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

Abbildungsverzeichnis

Tabellenverzeichnis

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

Weitere Kapitel

Die Gliederung hängt natürlich vom Thema und von der Lösungsstrategie ab. Als nützliche Anhaltspunkte können die Entwicklungsstufen oder -schritte zum Beispiel der Softwareentwicklung betrachtet werden. Nützliche Gesichtspunkte erhält und erkennt man, wenn man sich

- in die Rolle des Lesers oder
- in die Rolle des Entwicklers, der die Arbeit zum Beispiel fortsetzen, ergänzen oder pflegen soll,

versetzt. In der Regel wird vorausgesetzt, dass die Leser einen fachlichen Hintergrund haben, zum Beispiel Informatik studiert haben. Das heißt nur in besonderen, abgesprochenen Fällen schreibt man in populärer Sprache, sodass auch Nicht-Fachleute die Ausarbeitung prinzipiell lesen und verstehen können.

Die äußere Gestaltung der Ausarbeitung hinsichtlich Abschnittformate, Abbildungen, mathematische Formeln usw. wird in Kapitel 3 kurz dargestellt.

3

LaTeX-Bausteine

Der Text wird in bis zu drei Ebenen gegliedert:

1. Kapitel (`chapter{Kapitel} ),` indexKapitel
2. Unterkapitel (`\section{Abschnitt}`) und
3. Unterunterkapitel (`\subsection{Unterabschnitte}`).

3.1 Abschnitt

Text der Gliederungsebene 2.

3.1.1 Unterabschnitt

Text der Gliederungsebene 3. Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Text Beispiel für Quelltext

```
1     Prozess 1:
2
3     Acquire();
4        a := 1;
5     Release();
6     ...
7     Acquire();
8     if(b == 0)
9     {
10       c := 3;
11       d := a;
12    }
13    Release();
```

```
1       Prozess 2:
2
3       Acquire();
4           b := 1;
5       Release();
6       ...
7       Acquire();
8       if(a == 0)
9       {
10          c := 5;
11          d := b;
12      }
13      Release();
```

Größere Code-Fragmente sollten im Anhang eingefügt werden.

3.2 Abbildungen und Tabellen

Abbildung und Tabellen werden zentriert eingefügt. Grundsätzlich sollen sie erst dann erscheinen, nach dem sie im Text angesprochen wurden (siehe Abb. 3.1). Abbildungen und Tabellen (siehe Tabelle 3.1) können im (fließenden) Text (`here`), am Seitenanfang (`top`), am Seitenende (`bottom`) oder auch gesammelt auf einer nachfolgenden Seite (`page`) oder auch ganz am Ende der Ausarbeitung erscheinen. Letzteres sollte man nur dann wählen, wenn die Bilder günstig zusammen zu betrachten sind und die Ausarbeitung nicht zu lang ist (< 20 Seiten).

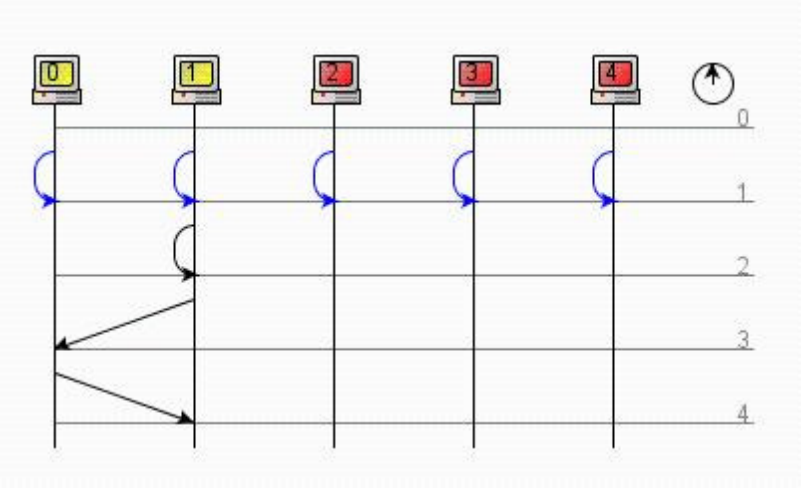

Abb. 3.1. Bezeichnung der Abbildung

3.3 Mathematische Formel

Mathematische Formeln bzw. Formulierungen können sowohl im laufenden Text (z.B. $y = x^2$) oder abgesetzt und zentriert im Text erscheinen. Gleichungen sollten für Referenzierungen nummeriert werden (siehe Formel 3.1).

