Informatik
Hauptcampus

H O C H
S C H U L E
T R I E R

Implementierung einer Enterprise Search Engine für das Dietrich Online Projekt

Implementation of an enterprise search engine for the Dietrich Online project

Florian Reitz

Bachelor-Abschlussarbeit

Betreuer: Professor Doktor Christoph Schmitz

Trier, den 15.10.2019 15.3.2020

Vorwort

Ein Vorwort ist nicht unbedingt nötig. Falls Sie ein Vorwort schreiben, so ist dies der Platz, um z.B. die Firma vorzustellen, in der diese Arbeit entstanden ist, oder einigen Leuten zu danken, die in irgendeiner Form positiv zur Entstehung dieser Arbeit beigetragen haben. Auf keinen Fall sollten Sie im Vorwort die Aufgabenstellung näher erläutern oder vertieft auf technische Sachverhalte eingehen.

Kurzfassung

In der Kurzfassung soll in kurzer und prägnanter Weise der wesentliche Inhalt der Arbeit beschrieben werden. Dazu zählen vor allem eine kurze Aufgabenbeschreibung, der Lösungsansatz sowie die wesentlichen Ergebnisse der Arbeit. Ein häufiger Fehler für die Kurzfassung ist, dass lediglich die Aufgabenbeschreibung (d.h. das Problem) in Kurzform vorgelegt wird. Die Kurzfassung soll aber die gesamte Arbeit widerspiegeln. Deshalb sind vor allem die erzielten Ergebnisse darzustellen. Die Kurzfassung soll etwa eine halbe bis ganze DIN-A4-Seite umfassen.

Hinweis: Schreiben Sie die Kurzfassung am Ende der Arbeit, denn eventuell ist Ihnen beim Schreiben erst vollends klar geworden, was das Wesentliche der Arbeit ist bzw. welche Schwerpunkte Sie bei der Arbeit gesetzt haben. Andernfalls laufen Sie Gefahr, dass die Kurzfassung nicht zum Rest der Arbeit passt.

The same in english.

Inhaltsverzeichnis

1 Einleitung und Problemstellung 1

2 Vergleich der Enterprise Search Engines 2
2.1 Apache Lucene Core 3
2.2 Terrier 4
2.3 Sphinx 4
2.4 Apache Solr 4
2.5 ElasticSearch 5
2.6 Fess 5
2.7 Algolia 5
2.8 Manticore Search 6
2.9 Xapian 6
2.10 Datafari 6
2.11 Tabellarischer Vergleich 7
2.12 Vorauswahl 7
2.12.1 Lucene Core 7
2.12.2 Sphinx 7
2.12.3 Solr 7
2.12.4 ElasticSearch 9
2.12.5 Fess 9
2.12.6 Algolia 9
2.12.7 Xapian 9
2.12.8 Datafari 9
2.12.9 Abschluss 10

3 Genauer Vergleich 11
3.1 Testsystem 11
3.2 Aufbau der Tests 11
3.2.1 Installation 11
3.2.2 Indexierung 12
3.2.3 Dokumentation 12
3.2.4 Absetzen einer Anfrage und Integration in PHP 12
3.3 Solr 13

3.3.1 Installation 13
3.3.2 Indexierung 14
3.3.3 Oberfläche 16
3.3.4 Dokumentation 16
3.3.5 Absetzen einer Anfrage und Integration in PHP 17
3.4 Datafari 18
3.4.1 Installation 18
3.4.2 Indexierung 18
3.4.3 Oberfläche 19
3.4.4 Dokumentation 20
3.4.5 Absetzen einer Anfrage und Integration in PHP 20
3.5 ElasticSearch 21
3.5.1 Installation 21
3.5.2 Indexierung 21
3.5.3 Oberfläche 22
3.5.4 Dokumentation 22
3.5.5 Absetzen einer Anfrage und Integration in PHP 22

4 Nutzung des Open Archives Initiative Protokolls für Metadaten 23
4.1 OAI Harverester 23
4.2 Support der Enterprise Search Engines 23
4.3 Auswertung 23

5 Zusammenfassung und Ausblick 25

Literaturverzeichnis 26

Glossar 28

Erklärung der Kandidatin / des Kandidaten 29

Abbildungsverzeichnis

3.1 Tabellenaufbau der Lemma-Administration Übersicht. 12
3.2 Frontend Ansicht der Lemma-Administration mit geladenen Buchstaben S (Ausschnitt). 13
3.3 Oberfläche der Indexierung mit Laufzeit. 16
3.4 Startseite der Weboberfläche von Solr. 17
3.5 Übersichtsseite des Querys in Datafari. 19
3.6 Kibana Integration in Datafari. 20
3.7 Dokumentationsseite für den JDBC Treibers von Datafari. 21

Tabellenverzeichnis

2.1 Feature-Vergleich der verschiedenen Enterprise Suchmaschinen 8

1

Einleitung und Problemstellung

Begonnen werden soll mit einer Einleitung zum Thema, also Hintergrund und Ziel erläutert werden.

Weiterhin wird das vorliegende Problem diskutiert: Was ist zu lösen, warum ist es wichtig, dass man dieses Problem löst und welche Lösungsansätze gibt es bereits. Der Bezug auf vorhandene oder eben bisher fehlende Lösungen begründet auch die Intention und Bedeutung dieser Arbeit. Dies können allgemeine Gesichtspunkte sein: Man liefert einen Beitrag für ein generelles Problem oder man hat eine spezielle Systemumgebung oder ein spezielles Produkt (zum Beispiel in einem Unternehmen), woraus sich dieses noch zu lösende Problem ergibt.

Im weiteren Verlauf wird die Problemstellung konkret dargestellt: Was ist spezifisch zu lösen? Welche Randbedingungen sind gegeben und was ist die Zielsetzung? Letztere soll das beschreiben, was man mit dieser Arbeit (mindestens) erreichen möchte.

2

Vergleich der Enterprise Search Engines

In ersten Schritt werden diverse Enterprise Search Engines evaluiert. Dafür wurde eine Anforderungsliste mit den Mitarbeitern erstellt. Die Systeme welche bei diesem Vergleich nach Features verglichen. Die Suchmaschinen, die am besten abschneiden werden anschließend aufgesetzt und genauer verglichen.

Diese Liste zeigt alle Basis-Funktionen, die für die Bibliothek hier K.O. Kriterien sind.

- Open Source oder Kostenlos
- Unterstützung von Facetten
- Ranking der Suchergebnisse
- Volltextsuche
- Support für PDF, SQL, XML
- Logging-Möglichkeit

Des Weiteren sind die folgenden Funktionen stark erwünscht, allerdings nicht ausschlaggebend zu Disqualifikation.

- Support für PostgreSQL
- Backup Funktionen
- Auto-Korrektur und Auto-Vervollständigung
- Security Features
- PHP-Support
- bezahlter Support

Durch die begrenzten finanziellen Mittel und die lange Projektlaufzeit besteht die Notwendigkeit eine kostenfreie, im besten Fall sogar eine Open Source Suchmaschine zu finden. Auch äußerst wichtig ist der Support für Facetten, da Dietrich-Online als Suchmaschine den Nutzer einige Tools zum Verfeinern seiner Suchergebnisse zur Verfügung stellen will. Da wir hier mit großen Datenmengen arbeiten ist das Ranking auch von größer Bedeutung. Es können nicht alle Daten gleichzeitig dargestellt werden, von daher sollten die besten Treffer auch zuerst angezeigt werden. Dabei ist die Transparenz sehr wichtig. Wie das Ranking funktioniert muss erklärt und auf der Webseite veröffentlicht werden. Die Volltextsuche wird es möglich machen auch nach Schlüsselwörtern im Titel oder Beschreibungen zu suchen. Der Support für die verschiedenen Dateiformaten ergibt sich dadurch, dass

dieses Projekt stark gewachsen ist. Es gibt viele Prozessschritte, welche auf denselben Daten in verschiedenen Formen arbeiten. Darunter werden alle Einträge im XML Format bearbeitet, es gibt alle Scans als PDF und für die Webseite sind alle Daten nochmals in der Datenbank vorhanden. Und alle Daten sind wichtig, da die Mitarbeitern-Module auf anderen Daten arbeiten, als der Nutzer. Dabei ergibt sich allerdings auch das Problem, dass sich Daten in der Datenbank doppeln werden. Dies wird sicherlich später nochmals von Bedeutung. Als letztes ist es noch wichtig, dass zumindest ein Fehler-Logging geboten wird, damit schnell und effizient Probleme mit dem System erkannt und gelöst werden können. Ein erweitertes Monitoring ist ein Bonuspunkt.

