Information Retrieval

2021-2-F1801Q110

Nina Singlan & Romain Michelucci

Project B: Customized Wikipedia search engine

GameDisplayer/custom-wiki-search-engine: Project B: Customized Wikipedia search engine (github.com)

Introduction

The project chosen is the creation of a customized Wikipedia search engine on three selected categories.

Different activities have been integrated. On the one hand, the “offline” part is composed of the creation of a text repository also known as dataset (which is described in the next section), the tokenization and indexing of the extracted documents and the creation of a topic profile. On the other hand, the “online” part consists of the process of basic searches, allowing users to perform a search specifying one or multiple fields, and advanced ones in order to analyze queries from specific topics and taking into account synonyms.

To this end, a Maven project has been set up using mainly Lucene for the search engine part, a python parser for the extraction and other NLP APIs for the document’s analysis.

Moreover, a graphic user interface (GUI) is available in order to provide to the user a better experience of the search engine.

Dataset

Creation approach:

The documents extracted come from Wikipedia.

First, from this website, we downloaded an XML file (WikiData.xml) based on the pre-selected categories (listed below). Then, we have created a custom Python parser (WikiDumpXMLtpCSV.py) using the WikiDumpReader. Finally, the maven project has been set up in a way that when it is compiled it automatically produces a new CSV file named WikiData.csv from the original XML file.

Description of the database:

There are 1518 documents. The three main topics are the following ones:

- History and Events
- Natural and Physical Sciences
- Religion & Belief Systems

4 different fields have been extracted from the documents extracted: the title, the abstract, the content and the topics (main and sub-topics).

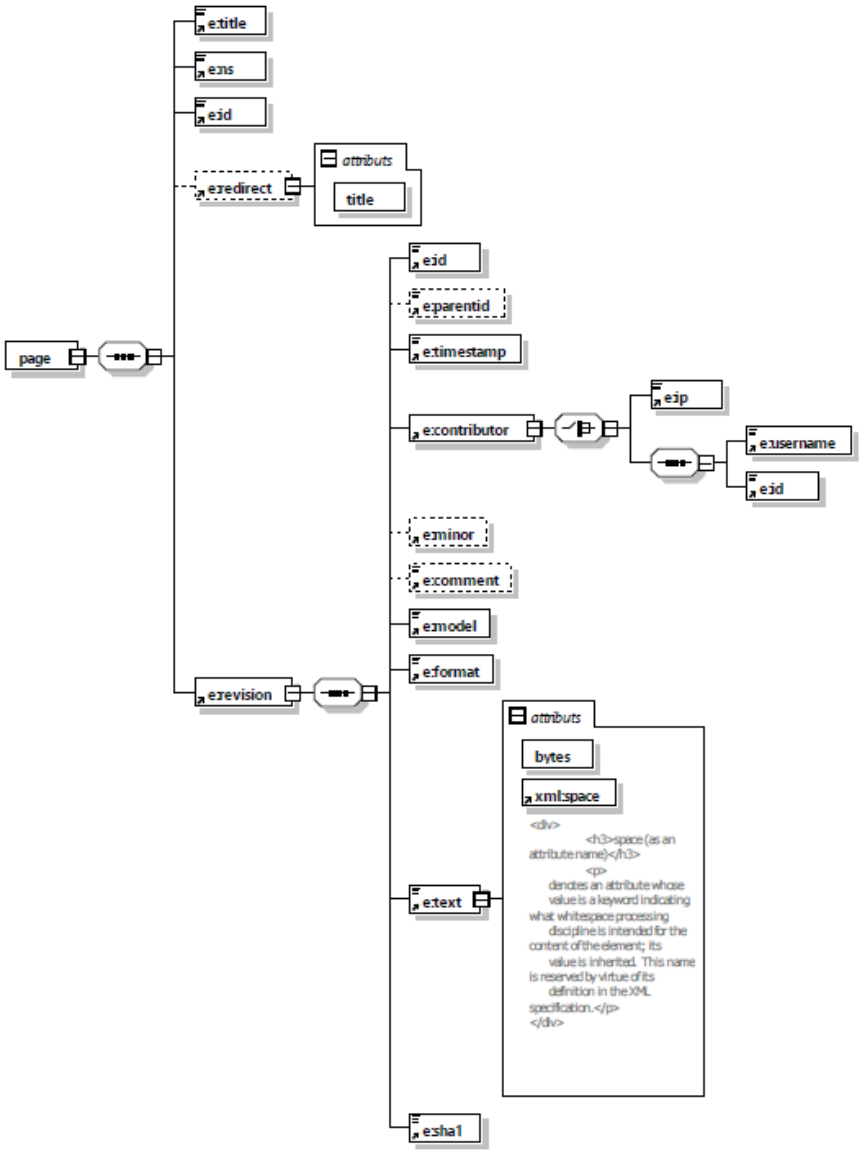
The average length of the title field is 53 characters, 1079 for the abstract and 9689 for the content. The average number of topics per document is around 5. (All these statistics may be found in the Statistics class of the project)

The number of documents per topic is, at first, unknown because a more general technique is used to extract this information (explained in the following section).

Dataset structure:

Below is an XML Diagram showing the structure of a data “page”, which is identified by his “title” and his “id”.

Excluding the title, all the fields of this project are coming from the “text” part of the data. And a major part of this structure is not utilized for the search engine purpose.



Search Engine

Description of the process:

1- **Tokenization and indexing of the documents**: After the CSV file is retrieved from the XML file, it is parsed in order to extract the four different fields. And each document is added to the index.

The title, abstract and content are stored in a TextField because it allows the analyzer to process the content's field. The topics are added in multiple StringFields of the same name.

