**Title**: CS132a spring 2017 Assignment 3: Building a simple Boolean Information Retrieval System

**Author**: Hao Wang

**Date**: 2/27/2017

**Description**: The program implements a simple Boolean information retrieval system supporting conjunctive (“AND”) queries over terms, and apply it to your Wikipedia movie corpus.

**Dependencies**:

OS: OSX(macOS Sierra)

Python:2.7.10

Flask (<http://flask.pocoo.org/)>

External Package: Stemming(<https://github.com/jedijulia/porter-stemmer/blob/master/stemmer.py)>. It is in my uploaded file called “stemmer.py”.

**Build Instructions**:

It includes two main steps.

The first step is building the index module. It includes python files to deal with creating the inverted index. At the end of the first step, it will built a new json file storing Inverted Index.

The second step will creates Flask UI and using Flask framework and build some python files to deal with user’s query, to load the needed term’s posting list from the Index’s json file and to load the needed articles from the corpus json file for showing in webpages.

**Run Instructions**:

You just need to run the “boolean\_index.py” without any input. It will load a json file called “file\_corpus.json” for running the system by default. If you would like to try my “test\_corpus.json”, you can change the path in “boolean\_index.py”.

**Modules**:

Index Module:

In the module, it is responsible for creating the index json file according to the corpus. It includes “control.py, index.py, stemmer.py”. In “control.py”, this is the python file which runs the inverted index system. I use the time package here to count the running time. In “index.py”, it has the different functions to deal with the corpus including tokenization, wiping off the punctuations, ignoring the stop words, stemming and lemmatization using an external python file, “stemmer.py”. In stemmer.py, this is an external python file downloaded from github ( <https://github.com/jedijulia/porter-stemmer> ). It is reliable because it creates according to an original paper.

Flask UI Module:

In the module, it is responsible for creating the UI webpages and get the user’s query and return the search result. To the front-end, there are three html files in templates. “index.html” is the homepage includes a search bar which user can input the query. “serp.html” returns the list of articles with titles and abstracts. “article.html” is used for displaying the details of a certain article the user clicks.

To the back-end, it includes “boolean\_index.py, entry.py, load\_article.py, search.py, stemmer.py.” In “boolean\_index.py”, it is the flask web framework file which is responsible for the framework of the website. In “search.py” it deals with a single query and conjunctive queries using tokenization, ignoring the stop words, stemming, lemmatization using an external python file, “stemmer.py” and intersection. It will return a posting list of for user’s query to the “boolean\_index.py”. In “load\_article.py”, it works for retrieve the article from the corpus according the posting id. In “entry.py”, it is an Entry class in the file which is useful to create the object for each article, which stores the fields like title, text, abstract. In stemmer.py, this is an external python file downloaded from github ( <https://github.com/jedijulia/porter-stemmer> ). It is reliable because it creates according to an original paper.

**Testing**: Considering the time consuming of the whole system, we just create a ten article corpus. After all sets, I run the programmer on thefilm campus. It takes less than 30 seconds to create the index in my new MacBook pro.

**Normalization Details:**

Stopping word list:

list = ['a', 'an', 'and', 'are', 'as', 'at', 'be', 'by', 'for', 'from', 'has', 'he', 'in', 'is', 'it', 'its', 'of', 'on', 'that', 'the', 'this', 'to', 'was', 'were', 'will', 'with']

