiziert mittels Map-Reduce<sup>7</sup> Bereiche für die der Crawler besonders aktiv wird. Dazu werden alle bekannten URLs durch Entfernung von Queries oder Fragmenten normalisiert, und in ihre Bestandteile wie Schema, Host und Pfad zerlegt. Beispiel 2.2 zeigt für die in Beispiel 2.1 besprochenen Links, wie durch aufsummieren dieser Bestandteile die Ausgabe des Plugins erzeugt wird.

### Beispiel 2.2: Berechnungsschritte des Statistik-Plugins

Die Map-Phase nimmt die Normalisierung der Links, sowie eine Umwandlung in die hierarchischen Bestandteile vor: **TODO MAIK BEISPIEL**

Die Reduce Phase summiert alle eintreffenden Paare bestehend aus Link-Bestandteil und count: **TODO MAIK BEISPIEL**

In der Ausgabe des Statistik-Plugins können schließlich sinnvoll Problemfälle identifiziert werden<sup>8</sup>. Um dies effizient durchführen zu können, bietet es sich an, dieses Plugin als Eingabe für existierende Standardwerkzeuge zu nutzen. So erlaubt die nach Anzahl sortierte sowie durch einen Mindest-Threshold und White-List gefilterte Ausgabe auf den ersten Blick potentielle Probleme zu identifizieren.

Im nächsten Schritt ist es notwendig, diese Problemfälle zu behandeln. Eine einfache, aber effiziente, Möglichkeit, dies umzusetzen, wird in Nutch in Form von [Regex-Url-Filtern](#) bereitgestellt. Darunter versteht man eine Liste von regulären Ausdrücken, welche jeweils um eine positive oder negative Markierung erweitert sind. Für eine URL werden die regulären Ausdrücke dann der Reihe nach ausgewertet. Der erste passende Ausdruck entscheidet, ob eine URL gecrawlt werden darf (falls der zugehörige Ausdruck positiv markiert ist), oder nicht<sup>9</sup>.

Die in Nutch vordefinierten Regex-Url-Filter verfolgen das Ziel, das Crawling auf sinnvolle Protokolle<sup>10</sup> und Dokument-Typen<sup>11</sup> zu beschränken. Um diese Filter systematisch korrekt zu erweitern wurde ein Testgetriebenes Vorgehen

<sup>6</sup> Das Statistik-Plugin kann in dem [entsprechenden Github-Repository](#) eingesehen werden.

<sup>7</sup> Einführungen in das Map-Reduce-Paradigma können hier eingesehen werden [\[1\]](#), [\[2\]](#), [\[3\]](#)

<sup>8</sup> Neben den in Beispiel 2.1 hervorgehobenen Problemen sei hier noch auf potentiell fehlende Seiten hingewiesen. Diese können durch einen Vergleich der aufsummierten Linkbestandteile mit der Anzahl der von Google indexierten Seiten entdeckt werden

<sup>9</sup> Aus diesem Grund ist es üblich, die Liste der Ausdrücke mit einem entsprechendem immer zutreffendem Eintrag abzuschließen

<sup>10</sup> HTTP, kein...

<sup>11</sup> Html-Seiten, keine CSS-Ressourcen, Bilder oder Videos

---

gewählt<sup>12</sup>. Dieses zeichnet sich durch die Möglichkeit aus, einzelne Bereiche getrennt voneinander zu konfigurieren, ohne ungewünschte Seiteneffekte zu erwarten.

Nach dem [Teile-und-herrsche-Verfahren](#) werden diesbezüglich für kleine Bereiche der zu crawlenden Webseiten einzeln anhand von Beispielen Konfigurationen vorgenommen. Dafür werden für diesen Bereich Positiv- und Negativ-Beispiele in Form von Urls zusammengetragen. Im nächsten Schritt werden die für diese Beispiele passenden Regex-Url-Regeln aufgestellt. Diese verfolgen das Ziel, dass sie sich für alle Positivbeispiele zu positiv auswerten, und für alle Negativbeispiele entsprechend zu negativ. Beispiel 2.3 verdeutlicht dies für die aus Beispiel 2.1 bekannten Urls.