Ein Support für PostgreSQL ist für dieses Projekt nicht so wichtig, allerdings könnte es sein, dass der Server später auch andere Datenbanken verwaltet. Dadurch wäre ein Support für diese Datenbankstruktur wünschenswert. Es gibt bei den Maschinen in der Bibliothek sowieso ein Backup-Maschinismus. Allerdings ist eine manuelle Backup-Lösung wünschenswert, um die Suchmaschine losgelöst zu sichern und gegebenenfalls auch einfach auf einen anderen Server umzuziehen zu können. Auto-Korrektur und Auto-Vervollständigung sind beide sehr interessant, um den Nutzer mehr Komfort-Funktionen bieten zu können, ohne selbst groß implementieren zu müssen. Die Sicherheitsfunktionen sind für die Suchmaschinen mit Web-Oberfläche interessant. Generell sollte der Server ja nur intern anzusprechen sein. Wenn es allerdings eine Web-Oberfläche gibt, kann es sein, dass diese per Reverse Proxy ansprechbar gemacht wird, um eine Administration aus dem Internet möglich zu machen. Daher wäre es gut, wenn der Server ein Login-System bietet. Einen PHP-Connector, welcher Objekte zum Umgang mit der Suchmaschine bietet, wäre auch wünschenswert. Allerdings bieten einige Suchmaschinen auch die Möglichkeit über JSON Anfragen an die Suchmaschine zu stellen. Es sollte zumindest eine der beiden Methoden verfügbar sein, damit die Suchmaschine einfach von PHP aus zu erreichen ist. Sollte es mal ein Problem geben, was nicht im Haus gelöst werden kann, wäre die Möglichkeit auf bezahlten Support von Vorteil.

2.1 Apache Lucene Core

Lucene Core ist eine Open Source Enterprise Search Engine von der Apache Foundation geschrieben in Java.

Das Lucene Projekt wurde im Jahre 1997 vom Entwickler Doug Cutting gestartet. 2001 ist es dann der Apache Foundation als Teil des Jakarta-Projekts beigetreten und wurde 2005 ein eigenes Hauptprojekt der Foundation. [1]

Lucene Core erfüllt alle der Grundanforderungen. Für das Monitoring gibt es eine Klasse, die es auch ermöglicht, dass langsame Query's geloggt werden. Zudem bietet es Support für PostgreSQL und Auto-Korrektur/Auto-Vervollständigung. Da es keine Web-Oberfläche besitzt, gibt es auch keine weiteren Sicherheitsfunktionen. Einen PHP-Connector gibt es leider auch nicht, man müsste daher mit PHP direkte Systemaufrufe an Java machen. Bezahlten Support gibt es hier nicht, da dieses Projekt zur Apache Foundation gehört. [2]

2.2 Terrier

Terrier ist eine Open Source Enterprise Search Engine geschrieben in Java. Entwickelt und gepflegt wird diese von der University of Glasgow. Sie existiert bereit seit 10 Jahren und besitzt, laut Webseite, eine breite Nutzerbasis. Terrier erfüllt leider nicht alle Grundanforderungen, da es keine direkte Möglichkeit gibt SQL zu indexieren. Es gibt allerdings eine Möglichkeit das SQL in JSON zu konvertieren und dieses dann in die Suchmaschine einzupflegen. Auch scheint es keinen Support für Facetten gegeben. [3]

2.3 Sphinx

Sphinx ist eine Suchmaschine entwickelt von Andrew Aksyonoff. Das Akronym steht für „SQL Phrase Index“.[4] Bis zur Version 2 wurde sie aktiv Open Source entwickelt. Ab Version 3 wurde die Entwicklung Closed Source. Auf der Github-Seite steht: „The sources for 3.0 will also be posted here when we decide to make those publicly available.“ [5], also gibt es kein genaues Datum ob und wann die Version 3 Open Source geht. Version 3.1.1 wurde im Oktober 2018 veröffentlicht und seitdem lässt sich auch nichts mehr über das Projekt finden. Von daher ist davon auszugehen, dass das Projekt nicht mehr weitergeführt wird.

Zu den Features ist festzuhalten, dass es keinen nativen PDF-Support in der Open-Source Version gibt, ab der Version 3 wurde jedoch ein Dokumenten-Speicher eingebaut. Allerdings werden die anderen Anforderungen alle erfüllt. Es existiert, laut Webseite, sogar ein bezahlter Support, allerdings ist fraglich, ob man mit der Firma noch in Kontakt treten kann. [6]

2.4 Apache Solr

Apache Solr ist eine, auf Lucene Core 2.1 basierende, viel eingesetzte Search Engine von der Apache Foundation. Sie erweitert Lucene Core, um ein grafische Benutzeroberfläche und einige weitere Funktionen. Die Entwicklung dafür begann 2004 als ein internes Projekt von CNET um eine bessere Suche für die eigene Webseite zu bieten. Später im Jahre 2006 hat CNET dann den Source Code an die Apache Foundation weitergegeben. Dadurch wurde es zu einem eigenen Projekt bei der Apache Foundation. Im Jahre 2009 wurde Solr dann in das Apache Lucene Projekt eingefügt. Dort wird es auch aktuell noch weiterentwickelt. [7]

Solr wird unter anderem von DuckDuckGo und Best Buy eingesetzt. Durch die Unterstützung von der Apache Foundation längerfristige Weiterentwicklung abzusehen.

Da Solr zur Apache Foundation gehört, ist es Open Source. Es bietet viele Funktionen von Haus, womit es alle Grundanforderungen erfüllt und besitzt darüber hinaus auch Support für fast alle Bonus-Features. Einzig und allein gibt es keinen bezahlten Support, dafür allerdings eine große Community, welche man durch einen Mailing Listen oder IRC erreichen kann. [8]

2.5 ElasticSearch

Eine weitere große Enterprise Search Engine ist ElasticSearch. Auch dieses Projekt arbeitet auf der Basis von Lucene. Zu den bekanntesten Kunden zählen Ebay und Adobe. Gestartet wurde das Projekt in den jungen 2000ern von Shay Banon, um eine Verwaltung für die Rezepte seiner Frau zu schaffen. Im Juni 2012 haben sich dann Logstash, ein Logging Dienst, Kibana, ein UI für ElasticSearch, und ElasticSearch zusammengetan. So enstand der ELK-Stack. Die enstandene Firma nennt sich: ElasticSearch Incorporated. Seitdem wurden der Produktkatalog stetig erweitert und die Produkte weiterentwickelt. Viele der weiteren Produkte sind allerdings nicht mehr Open-Source oder kostenlos. Der ELK-Stack ist allerdings weiterhin, und es wurde versprochen, dass es so bleibt, kostenlos und ElasticSearch zudem auch als Open-Source Variante zu haben. Eine genauere Aussage, welche Features nur in der kostenlosen und nicht in der Open-Source Variante zu finden sind, finden sich in der Tabelle 2.1.

ElasticSearch erfüllt alle der Grundanforderungen, auch in der Open-Source Variante. Auch viele der optionalen Features kann man in der Open Source Variante genießen. Einzig die Sicherheitsfunktionen, wie rollen-basierte Authentifizierung sind der kostenlosen Variante vorbehalten. Eine Möglichkeit auf bezahlten Support besteht auch. [9]

2.6 Fess

Fess ist eine Enterprise Search Engine basierend auf ElasticSearch entwickelt von dem japanischen Unternehmen CodeLibs. Die Suchmaschine ist komplett Open-Source und wird unter der Apache-Lizenz entwickelt.

Die Suchmaschine erfüllt alle Grundanforderungen. Darüber hinaus bietet es Support für PostgreSQL, Backups (sogar über die Web-Oberfläche) und Auto-Korrektur und Vervollständigung. Es gibt keinen direkten PHP Support, allerdings können anfragen über JSON geschickt werden. Ein bezahlter Support ist auch über die Firma N2SM Incorporated. [10] möglich. Bei dieser Arbeiten anscheinend auch einige der Entwickler von Fess. Sicherheitsfunktionen werden über rollen-basierte Authentifizierung mitgeliefert. [11]

2.7 Algolia

Algolia ist eine cloud-basierte Search Engine, welche unter anderem von Twitch und Lacoste verwendet wird. Die Suchmaschine wird hierbei als SAAS (Software as a Service) angeboten. Hierbei lädt man die Daten auf Algolia Server und dann daraufhin über eine API-Schnittstelle die Suchen auf den Daten in der Cloud ausführen.

Sie erfüllt alle Grundanforderungen, wobei allerdings in der kostenlosen Variante grade einmal 10 Tausend Einträge und 50 Tausend Operationen im Monat erlaubt sind. Auch die optionalen Anforderungen werden so weit alle erfüllt. Der

bezahlte Support wird ab der Starter Edition für 30 Dollar im Monat mitgeliefert. [12]

2.8 Manticore Search

Manticore Search Engine ist eine Open-Source Lösung basierend auf Sphinx 2.3. Nachdem Sphinx Closed Source gegangen ist, wurde auf der letzten offenen Version die erste Version von Manticore Search entwickelt. Zu den großen Kunden zählen unter anderem Craigslist und Boardreader.

Manticore erfüllt fast alle Grundanforderungen, allerdings ist kein nativer PDF-Support gegeben. Es muss daher auf eine Konvertierung der Daten auf XML gesetzt werden. Es findet sich außerdem eine Unterstützung von PostgreSQL, sowie Auto-Korrektur und Vervollständigung. Auch werden Log-Dateien produziert. Zuletzt gibt es noch eine Option auf bezahlten Support. Die Supportkosten sind dabei direkt auf der Webseite angegeben und belaufen sich auf 3000 Dollar im Jahr für den Standard Support. [13]

2.9 Xapian

Xapian ist eine Open-Source Enterprise Suchmaschine, welche von Zeit-Online, der Universitätsbibliothek Köln und der Debian Webseite genutzt wird. Die Suchmaschine basiert auf Open Muscat, einer Suchmaschine, welche an der Cambridge Universität in den 1980ern von Dr. Martin Porter entwickelt wurde. In 2001, als Open Muscat Closed Source ging, haben sich einige Entwickler die letzte offene Version genommen und diese weiterentwickelt.