2- **Creation of the topic profile**: The first step is the *topic extraction*, each document belonging to the right topic is stored in a list of documents. To "detect" the topic we have decided to implement a general technique that may be used in other cases (not only with these documents). To this end, the (sub and main) topics automatically extracted from the document are tokenized and lemmatized (thanks to the Stanford NLPCore API).
Then, a general set of stems is defined (using the PorterStemmer) and if one of the tokens matches one of the stems, the document is stored in the corresponding topic list.
As mentioned earlier, we obtained 559 documents classified as history, 893 as science and 225 as religion. Notice that 156 documents have been classified in different topics and 13 are not classified at all because the topics do not correspond to any of the stems defined. [Appendix]

To identify the worlds that are more relevant for the topics, different metrics have been calculated for all the terms referring to its document's topic. [Appendix]

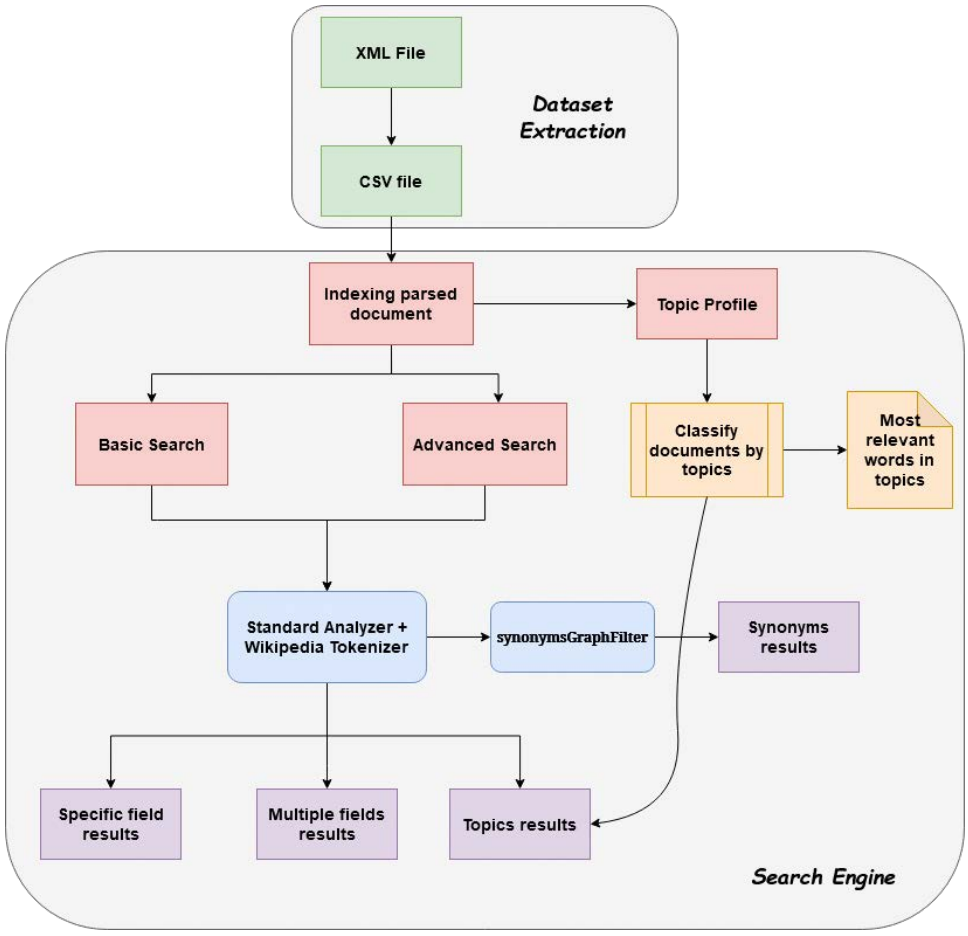
3- **Basic searches:** These searches are allowing the user to search queries, simple or more complex ones like phrases, regular expression, combination of queries, on one or multiple fields. In order to do this, a new analyzer was created based on the model of the Standard one. This analyzer proceeds using WikipediaTokenizer, which is considered as a standard tokenizer aware of the Wikipedia syntax. It is also removing upper case letters and stop words.
The results are classified according to the best score (ScoreDoc.score) obtained and displayed in this order.

4- **Advanced searches:** Concerning the ranking of the pages considering the topics selected by the user, the choice given is among History, Religion and Sciences.
The results obtained in the basic searches are re-classified taking into account the topic choice of the user and the score of the documents (the documents which correspond the