#### Beispiel 2.3: Erstellung der Regex-Url-Regeln für den Dokumentenserver

```
# Positivbeispiele: Stelle sicher, dass die Dokumente gecrawlt werden
http://lips.informatik.uni-leipzig.de/browse/results?page=4
http://lips.informatik.uni-leipzig.de/files/thesis.pdf
http://lips.informatik.uni-leipzig.de/pub/2017-0

# Negativbeispiele: Verhindere, dass die Filterfunktion genutzt wird
http://lips.informatik.uni-leipzig.de/browse/results/taxonomy

# Regex-Url-Regeln, welche diese Anforderungen erfüllen
+http://lips.informatik.uni-leipzig.de/(pub|files)/.*
+http://lips.informatik.uni-leipzig.de/browse/results.*
-http://lips.informatik.uni-leipzig.de.*
```

Die Konfigurationen der einzelnen Bereiche werden schließlich [Bottom-up](#) zu einer einzigen Url-Filter-Liste vereinigt. Für diese Liste werden anschließend alle Positiv- und Negativ-Beispiele evaluiert. Nur wenn dabei für alle Beispiele das spezifizierte Verhalten beobachtet wird, ist diese Url-Liste valide und kann verwendet werden.

Durch diese Hilfswerkzeuge wurde im Rahmen der Arbeit die Site der Universität Leipzig anhand ihrer Fakultäten auf die einzelnen Bearbeiter aufgeteilt. Mit einer Reihe von Probe-Crawlings konnte die vollständige Regex-Url-Filter-Liste erstellt werden. Mithilfe dieser Vorbereitung konnten die für den Index verwendeten Dokumente unterbrechungsfrei gecrawlt werden. Dafür konnten die entstandenen Positiv-Beispiele direkt als [Seeds](#) eingesetzt werden.

---

<sup>12</sup> Die Implementierung findet sich in dem Github-Repository [check-nutch-regex-urlfilter](#)

---

## 2.2 Reproduzierbare Erzeugung eines Lucene Index mit Docker und Solr

Um die gecrawlten Dokumente in der in Abschnitt ?? beschriebenen Suchmaschine verfügbar zu machen, müssen sie in einen Lucene Index überführt werden.

Dafür wird im ersten Schritt eine sogenannte Conversion durchgeführt [CMS09a]. Dabei werden im vorliegenden Fall gecrawlte Dokumente, welche in den unterschiedlichsten Formaten vorliegen können<sup>13</sup> in ein einheitliches Format umgewandelt. Auf eine mögliche Reduzierung der Dokumente auf deren Main-Content wurde verzichtet<sup>14</sup>. Realisiert wurde die Conversion mit [Apache Tika](#).

Anschließend wird mit Nutch eine [Inversion der Links](#) durchgeführt. Dabei werden für alle Dokumente die Texte der eingehenden Links in dedizierten Feldern an dem Dokument abgespeichert. Im nächsten Schritt wird eine [Near Duplicate Detection](#) durchgeführt, welche näher in Abschnitt 2.3 erläutert wird.

Nutch kann den Lucene Index nicht selbstständig erzeugen. Aus diesem Grund wird mit einem [Solr Server](#) ein Index über die gecrawlten Dokumente erzeugt. Diesbezüglich wird mit Solr eine Text Transformation [CMS09a] durchgeführt. Diese umfasst die [Entfernung](#) von deutschen und englischen Stoppwörtern, sowie ein Stemming mit dem [Porter Stemmer](#). Die eigentliche Indexerzeugung mit Solr schließt diesen Prozess ab. Dabei werden neben den Dokument-Statistiken [CMS09a] für das spätere Scoring von Dokumenten auch Vorbereitungen zur Erzeugung von Snippets vorgenommen. Der so erzeugte Index kann unmittelbar in Lucene eingesetzt werden.

Die hier beschriebene Erzeugung des Index aus den gecrawlten Dokumenten ist von vielen Parametern abhängig. Dadurch wird eine häufige Wiederholung dieses Prozesses sinnvoll, um die Auswirkung von Anpassungen an verschiedenen Parametern zu prüfen. Um den Aufwand dafür gering zu halten, wurde der komplette Workflow reproduzierbar automatisiert<sup>15</sup>.

Dafür wird als Eingabe eine Menge von Nutch-Crawl-Directories erwartet. Diese werden im ersten Schritt bereinigt, um ein erneutes Parsing der rohen Dokumente durchzuführen. Danach werden die einzelnen, oben beschriebenen Ar-

---

<sup>13</sup> Häufigstes Format war HTML, aber auch PDF, Word oder Power Point waren vertreten.