Sie erfüllt alle der Grundanforderungen, wenn auch Logging nur im Grundsinne erfüllt wird, da nur Fehlermeldungen ausgegeben werden. Des Weiteren bietet die Suchmaschine Support für PostgreSQL. Auch eine Replikations-Funktion wird mitgeliefert. Sie bietet auch Auto-Korrektur und Auto-Vervollständigung. Ein Login-System mit Sicherheitsfunktionen gibt es durch das Fehlende Frontend Administration nicht. Es gibt allerdings die Möglichkeit mit Omega eine CGI-Suche zu nutzen. Diese Suche bietet allerdings keine Administration, sondern nur eine grafische Oberfläche für Suchanfragen.

Auch gibt es eine Möglichkeit für bezahlten Support. Auf der Webseite werden zwei Firmen angegeben, welche bezahlten Support bieten. Allerdings funktioniert der Link aktuell nur für eine der beiden Firmen aktuell. Zudem ist ein PHP-Connector für die Suchmaschine vorhanden, was die Einbindung ist das Projekt vereinfacht. [14]

2.10 Datafari

Datafari ist eine Open-Source Enterprise Suchlösung vom französischen Entwickler France Labs. Das Entwicklerstudio wurde 2011 gegründet und hat sich es sich zum

Ziel gemacht, die beste Open-Source Enterpise Suchlösung zu erstellen. [15] Als Fundament dafür wurde hierbei Solr verwendet. Dies wurde dann mit dem ELK-Stack für die Analyse gemischt. Zu den Kunden zählt unter anderem das Linux Magazin, welches diese Suchmaschine in einer ihrer Ausgaben vorstellt. [16]

Die Suchmaschine erfüllt alle Grundanforderungen. Darüber hinaus bietet sie auch Support für PostgreSQL, Auto-Korrektur und Vervollständigung, sowie den bezahlten Support. Eine Backup-Funktion gehört zu den Premium-Funktionen, genauso wie erweiterte Sicherheitsfunktionen. Allerdings ist zumindest die Rollenbasierte-Authentifizierung auch in der Open-Source Variante zu haben. Einen direkt PHP-Connector gibt es nicht, allerdings wird eine HTTP-API zu verfügung gestellt, welche es ermöglicht per POST-Request Anfragen zu stellen. [17]

2.11 Tabellarischer Vergleich

Alle Suchmaschinen die zumindest die Grundanforderungen erfüllen, werden hier in der Tabelle 2.1 nun nochmals aufgeführt für einen leichteren Vergleich.

2.12 Vorauswahl

Nach einem ersten Feature-Vergleich haben nur 7 Suchmaschinen die Grundanforderungen erfüllt. Davon werden nun 4 Stück in den genaueren Vergleich genommen, bei dem die Systeme nun aufsetze und teste. Ich gehe nun die Suchmaschinen der Reihe nach durch und gebe zur jeder eine Begründung warum oder warum sie es nicht in den genauen Vergleich geschafft hat.

2.12.1 Lucene Core

Lucene Core scheidet dadurch aus das es zum einen keine direkte Schnittstelle liefert, die gut mit PHP zur erreichen ist. Die einzige Möglichkeit wären direkte System-Calls, wodurch es schwerer ist, die Systeme voneinander zu separieren, zum Beispiel auf verschiedenen Server zu legen. Zum anderen gibt es für Lucene Core ja eine Erweiterung, namentlich Solr, welches alle diese Probleme löst. [2]

2.12.2 Sphinx

Sphinx wäre eine interessante Alternative gewesen, allerdings durch den Kommunikationsverlust und die gestoppten Updates (Es gab schon seit über einem Jahr kein Update mehr), ist dieses Projekt wohl als tot anzusehen. [6]

2.12.3 Solr

Wie schon bei Lucene Core kurz angesprochen, liefert Solr viele der Funktionen, die in diesem Umfeld benötigt werden, direkt mit. Dazu besitzt es eine Web-Oberfläche

	LC	**SH**	**AS**	**ES**	**FE**	**AG**	**XP**	**DF**
Open Source oder Kostenlos	x	x	x	x	x	x	x	x
Unterstützung von Facetten	x	x	x	x	x	x	x	x
Ranking der Suchergebnisse	x	x	x	x	x	x	x	x
Volltextsuche	x	x	x	x	x	x	x	x
Support für PDF, SQL, XML	x	x*	x	x	x	x	x	x
Monitoring / Logging	x	x	x	x	x	x	x?	x
Support für PostgreSQL	x	x	x	x	x	x	x	x
Backup	-	-	x	x	x	x+	-	-
Auto-Korrektur und Vervollständigung	x	x	x	x	x	x	x	x
Security Features	-	-	x-	x*	x	x	-	x
PHP Support	-	x	x	x	-	x	x	-
bezahlter Support	-	x	-	x	x	x	x	x
unter aktiver Entwicklung**	x	-	x	x	x	x	x	x
offizielles Docker Image	-	-	x	x	x	-	-	x
Synonym Support	x	x	x	x	x	x	x	x
Web-Interface	-	-	x	x	x	x	-	x
Plugin Support	-	x	x	x	x	-	-	-
JSON oder RESTful API	-	x*	x	x	x	-	x–	x
SQL-Like Query Support	-	x	-	x	-	-	-	-

Tabelle 2.1. Feature-Vergleich der verschiedenen Enterprise Suchmaschinen

* = Feature nur in der kostenlosen Variante verfügbar.
** = Update innerhalb des letzten halben Jahres
– = Nur mit Omega CGI installiert
+ = Anbieter kümmert sich um das Feature
- = Funktion nur per Plugin Implementiert

Die Tabelle vergleicht einige Features der ausgewählten Search Engines. Dabei wurden die Namen aus Platzgründen wie folgt abgekürzt:

- LC = Lucene Core 2.1
- SH = Sphinx 2.3
- AS = Apache Solr 2.4
- ES = ElasticSearch 2.5
- FE = Fess 2.6
- AG = Algolia 2.7
- XP = Xapian 2.9
- DF = Datafari 2.10

zur Administration. Durch die aktive Entwicklung unter der Apache-Lizenz und die große Community ist auch eine Langzeit-Entwicklung sehr wahrscheinlich. Daher ist Solr die erste der vier Kandidaten für das genauere Testen. [8]

2.12.4 ElasticSearch

Auch ElasticSearch basiert auf Lucene, ist aber im Gegensatz nicht komplett Open-Source und bietet auch eine kommerzielle Version an, was allerdings auch bedeutet, dass es bezahlten Support gibt. Die Community und der Kundenkreis ist auch groß, was eine Weiterentwicklung sehr wahrscheinlich macht. Auch diese Suchmaschine bietet eine Web-Oberfläche mit besonderem Augenmerk auf die Visualisierung der Daten, was für spätere Administratoren einen einfacheren Einstieg in die Administration liefern könnte. Daher wird auch ElasticSearch den genaueren Vergleich mit eingebunden. [9]

2.12.5 Fess

Fess ist eine Suchmaschine, welches auf ElasticSearch basiert, was ja seinerseits auf Lucene basiert. Von den Funktionen her bietet Fess, dank der Basis, viele Möglichkeiten. Es gibt auch kommerziellen Support, allerdings nur von einer japanischen Firma. Dadurch kann es schwere werden mit dem Support in Kontakt zu treten, was mich dazu veranlasst Datafari 2.10 dieser Suchmaschine vorzuziehen. Von der Idee her machen die beiden Firmen ja etwas ziemlich Ähnliches. [11]

2.12.6 Algolia

Als einzige Cloud-only Lösung im Vergleich, bietet Algolia eine interessante Alternative. Leider sind im kostenlosen Bereich nicht genügend Einträge speicherbar. Auch sind 50.000 Operationen zu wenig für die das Dietrich-Online Projekt. Von daher fällt diese Suchmaschine durch diese Limitationen raus. [12]

2.12.7 Xapian

Xapian ist als einzige Suchmaschine ohne Web-Administration im engeren Vergleich. Durch die Nutzung der Suchmaschine für die Bibliothek Köln gibt es einen Kunden der Software, welcher einen ähnlichen Anwendungsfall besitzt. [18] Dadurch und die Erfüllung vieler weiterer Kriterien kommt diese Suchmaschine auch in die engere Auswahl. [14]

2.12.8 Datafari

Datafari ist der letzte Kandidat, der es in die engere Auswahl schafft, wie oben schon erwähnt gewinnt diese Suchmaschine gegen Fess, durch die Entwicklung in Frankreich und der daher besser zu erreichende Support. Darüber hinaus ist es interessant zu sehen, ob das Entwicklerstudio schafft Solr sinnvoll zu erweitern und die Datenaufbereitung mit ElasticSearch so zu liefern, dass sich die Suchmaschine trotzdem noch wie aus einem Guss anfühlt. [17]

2.12.9 Abschluss

Daraus ergibt sich nun, dass die folgenden Suchmaschinen es in die genauere Auswahl geschafft haben: die beiden Platzhirsche Apache Solr 2.4 und ElasticSearch 2.5, sowie Datafari 2.10 als Open-Source Erweiterung von Solr und Xapian 2.9 als Lucene-freie Alternative. Ich hätte auch gerne eine Suchmaschine, welche auf Sphinx basiert dabei gehabt, allerdings, war die einzige noch aktive Alternative zu Sphinx direkt Manticore Search. Diese hat aber bisher noch keinen PDF-Import, welcher leider zwingend erforderlich ist.