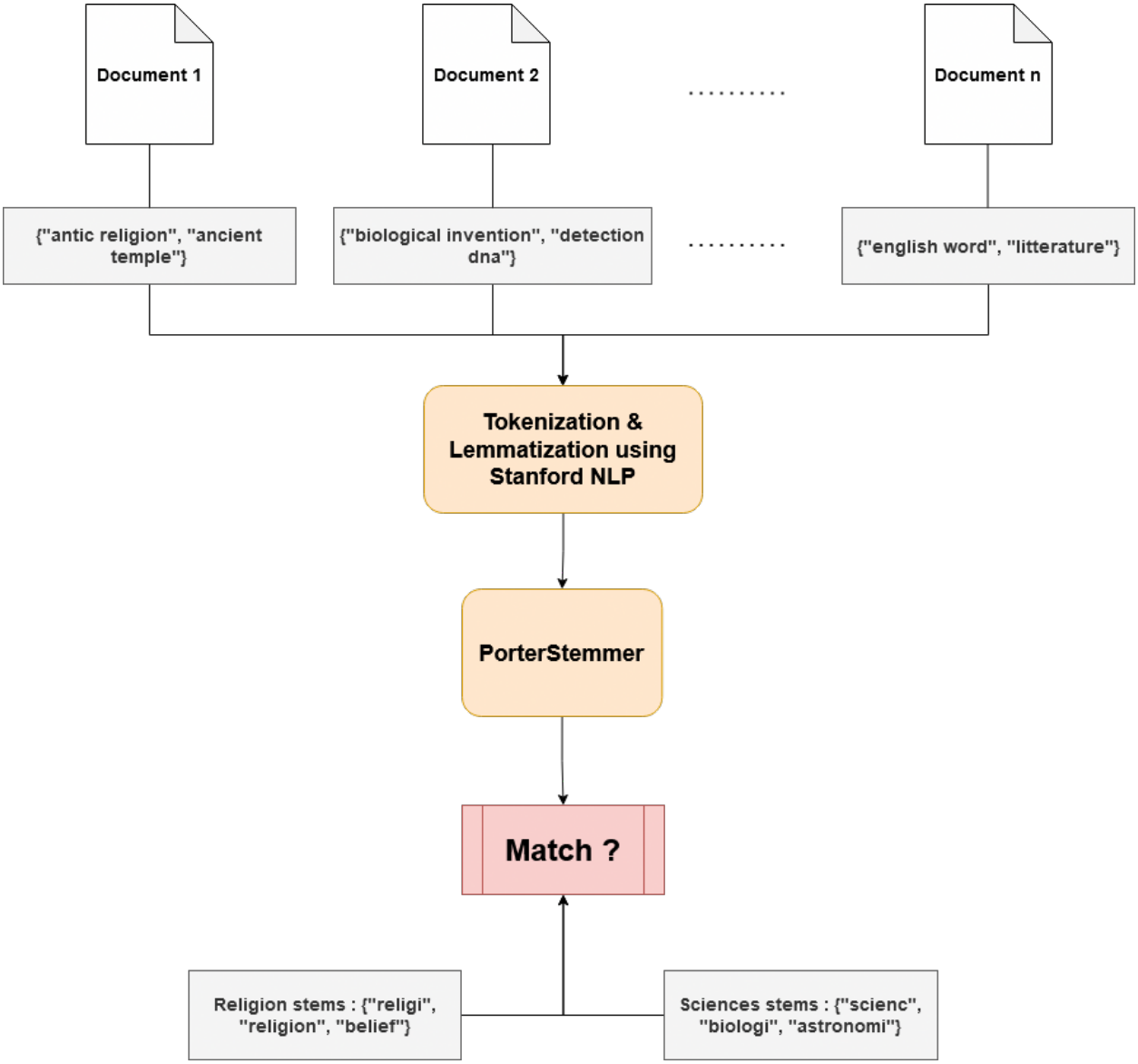
best to the intersection of these two constraints/criteria are preferably chosen).
Using the topic profile, the documents present in the topic(s) selected by the user and having the best score are ranked at the top, then are the ones that have at least one topic in common and finally the ones that do not correspond with any topic (always ranked by score).
Regarding the synonyms, the user is now allowed to choose to use synonyms in his query. This means he can search for “religion” and obtain results with the term “faith”.
To this end, a new analyzer is modeled on the previous one adding a synonym filter (synonymsGraphFilters). WordNet is used, allowing the knowledge of synonym relation between words.

Data flow Diagram:

Below, there is a diagram showing the data flow.



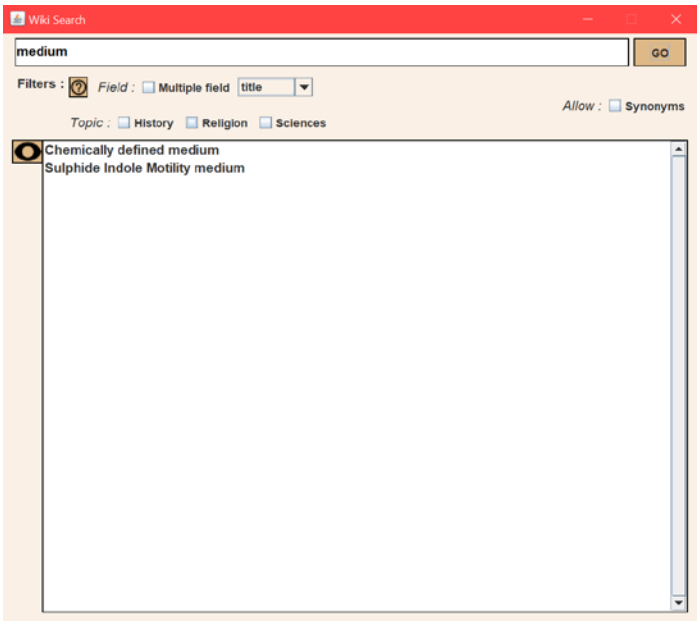
There is also a diagram showing how the topic detection is working. The first document will be classified in the Religion topic thanks to the “religion” stem, the second document in the Sciences topic thanks to the “biologi” stem and the last document will not be classified because it will not match any stem.



Demonstration plan

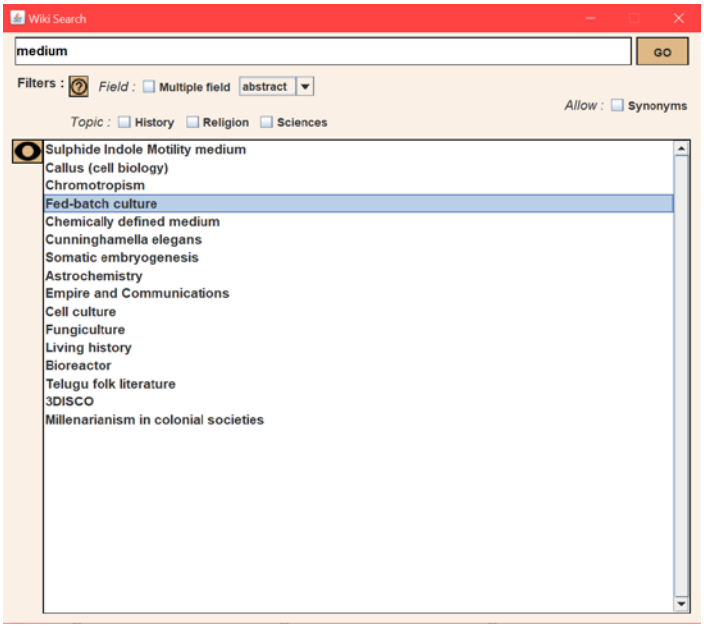
1- **Textual search on a specific field:** For this example, the simple query used is the word "medium".

