Informatik
Hauptcampus

HOCHSCHULE TRIER

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: Titel Vorname Name

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).

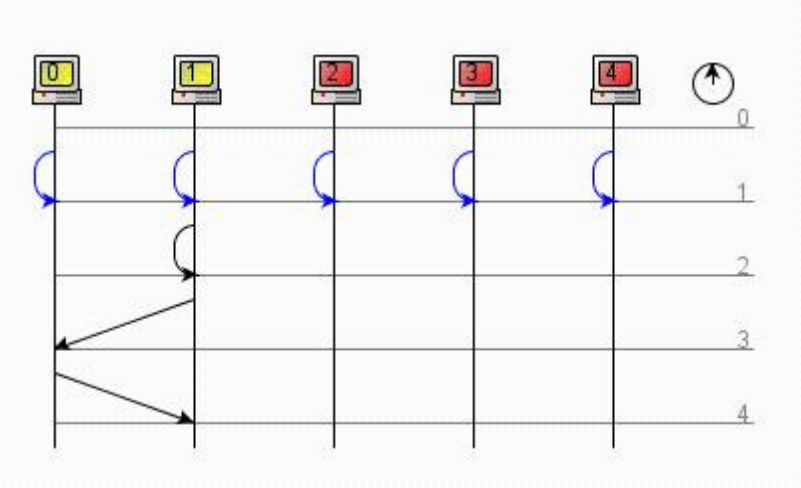

Abb. 3.1. Bezeichnung der Abbildung

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

Das System muss Open-Source und kostenlos sein. Es müssen Facetten unterstützt werden, um eine einfache Möglichkeit zu bieten die Ergebnisse zu Filtern.

5.1 Lucene Core

5.2 Solr

Apache Solr ist eine viel eingesetzte Search Engine von der Apache Foundation. Sie basiert auf Apache Lucene und erweitert dieses um ein einfacheres Interface. 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.

[The19]

5.3 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]

Sollte man einen höheren Plan auswählen kriegt man auch Support.

Zu ElasticSearch kann noch Kibana installiert werden, um eine Visualisierung der Daten zu bekommen und den Elastic Stack zu navigieren.

[Elaa]

5.4 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]

5.5 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.

5.6 Vorauswahl

Einen genaueren Feature-Vergleich sehen sie hier in Tabellenform.

Nach einem ersten Überblick wurden nun Aufgrund der Auswahlkriterien diese Systeme zum genaueren Vergleich ausgewählt:

	Lucene Core	**Solr**	**Manticore**	**ElasticSearch**	**Xapian**
Volltext-Suche	x	x	x	x	x
Facetten	x	x	x	x	x
Open Source	x	x	x	x*	x
Ranked Suche	x	x	x	x	x
Auto-Korrektur	x	x	x	x	x
Wildcard Suche	x	x	x	x	x
Query Vorschläge	x	x	x	x	x
Synonym Support	x	x	x	x	x
Boolesche Operatoren	x	x	x	x	x
PHP Support	-	x	x	x	x
(Fast) Echtzeit Indizierung	x	x	x	x	x
Web-Interface	-	x	-	x*	-
Monitoring / Logging	-	x	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

- Apache Solr
- Manticore Search
- ElasticSearch
- Sphinx

Literaturverzeichnis

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.*

The19. *Apache Solr*, 26.07.2019.

Sachverzeichnis

A

Glossar

ESE	Enterprise Search Engine
Facetten	Filter in Bibliothekarssprache

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