Similarity analysis of document contents

Lecture: Wissensmanagement

Semester: Summer semester 2017

Authors: Jan-Hendrik Hemmje, Johannes Giere

01.07.2017

Goal

The project aims to offer a tool, which can be used to analyze text documents, to detect similarities between documents and to present the results.

To this end, the documents will be divided into terms and their similarity will be calculated, based on their TF-IDF values. Because different document types will be analyzed, the indexing software “Apache Solr” will be used. This software is based on “Apache Lucene” and uses “Apache Tika” additionally. Apache Tika can parse different document types (TXT-, PDF- & Word-files) and forward them to Solr. On these grounds, “Apache Solr” is perfectly suited for this project. The calculated similarities will be presented in a graph. Furthermore the user can enter search keys and similar documents will be shown to him.

Executing the software

The project rests on two programs. Apache Solr must be preinstalled and the custom tool (written in Java) must be downloaded from the Github repository. Before starting the Java application, it is necessary to configure Solr. Solr can be downloaded from the website http://lucene.apache.org/solr/ or from the archive https://archive.apache.org/dist/lucene/solr/6.5.0/.

It is required that a Linux OS and Solr Version 6.5.0 are being used. Newer versions of Solr may work as well. In the Java application is a sub folder called “scripts”. The scripts in this folder can be used to configure Solr automatically. If Solr has not been extracted in “/opt/solr-6.5.0“, the file path needs to be changed in the scripts. The scripts must be executed on the same computer, Solr has been extracted.

The scripts start Solr, create a new field type and replace the type of an existing field with the new created field type. At the beginning, Solr has to be started with the script “startAsCloud.sh“. Afterwards, a collection has to be created on the Web-Interface of Solr. The Interface can be accessed on “http://localhost:8983/solr“. Navigating to “Collections“ → “Add Collection“, ends to a form where a collection can be created. The collection must be called “wm“ and the “config set“ must be “gettingstarted“. The remaining properties can be left on their default values. In the end, “addNewFieldType.sh“ needs to be executed, followed by “replaceFieldType.sh“. The execution process will be successful, if the returned “status” is 0.

After that die Java application can be configured. If Solr is not provided from the same computer or from port 9983, the variable “CONNECTION_STRING” in class “src/main/java/solr/SolrConnection.java“ must be edited. In the next step, the maven packets have to be installed. If the installation is successful, the application can be started, by executing the class “view.App“. Alternatively, the source code can be compiled with Maven.

Using the tool

After the start of the tool, multiple files can be selected and imported by clicking on “Datei auswählen” in the upper left corner. The imported files will be displayed in the list below. By clicking on “Importieren”, the selected files will be imported to Solr. The import process is finished, if the selected files have been removed from the list. Depending on the amount of files, the process can range from a few seconds to minutes.

By clicking on “Lade Abstandsgraphik“ the distance matrix will be calculated and displayed. On the upper right of the software is a graph, where every single point represents one document. While hovering over a point in the graph, the title of the document is shown.

To receive a list of documents that are similar to a specific document, a document can be selected from a drop-down-control. The control can be found below “Dokumentenähnlichkeit”. By clicking on “Suchen”, the list is being generated. The selected document will be seen on the first position

Moreover, the user can search for a term. Below “Suchterm“ a term can be entered. The list on the right is sorted by accordance and similar documents will be displayed on the top, as well.

Algorithms

To calculate the similarity of documents, an inverted index is created. Across this index, an IDF value and TF-IDF value can be created for every term in a document. The cosine similarity computes the similarity of documents and user queries. The similarity algorithm uses the TF-IDF values.

To display the documents in a distance matrix, the computed similarities are transformed and an external library is used. The library “mdsj.jar“ can compute the matrix.

Issues

Using the parsing functionality of Apache Tika, was problematic. Tika is integrated into Solr, but it creates significant issues with files that are not TXT type.
PDF and Word files can be received by Solr and forwarded to Tika. Tika is responsible to parse the documents so that Solr can extract the terms. Unfortunately, the documents are being parsed insufficient. The parsing process creates letter combination that are absolutely unique. They are unique in that way that they do not occur in any other document. This means, that plenty of terms are equal in the source documents, but are being parsed in a way that makes them distinct to any other term. For instance, letters are removed from terms, but the process of removing the letters is not consistent.

Because stemming is enabled on the field type, the authors assumed the stemming to be the issue. Whereas Solr can parse TXT files without the help of Tika, PDF files have been converted with Calibre and indexed by Solr to prove the assumption. The created terms did not have the observed issue.

Therefore, it has been concluded that the process of Tika does not work satisfactorily. For that reason, it is recommended to use TXT files in a productive environment only.