3

Genauer Vergleich

In diesem Kapitel werden die vorher ausgewählten Suchmaschinen genauer verglichen. Dafür werden alle vier Suchmaschinen aufgesetzt, um einen Ersteindruck zu erstellen. Da ich dieses Projekt nicht nach meiner Bachelor-Arbeit weiter betreuen kann, ist es auch wichtig zu schauen, wie leicht es für einen neuen Administrator ist, sich in dieses System einzuarbeiten. Deshalb wird ein besonderes Augenmerk auf die Dokumentation und Oberfläche, insofern vorhanden, gelegt. Die genaueren Kriterien werden nun im Folgenden mit Erklärungen aufgeführt.

3.1 Testsystem

Das Testsystem besitzt die folgende Spezifikationen:

- CPU: 4 Kerne
- RAM: 16 Gigabyte
- Festplattenspeicher: 20 GB
- Betriebssystem: Ubuntu 18.04.03 LTS

Auf das System wird zudem die MariaDB Datenbank von Dietrich-Online Projekt als Datenquelle eingespielt. Zudem mussten einige Programme während der Vorbereitung des Servers durch die Administratoren aufgespielt werden. Eine genaue Liste findet sich im Anhang. Diese Programme werde daher auch als gegeben bewertet.

3.2 Aufbau der Tests

3.2.1 Installation

Im ersten Schritt wird die Installation bewertet, dabei wird geschaut, wie einfach es ist die Software zu installieren. Dabei wird der Installationsprozess kurz beschrieben. Existiert zum Beispiel ein Installations-Wizard? Wie viel muss manuell in den Dateien geändert werden? Müssen viel externe Programme nachinstalliert werden?

3.2.2 Indexierung

Hierbei wird geschaut, wie einfach die Indexierung von den Daten aus der Datenbank ausfällt. Dabei wird auch geschaut, ob es möglich ist, Daten direkt von der Oberfläche zu indexieren und ob man die Indexierung in einen Zeitplan legen kann.

3.2.3 Dokumentation

Im dritten Schritt wird die Dokumentation analysiert. Hierbei wird das Augenmerk auf die Übersichtlichkeit und Verständlichkeit gelegt. Da in diesen Kurztest nicht alle Bereiche der Dokumentation genau durchgelesen und daraufhin auch Testweise implementiert werden können, werden nur die Seiten, die von Relevanz für die obigen Schritte sind bewertet.

3.2.4 Absetzen einer Anfrage und Integration in PHP

Im letzten Schritt werden eine Query in PHP abgesetzt. Dabei wird die Zeit gemessen, wie lange die Query braucht um die Daten zu liefern.

Die dabei verwendete Query ist die bisher am langsamsten laufenden Query der Dietrich-Online Projektes. Sie ermittelt alle Lemmata vom Buchstaben S und baut alle Daten, die zur Anzeige benötigt werden zusammen 3.2. Die Tabellen die für diese Ansicht gebraucht werden, sind in diesem Diagramm 3.1 zu finden. Um genau zu sein, sind es zwei Querys. Die erste findet alle IDs der Lemmata und der zweite baut auf dieser Liste die Daten zusammen.

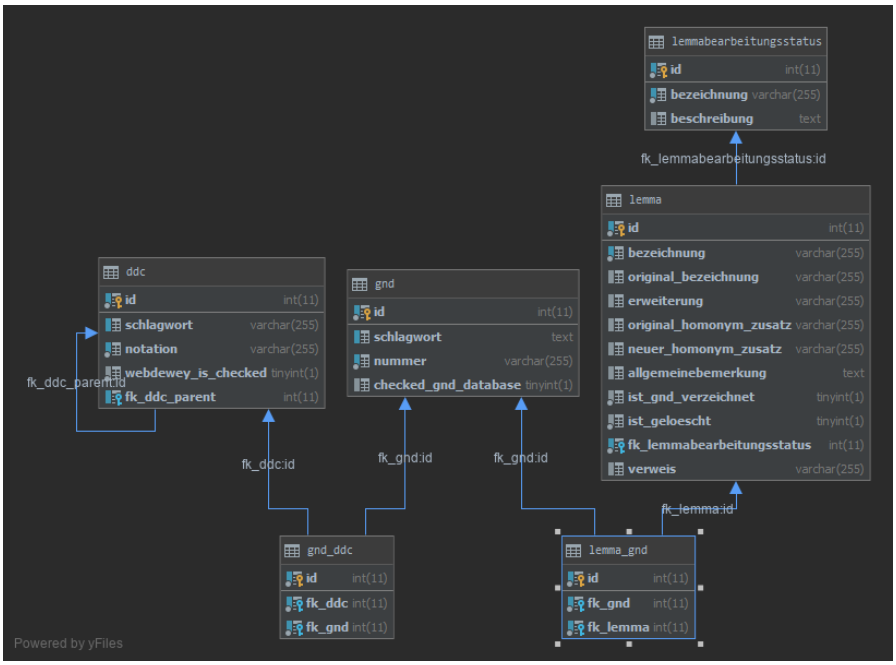


Abb. 3.1. Tabellenaufbau der Lemma-Administration Übersicht.

Abb. 3.2. Frontend Ansicht der Lemma-Administration mit geladenen Buchstaben S (Ausschnitt).

```
1       SELECT
2       lemma.id
3   FROM lemma
4   WHERE
5       lemma.bezeichnung LIKE 'S%'
6       AND lemma.ist_geloescht = 0
7   ORDER BY
8       lemma.bezeichnung ASC,
9       lemma.id ASC;
```

Im zweiten Schritt werden dann die gerade geholten ID's mit vielen JOIN's für die Darstellung vorbereitet.

```
1   SELECT  lemma.id,
2           [...] #Lemma, GND und DCC-Spalten
3   FROM lemma lemma
4
5   INNER JOIN lemmabearbeitungsstatus lemmaBStatus
6   ON lemma.fk_lemmabearbeitungsstatus = lemmaBStatus.id
7
8   LEFT JOIN lemma_gnd lemma_gnd_map ON lemma.id = lemma_gnd_map.fk_lemma
9   LEFT JOIN gnd gnd ON lemma_gnd_map.fk_gnd = gnd.id
10  LEFT JOIN gnd_ddc gnd_ddc_map ON gnd.id = gnd_ddc_map.fk_gnd
11  LEFT JOIN ddc ddc ON gnd_ddc_map.fk_ddc = ddc.id
12  WHERE lemma.id IN ([Array of Lemma IDs])
13  ORDER BY lemma.bezeichnung ASC, lemma.id ASC;
```

3.3 Solr

3.3.1 Installation

Als Systemvoraussetzungen ist eine Java Version > 8 gegeben. Ich habe mich hierbei für OpenJDK 11 entschieden. Nach dem ersten Starten wurden 2 Warnungen gemeldet, dass die User-Limits für Solr zu gering sind 3.3.1. Nachdem diese entsprechend erhöht wurden, verschwanden die Warnungen.

```
1    *** [WARN] *** Your open file limit is currently 1024.
2    It should be set to 65000 to avoid operational disruption.
3
4    *** [WARN] ***  Your Max Processes Limit is currently 63918.
5    It should be set to 65000 to avoid operational disruption.
```

Um Solr im Entwicklermodus auszuführen, kann das entpackte Programm einfach mit `bin/solrstart` gestartet werden. Bei der richtigen Installation installiert sich Solr als Service und legt einen eignen Nutzer an. Ein entsprechendes Installations-Skript findet sich dafür im entpackten Solr-Ordner.

3.3.2 Indexierung

Um mit der Indexierung starten zu können, muss zuerst in sogenannter Core erstellt werden. Dieser ist ein Index mit dazugehörigen Transaktions-Log und den Konfigurationsdateien. Nur mit diesen ist es möglich Dateien zu indexieren und auf ihnen zu suchen. Nach der Erstellung lässt sich der Core nun auch über die Oberfläche einsehen und zum Teil konfigurieren.