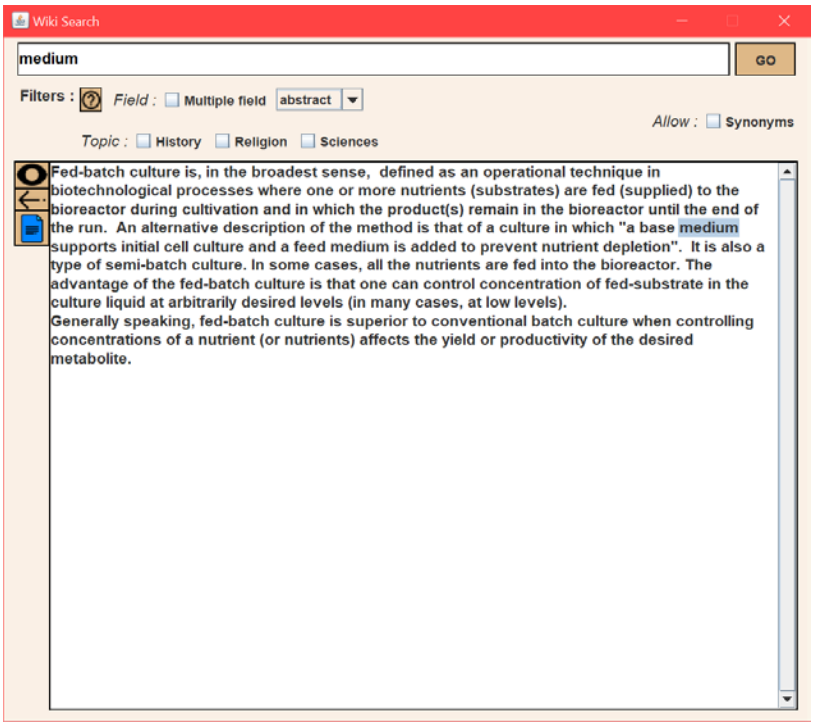
First, the result for the search on the "title" field:



The query is present in all the titles.

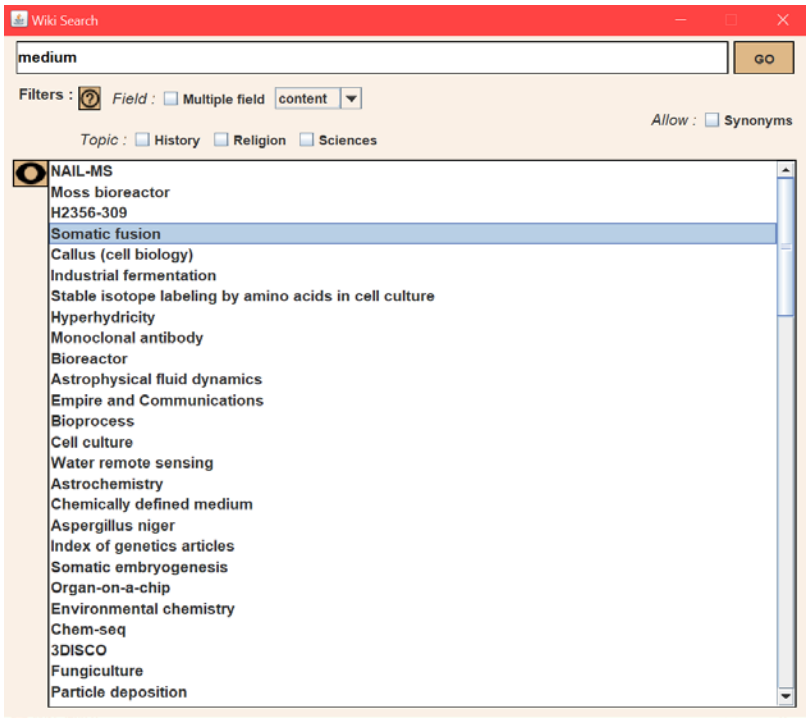
Second, there is the result of the abstract field. Here the search is done on a bigger text, so we can see that more documents are returned.

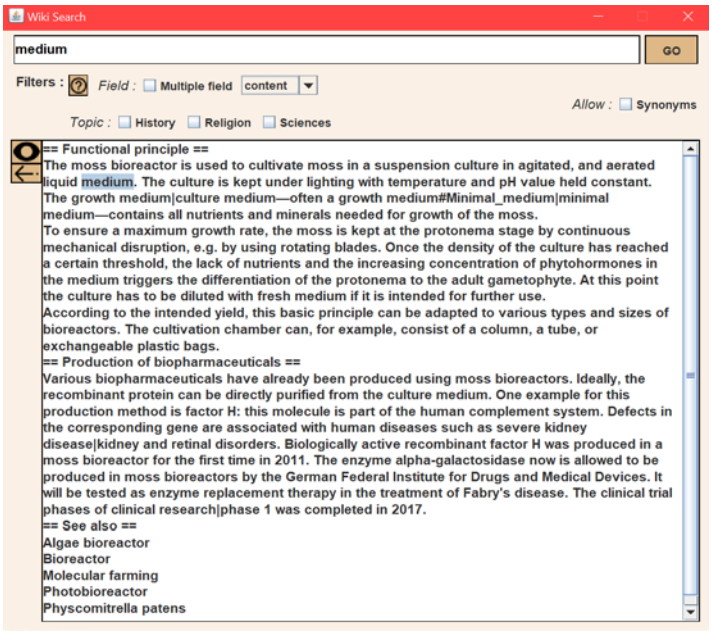




On this example, “medium” is not in the title, but it appears in the abstract.

Finally, there is also an example on the “content” field. As for the previous example, the term does not appear in the title, but on the text content of the page.