<sup>14</sup> Die Extraktion des Main-Content ist mit Apache Tika unmittelbar möglich. Jedoch erschien es plausibel, dass für die gecrawlten Webseiten mit vergleichsweise wenig Rauschen zu rechnen ist.

<sup>15</sup> Das entsprechende Github-Repository mit den Scripten und der Konfiguration ist [Nutch-Tools](#)

beitsschritte durchgeführt. Sobald die Indexerzeugung beginnt, wird der dafür notwendige Solr-Server mit Docker bereitgestellt. Da die notwendige Konfiguration sowie das Index-Schema in den Container gemountet werden, ist auch dieser Schritt einfach wiederholbar.

## 2.3 Data Cleaning? Aggregation von Fast-Duplikaten

Duplikate und Fast-Duplikate treten in vielen Situationen auf [CMS09b]. Im Rahmen dieser Arbeit war die entsprechende Zielsetzung, diese transparent an den Nutzer zu kommunizieren. Abbildung 1 stellt dies an einem Beispiel dar.

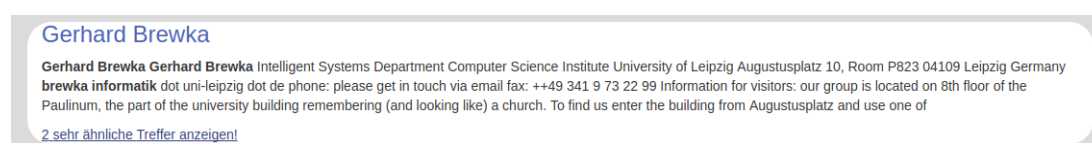


Abbildung 1: Ein Dokument mit 2 Fast-Duplikaten

Um dies zu ermöglichen, müssen zuerst die (Fast-)Duplikate identifiziert werden. Dafür wurde das [TextProfile-Signature](#)-Verfahren verwendet. Dieser Algorithmus weist einem Dokument einen Fingerprint zu. Dabei werden ähnliche Dokumenten auf den gleichen Wert abgebildet. Algorithmus 1 zeigt dies im Pseudocode. In Beispiel 2.4 wird der Fingerprint für einen Text berechnet.

### Algorithmus 1 : Text Profile

**Input** : Document  $\mathcal{D}$   
**Output** : Fingerprint for Document  $\mathcal{D}$   
1 TBA  
2 TBA  
3 TBA

### Beispiel 2.4: Berechnung des Fingerprints mittels TextProfileSignature

TODO MAIK

Nach der Zuweisung des Fingerprints werden entsprechend dem Discovery Szenario alle Paare von Fast-Duplikaten gesucht [CMS09b]. Durch dieses Vorgehen entstehen Gruppen ähnlicher Dokumente. Für jede solche Gruppe wird ein Repräsentant gewählt<sup>16</sup>. Dieser wird um die Urls der anderen Gruppenmitglieder angereichert. Abschließend werden die restlichen Dokumente einer Gruppe gelöscht<sup>17</sup>.

<sup>16</sup> Das längste Dokument wird der Vertreter.

<sup>17</sup> Die Implementierung ist in dem Modul [Custom Index Cleaning](#) zu finden.



---

## 3 Verarbeitung von Anfragen

Die Verarbeitung von Anfragen<sup>18</sup> wird im Allgemeinen durch drei Komponenten realisiert [CMS09a]. Auch die im Rahmen dieser Arbeit erstellte Suchmaschine bildet davon keine Ausnahme. Dementsprechend werden in den folgenden Abschnitten die Bestandteile User Interaction, Ranking und Evaluation vorgestellt.

### 3.1 User Interaction [CMS09a]

Die Komponente zur Nutzerinteraktion bietet eine Schnittstelle<sup>19</sup> zwischen dem Benutzer und der Suchmaschine. Um diesem Nutzer eine gewohnte Usability inklusive intuitiver Bedienung zu ermöglichen, wurden die in der Praxis verbreiteten Standards eingehalten.

Dementsprechend muss einem Nutzer das Absenden von Anfragen ermöglicht werden. Dafür ist es üblich, eine Landing Page mit prominent mittig platzierter Eingabemaske auszuliefern []. Abbildung ?? zeigt dies für die im Rahmen dieser Arbeit entwickelte Suchmaschine.

