

Pony: An Information Retrieval System

Indexing and vector space searching

Giannis Apostolidis
japostol@csd.uoc.gr
AM: 814

Michalis Athanasakis
michath@csd.uoc.gr
AM: 858

1 Abstract

Information retrieval systems are continuously gain attention through the last fifteen years after the explosion of the Internet. A good performing search engine is always useful and can introduce new understanding of a dataset. We introduce *Pony*, a modern offline indexing and search engine for documents.

2 Introduction

Our implementation is separated under three basic components. Indexing, Searching, Graphics User Interface(GUI). Each component can provide functionality separated from the rest, except GUI. In the phase of indexing the system is guided to a directory of documents for indexing and performs an indexing algorithm described below to finally store a custom file database to the hard drive. In the phase of searching the program retrieves the database stored from the indexing phase and performs searches using two different algorithms. The program can be executed from a terminal but also provides a GUI functionality to improve usability. Under the GUI mode the user can perform both indexing and searching with just importing the directory of documents to be searched. All parts of the program are implemented with Java.

3 Internals

For the need of this project we implemented a custom database to store all the necessary information without spending unnecessary size on the disk. The database is separated in three big parts that can be accessed separately from the disk. This enables us to fetch things from the disk when is only necessary during the searching phase. The only part that is on memory while the phase of search is the database of the words(terms) that

our system has indexed. The below figures provides an abstract idea on how our system is implemented and how a term is indexed and searched respectively.

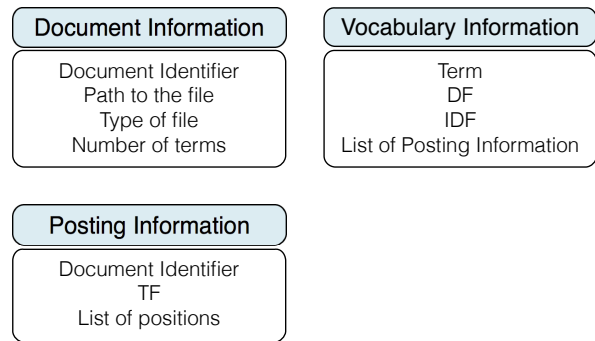


Figure 1: Structure of information entries

The backbone of the database is the Vocabulary database where all unique terms are indexed and later searched. The Vocabulary database is populated during the indexing phase with with all unique terms located inside the document collection. Inside each Vocabulary Entry(Vocabulary Information in the figure) the system store some information and some pointers to other parts of the database for fast access. For each term there is a list of Posting information which it provides more specific information. Posting information holds the document identifier, calculations and the locations of each sighting of the term inside the specific document. Creating a Posting information list for each term essentially provides the system with information about all the sightings of each term. With the usage of the document identifier we can later retrieve the document information, retrieve the file and complete the search. The final output provides plenty of information about the location and the similarity of the returned document to the specific search we request.

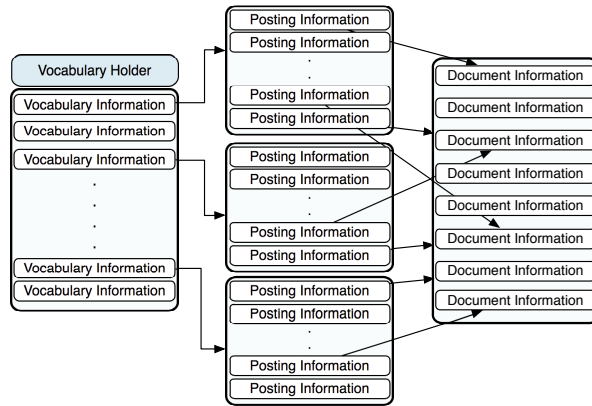


Figure 2: Database structure

When we use the term *list* we imply a data structure responsible to store the information. The actual implementation for each specific data structure is chosen after considering each special characteristic of that part of the database and other factor, performance on writing and on reading.

4 Indexing

The phase of *Indexing* is the phase where the database is initialized using a specific document collection provided by the user. This phase can be done both by terminal or the GUI our system offer. The system has the ability to index documents type of *.txt* with UTF-8 encoding. This provides the ability to index documents in languages different of English. Pony also offer the functionality of stop words, words that when located in the documents are ignored. Along with the stop words specific types of data are ignored. The user has the ability to select a directory with the stop words files of his choice to feed the system to alter the indexing terms. After the valid words have been processed another part of the algorithm is to provide some *stemming* functionality to improve the indexing. The list below provides the types and the behavior the system has on them.

Numbers All digits are ignored from the search. In the case a digit is located inside a word during indexing then the word is separated into two words, the word before the digit and the word after. After that, each of the newly created word is again processed to finally separate all the words from the numbers. All the terms are then processed into indexing.

Symbols A large variety of symbols are ignored during the search. In the case a symbol is located inside a word during indexing then the word is separated into two words, the word before the symbol and the

word after. After that, each of the newly created word is again processed to finally separate all the words from the symbols. All the terms are then processed into indexing.

List of all the symbols can be located inside the directory: `/src/PonyIndexer/stopWords.java`

5 Searching

Searching is the second phase of this system. After all the documents are indexed the user can search in them. The search returns a bundle of information to help the user understand how similar is the document that was returned regarding his search terms. Along with that snippet of text of the documents are returned along with the specific offset from the start of the file where the term is located. The search engine provides the same functionality with the usage of stop words and by ignoring the same numbers and symbols. Also, the search engine ignores the capital and lower letters improving the returning results. In some languages like Greek also punctuation marks are also ignored to offer another improvement. Finally, the search terms are also being stemmed to offer compatibility between indexing and searching.

The actual searching algorithm has been implemented using two separated models. The one model is the Vector Space Model and the other is based on the *OKAPI BM25* standard.

6 GUI

The system provides command line functionality but the user is advised to prefer the GUI interface for better optical and structured results. The GUI provides functionality to both index and search documents. If a document indexing database does not exist the GUI being provided with the documents collection directory and a stop words directory can index them and "export" a database. If the database already exists in the specified directory the search engine loads it and is ready to provide search functionality. After the search the results are being displayed inside the same GUI along with a variety of information.

7 Build instructions

Our system was deployed with: Netbeans 8.0 and Java 1.7.0_45. Building instructions should be similar to other versions of Netbeans.

The following steps show to import the source and dependencies into a Netbeans project and build it from there.

1. We create a new Netbeans Java project by
File > New Project...> Java Project
with Existing Sources
2. We name the project Common and the store location
and press Next
3. We add the source folder located at the Common
folder using Add folder...
4. After the project is created, we import the Stemmer
library located at Common
dependencies.
5. We follow the steps 1 to 3 to create another
project under the name "PonyIndexer" and we im-
port the existing source files from the directory
PonyIndexer
src.
6. We again import the Stemmer library and this time
we also import our created project "Common" by
right-click on Libraries of the "PonyIndexer"
project and by following the steps of choice Add
Project...
7. Final step is to build the project. Almost every pack-
age has it's own main and we can build the GUI
separately.

8 Run instructions

- If you use Netbeans just Build & Run.
- For a command line environment we simply change
directory to
dist and we run the command `java -jar
NameOfJar.jar`. Please be make sure to copy all
the necessary libraries.
- For the GUI, simply double-click the jar file.