CE706-7-AU

INFORMATION RETRIEVAL

ASSIGNMENT 1

Víctor Pérez Cester (vp19885@essex.ac.uk)
Joel Valiente Sanchez (jv19228@essex.ac.uk)

Index

Team Members

We’ve developed these assignment joining efforts. The team Members are

1. Joel Valiente (jv19228@essex.ac.uk) -> 1900289
2. Victor Perez (vp19885@essex.ac.uk) -> 1900232

Project Environment

The project is developed using the **Python** programming language using the latest stable compiler 3.7.6. The project has some module dependencies such as pandas, nltk and numpy among others. We provide a requirements file detailing all the modules you must install to run the code successfully. Further details on how to run the code can be found in the below subsections.

Project Structure

The Project consists of a src/ folder and an output/ folder along with some python files at the root directory. Here we list and explain the aim of every module.

At the root directory we can find the following folders and files

- **src/** : This folder contains all the source code used for these assignment. It contains 6 python modules, each of them aims to solve a task of the assignment.
- **output/** : This folder contains all the output files generated after the program execution. For each of the tasks in the assignment, we will generate a text or csv file and write the output of the task.
- ***outputUrl1/*** : This folder contains the output files generated after executing the first url.
- ***outputUrl2/*** : This folder contains the output files generated after executing the second url.
- **main.py** : This python module will solve the assignment when executed from the command line terminal by the user.
- **definitions.py** : These python module defines some global variables such as file paths that other modules may need.
- **requirements.txt** : Specifies the requirements and dependencies of the system
- ***README.md** :* Brief summary of the work we’ve done and how to run the code

Inside the **src/** folder we can find these python modules :

- **KeywordsFilter** : *Parses a given URL and extracts all the visible information.*
- **Parser** : *Parses a given URL and extracts all the visible information.*
- **Pos** : *Computes the POS tags for every token in the tokens list and writes this in the output directory.*
- **Preprocessing** : *Preprocess a text. Splits it in lines and performs tokenization. Removes punctuations and perform lemmatization. Returns the documents in the text and the tokens.*
- **Tokenization** : *Defines a series of methods to tokenize texts, perform lemmatization and remove punctuation characters.*
- **UrlDownloader** : *Downloads a given URL. Throws an exception if the URL is not reachable.*

- ***Stemming** : Computes the stems for every token.*

How to Run

To run the project you must install all the dependencies first by running the following command on your linux terminal.

```
pip3 install -r requirements.txt
```

Once all the dependencies are installed, you can execute the code as above.

```
python3 main.py --url <your_url_to_parse>
```

If you want the system to be verbose you can add the --verbose flag as above.

```
python3 main.py --url <your_url_to_parse> --verbose
```

Once the code finished the execution you can check the outputs inside the **output/** folder

Implementation

Engineering a Complete System

We've created a robust and reliable system than can be executed from the command line. It supports command line arguments so that the url to download can be specified from the command line as shown in the previous section. The **main** module is the one who manages all the execution and distribute the workload for every of the modules dedicated to solve a concrete task.

At first, it will create an instance of the UrlDownloader class and delegate to these module the downloading of the url html. If the URL is not reachable it will throw an exception and end the execution

HTML Parsing

The Parser module is dedicated to extract the HTML tags from the raw html. Using the python module BeautifulSoup we extract all the visible information of the hml, getting rid of text appearing in tags such as style, script, head, etc

After extracting all the information from the tags, it will return it as a raw string.

Pre-Processing

The preprocessing module is dedicated to preprocess the webpage text so that it can be handled by next modules. The preprocessing task involve the following subtasks

- **Documents** : *Split the raw text into documents. We say that a document is a line contained in the raw text. A line can be any group of words that have a dot at the end or a line break.*
- **Tokenization** : *For every document, tokenize it and remove the punctuation characters.*

In both documents and tokens the english stop words are removed. After this final step, it returns a list of documents and a list of tokens

POS Tagging

The POS module is dedicated to label each token with its POS tag. It receives a list of tokens generated in the previous step (Pre-Processing). For every token in the list computes its POS tag by using the NLTK python module.

Writes the tags for each token to an output file and return the tags.

Selecting Keywords

The KeywordsFilter module is dedicated to determine the best keywords and sentences by computing the TF*IDF measure. As we have seen in class, TF-IDF measure construct a table where each cell contains the TF-IDF value for a given word of the vocabulary in the current document. The vocabulary is defined as the set of unique words of tokens. In order to compute the TF-IDF table, this module first computes the Term Frequency table (TF) which is a dictionary containing for each document its vector space ,then the Document Frequency (DF) for each term belonging to the vocabulary, using DF the Inverse Document Frequency (IDF) is computed and, finally, the TF-IDF table is computed (multiplying each TF value by the IDF of the current word). If you take a look at the CSV file generated in this task you will easily see what is done in this task.

