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).

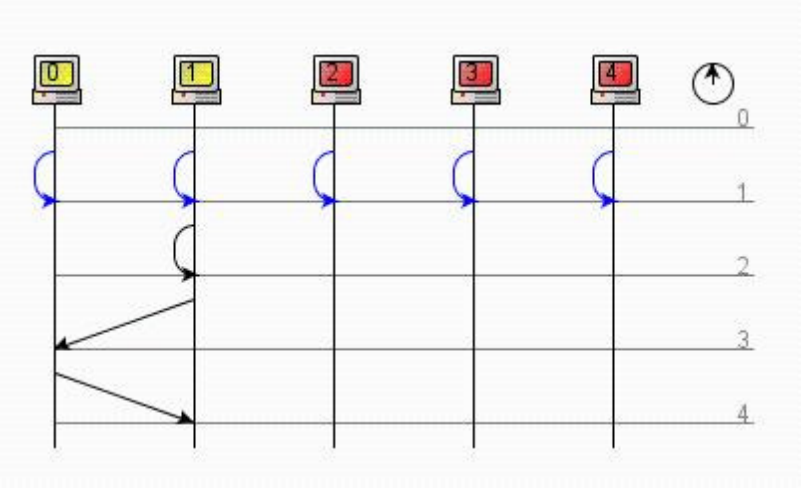

Abb. 3.1. Bezeichnung der Abbildung

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 diesem ersten Schritt werden sich diverse Enterprise Search Engine Systeme angeschaut und evaluiert. Dafür wurde eine Liste mit Anforderungen an das System erstellt. Die Anforderungen sehen wie folgt aus:

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

Des Weiteren sind folgende Features ein Bonus:

- Support für PostgreSQL
- Backup
- statistische Auswertung
- Auto-Korrektur und Auto-Vervollständigung
- Login System mit Security
- bezahlter Support

5.1 Apache Lucene Core

Lucene Core ist eine Open Source Enterprise Search Engine von der Apache Foundation geschrieben in Java. Sie bildet die Basis für Solr, auf welches später noch eingegangen wird. Dabei bietet Lucene Core allerdings auch schon einige interessante Features, unter anderem eine Ranked Suche und Facettierung. 5.1

Das Lucene Projekt wurde im Jahre 1999 vom Entwicker Doug Cutting gestartet. Es ist im September 2001 der Apache Foundation beigetreten, als ein der Jakarta Familie. 2005 wurde es dann zu einen eigenen Top-Level Projekt.

[The19a]

5.2 Terrier

5.3 Sphinx

5.4 Apache Solr

Apache Solr ist eine viel eingesetzte Search Engine von der Apache Foundation. Sie basiert auf Apache Lucene Core und erweitert dieses um ein Interface. Die Entwicklung dafür begann 2004 als ein internes Projekt von CNET. Im Jahre 2006 hat CNET den Source Code an die Apache Foundation weitergegeben und es wurde zu einem großen Part des Lucene Projektes. Es wird unter anderem von DuckDuckGo und Best Buy eingesetzt. Durch die Apache Foundation ist ein stabiler Langzeitpartner gegeben. Solr arbeitet auf einer REST-like API, welches es ermöglicht eine einfache Schnittstelle für das Dietrich Projekt bereitzustellen. Darüber hinaus bietet es Support für Facetten. Als Bonus hat es noch Support für Suchvorschläge, was für das Dietrich Projekt auf von Interesse wäre.

Apache Solr hat keinen bezahlten Support, sondern einen kostenfreien Community Support.

[The19b]

5.5 ElasticSearch

Eine weitere großes Enterprise Search Engine ist ElasticSearch. Auch dieses Projekt arbeitet auf der Basis von Lucene. Zwei der bekannten Kunden sind Adobe und Ebay. 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 Engine ist eine Open-Source Solution basierend auf Sphinx 5.3. 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 Grund Anforderungen, 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. Es gibt auch einen Query-Log. 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.

[Man]

5.7 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 geladen und diese weiterentwickelt.

Sie erfüllt alle der Grundanforderungen, wenn auch Logging nur im Grundsinne erfüllt wird, da nur Errors geschmissen 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.

[XAP19]

5.8 Vorauswahl

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.

	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	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.

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*.

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

The19b. *Apache Solr*, 26.07.2019.

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