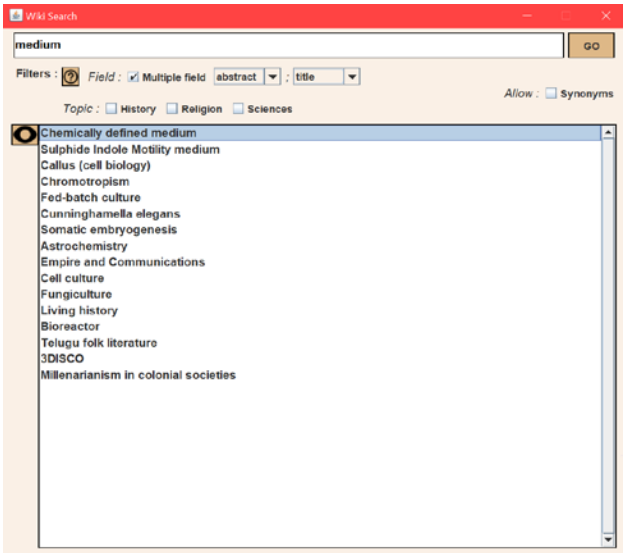




This is not shown on these examples, but the user can also ask for a more complex query like multiple word query or even containing regular expressions.

2- **Textual search on a combination of fields:** The user can also choose to filter his query selecting multiple fields.

For this option, there is a choice, the user can select multiple fields and look for less queries than fields. In this case, the search engine utilizes the last query for other fields.

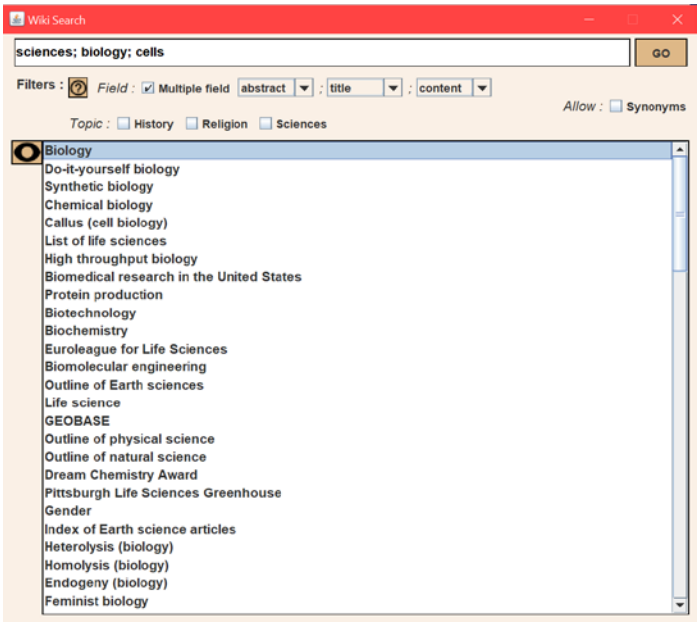


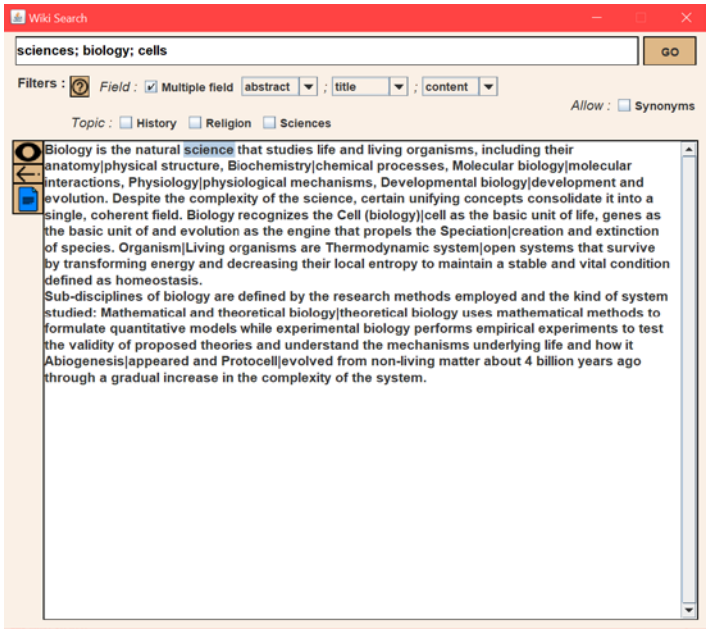
Using the previous example, it appears that the search engine gives priority to the document

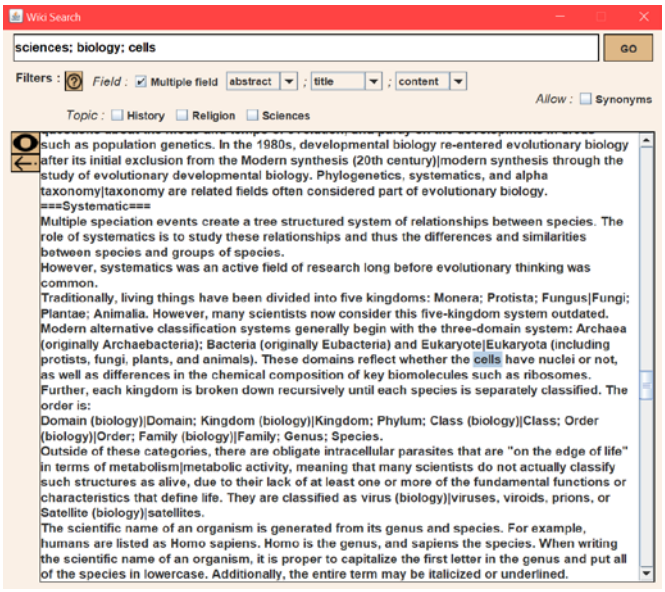
where the query is satisfied for all the fields selected. But it is also returning the documents that correspond to the query for just one field.

Or the user can even write as much queries as fields. Here, the user is looking for “sciences” in the abstract, “biology” in the title and “cells” in the content.

As for the previous example, the search engine is returning first the most relevant document which contains all the queries in all the fields.





