732A47 Text Mining
Fang Wei-Kleiner

Lab work for Information Retrieval
Simple App Search Engine in Python

Purpose

After completing this lab you should have knowledge on basic data structure for web information retrieval such as inverted index, as well as programming techniques such as web crawling, tf-idf weighting and ranked query answering.

Task description

The number of apps developed for the web and the mobile devices is increasing tremendously. Therefore efficient and effective search engine for apps is of vital importance. In this lab, you will learn to construct a simple Android app search engine in Python.

For each App there is a piece of text describing it. Your task is to extract this kind of information from the specific websites and build inverted index over the documents. Moreover, the term frequency and document frequency are computed and stored in the inverted index. Based on this index structure, the search engine should take the keyword query as input and evaluate the query over the index using the ranked query algorithm we learned from the class.

Instruction

Step1: App crawling
You will access the app websites such as Google Play (https://play.google.com/store) and AppBrain (http://www.appbrain.com/) to obtain the desired description texts for the apps. There is no constraint on which kind of apps you choose. The number of apps should not be less than 1000. Store the description text in the files.

Please note that you need to extract full description of the apps from the above website, instead of the snippets. Take the example of http://www.appbrain.com. By browsing the category, you could get the links of the items. For instance, following the link

http://www.appbrain.com/apps/popular/tools/?o=10

you will get in the html file the blocks like this

```
<div>

<a href="/app/flashlight/com.intellectualflame.ledflashlight.washer" class="app-result app-result-medium safelink" style="display:block;float:left;">
        <div class="table" style="width: 100%"><div class="tr">
                <div class="app-result-medium-image td">
```

```
                        <img
src="http://lh5.ggpht.com/CcIcUa1HqbeJGu8BF_Q72dxDKeY3ThBJhzVIPhVaIvDhXBtYfxJordJaIA2P35VxEe91Nn_A8kE21MRiBN0UQlB_=s48" />
                    </div>
                    <div class="td" style="vertical-align: top;">
                        <div class="table">
                            <div class="tr"><div class="td" style="vertical-align: top;">
                                <div class="browse-emph app-result-medium-title">Flashlight</div>
                            </div></div>
                            <div class="tr"><div class="td" style="vertical-align: top;">
                                <div class="app-result-score" title="AppBrain Score (0-100)" style="display:inline-block; margin-right: 3px; background-color: #8eb03f">92</div>
                                <div class="app-result-desc-low" style="display: inline-block">Free</div>
                                <div class="app-result-desc-low" style="display: inline-block; margin: 0 3px;">|</div>
                                <div class="app-result-desc-low" style="display: inline-block">10,000,000+</div>
                            </div></div>
                        </div>
                    </div>
                </div></div>
                <div class="app-result-desc-mid" style="margin-bottom: 5px;">Intellectual Flame Co., Ltd.</div>
                <div class="app-result-desc-low">The brightest, fastest, and most handy flashlight you will ever have! The one you will never forget to bring when in nee</div>
            </a>

        </div>
```


---

which describes the first app in the list.
The task is to identify these blocks and follow the item links to obtain the full description.
To identify them, one can use certain pattern matching tricks, such as searching for strings starting with
" <a href="/app/"
Of course it is your freedom to decide which kind of pattern you are going to use.

Step2: Index construction
Build inverted index on the texts. You will use NLTK to preprocess the text, such as tokenizing, normalizing, etc. Compute and store tf, df in the inverted index.

Step3: Query processing
Write a ranked query processor using vector space model. The input parameter is the set of keywords and integer k, for top-k query answering. The query processor should compute the vector space similarities of the query to the documents. Top-k documents are returned according to the ranked similarity values.

Submission

Write a short report describing your algorithm and the experimental analysis.