Prozesse	**Zeit** $\rightarrow$			
P_1	$W(x)1$			
P_2		$W(x)2$		
P_3		$R(x)2$		$R(x)1$
P_4			$R(x)2$	$R(x)1$

Tabelle 3.1. Bezeichnung der Tabelle

$$e_i = \sum_{i=1}^{n} w_i x_i \tag{3.1}$$

Entscheidungsformel:

$$\psi(t) = \begin{cases} 1 & 0 <= t < \frac{1}{2} \\ -1 & \frac{1}{2} <= t < 1 \\ 0 & sonst \end{cases} \tag{3.2}$$

Matrix:

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \tag{3.3}$$

Vektor:

$$\overline{a} = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} \tag{3.4}$$

3.4 Sätze, Lemmas und Definitionen

Sätze, Lemmas, Definitionen, Beweise, Beispiele können in speziell dafür vorgesehenen Umgebungen erstellt werden.

Definition 3.1. *(Optimierungsproblem)*
Ein Optimierungsproblem $\mathcal{P}$ *ist festgelegt durch ein Tupel* $(I_\mathcal{P}, sol_\mathcal{P}, m_\mathcal{P}, goal)$ *wobei gilt*

1. $I_\mathcal{P}$ *ist die Menge der Instanzen,*
2. $sol_\mathcal{P} : I_\mathcal{P} \longmapsto \mathbb{P}(S_\mathcal{P})$ *ist eine Funktion, die jeder Instanz* $x \in I_\mathcal{P}$ *eine Menge zulässiger Lösungen zuweist,*
3. $m_\mathcal{P} : I_\mathcal{P} \times S_\mathcal{P} \longmapsto \mathbb{N}$ *ist eine Funktion, die jedem Paar* $(x, y(x))$ *mit* $x \in I_\mathcal{P}$ *und* $y(x) \in sol_\mathcal{P}(x)$ *eine Zahl* $m_\mathcal{P}(x, y(x)) \in \mathbb{N}$ *zuordnet (= Maß für die Lösung* $y(x)$ *der Instanz* x*), und*
4. $goal \in \{min, max\}$.

Beispiel 3.2. MINIMUM TRAVELING SALESMAN (MIN-TSP)

- $I_{MIN-TSP} =_{def}$ s.o., ebenso $S_{MIN-TSP}$
- $sol_{MIN-TSP}(m, D) =_{def} S_{MIN-TSP} \cap \mathbb{N}^m$
- $m_{MIN-TSP}((m, D), (c_1, \ldots, c_m)) =_{def} \sum_{i=1}^{m-1} D(c_i, c_{i+1}) + D(c_m, c_1)$
- $goal_{MIN-TSP} =_{def} min$

□

Satz 3.3. *Sei $\mathcal{P}$ ein **NP**-hartes Optimierungsproblem. Wenn $\mathcal{P} \in$ **PO**, dann ist **P = NP**.*

Beweis. Um zu zeigen, dass **P** = **NP** gilt, genügt es wegen Satz A.30 zu zeigen, dass ein einziges **NP**-vollständiges Problem in **P** liegt. Sei also $\mathcal{P}'$ ein beliebiges **NP**-vollständiges Problem.

Weil $\mathcal{P}$ nach Voraussetzung **NP**-hart ist, gilt insbesondere $\mathcal{P}' \leq_T \mathcal{P}_C$. Sei R der zugehörige Polynomialzeit-Algorithmus dieser Turing-Reduktion. Weiter ist $\mathcal{P} \in$ **PO** vorausgesetzt, etwa vermöge eines Polynomialzeit-Algorithmus A. Aus den beiden Polynomialzeit-Algorithmen R und A erhält man nun leicht einen effizienten Algorithmus für $\mathcal{P}'$: Ersetzt man in R das Orakel durch A, ergibt dies insgesamt eine polynomielle Laufzeit. □

Lemma 3.4. *Aus **PO** = **NPO** folgt **P** = **NP**.*

Beweis. Es genügt zu zeigen, dass unter der angegeben Voraussetzung KNAPSACK ∈ **P** ist.

Nach Voraussetung ist MAXIMUM KNAPSACK ∈ **PO**, d.h. die Berechnung von $m^*(x)$ für jede Instanz x ist in Polynomialzeit möglich. Um KNAPSACK bei Eingabe (x, k) zu entscheiden, müssen wir nur noch $m^*(x) \geq k$ prüfen. Ist das der Fall, geben wir 1, sonst 0 aus. Dies bleibt insgesamt ein Polynomialzeit-Algorithmus.

□

3.5 Fußnoten

In einer Fußnote können ergänzende Informationen[1] angegeben werden. Außerdem kann eine Fußnote auch Links enthalten. Wird in der Arbeit eine Software (zum Beispiel Java-API[2]) eingesetzt, so kann die Quelle, die diese Software zur Verfügung stellt in der Fußnote angegeben werden.

3.6 Literaturverweise

Alle benutzte Literatur wird im Literaturverzeichnis angegeben[3]. Alle angegebene Literatur sollte mindestens einmal im Text referenziert werden.