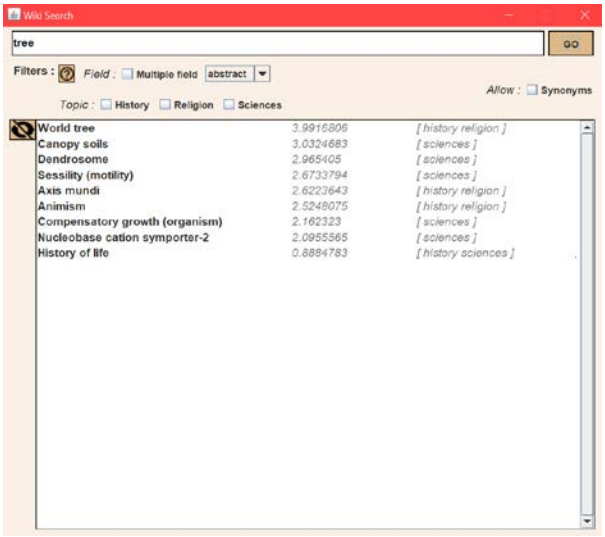


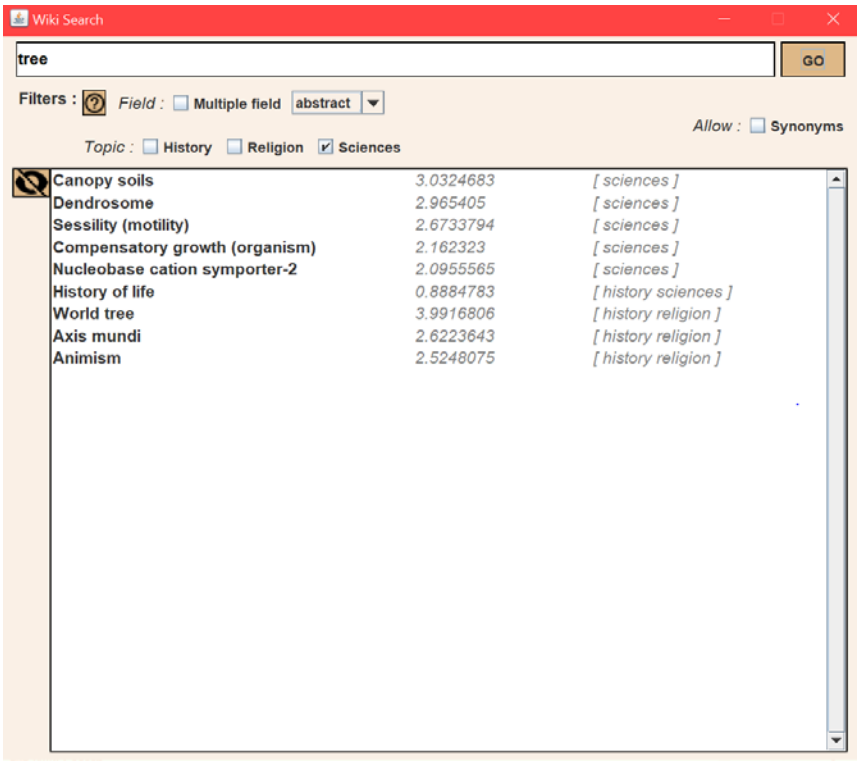
As for the query in one field, the search engine is also taking into account more complex queries.

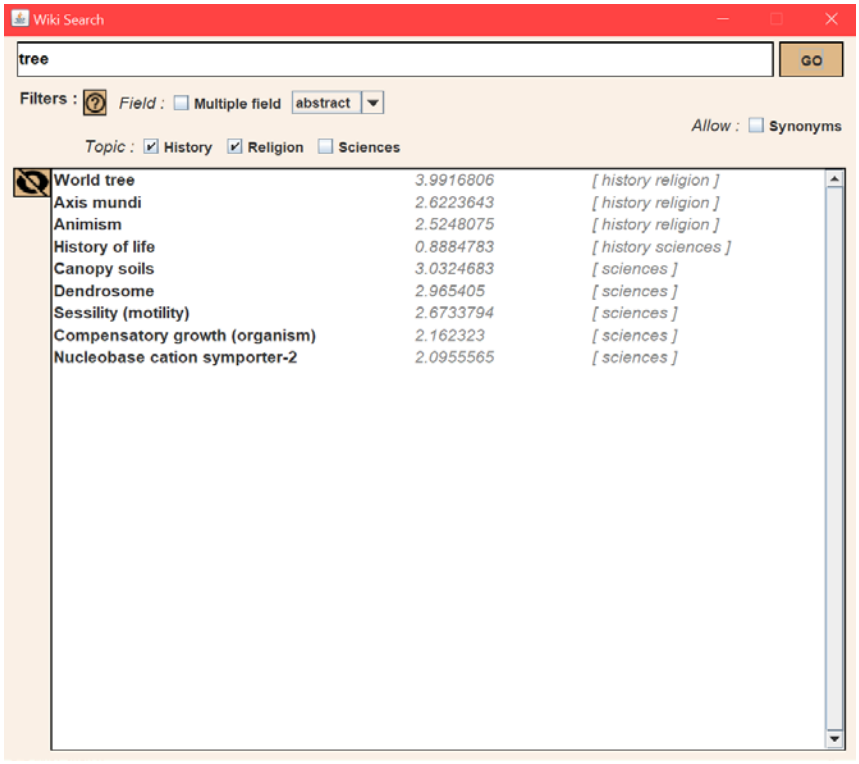
3- **Rank the pages taking into account the topics selected by the user:** The topic profiles are used in order to give a more accurate score.

The user can now choose one or multiple topics, and there are used to rank the results.

First, the user searches “tree” without selecting any topic. One document is selected as the best one and is shown first. Now, is selecting “sciences” as topic, a document less relevant without this selection becoming the better one. The same thing is happening if the user chooses “religion” and “history”.