Damit Solr nun die Daten von der Datenbank liest, muss ein DataImportHandler (DIH) 3.3.2 geschrieben werden. In diesen werden die Daten, welche Indexiert werden sollen, mit MySQL-Queries eingelesen. Das System setzt dabei auf eine XML-Struktur mit sogenannten Entitys. Diese besitzen jeweils mehre Attribute, wie den Namen, welcher auf der Oberfläche zur Indexierung angezeigt wird, den MySQL-Query mit dem die Daten gelesen werden und einen Delta-Query, welcher dazu dient, nur die neuen Einträge zu laden. Der Delta-Query benötigt hierbei eine eigene Zeitstempel-Spalte in der Datenbank, welche angezeigt, wann die Spalte das letzte Mal editiert wurde. Da die Tabellen im Projekt eine solche Spalte nicht besitzen können die Delta-Querys nicht getestet werden. Innerhalb des Entity Elements gibt es entweder weitere Entitys, dazu gleich mehr, oder Field-Elemente. Diese besitzen ein Attribut, welches die Spalte der Tabelle ausweist und einen Namen, der das zugehörige Solr-Schema-Element ausweist. Entitys können unbegrenzt ineinander verschachtelt werden. Damit Änderungen an einer verschachtelten Entity nach oben richtig weitergegeben werden, gibt es Parent-Delta-Querys. Diese geben die betroffenen Werte an die übergeordnete Entity weiter. Dafür führt der Parent-Delta-Query einen Aufruf an die überliegende Entity-Tabelle aus, in der er mithilfe der Fremdschlüssel-IDs in den betroffen Zeilen herausfindet.

Der DataImportHandler muss, bevor er benutzt werden kann, jedoch noch mit dem Core verbunden werden, dafür wird er, zusammen mit einem JDBC-Treiber in die solrconfig.xml eingetragen. Bei den JDBC-Treiber habe ich mich bei diesem Beispiel für den Treiber von MariaDB entschieden. Damit es nicht deswegen zu Laufzeit-Unterschieden bei der Indexierung kommen kann, werde ich diesen Treiber bei allen Systeme, dies es zulassen, den MariaDB-Treiber verwenden.

```
1      <entity name="lemma"
2        query="select * from lemma"
3        deltaQuery="select eid from lemma
4          where last_modified > '${dataimporter.last_index_time}'">
5        <field column="bezeichnung" name="bezeichnung" />
6      [...]
```

```
7       <entity name="lemma_gnd"
8         query="select * from lemma_gnd where fk_lemma='${lemma.id}'"
9         deltaQuery="select * from lemma_gnd
10          where last_modified > '${dataimporter.last_index_time}'"
11        parentDeltaQuery="select * from lemma
12          where id=${lemma_gnd.fk_lemma}">
13
14          <entity name="gnd"
15            query="select * from gnd where id = '${lemma_gnd.fk_gnd}'"
16            deltaQuery="select * from gnd
17              where last_modified > '${dataimporter.last_index_time}'"
18            parentDeltaQuery="select * from lemma_gnd where fk_gnd=${gnd.id}">
19            <field column="nummer" name="gnd_nummer" />
20            <field column="schlagwort" name="gnd_schlagword" />
21            [...]
22          </entity>
23        </entity>
24      </entity>
```

Wie schon eben angesprochen, muss das Solr-Schema für die entsprechende Elemente auch angepasst werden. Dieses Schema dient dazu die Dateitypen für eine möglichst gute Indexierung auszuweisen. Dafür wird zuerst der Dateityp für die Tabellen-Spalte angegeben. Hierbei werden bei den Grundtypen, zum Beispiel unter anderem String und Text_de gelistet. Ohne genauer darüber nachzudenken, habe ich angenommen, dass beide gleichwertig sind und nur für spätere Abfragen auf Sprachen eine Relevanz besitzen. Als ich allerdings eine Abfrage stellte, die alle Lemmata mit den Buchstaben S finden sollte, bekam ich mehr Ergebnisse als erwartet. Dies liegt daran, dass Text_de, das Feld aus Volltext ausweist. Bei Volltexten wird jedes Wort einzeln betrachet und so kamen Lemma, in welchen irgendein Wort mit S begann in meine Auflistung.

Auch muss darauf geachtet werden, ob der Typ String oder Strings angegeben wird. Letzteres weist das Feld als multiValued aus. Dieser bildet eine 1 zu N Beziehung ab. So kann man zusammen mit der Verschachtelung des DataImport-Handlers auch N zu M Beziehungen abbilden.

Es gibt mehrere Möglichkeiten diese Einträge auszuweisen. In diesem Fall habe ich die Einträge über die Administrations-Oberfläche angelegt. Es ist allerdings auch möglich eine eigene Schema-Datei zu erstellen. Diese Methode soll allerdings nicht mehr verwendet werden, da es die Möglichkeit gibt, diese Einträge per API einlesen zu lassen. Dadurch wird direkt überprüft, ob die Einträge formal stimmen. So können keine fehlerhaften Schemata gebaut werden. Die Einträge, welcher über die API oder die Administrations-Oberfläche gestellt werden, werden in einer Datei namens managed_schema 3.3.5 im XML-Format angelegt.

```
1
2       [...]
3       <field name="ddc_webdewey_is_checked" type="boolean"
4           uninvertible="false" indexed="true" stored="true"/>
5       <field name="description" type="text_de" uninvertible="false"
6           multiValued="true" indexed="true" stored="true"/>
7       <field name="erweiterung" type="text_de"
8           uninvertible="false" indexed="true" stored="true"/>
9       [...]
```

Die Indexierung lief eine Minute und 34 Sekunden für rund 14 Tausend Einträge 3.3. Dabei wurde der gegebene Arbeitsspeicher nicht komplett ausgenutzt, was

darauf schließen lässt, dass die Datenbank der limitierende Faktor war. Die hohe Anzahl der Abfragen ist darauf zurückzuführen, dass Solr keine Joins verwendet, sondern bei jeder verschachtelten Entity die gesamten Tabellen wieder und wieder passenden Einträgen durchsucht.

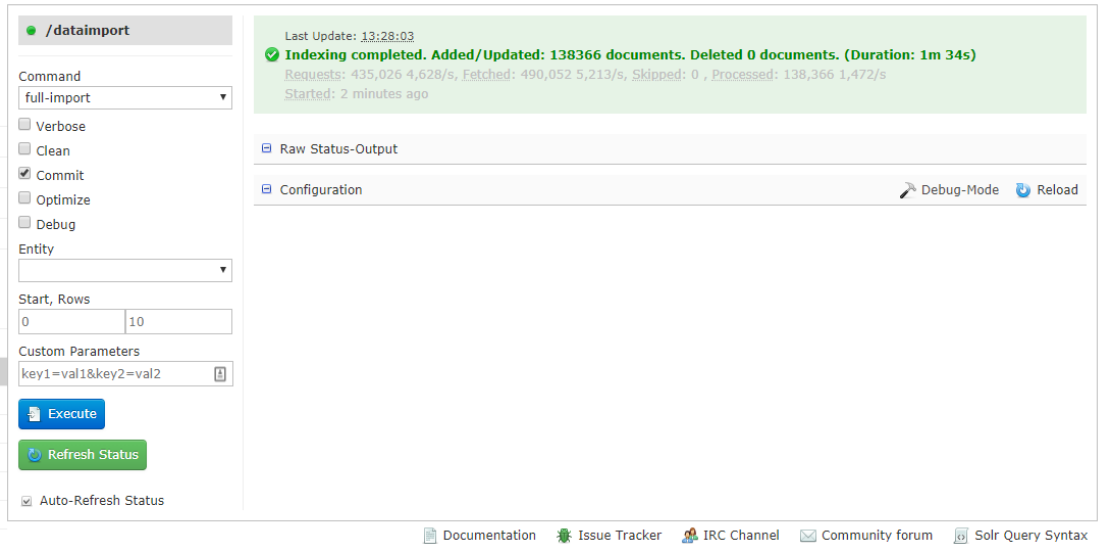


Abb. 3.3. Oberfläche der Indexierung mit Laufzeit.

3.3.3 Oberfläche

Der Startseite des Solr-Systems bietet direkten Einblick in auf die Auslastung des Systems 3.4. Der Fehler-Log ist auch sehr einfach mit einem Klick zu erreichen. Um an die Statistiken von dem aktuellen Core zu kommen, kann dieser aus einen Drop-Down-Menu ausgewählt werden. Positiv anzumerken ist, dass es möglich ist, Schema Einträge direkt in der Weboberfläche zu löschen und anzulegen. Leider ist es jedoch nicht möglich, den DataImportHandler direkt zu verändern, ohne weitere Einstellungen im System vorzunehmen. Es gibt eine Möglichkeit Querys direkt über den Web-Client zu senden, was zum Testen und debuggen sehr nützlich ist. Auch bei der Indexierung kann ein Debug-Modus dazu geschaltet werden 3.3. Zudem besteht die Möglichkeit die Konfigurationsdateien des Cores auf der Weboberfläche einzusehen. Die Dateien dort direkt zu editieren, ist jedoch nicht möglich. Es gibt keine Möglichkeit Updates direkt über die Weboberfläche einzuspielen, zudem ist die Seite auch nicht responsive geschrieben.