The TF-IDF table is used to index the word and the documents. The following table shows the structure that the CSV file has in order to proceed with the explanation:

Word	**IDF**	**TF-IDF doc1**	**TF-IDF doc...**	**TF-IDF docN**
word1	value	value	value	value
word...	value	value	value	value
wordN	value	value	value	value

For indexing the words, for each word in the vocabulary we read the corresponding row in the CSV file, sum all numerical values (and subtract the idf as it also appears in the row). Then store in a dictionary the word as the key and the sum of TF-idf as the value and sorte that dictionary in descending order by the value.

For indexing the words, each document column is read and all the TF-IDF values are summed. Store it in a dictionary where the key is the document itself and the value is the TF-IDF values of the words appearing on it. The dictionary is sorted again in descending order by the value. As you can imagine, the more words a document has the highest value it will have. This is not the best ranking for documents but it is the easiest one regarding the assignment instructions.

Morphological Analysis

The Stemming module is dedicated to compute the stems for every token. It uses the WordNetLemmatizer class from the nltk module to compute the stems. In order to apply stemming to each word in the vocabulary (the method ***getVocabulary()*** from KeywordsFilter module returns it) the stem is computed and it is stored in a file.

Experimentation

We tried out system performance on some web pages. Here we present the results we obtained.

For the http://www.multimediaeval.org/mediaeval2019/memorability/ url we've obtained the following results

Top 10 Words	**TF-IDF**
media	54.9
20	49.4
ability	46.4
video	46.1
memorability	41.8
multi	38.2
mediaeval	35.9
multimedia	35.7
predict	32.3
videos	29.5

Top 10 Sentences	TF-IDF
researchers find task interesting work areas human perception impact multimedia perception image video interestingness memorability attractiveness aesthetics prediction event detection multimedia affect perceptual analysis multimedia content analysis machine learning though limited	152.9
come set pre-extracted features dense sift hog descriptors lbp gist color histogram mfcc fc7 layer alexnet c3d features etc	92.4
effective memorability prediction models also push forward semantic understanding multimedia content putting human cognition perception center multimedia analysis	89.4
dataset composed 10,000 soundless short videos extracted raw footage used professionals creating content particular commercials	78.5
applications make use information predicted memorability important understand memorability well enough able avoid systems hyper-optimized viewer responses	76.9
like aspects related video relevance aesthetics interestingness memorability regarded useful help make choice competing videos	75.0
video consists coherent unit terms meaning associated two scores memorability refer probability remembered two different durations memory retention	73.1
task participants provided extensive dataset videos accompanied memorability annotations well pre-extracted state-of-the-art visual features	68.1
motivation task growth recent years amount multimedia content shared consumed via online platforms	67.4
increasingly important understand makes videos memorable order keep use automatic processing techniques evenhanded	66.3

For the https://sites.google.com/view/siirh2020/ url we've obtained the following results:

Top 10 Words	TF-IDF
de	52.6
university	37.8
ir	37.09
2020	31.0
usa	30.2
per	28.8
work	28.1
lu	26.9
workshop	25.2
health	25.2

Top 10 Sentences	TF-IDF
health-related content particularly challenging due specialized domain language implicit differences language characteristics depending content type patient-generated content like discussion forum blogs internet sources healthcare documentation clinical records professional scientific publications clinical practice guidelines clinical trials documentation etc	196.4
planned workshop functions venue different types contributors mainly task providers solution providers meet together exchange experiences	100.4
covering ir technologies heterogeneous health-related content open multiple languages particular interest exploitation structured controlled vocabularies entity linking	82.1
moreover also critical provide search solutions non-english content well cross-language multilingual ir solutions	76.3
workshop forum community present discuss current future directions area based experience results obtained bioasq task	76.1
increasing interest exploiting vast amount rapidly growing content related health means information	71.6

retrieval deep learning strategies	
submissions must written english following springer lncs author guidelines submitted pdf files easychair	63.6
expect investigation topics task continue workshop based new insights obtained discussions workshop	63.5
use specialized machine translation advanced deep learning approaches improving health related search results	61.3
thierry hamon limsi cnrs université paris-saclay orsay université paris 13 villetaneuse france	58.5

Since the output for each stage of the pipeline is too long to paste it here we encourage you to run the code and check the output/ folder where you will find the output for every pipeline stage.

There is a folder containing full output for each url provided. Check these folders to see the full files.

Discussion

We believe that our system works correctly and performs good. We've built a robust and reliable system to download a given URL and compute the best indexing keywords and sentences so that an Information Retrieval System can find user queries using the computed keywords and sentences.
To determine the system efficiency and performance, an evaluation step should be carried out.

Also, our system determines the best sentences as the sum of the TF_IDF of every token in the sentence. Therefore, longer sentences become better ranked by our system. We believe that these way of ranking sentences is not the best but works acceptable.

Improvements

A future work would consist in determining the system efficiency and performance by using evaluation techniques. Also, doing some more research on how to rank sentences and build a better model for ranking sentences.

Also, we could improve the system language recognition so that it only processes english documents and tokens by using the NLTK module.

Moreover, we'd like to improve the keywords ranking by classifying them into groups; such as dates, places, nouns ... so that when a query is performed a better search can be applied.
For example, for a given URL we could specify the top keywords relating dates, locations, and other types of descriptors. Then when the user performs a query, the system would convert that query into a set of parameters such as date, location, etc ...