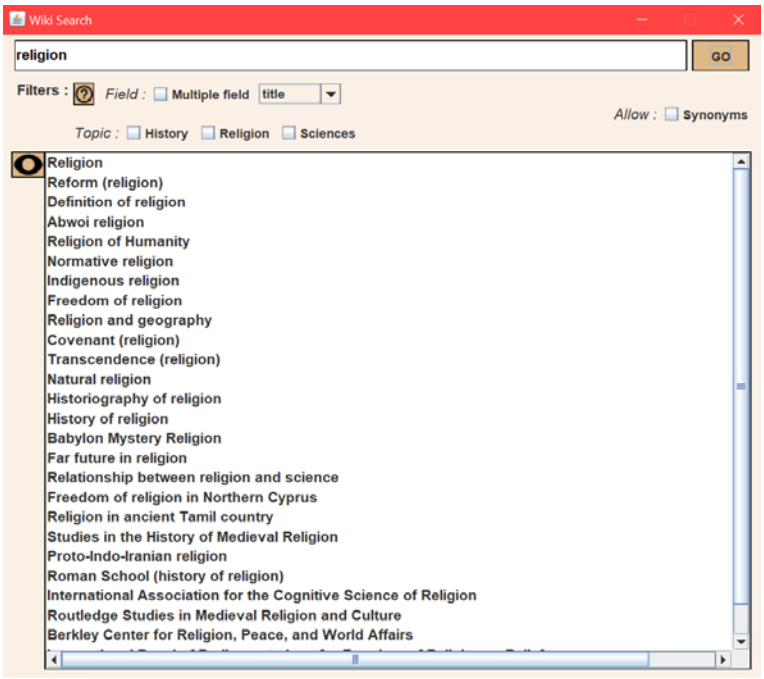


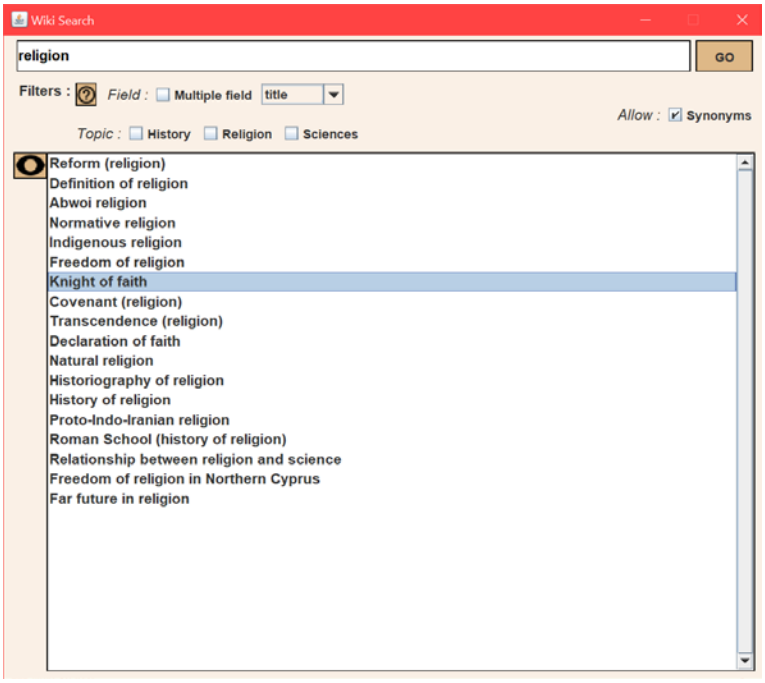
As for previous examples, the search engine handles more complex queries.

4- **Expand the search adding synonyms of the words in the query:** The user can also select

an option allowing the search by synonyms.

Here, he is looking for “religion” in the title. But he is not really satisfied by the results. After allowing the search by synonyms we can see the word “faith” appearing in the title.