[1] Informationen die für die Arbeit zweitrangig sind, jedoch für den Leser interessant sein könnten.
[2] `http://java.sun.com/`
[3] Dazu wird ein sogenannter bib-File, literatur.bib verwendet.

4

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.

5

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 diesen Vergleich nach Features am besten abschneiden werden anschließend aufgesetzt, getestet und genauer verglichen.

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

Des Weiteren sind die folgenden Funktionen auch wichtig, allerdings keine K.O. Kriterien:

- 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 Open Source Suchmaschine zu finden. Auch äußerst wichtig ist der Support für Facetten, da viele Dietrich-Online als Suchmaschine den Nutzer einige Tools zum Verfeinern seiner Suchergebnisse zur Verfügung stellen will. Das Ranking der Suchergebnisse ist vor allem für die Transparenz wichtig, da hier mit vielen Daten gearbeitet wird, welche allerdings nicht alle gleichzeitig dargestellt werden können. Dieses Problem kann mit einer Gewichtung der Suchergebnisse behoben werden, welches dann auf der Seite für Transparenz veröffentlicht wird. Die Volltextsuche wird es möglich machen auch nach Schlüsselwörtern im Titel zu suchen. Der Support von den 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. Dadurch gibt sich auch das Problem, dass Daten mehrfach vorliegen welches sicherlich Später von größerer Bedeutung sein wird. Als letztes ist es noch wichtig, dass eine Logging-Möglichkeit geboten wird, damit schnell und effizient Probleme mit dem System erkannt und gelöst werden können.

Ein Support für PostgreSQL ist für dieses Projekt nicht so wichtig, allerdings könnte es sein, dass der Server später auch andere Datenbanken verwaltet. Der Server wird sowieso täglich durch einen Automatismus gesichert. Allerdings ist eine manuelle Backup-Lösung wünschenswert, um die Suchmaschine losgelöst von den Server zu sichern und gegebenenfalls auch einfach auf einen anderen Server umzuziehen. Auto-Korrektur und Auto-Vervollständigung sind beide sehr interessant, um den Nutzer mehr Komfort-Funktionen bieten zu können. Der Server sollte ja generell nur intern Ansprechbar sein. Allerdings gibt es für manche der Suchmaschinen eine Web-Oberfläche, da wäre es wichtig eine sichere Verbindung dorthin und gegebenenfalls eine Möglichkeit der Mehrfaktor-Authentifizierung zu haben. Einen PHP-Connector, welcher Objekte zum Umgang mit der Suchmaschine bietet, wäre wünschenswert. Allerdings bieten einige Suchmaschinen auch die Möglichkeit über JSON Anfragen an die Suchmaschine zu stellen. Allerdings sollte zumindest eine der beiden Möglichkeiten gegeben sein. Zuletzt wäre für akute Probleme ein bezahlter Support, der dafür schneller reagiert wünschenswert.

5.1 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. [Wik19b]

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. [The19a]

5.2 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 hat eine breite Nutzerbasis laut Webseite. 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 ist kein Support für Facetten gegeben.

[MC19]

TERRIER HAT LOGGING KEIN SUPPORT FÜR SQL -¿ SQLXML??

5.3 Sphinx

Sphinx ist eine Suchmaschine entwickelt von Andrew Aksyonoff. 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.“ [sphb], 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 gibt. 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. Da es auch für Sphinx kein Frontend gibt, ist eine Authentifizierung nicht notwendig. [Spha]

5.4 Apache Solr

Apache Solr ist eine, auf Lucene Core 5.1 viel eingesetzte Search Engine von der Apache Foundation. Sie basiert auf Apache Lucene Core und erweitert dieses um ein grafische Benutzeroberfläche und einige Features. 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 wurden Solr dann in das Apache Lucene Projekt eingefügt. Dort wird es auch aktuell noch weiterentwickelt. [Wik19a]

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 und hat zudem auch noch Support für Plugins. Es erfüllt alle Grundanforderungen 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 Liste oder IRC erreichen kann.

[The19b]

5.5 ElasticSearch

Eine weitere großes 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. Alle kamen zusammen in der 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 kostenlos und ElasticSearch zudem auch als Open-Source Variante zu haben. Für den Vergleich wird hier nun die Kostenfreie und nicht die Open-Source Variante beleuchtet. Eine genauere Aussage, welche Features nur in der kostenlosen und nicht in der Open-Source Variante zu finden sind, finden sich in der Tabelle 5.1.

// HIER WEITERMACHEN!!