3.3.4 Dokumentation

Die Dokumentation war bei diesem kurzen Test meine Hauptquelle. Die Installation ist dort genau beschrieben. Positiv aufgefallen ist mir dabei vor allem die genaue Beschreibung der Systemanforderungen. Es wurde diverse Java-Versionen getestet und dort aufgeführt. Generell gibt es für alle Themen eine kleine Übersichtsseite,

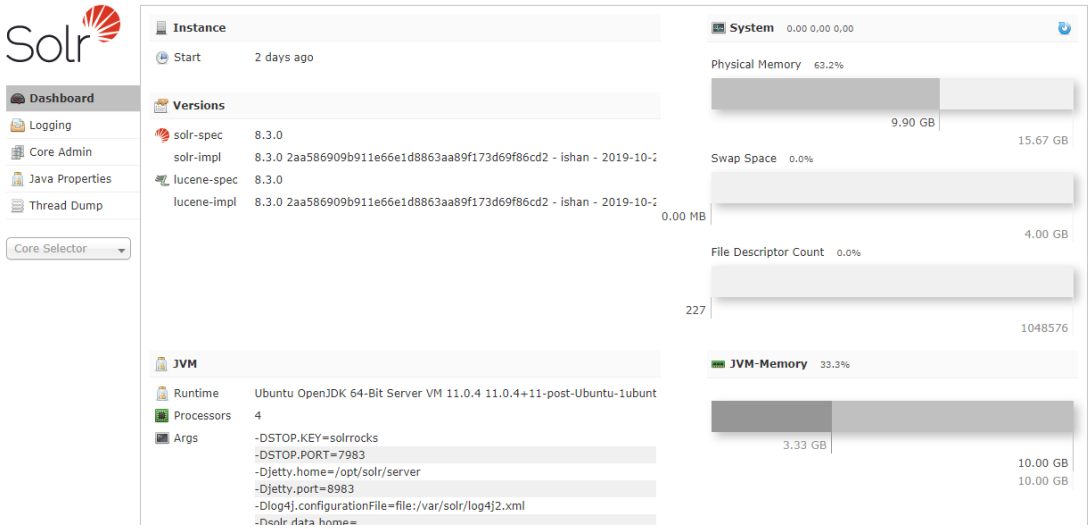


Abb. 3.4. Startseite der Weboberfläche von Solr.

welche die grundlegenden Funktionen erklärt, ohne sich dabei in Details zu verlieren. Die Seite für den DataImportHandler hat anhand eines Beispiels gut die Struktur erklärt. Allerdings wäre ein Verweis, dass für die DataImportHandler-Attribute noch extra ein Solr-Schema-Attribut benötigt wird, schön gewesen. Dies habe ich erst durch einen Blog [19] herausgefunden und richtig verstanden. Die Dokumentation ist, soweit ich das gesehen habe, gut bebildert und bietet einen guten Einstiegspunkt in das System.

3.3.5 Absetzen einer Anfrage und Integration in PHP

Um nicht direkt mit der JSON-API arbeiten zu müssen, gibt es diverse Bibliotheken, die ein wenig der Arbeit abnehmen. Eine der größten ist hierbei Solarium, welches sich mit Composer installieren lässt. Dies passt sehr gut zum Dietrich-Online Projekt, da auch dieses auf Composer setzt. Die Query ist herbei sehr einfach, da die Daten beim Import schon dementsprechend indexiert wurden.

```
1
2     [...] # Imports and variable declarations
3
4     $config = array(
5         'endpoint' => array(
6             'localhost' => array(
7                 'host' => '136.199.34.55',
8                 'port' => 8983,
9                 'core' => 'dietrich'
10    )));
11    $queryText = 'original_bezeichnung:S*';
12    $solr = new Client($config);
13    $query = $solr->createSelect();
14    $query->setQuery($queryText);
15    $query->setRows(2147483647);
16    [...] # Loop with Timer
17    $resultSet = $solr->select($query);
18    $count = $resultSet->count();
19    [...] # Output Runtime
```

Interessant bei der Verbindung ist, dass die Anzahl der Zeilen die von Solr geladen werden, standardmäßig auf 10 limitiert sind. Erst mit setRows kann die Anzahl erhöht werden. Ich für diesen Test den maximalen Integer-Wert gewählt, um immer alle Ergebnisse zu bekommen. Damit ich nun einen guten Median-Wert erhalte habe ich die Abfrage 100 mal laufen lassen. Dabei lief die Abfrage durchschnittlich 1.01 Sekunden um die 15838 Ergebnisse herauszusuchen. Eine Vergleichstabelle, wie sich Solr dabei mit den anderen Suchmaschinen schlägt, finden Sie hier. !!VERGLEICH ANFÜGEN!!

3.4 Datafari

3.4.1 Installation

Für Datafari musste folgende Software nachinstalliert werden: Java 8 und JQ, ein JSON-Prozessor. Damit die Installation richtig durchläuft, muss die JAVA_HOME-Variable erstellt werden. Insofern Datafari nicht unter Root laufen soll, muss noch ein besonderer Nutzer mit Root Rechten angelegt werden. Dieser muss wie schon bei Solr höhere User-Limits erhalten. Datafari installiert sich selbst durch eine DEB-Datei. Während der Installation erscheint ein kurzer Setup-Dialog, welcher einen durch die Konfiguration führt. Das Starten des Server geschieht daraufhin durch ein Script im Installationsordner.

3.4.2 Indexierung

Damit eine Indexierung durchgeführt werden kann, muss bei Datafari ein sogenanntes Repository angelegt werden. In diesem wird die Datenbank-Verbindung eingetragen. Dabei ist es wichtig, dass vorher der Treiber korrekt installiert wird. Es kam bei mir dabei zu Problemen. Das auf Apache ManifoldCF basierende System akzeptiert nur MySQL-JDBC Treiber. Da der MariaDB-Treiber einen anderen Klassennamen in Java verwendet, funktioniert dieser nicht.

> This connection type cannot be configured to work with other databases than the ones listed above without software changes. [20, S. 61]

Deswegen musste ich für diesen Test den MySQL-Treiber von Oracle verwendet. Nachdem der Treiber korrekt installiert ist und das Repository erstellt ist, kann nun einen Job zu Indexierung der Einträge gestartet werden. In diesem werden die Queries und der Zeitplan konfiguriert. Im ersten Schritt wird das Repository ausgewählt und das Ziel, in diesem Fall also Solr. In dem Tab Queries lassen sich dann diverse Querys bauen. Der erste ist der Seeding Query, welche eine Art Delta-Query für dieses System ist und natürlich der Data-Query, welcher die Daten aus der Datenbank lädt. Dabei werden mehrere Variablen definiert, damit der Query korrekt von ManifoldCF erkannt wird. Zuerst einmal das Feld: IDCOLUMN, welches die ID enthält, dann URLCOLUMN, welches einen Hyperlink für diesen Eintrag enthält. Da hier keine solche Spalte gegeben ist, wird einfach nochmal die ID mitgegeben, was so in einen Screenshot in der Dokumentation zu sehen ist.

und als letztes die DATACOLUMN, welche alle Daten konkateniert enthält. Um das System zu testen habe ich allerdings erstmal nur eine Zeile in die DATACOLUMN geschrieben 3.5 Die Konkatenation ist vorgegebene die Methode aus der ManifoldCF-Dokumentation. [20, S. 97] Dies ist für unseren Zweck leider keine gute Datenstruktur. Sind alle Querys eingetragen, kann die Indexierung beginnen. Dafür wird der Job in der Oberfläche manuell gestartet, insofern kein Zeitplan konfiguriert ist. In meinen Test kam es dabei allerdings zu Problemen, die Indexierung erfolgte nicht korrekt und blieb immer am Ende hängen. Der Log zeigte ein "Ready for processingän, machte dort allerdings nicht weiter. Einen Eintrag in der Dokumentation oder generell im Internet konnte ich zu diesem Problem nichts finden. Auch eine Reduktion der Einträge auf nur 125 hat das Problem leider nicht lösen können. Deswegen breche ich an dieser Stelle den Test ab.

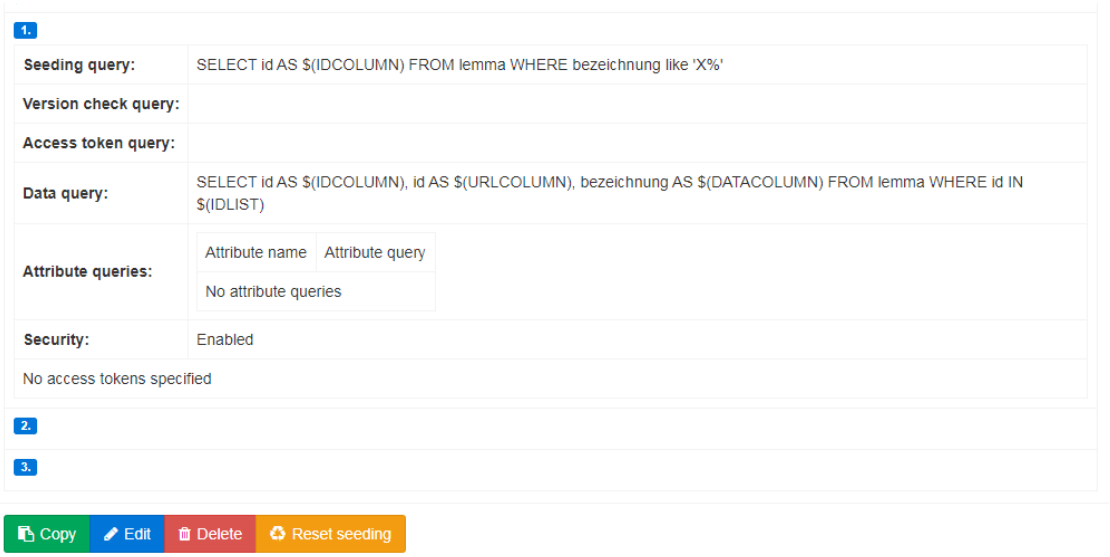


Abb. 3.5. Übersichtsseite des Querys in Datafari.