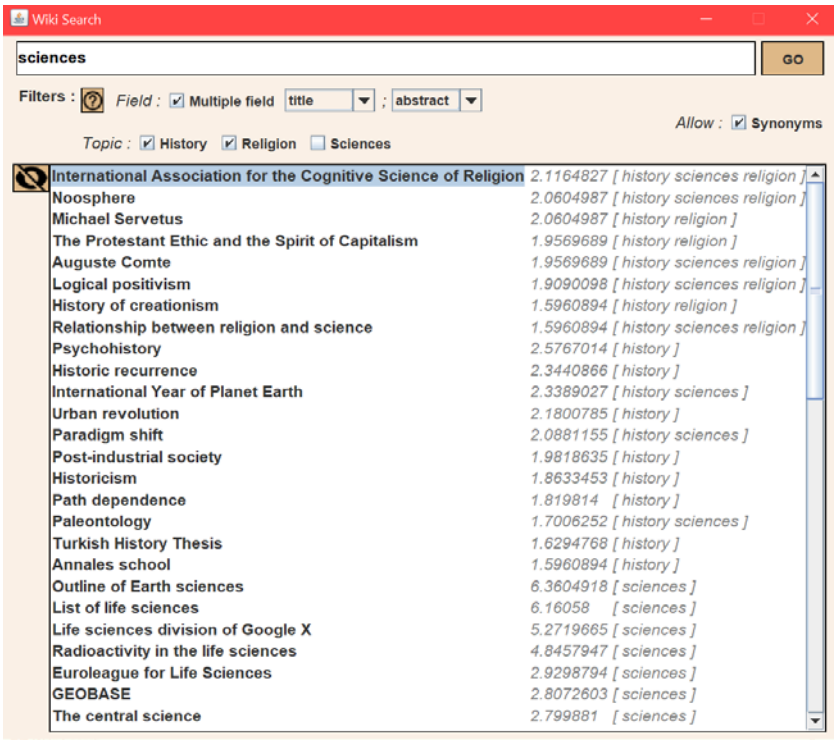


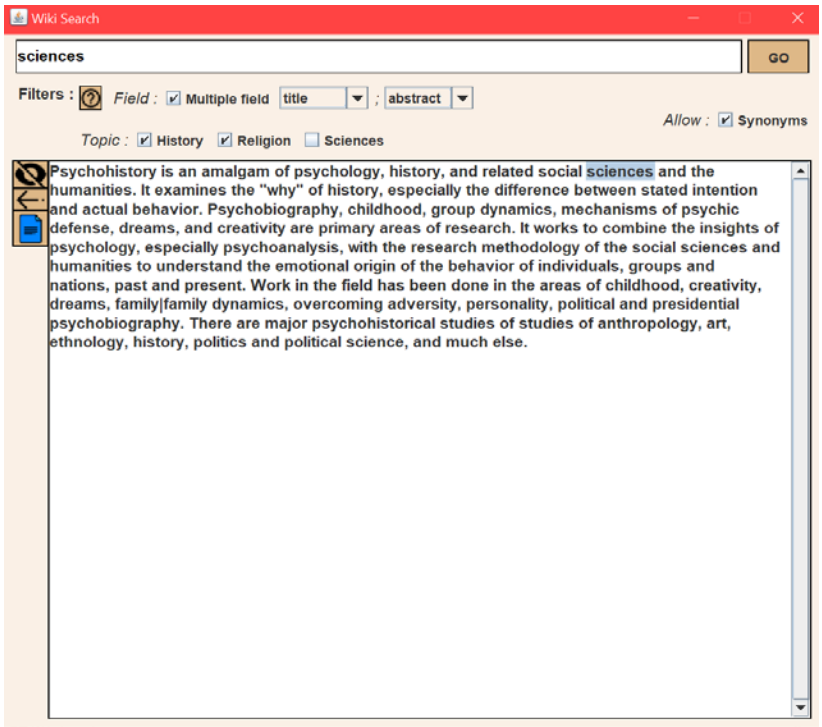


5- **Combination of all possibilities:** Here, there is an example of a query using a multiple

field search, ranking by topics, and allowing the search by synonyms.

The user is looking for “sciences” or synonyms in “title” and “abstract” with a preference for topics “History” and “Religion”.





Appendices

Appendix most relevant words per topic:

Table of Top 10 words of the History topic according to three different metrics (TF, IDF, TF-IDF)

	Word:TF	Word:IDF	Word:TF-IDF
#1	history:581	history:0.3010299956639812	**history:174.89842748077308**
#2	world:238	book:0.6020599913279624	**social:153.07443094958813**
#3	theory:234	world:0.6020599913279624	**historical:145.38576090189193**
#4	social:219	century:0.6020599913279624	**world:143.29027793605505**
#5	book:209	theory:0.6020599913279624	**theory:140.8820379707432**
#6	historical:208	historical:0.6989700043360189	**religion:133.65731807480765**
#7	century:189	new:0.6989700043360189	**society:130.04495812683987**
#8	new:158	social:0.6989700043360189	**war:129.77698128374817**
#9	religion:148	people:0.6989700043360189	**ancient:127.33568816586404**
#10	ancient:145	united:0.7781512503836436	**book:125.83053818754414**

Table of top 10 words of the Sciences topic according to three different metrics (TF, IDF, TF-IDF)

	Word:TF	Word:IDF	Word:TF-IDF
#1	dna:511	biology:0.6020599913279624	**dna:397.6352889460419**
#2	cell:407	research:0.6989700043360189	**cell:316.70755890614294**
#3	cells:370	cell:0.7781512503836436	**cells:287.91596264194817**
#4	chemical:327	including:0.7781512503836436	**chemistry:263.7022762016475**
#5	biology:304	chemical:0.7781512503836436	**chemical:254.45545887545146**
#6	chemistry:292	dna:0.7781512503836436	**earth:237.43753221607534**
#7	science:271	biological:0.7781512503836436	**science:229.0215688438636**
#8	research:268	known:0.7781512503836436	**gene:223.0**
#9	biological:228	use:0.7781512503836436	**protein:215.6588071332874**
#10	earth:228	based:0.7781512503836436	**research:187.32396116205305**

Table of top 10 words of the Religion topic according to three different metrics (TF, IDF, TF-IDF)

	Word:TF	Word:IDF	Word:TF-IDF
#1	religion:279	religion:0.3010299956639812	**religion:83.98736879025076**
#2	religious:244	religious:0.3010299956639812	**religious:73.45131894201141**
#3	religions:113	world:0.47712125471966244	**philosophy:73.14621753606251**
#4	world:102	religions:0.6020599913279624	**freedom:71.58991503529522**
#5	god:99	belief:0.6020599913279624	**jesus:71.36603743993989**
#6	belief:96	many:0.6020599913279624	**god:69.19803042926587**
#7	philosophy:94	god:0.6989700043360189	**religions:68.03277902005975**
#8	freedom:92	often:0.6989700043360189	**science:63.0**
#9	church:76	history:0.6989700043360189	**church:59.13949502915692**
#10	christianity:73	human:0.6989700043360189	**belief:57.79775916748439**

Metrics used:

The Term frequency (TF) used is the *raw count* of a term in a document, i.e., the number of times that term t occurs in document d: $tf(t,d) = f_{t,d}$

The Inverse document frequency (IDF) used is a smoothed version of the theory formula $idf(t,D) = \log\left(\frac{N}{|\{d \$ D : t \$ d\}|}\right)$ where 1 is added to both the numerator and the denominator inside of the logarithm function.

The Term Frequency-Inverse document frequency (TF-IDF) is calculated as $tfidf(t,d,D) = tf(t,d) * idf(t,D)$

Some observations of the results obtained:

- The term frequency already shows interesting results (even if some useless words have been added to the stop_words file).
- The IDF results make appeared some common words that are not relevant such as “many” in #6 of the religion topic or “based” in #10 of the sciences topic)
- The TF-IDF smoothes the results and shows more strong results that can be seen as even more relevant than TF only (and obviously IDF)

Note:

The full list of terms is available in the src/main/resources/topics_profile/*topic*_occurences.txt, *topic*_idfs.txt and *topic*_tfidfs.txt (with *topic* = history or religion or sciences)