Nachdem ein Nutzer seine Anfrage spezifiziert hat, werden ihm die Ergebnisse präsentiert. Dafür erhält die Schnittstelle eine für die Query gerankte Liste von Dokumenten von der Ranking-Komponente. Diesbezüglich ist es gängig, einem Benutzer für kleinere Ausschnitte aus der Dokumentliste jeweils ausgewählte Metadaten in Verbindung mit einem für die Anfrage besonders relevanten Textausschnitt<sup>20</sup> zu präsentieren []. Relevante Wörter werden dabei speziell hervorgehoben. Die unter Berücksichtigung dieser Punkte entstandene Search Engine Result Page<sup>21</sup> wird in Abbildung ?? vorgestellt.

In der Regel wird einem Nutzer die Formulierung und Spezialisierung seiner Anfragen durch verschiedene Hilfsmittel erleichtert ?. In dem vorliegenden Projekt wurde eine Query Suggestion implementiert, welche Vervollständigungsvorschläge auf Basis populärer Anfragen liefert. Die dafür notwendige, dynamische Benutzerschnittstelle wird in Abbildung ?? gezeigt. Verwandte Maßnahmen zur Verbesserungen der Benutzbarkeit wie Spell Checking oder Query Refinement wurden zugunsten anderer Features nicht umgesetzt.

---

<sup>18</sup> Auch Query Processing

<sup>19</sup> Das entsprechende Modul ist TODO ADD LINK TO REPOSITORY

<sup>20</sup> Sogenannte Snippets

<sup>21</sup> SERP

---

Um den Suchmaschinenservice einem möglichst breiten Nutzerkreis zuzuführen, sind die Komponenten zur Benutzerinteraktion in Form eines Webfrontends realisiert. Für die Entwicklung werden die jeweils aktuellen Standards HTML5, CSS3 sowie JavaScript eingesetzt. Ausgeliefert werden diese Bestandteile durch einen Express-Webserver, welcher im Rahmen einer NodeJS-Anwendung betrieben wird. Aus Kompatibilitätsgründen werden Teile der Bootstrap-Library eingebunden. Falls ein Browser die History-API unterstützt, wird diese zur Bereitstellung von Deep-Links ohne einen Full-Page-Refresh geeigneter Seiten<sup>22</sup> verwendet.

Der Zugriff auf die Ranking-Komponente (siehe Abschnitt 3.2) und Query-Suggestions wird durch entsprechende [REST-Endpunkte](#) ermöglicht. Bereitgestellt werden diese durch eine [Spring-Boot-Anwendung](#). Die Query-Suggestions unterscheiden dabei zwischen nutzerbezogenen und globalen Vorschlägen. Durch nutzerbezogene Vorschläge ist es einem Nutzer möglich, von ihm bereits getätigte Anfragen zu wiederholen. Globale Vorschläge identifizieren innerhalb der Suchmaschine populäre Anfragen mit Hilfe von Logdaten. Für eine sinnvolle initiale Menge von globalen Vorschlägen wurden semantisch passende Vorschläge von Google gecrawlt und eingepflegt<sup>23</sup>.

## 3.2 Ranking [[CMS09a](#)]

Die Ranking-Komponente ist der Kern jeder Suchmaschine. Sie erzeugt für eine Query aus der User-Interaction-Komponente eine gerankte Liste von Dokumenten.

Dabei beeinflussen deren Effizienz<sup>24</sup> und Effektivität<sup>25</sup> die Nützlichkeit der Suchmaschine wesentlich. Um für beide Anforderungen einen sinnvollen Ausgangspunkt zu schaffen, wurde für das Ranking [Lucene](#) verwendet. Diese Software stellt einen geeigneten Einstieg dar, da darauf andere, in dieser Domäne verbreitete Softwarekomponenten aufbauen<sup>26</sup>. Verschiedene [Ranking-Algorithmen](#)<sup>27</sup> sind in Lucene verfügbar.

---

<sup>22</sup> About + Statistik mit Link zum Quellcode

<sup>23</sup> Der Quellcode für den Crawler ist [TODO](#) link zum Code. Damit wurden 2639 Vorschläge generiert

<sup>24</sup> Die Verarbeitung vieler Anfragen in kurzer Zeit.

<sup>25</sup> Die Qualität des Rankings: Kann die Suchmaschine relevante Informationen finden?