3.4.3 Oberfläche

Die Oberfläche von Datafari ist dreigeteilt. Zum einen gibt eine Such-Oberfläche, welche sich ohne Anmeldung erreichen lässt. Als Zweites findet sich eine Administrationsoberfläche, welche erst eingesehen werden kann, sobald man eingeloggt ist. Dort findet man diverse Einstellungen für die Suchmaschinen, wie Synonyme oder die Facetten-Konfiguration. Auch sind dort die Logs einzusehen, welche durch die Einbindung der Oberfläche von Kibana 2.5 aus ELK-Stack angezeigt werden. Die dritte Oberfläche ist die Einstellungsseite für die Datacrawler. Dies ist eine modifizierte Oberfläche von Apache ManifoldCF. Generell sind die Menüs sehr übersichtlich, auch wenn die Einbindung von anderen Anwendungen keine Ideale Lösung darstellt. Es lassen sich keine Updates direkt über die Oberfläche einspielen. Die Such-Seite und die Seite für die Erstellung der Datacrawler sind Responsive, während die Administrationsoberfläche bei kleineren Bildschirmgrößen

das Menü versteckt und die Seite somit unnutzbar macht. Update können auch hier nicht über die Oberfläche eingespielt werden.

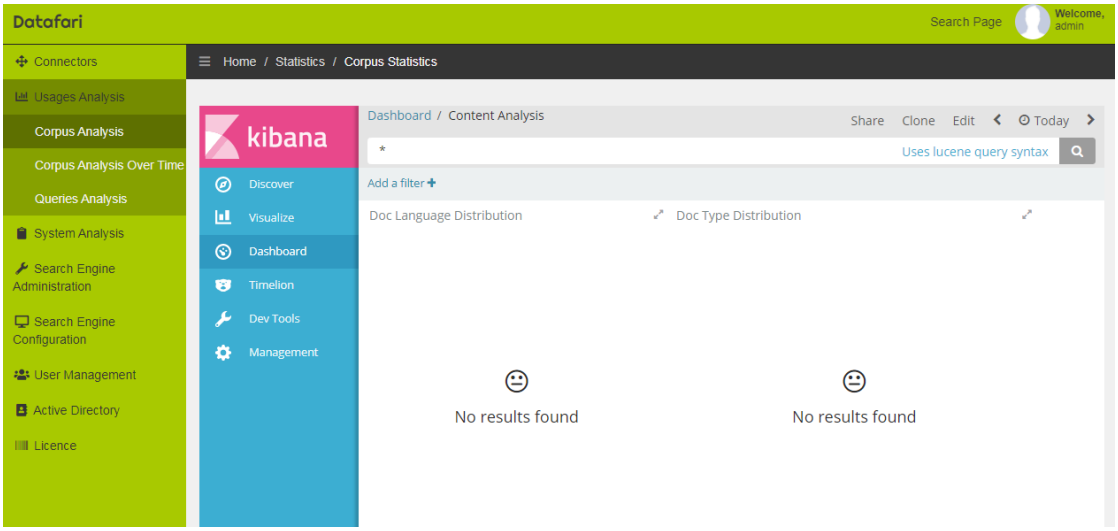


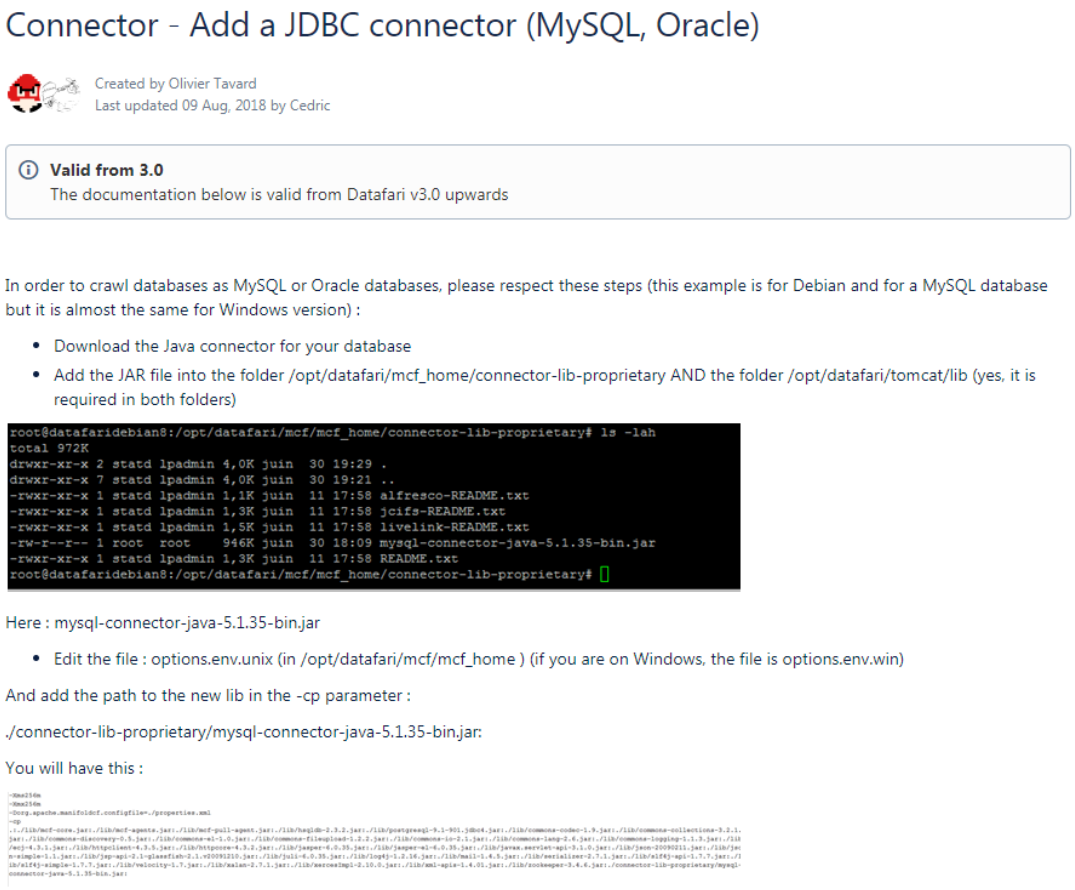
Abb. 3.6. Kibana Integration in Datafari.

3.4.4 Dokumentation

Die Dokumentation geht sehr genau auf die Installation des Systems ein, dabei werden alle Konfigurationsaspekte beleuchtet. Zum Beispiel wird beschrieben, wie die User Limits erhöht werden, oder die JAVA_HOME-Variable korrekt gesetzt wird. Allerdings merkt man an manchen Stellen, dass die Dokumentation nicht von nativen Englischsprechenden geschrieben wurde, da die Grammatik nicht immer stimmt. Allerdings hat dies nie zu Problemen oder Verwechslungen geführt. Bei der Dokumentation zum Einrichten des JDBC-Treibers finden sich einige Probleme 3.7. Zum einen sind beide Pfade, die in dem Text angegeben sind, falsch. Einer davon wird sogar richtig in dem Screenshot direkt darunter angezeigt. Und zum anderen ist der zweite Screenshot so niedrig aufgelöst, dass sich nicht viel erkennen lässt. Dies passiert auch, wenn das Bild in einen neuen Tab geladen wird. Generell ist die Dokumentation für den Umgang mit Datenbanken nicht sehr ausführlich. Die Erklärungen, wofür die Variables bei der Erstellung eines Jobs stehen, musste ich in der Dokumentation von ManifoldCF nachlesen. Die Dokumentation ist im aktuellen Stand nicht sauber strukturiert. Sie gibt das Gefühl, dass es sich eher um eine Sammlung verschiedener Artikel, welche Intern genutzt wurden, handelt.

3.4.5 Absetzen einer Anfrage und Integration in PHP

Durch das fehlgeschlagene Einlesen der Daten konnte dieser Test leider nicht ausgeführt werden.



Connector - Add a JDBC connector (MySQL, Oracle)

Created by Olivier Tavard
Last updated 09 Aug, 2018 by Cedric

Valid from 3.0
The documentation below is valid from Datafari v3.0 upwards

In order to crawl databases as MySQL or Oracle databases, please respect these steps (this example is for Debian and for a MySQL database but it is almost the same for Windows version) :

- Download the Java connector for your database
- Add the JAR file into the folder /opt/datafari/mcf_home/connector-lib-proprietary AND the folder /opt/datafari/tomcat/lib (yes, it is required in both folders)

```
root@datafaridebian8:/opt/datafari/mcf/mcf_home/connector-lib-proprietary# ls -lah
total 972K
drwxr-xr-x 2 statd lpadmin 4,0K juin  30 19:29 .
drwxr-xr-x 7 statd lpadmin 4,0K juin  30 19:21 ..
-rwxr-xr-x 1 statd lpadmin 1,1K juin  11 17:58 alfresco-README.txt
-rwxr-xr-x 1 statd lpadmin 1,3K juin  11 17:58 jcifs-README.txt
-rwxr-xr-x 1 statd lpadmin 1,5K juin  11 17:58 livelink-README.txt
-rw-r--r-- 1 root  root    946K juin  30 18:09 mysql-connector-java-5.1.35-bin.jar
-rwxr-xr-x 1 statd lpadmin 1,3K juin  11 17:58 README.txt
root@datafaridebian8:/opt/datafari/mcf/mcf_home/connector-lib-proprietary#
```

Here : mysql-connector-java-5.1.35-bin.jar

- Edit the file : options.env.unix (in /opt/datafari/mcf/mcf_home) (if you are on Windows, the file is options.env.win)

And add the path to the new lib in the -cp parameter :

./connector-lib-proprietary/mysql-connector-java-5.1.35-bin.jar:

You will have this :

Abb. 3.7. Dokumentationsseite für den JDBC Treibers von Datafari.

