Use structured text retrieval to search for scientific works

Betreuer:
Dipl.-Ing. Dr.techn. Roman Kern

Thomas Mauerhofer (1031957)

Graz, am 17. Mai 2017

Inhaltsverzeichnis

1 Introduction

In der Gegenwart gehört die Suche nach Informationen via Internet und im Besonderen Google, zum Alltag der meisten Menschen. Hatte Google im Jahr 1998 noch 10.000 Suchanfragen am Tag, wurde dieselbe Zahl an Anfragen 2006 in einer Sekunde getätigt. (Quelle: http://www.internetlivestats.com/google-search-statistics/ –> Zugriff am 15.05.2017). Dadurch ist es nur naheliegend, dass auch die Anzahl an zugängichen Informationen stetig zunimmt. Täglich werden z.B. unzählige neuen Webseiten erstellt, Artikel geschrieben und wissenschaftliche Arbeiten veröffentlicht. Um diese Menge an Informationen zu managen und zwischen relevanten und nicht relevanten Quellen zu unterscheiden, verwendet man unterschiedliche Suchmaschinen. Dabei soll die Suche simpel aber dennoch genau sein.

Betrachtet man Suchmaschinen für wissenschaftliche Arbeit und Forschung existiert eine große Auswahl an Möglichkeiten. Eine der bekanntesten Suchmaschinen im wissenschaftlichen Bereich ist wohl Google Scholar, welches eine einfache Eingabe zur Suche verwendet und die gefundenen Arbeiten entsprechend ihrer Relevanz listet. Diese Listungen enthalten verschiedene Dateiformate aus unterschiedlichen Jahren zu diversen Themen, die nicht immer die Kriterien der Suchanfrage wiedergeben. Gerade für wissenschaftliche Arbeiten und Forschung ist es allerdings wichtig, ein präzises Suchergebnis zu erhalten.

Diese Arbeit befasst sich mit der Verbesserung von Suchanfragen und deren Ergebnissen für wissenschaftliche Arbeiten in PDF Format. Recherchiert man z.B. einen Autor, erhält man mit den gängigen Suchmaschinen nicht nur die veröffentlichten Artikel und Bücher, sondern häufig auch alle Quellen in welchen dieser zitiert wurde. Ziel dieser Arbeit ist es die Eingabe des Suchbegriffes robuster zu gestalten, die Usability der Suchmaschine zu erhöhen und so treffsicherere Ergebnisse zu liefern. Dies geschieht im Hilfe von einfachen search queries Strukuren, durch selbsterklärendes Front End, Spelling Checks und der Bearbeitung und Beurteilung im Back End.

2 Related Work

Wie bereits in der Introduction erwähnt befasst sich die geplante Masterarbeit mit der Verbesserung von Suchanfragen und deren Ergebnissen für wissenschaftliche Arbeiten in PDF Format. Laut [2] sind alle wissenschaftliche Arbeiten in ähnlichen Strukturen aufgebaut. Diese Strukturen unterteilen sich in capters, sections, subsections and so on, welche sich gut in dem [4] beschriebene Structured Text Retrieval Model anwenden lassen. Dieses Model beschreibt den Umgang von Suchanfragen und deren Ergebnissen für wissenschaftliche Arbeiten, sowie deren Bearbeitung und Beurteilung im Backend durch die Verwendung der Metainformation über die Struktur des Dokuments. Um die Dokumente nach ihrer Priorität zu listen kommen verschiedene Ranking strategies z.B. Jelinek-Mercer smoothing, zum Einsatz. Durch Erweiterungen wie contextualization or aggregation strategies werden die Ergebnisse des rankings noch verbessert.

Um einfache und robuste search queries Strukuren zu gewährleisten wird die Syntax

wie in [1] gestaltet. Dabei bestehen die queries aus mehreren Termen, welche in der Form *label*:*keyword* gegliedert sind. Zusätzlich können Termen ein + vorangestellt werden, um sie besonders hoch zu priorisieren. Durch diese Struktur wird das user interface dann mit einer erweiteren Suche ausgebaut, um die usability stark zu erhöhen, und ein selbsterklärendes Front End zu schaffen.