Stemming: an external library using porter-stemmer

GitHub: <https://github.com/jedijulia/porter-stemmer>

**Query example**:

Query: love

Test: single query



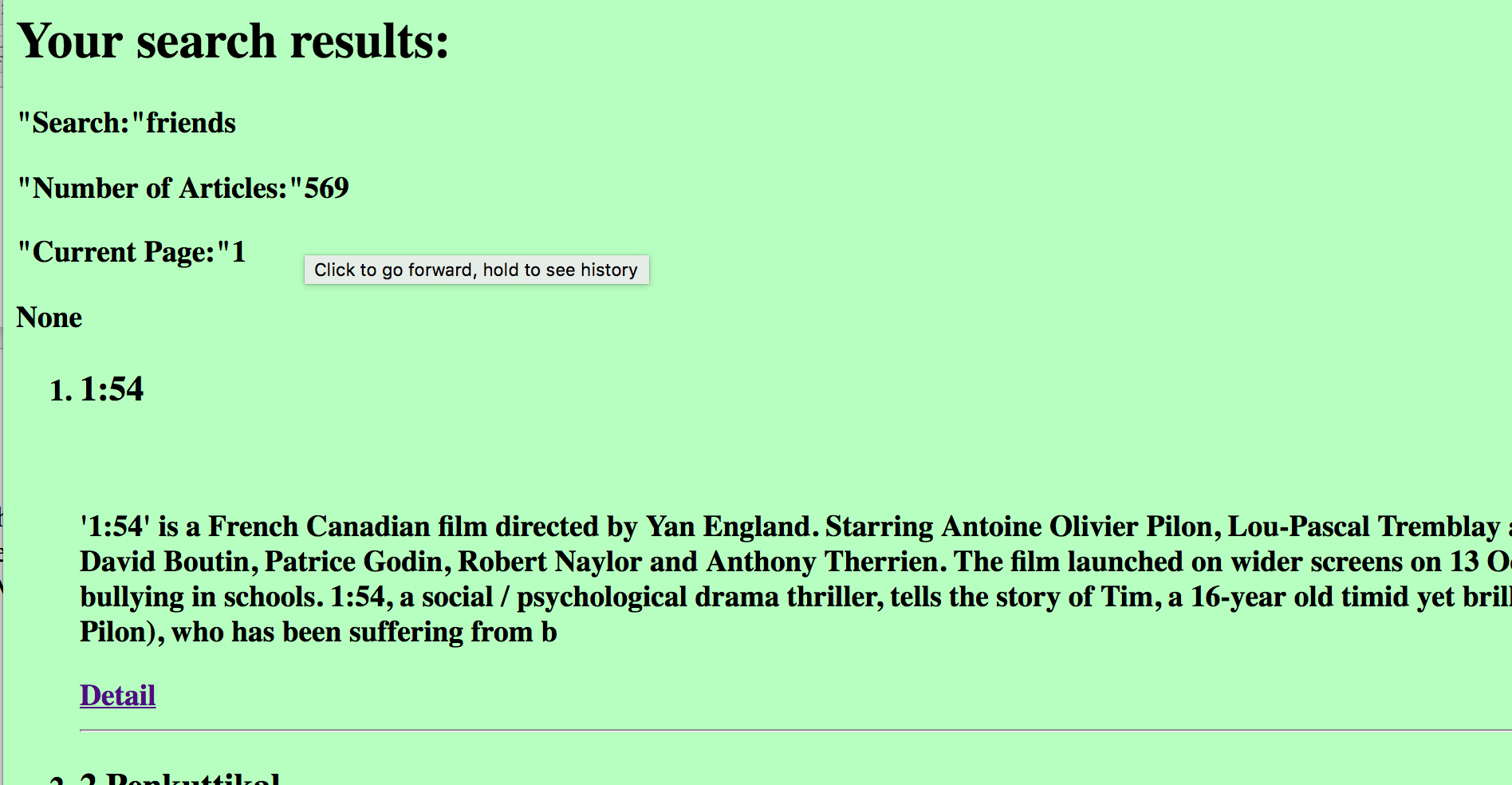
Query: an love story

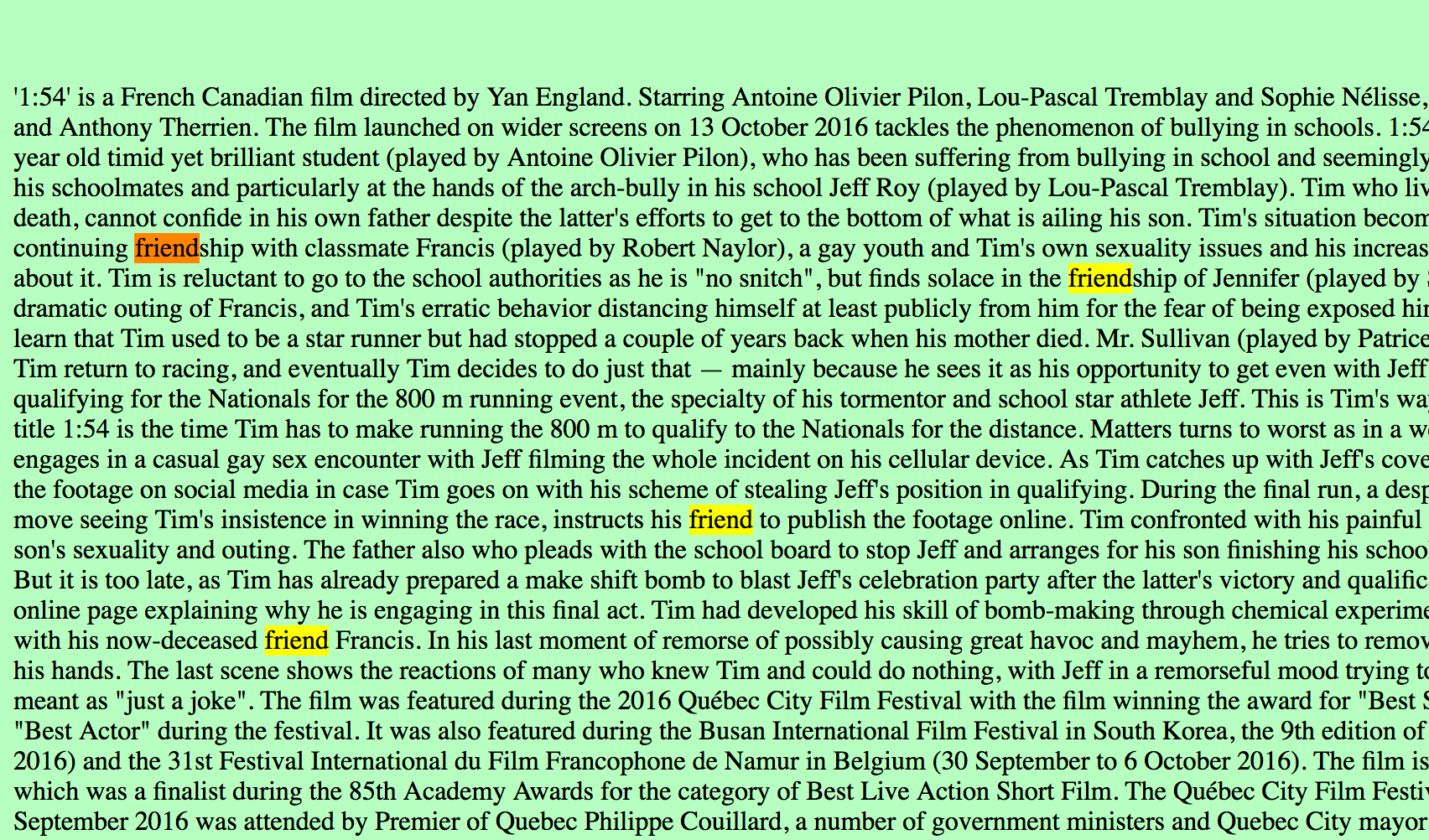
Test: the intersection of conjunctive queries and the function of ignoring stop words.



Query: friends

Test: stemming





Query: cutting

Test: lemmatization



