Practical lessons: 01

(Information retrieval perspective)

Main tasks:
Assuming that you have a set of topics and a set of documents (corpus).
You have to work on the following tasks (A, B, and C):

A: Parsing the topics and corpus

1. **Parse** the topics file
2. **Parse** all **html** documents and **metadata** from a corpus file
3. **Extract** the clean text from all the parsed html documents
4. **Save** the clean text of all the documents including metadata to a **file.**

You can use the following program to implement the above subtasks (from A_1 to A_4):
"parsing_topics_documents.py"

The following command will display the parameters that the script accepts:

```
python parsing_topics_documents.py -h
```

Output:

```
Parsing arguments/input parameters ...
usage: parsing_topics_documents.py [-h] -corpus-path [CORPUS_PATH]
                              -topics-path [TOPICS_PATH]
                              -topics-documents-path [TOPICS_DOCUMENTS_PATH]
A program to parse a list of documents from a file for a set of topics optional arguments:
-h, --help show this help message and exit
-corpus-path [CORPUS_PATH], --corpus_path [CORPUS_PATH] The corpus file path
-topics-path [TOPICS_PATH], --topics_path [TOPICS_PATH] The topics file path
-topics-documents-path [TOPICS_DOCUMENTS_PATH The topic to documents output file path
```

This script actually accepts three parameters such as corpus-path, topics-path, and topics-documents-path.

corpus-path: the path of the file which contains all the html documents.

topics-path: the path of the file containing all the topics.

topics-documents-path: The clean documents will be stored in this file.

You can run the following command with the necessary parameters:

```
python parsing_topics_documents.py -corpus-path data/WT10G_LMDIR_TOP10 -topics-path data/topics451-550.txt.csv -topics-documents-path output/topics_documents
```

NB:

1. Open the input files in “data/WT10G_LMDIR_TOP10” and “data/topics451-550.txt.csv”. See the format of the data.

2. Your python interpreter should have the package “BeautifulSoup” installed.

3. Open the output file in “output/topics-documents” and see the parsed documents.

B: Indexing the corpus

1. **Load** the parsed clean text of all the documents including metadata
2. **Index** the documents and **keep the metadata** of the document
3. **Tokenize** a document into terms
4. **Save** the document’s metadata to **file**
5. **Save** the **Inverted index** to **file**

You can use the following program to implement the above subtasks (from B_1 to B_5): “indexing_documents.py”

The next command will display the parameters that the script accepts:

```
python indexing_documents.py – h
```

This program accepts four parameters, including topics-path, topics-documents-path, documents-metadata-path, and documents-index-path.

topcis-path*: the path of the file containing all the topics.*
topics-documents-path*: the path of the file of clean documents including metadata.*
documents-metadata-path*: the path of the file where documents’ metadata will be stored.*
documents-index-path*: the path of the file where inverted-index will be stored.*

You can run the following command with the necessary parameters:

```
python indexing_documents.py -topics-path data/topics451-550.txt.csv -topics-documents-path
output/topics_documents -documents-metadata-path output/documents_metadata -documents-index-path
output/terms_postinglist
```

NB:

1. Open the output file in "output/documents_metadata" and "output/terms_postinglist". Now, observe the format of the data.

2. Your python interpreter should have the package "nltk" installed including WordNet.

C: Extracting features

1. **Load** documents metadata from a file
2. **Load** the **Inverted-index** from a file
3. **Extract** topics' **features**
   a. **DF** (document frequency)
   b. **IDF** (Inverse document frequency)
4. **Save** the topics and the extracted features to a file

You can use the following program to implement the above subtasks (from C_1 to C_4): "feature_extraction.py"

The next command will display the parameters that the script accepts:

```
python feature_extraction.py – h
```

This program accepts four parameters such as topics-path, documents-metadata-path, documents-index-path, and topics-features-path.

topcis-path: the path of the file containing all the topics
documents-metadata-path: the path of the file containing documents' metadata.
documents-index-path: the path of the file containing inverted-index.
topics-features-path: the path of the file where topics and its features will be saved

You can run the following command with the necessary parameters:

```
python feature_extraction.py -topics-path data/topics451-550.txt.csv -documents-metadata-path output/documents_metadata -documents-index-path output/terms_postinglist -topics-features-path output/topics_features
```

NB:

1. Open the output file in “output/topics_features.” See the format of the data.

Exercises:

1. Extract the following features from documents
   a. TF (Term frequency)
   b. NTF (Normalized term frequency)
   c. TF-IDF (Term frequency-inverse document frequency)
   d. BM25 (Best match 25 model)
   e. LM (Language model with dirichlet smoothing)
2. Extract the following features from query:
   a. CF (Corpus frequency)
   b. NCF (Normalized corpus frequency)
3. Include the document length into the feature list