<sup>26</sup> Insbesondere die verbreiteten Tools [Elasticsearch](#) und [Solr](#) basieren auf Lucene. Deren erweiterter Funktionsumfang, wie Beispielsweise eine [horizontale Skalierung](#), wurde im Rahmen dieser Arbeit nicht benötigt.

<sup>27</sup> Und damit auch [Retrieval-Modelle](#)

---

Davon wurde [BM25F](#) eingesetzt, da es als Baseline für modernere Ranking-Algorithmen fungiert und für allgemeine Dokumentsammlungen bessere Ergebnisse<sup>28</sup> erzielt als die verfügbaren, [klassischen Vektor-Modelle](#) [[BYRN10](#)]. Dieses [Retrieval-Modell](#) besitzt die Fähigkeit, mehrere Felder<sup>29</sup> in die Berechnung des Scores einfließen zu lassen. Entsprechend wurden alle verfügbaren Felder einbezogen<sup>30</sup>. Da eine sinnvolle Wahl der Attribute<sup>31</sup> und deren Gewichtung im Allgemeinen in Verbindung mit der User-Relevanz steht, wurden alle Parameter bei ihren Standards belassen.

Das entsprechende Vorgehen für ein Tuning basierend auf Nutzer-Feedback wird in Abschnitt ?? vorgestellt.

### 3.3 Evaluation [[CMS09a](#)]

Die Aufgabe der Evaluationskomponente ist es, Effizienz und Effektivität zu monitoren. Dafür muss eine Aufzeichnung des Nutzerverhaltens sowie ausgewählter Systemmetriken vorgenommen werden.

Insbesondere zur Aufzeichnung des Nutzerverhaltens ist eine Identifikation der Nutzer notwendig. Darauf aufbauend kann protokolliert werden, welcher Nutzer welche Anfragen ausgeführt hat, welche Ergebnisseiten oder Query-Suggestions einem Nutzer präsentiert worden, sowie gegebenenfalls welche Ergebnisse oder Query Suggestions ausgewählt wurden<sup>32</sup>.

Bei einer technischen Realisierung dieser Punkte fällt auf, dass es sich um sogenannte [Cross-Cutting Concerns](#) handelt. Ein bewährtes Mittel derartige Funktionalitäten zu realisieren, ohne die Komplexität der betroffenen Systemteile unnötig zu erhöhen, stellt die Aspektorientierte Programmierung<sup>33</sup> dar []. Die

---

<sup>28</sup> Entsprechendes Tuning der von BM25F verwendeten Parameter vorausgesetzt [[BYRN10](#)].

<sup>29</sup> Felder werden auch als Attribute oder Features bezeichnet.

<sup>30</sup> Alle Felder mit unmittelbarem Bezug zu dem Inhalt der Dokumente. Dies sind der vollständige Text eines Dokuments, sowie deren Titel, URL und Anchor-Texte eingehender Links.

<sup>31</sup> Einschließlich eventuell abgeleiteter Features.

<sup>32</sup> Das erweitern dieser Informationen um die entsprechenden Event-Zeitpunkte erlaubt eine sinnvolle Rekonstruktion des Nutzerverhaltens.

<sup>33</sup> AOP

---

darin verwendeten Aspekte definieren, was<sup>34</sup>, wann<sup>35</sup> und wo<sup>36</sup> auszuführen ist [].

In dem für die Interaction-Komponente verwendeten Framework werden alle Endpunkte über speziell annotierte Methoden bereitgestellt. Diesen ist gemein, dass sie ein [POJO](#)<sup>37</sup> als Model für die Antwort zurückgeben. Dieses Modell wird je nach Endpunkt später in ein HTML-Template<sup>38</sup> gerendert oder JSON-serialisiert.

Unter diesen Voraussetzungen eignen sich diese Methoden hervorragend, um sie durch Aspekte um die gewünschten Funktionalitäten zu erweitern<sup>39</sup>. Dazu wird ein Effizienz-, ein User-Identification-, sowie ein Logging-Aspekt eingesetzt. Der Effizienz-Aspekt misst dabei Beispielfür weitere Systemmetriken die Bearbeitungszeit von Anfragen, und erweitert das zurückgegebene Model entsprechend. Der User-Identification-Aspekt stellt vor dem Aufruf jeder Endpunkt-Methode sicher, dass der Request mit einer eindeutigen Identifikation des Nutzers versehen ist. Dies wird über einen Cookie realisiert. Der Logging-Aspekt protokolliert ausnahmslos alle verfügbaren Informationen. Dazu notiert er für jeden Endpunkt den Request des Clients zusammen mit dem zurückzugebenden POJO-Model<sup>40</sup>.