Der letzte Punkt um die usability zu verbessern sind spelling checks. Diese können wie in [5] beschrieben implementiert werden. Hierbei wird mithilfe der Levenshtein Distance ermittelt, ob ein Wort korrekt geschrieben wurde. Falls dies nicht der Fall ist werden ähnliche Wörter gesucht, und vorgeschlagen.

3 Aufbau

Im ersten Schritt wird eine Grundstruktur für das Projekt geschaffen. Durch das microframework Flask (`http://flask.pocoo.org/`) lässt sich das Frontend in Javascript, und das Backend in Python schreiben. Durch einfache requests lassen sich Eingaben von der Userseite zur Serverseite schicken. Dort wird die Anfrage dann bearbeitet, und ein response versendet. Dies bildet ein solides Grundgerüst für das structured text retrieval model.

Python besitzt eine eigene library um Verbindungen zu MySQL Datenbanken herzustellen. Diese wird verwendet um eine Datenbank wie in [6] beschrieben zu erschaffen. Danach wird die Datenbank mihilfe des in [2] beschriebenen Tools mit wissenschaftlichen Arbeiten befüllt.

Nun sind alle nötigen Teile vorhanden, um effizient in wissenschaftlichen Arbeiten zu suchen, und ein raking system wie in [3, 4] beschrieben zu implementieren.

Als letzter Schritt Spelling checks auf den Suchanfragen

4 Gliederung

1. Introduction
2. Related Work
   a) Unsupervised document structure analysis
   b) Structured Text Retrieval
   c) Structured-Text Retrieval System with an Object-Oriented Database System
   d) Search queries
   e) Real-word error detection and correction
3. Implementierung
   a) Basisstruktur
   b) Aufbau der Datenbank
   c) Userinterface

   d) Satisfaction of searchqueries
   e) Ranking system
   f) Spelling checks

4. Result

5. Conclusion

5 Auswahlbibliografie

Literatur

[1] Sara Cohen, Jonathan Mamou, Yaron Kanza, and Yehoshua Sagiv. Xsearch: A semantic search engine for xml. In Johann Christoph Freytag, Peter C. Lockemann, Serge Abiteboul, Michael J. Carey, Patricia G. Selinger, and Andreas Heuer, editors, *VLDB*, pages 45–56. Morgan Kaufmann, 2003.

[2] Stefan Klampfl, Michael Granitzer, Kris Jack, and Roman Kern. Unsupervised document structure analysis of digital scientific articles. *Int. J. on Digital Libraries*, 14(3-4):83–99, 2014.

[3] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[4] Berthier Ribeiro-Neto and Ricardo Baeza-Yates. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

[5] Pratip Samanta and Bidyut B. Chaudhuri. A simple real-word error detection and correction using local word bigram and trigram. In *ROCLING*. Association for Computational Linguistics and Chinese Language Processing (ACLCLP), Taiwan, 2013.

[6] Tak W. Yan and Jurgen Annevelink. Integrating a structured-text retrieval system with an object-oriented database system. In Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *VLDB*, pages 740–749. Morgan Kaufmann, 1994.

6 Zeitplan

Arbeitsschritt	Aufgabe im Detail	Deadline
Implementierungsphase	Erstellen einer Basisstruktur	
	Datenbank	
	Import von PDFs via pdf-extractor	
	Satisfaction of search queries	
	Simple ranking system	
	Improve ranking system	
	Implement spelling checks	
	Improve Userinterface	
Schreibphase	Introduction	
	Related Work	
	Unsupervised document structure analysis	
	Structured Text Retrieval	
	STR with an Object-Oriented Database System	
	Search queries	
	Spelling Checks	
	Implementierung	
	Basisstruktur	
	Aufbau der Datenbank	
	Userinterface	
	Satisfaction of searchqueries	
	Ranking system	
	Spelling checks	
	Result	
	Conclusion	
	Korrektur	