Auch ElasticSearch stellt eine REST-API bereit, allerdings gibt es auch Klienten für viele Programmiersprachen, darunter PHP. Dabei kann auch eine SQL-like Sprache verwendet werden, insofern auf Open-Source verzichtet wird und der Basic Plan genommen wird. [Elab]

Sie bietet den Vorteil, dass Support dazugebucht werden kann. Für eine grafische Oberfläche gibt es Kibana. Dies bietet nicht nur eine Oberfläche für Elastic Search, sondern lässt sich auch mit den Logstash einer Logging-Engine verbinden. Da aktuell auch noch nach einer Logging Lösung gesucht wird, ist dies ein nicht zu unterschätzender Pluspunkt.

[Elaa]

5.6 Manticore Search

Manticore Search ist eine Open-Source Option ESE, welche unter anderem Craigslist zu seinen Kunden zählen kann. Sie basiert auf Sphinx, spaltet sich jedoch immer weiter davon ab, da Sphinx vom Open Source Modell weg gegangen ist.

Sie bietet auch eine Auto-Korrektur und Support für Suchvorschläge. Auch wird SQL und JSON zur Anfragenstellung unterstützt.

[Man]

5.7 Xapian

Xapian ist eine Open-Source ESE, welche unter der GPL Lizenz verfügbar ist. Sie basiert auf Open Muscat. Eine Search Engine mit Wurzeln an der Cambridge University in den 1980ern. Nachdem Muscat umbenannt wurde und closed Source gegangen ist, haben einige Entwickler die letzte öffentlich zugängliche Version übernommen und weiterentwickelt. Sie bietet Support für Facetten, Wildcard Suche und Ranked Suche. Auto-Korrektur ist auch verfügbar.

[XAP19]

5.8 Vorauswahl

	LC	**AS**	**MC**	**ES**	**XP**
Open Source	x	x	x	x*	x
Unterstützung von Facetten	x	x	x	x	x
Ranking der Suchergebnisse	x	x	x	x	x
statistische Auswertung (Metriken)	-	x	x	x	-
Volltextsuche	x	x	x	x	x
Support für PDF, SQL, XML	x	x	x	x	x
Monitoring / Logging	-	x	x	x*	-
Support für PostgreSQL	x	x	x	x	x
Backup	-	x	-	x*	-
Auto-Korrektur	x	x	x	x	x
Auto-Vervollständigung	x	x	x	x	x
Login System und Security	-	x	-	x*	-
bezahlter Support	-	-	x	x	-
Wildcard Suche	x	x	x	x	x
Query Vorschläge	x	x	x	x	x
Synonym Support	x	x	x	x	x
PHP Support	-	x	-	x	x
(Fast) Echtzeit Indizierung	x	x	x	x	x
Web-Interface	-	x	-	x	-
Plugin Support	x	x	x	x	-
JSON Interface	x	x	x	x	-
SQL-Like Query Support	x	-	x	x	-

Tabelle 5.1. Feature-Vergleich der verschiedenen Enterprise Search Engines

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

- LC = Lucene Core
- AS = Apache Solr
- MC = Manticore Search
- ES = Elastic Search
- XP = Xapian

Der Stern bedeutet, dass ein Feature nur in der Kostenlosen und nicht der Open Source Variante verfügbar ist.

Nach einem ersten Überblick wurden nun Aufgrund der Auswahlkriterien diese Systeme zum genaueren Vergleich ausgewählt: Apache Solr, Manticore Search, ElasticSearch (in der kostenlosen Version) und Xapian. Lucene Core wird nicht genauer untersucht, da Solr ein umfassenderes Paket bietet, welches den gestellten Anforderungen mehr entspricht.

6

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. [Deu]

6.1 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. [Deu]

6.2 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.

6.3 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 einzubauen. Es müsste ein Server vor die Datenbank installiert werden und ein Harverester vor der ESE. Dies ist ein großer Mehraufwand. Sollte allerdings die ESE ein übergreifendes System werden, kann darüber nachgedacht werden, die anderen Datenbanken per Harverester anzusprechen.

Literaturverzeichnis

Deu. *OAI-Schnittstelle*.

Elaa. *Elasticsearch: Verteilte RESTful-Suchmaschine und -Analytics Engine — Elastic*.

Elab. *Subscriptions · Elastic Stack Products & Support — Elastic*.

Man. *Manticore Search – open source text search engine for big data and stream filtering*.

MC19. McCreadie, Richard und Craig Macdonald and Jie Peng: *Terrier IR Platform - Homepage*, 25.01.2019.

Spha. *Sphinx — Open Source Search Engine*.

sphb. *Sphinx search server*.

The19a. *Apache Lucene - Apache Lucene Core*, 26.07.2019.

The19b. *Apache Solr*, 26.07.2019.

Wik19a. *Apache Solr - Wikipedia*, 14.10.2019.

Wik19b. *Apache Lucene - Wikipedia*, 27.10.2019.

XAP19. *The Xapian Project*, 14.10.2019.

Sachverzeichnis

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