Die vom Logging-Aspekt protokollierten Informationen können sowohl Online<sup>41</sup>, als auch Offline analysiert werden. Um beides zu ermöglichen, wurde mit [Apache Kafka](#) ein ... mit persistenter Datenhaltung eingesetzt. Dazu publiziert der Logging-Aspekt seine Daten als Events nach Kafka. Um diesen Logging-Aspekt so einfach wie möglich zu halten, bereitet ein [Stream Processor](#) diese Daten weiter auf<sup>42</sup>. Online-Analysen wie die Query-Suggestion können sich nun unmittelbar an die für sie interessanten Events subscriben. Offline Analysen können die persistierten Events gleichzeitig in einem Batch-Vorgang verarbei-

---

<sup>34</sup> In der zugehörigen AOP Terminologie definiert der zu einem Aspekt gehörende Advice die Aufgabe, also was von dem Aspekt zu erledigen ist. []

<sup>35</sup> Der Advice eines Aspekts definiert neben dem was auch zu welchen Zeitpunkten ein Aspekt auszuführen ist. Unter anderem kann vor, nach, oder das wrappen einer Zielmethode spezifiziert werden [].

<sup>36</sup> Durch sogenannte Pointcuts werden eine oder mehrere Methoden definiert, an die der Advice gewoben wird [].

<sup>37</sup> Jeweils definiert in...

<sup>38</sup> Spezifiziert in ....

<sup>39</sup> Schließlich lassen sie sich eindeutig und generisch über die notwendigen Endpunkt-Anotationen identifizieren

<sup>40</sup> Mit diesem Vorgehen lassen sich alle angesprochenen Informationen zur Rekonstruktion des Nutzer-Verhaltens erheben, da insbesondere die Links zu den Dokumenten auf dem Server durch Weiterleitungen aufgelöst werden.

<sup>41</sup> Beispielfür wurde das im Rahmen dieser Arbeit für die Query-Suggestions vorgenommen

<sup>42</sup> auf die richtige bahn werfen, Siehe modul...

---

ten. Als Beispiel dafür stellt Abbildung ?? einen mit Python realisierten Import der Events nach [Neo4j](#) zur Visualisierung dar.

Die hier vorgestellte Evaluationskomponente funktioniert für alle Clients, welche Cookies erlauben und den Referrer-Header korrekt setzen.

---

## **4 Analyse der Logdaten**

Bla

### **4.1 Durchführung eines Laborexperiments**

Bla

---

## Literatur

- [BYRN10] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval: The concepts and technology behind search*, chapter Chapter 3.5: Alternative Probabilistic Models, page 107. Pearson, 2010.
- [CMS09a] W. Bruce Croft, Donald Metzler, and Trevor Strohman. *Search Engines Information Retrieval in Practice*, chapter Chapter 2: Architecture of a Search Engine, pages 13–30. Pearson, 2009.
- [CMS09b] W. Bruce Croft, Donald Metzler, and Trevor Strohman. *Search Engines Information Retrieval in Practice*, chapter Chapter 3: Crawls and Feeds, pages 31–73. Pearson, 2009.
- [MRS09] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *An Introduction to Information Retrieval*, chapter Chapter 20: Web crawling and indexes, pages 443–459. Cambridge University Press, 2009.
- [nut13] Apache nutch documentation: invertlinks, 2013. [Online erhältlich unter <https://wiki.apache.org/nutch/bin/nutch%20invertlinks>. Abgerufen am 25.03.2018].
- [wik18a] Free software license — Wikipedia, the free encyclopedia, 2018. [Online erhältlich unter [https://en.wikipedia.org/wiki/Free\\_software\\_license](https://en.wikipedia.org/wiki/Free_software_license). Abgerufen am 22.03.2018].
- [wik18b] Free software license — Wikipedia, the free encyclopedia, 2018. [Online erhältlich unter <https://de.wikipedia.org/wiki/Facettensuche>. Abgerufen am 25.03.2018].