3.5 ElasticSearch

3.5.1 Installation

Die Installation ist bei ElasticSearch dreigeteilt. Um ElasticSearch in dem Umfang nutzen zu können, wie es hier gewünscht ist, muss zum einen ElasticSearch, Kibana als auch Logstash installiert werden. ElasticSearch ist hierbei das Kernstück und dient als Datenbank. Kibana ist eine grafische Benutzeroberfläche für ElasticSearch und Logstash stellt die Brücken zwischen der MySQL-Datenbank und ElasticSearch dar. Während ElasticSearch Java mitgeliefert hat, muss für Logstash Java Version 8 oder 11 nachinstalliert werden. Um die drei Dienste für den Development Modus zu installieren, mussten nur die Archive entpackt und die entsprechenden Anwendungen gestartet werden. Ohne die Konfigurationsdateien zu ändern, konnten die Anwendungen direkt miteinander kommunizieren.

!!Richtige Installtion!!

3.5.2 Indexierung

Um nun Daten zu indexieren, muss in einer Conf-Datei in Logstash definiert werden, wie und welche Daten gelesen und weitergegeben werden sollten 3.5.2. Die

Datei kann direkt Querys gegen die Datenbank stellen. Dabei trat zu Beginn allerdings ein Fehler auf, welcher nur damit behoben werden konnte, dass der MariaDB-Treiber direkt im Kern von Logstash mitgeladen wird. Deswegen ist der Pfad zur Treiber-Bibliothek in der Datei auch leer. In den Block Output definiert man nun das Ziel. !!TODO FURTHER!!

literateliterate

```
1   input {
2     jdbc {
3       jdbc_validate_connection => true
4       jdbc_driver_library => ""
5       jdbc_driver_class => "Java::org.mariadb.jdbc.Driver"
6       jdbc_connection_string =>
7           "jdbc:mariadb://localhost:3306/dietrichonline"
8       jdbc_user => "USER"
9       jdbc_password => "PW"
10      statement => "MYSQL-Query"
11      schedule => "0 */6 * * *"
12    }
13  }
14
15  output {
16    stdout { codec => json_lines }
17    elasticsearch {
18      document_id => "%{id}"
19      document_type => "lemma"
20      index => "lemma"
21      hosts => "localhost:9200"
22    }
23  }
```

3.5.3 Oberfläche

Es gibt einen Update Knopf!

3.5.4 Dokumentation

3.5.5 Absetzen einer Anfrage und Integration in PHP

4

Nutzung des Open Archives Initiative Protokolls für Metadaten

Das Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) ist ein Protokoll zum Austausch von Metadaten. Dabei werden Anfragen per GET oder POST-Request angefragt. Als Antwort erhält man im Folgenden ein XML-Dokument. So können die Metadaten mit bestimmten Facetten abgefragt werden (zum Beispiel Autor). Dabei geht es allerdings darum primär darum Änderungen weiterzugeben. So können durch dieses Protokoll neue Einträge oder Änderungen in der Datenbank weitergeben werden. [21]

4.1 OAI Harverester

Ein OAI Harverester ist ein Programm, welches durchgehend einen Abgleich der Daten vollführt. Dabei lässt es sich die Änderungen mit einem List-Befehl von dem Server geben und gleicht diese danach mit der eigenen Struktur ab. Sollten dabei Unterschiede festgestellt werden, werden daraufhin die Änderungen auch beim Harverester eingefügt. So steht der Harverester immer mit dem Server auf einen Stand. [21]

4.2 Support der Enterprise Search Engines

Bei den vorhin genannten Enterprise Search Engines gibt es keine mit nativen OAI Harverester Support. Es gibt die Möglichkeit für manche der Suchmaschinen ein solches Verhalten mithilfe von Plugins zu implementieren. Allerdings sind die meisten dieser Add-ons auch schon veraltet.

4.3 Auswertung

Durch eine fehlende Basisimplementierung des Protokolls in den einzelnen Suchmaschinen und der Möglichkeit eines direkten Zugriffs auf die Datenbank, sehe ich keinen Grund dieses Protokoll zu verwenden. Es müsste ein Server vor die Datenbank installiert werden und ein Harverester vor der ESE. Dies ist ein großer Mehraufwand, welcher bei diesem Anwendungsfall nicht notwendig ist. Sollte allerdings

diese Suchmaschine ein übergreifendes System werden, kann darüber nachgedacht werden, die anderen Datenbanken per OAI-Harverester anzusprechen.

5

Zusammenfassung und Ausblick

In diesem Kapitel soll die Arbeit noch einmal kurz zusammengefasst werden. Insbesondere sollen die wesentlichen Ergebnisse Ihrer Arbeit herausgehoben werden. Erfahrungen, die zum Beispiel Benutzer mit der Mensch-Maschine-Schnittstelle gemacht haben oder Ergebnisse von Leistungsmessungen sollen an dieser Stelle präsentiert werden. Sie können in diesem Kapitel auch die Ergebnisse oder das Arbeitsumfeld Ihrer Arbeit kritisch bewerten. Wünschenswerte Erweiterungen sollen als Hinweise auf weiterführende Arbeiten erwähnt werden.

Literaturverzeichnis

1. “Apache lucene - wikipedia,” 27.10.2019. [Online]. Available: https://en.wikipedia.org/w/index.php?oldid=915250662
2. “Apache lucene - apache lucene core,” 26.07.2019. [Online]. Available: https://lucene.apache.org/core/
3. R. McCreadie and Craig Macdonald and Jie Peng, “Terrier ir platform - homepage,” 25.01.2019. [Online]. Available: http://terrier.org/
4. “Sphinx — open source search server.” [Online]. Available: http://sphinxsearch.com/docs/manual-2.3.2.html#intro
5. “Sphinx search server.” [Online]. Available: https://github.com/sphinxsearch/sphinx
6. “Sphinx — open source search engine.” [Online]. Available: http://sphinxsearch.com/
7. “Apache solr - wikipedia,” 14.10.2019. [Online]. Available: https://en.wikipedia.org/w/index.php?oldid=915250761
8. “Apache solr,” 26.07.2019. [Online]. Available: https://lucene.apache.org/solr/
9. “Elasticsearch: Verteilte restful-suchmaschine und -analytics engine — elastic.” [Online]. Available: https://www.elastic.co/de/products/elasticsearch
10. “Professional support — n2sm, inc.” [Online]. Available: https://www.n2sm.net/en/support/fess_support.html
11. “Fess installation guide,” 31.10.2019. [Online]. Available: https://fess.codelibs.org/13.4/install/index.html
12. “Fast, reliable and modern search and discovery.” [Online]. Available: https://www.algolia.com/
13. “Manticore search – open source text search engine for big data and stream filtering.” [Online]. Available: https://manticoresearch.com/
14. “The xapian project,” 14.10.2019. [Online]. Available: https://xapian.org/
15. F. Labs, “About << france labs: Open source enterprise search,” 2018. [Online]. Available: https://www.francelabs.com/en/about.html
16. Michael Brandenburg, “Suchtrupp: Eine eigene suchmaschine bauen (teil 1),” *LINUX-Magazin: Die Zeitschrift für LINUX-Professionals*, no. 11, pp. 62–68, 2019. [Online]. Available: https://www.linux-magazin.de/ausgaben/2019/11/datafari/

17. F. Labs, “Datafari enterprise search.” [Online]. Available: https://www.datafari.com/en/index.html
18. “Xapian users,” 14.10.2019. [Online]. Available: https://xapian.org/users
19. Iqubal Mustafa Kaki, “Solr indexing - mariadb table data into apache solr,” 2016. [Online]. Available: https://erimkaki.blogspot.com/2016/01/solr-indexing-mariadb-table-data.html
20. Apache Software Foundation, “Manifoldcf- end-user documentation.” [Online]. Available: https://manifoldcf.apache.org/release/release-2.14/en_US/end-user-documentation.pdf
21. “Oai-schnittstelle,” 31.05.2019. [Online]. Available: https://www.dnb.de/DE/Professionell/Metadatendienste/Datenbezug/OAI/oai_node.html

A

Glossar

ESE	Enterprise Search Engine
Facetten	Filter in Bibliothekarssprache
OAI	Open Archives Initiative

B

Erklärung der Kandidatin / des Kandidaten

☐ Die Arbeit habe ich selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Datum	Unterschrift der Kandidatin / des Kandidaten