EECE 5698
Assignment 1: Text Analyzer

Preparation. Create a folder on discovery named after your username under the directory `/scratch`. You can do so by logging in to discovery and typing

```
mkdir /scratch/$USER
```

Copy the directory

```
/scratch/EECE5698/Assignment1
```

to the folder you just created. You can do so by typing:

```
cp -r /scratch/EECE5698/Assignment1 /scratch/$USER/
```

Make the contents of this directory private, by typing:

```
chmod -R go-rx /scratch/$USER/Assignment1
```

The directory contains (a) a python file called `TextAnalyzer.py`, (b) a directory called `masc_500k_texts` containing the American National Corpus (ANC).[1] In this assignment, you are asked to modify the provided code `TextAnalyzer.py` and use it to analyze this corpus of data. You must:

1. Provide a report, in `pdf` format, outlining the answers of the questions below. The report should be type-written in a word processor of your choice (e.g., MS Word, Latex, etc.).
2. Provide the final code in `TextAnalyzer.py` you wrote, that implements the full functionality described below.

The report, along with your final code, should be uploaded to Blackboard.

[1] `http://anc.eestes.com/`

Question 0: Go to the directory that contains `TextAnalyzer.py` and run the following from the command prompt:

```
python TextAnalyzer.py --help
```

What does this print? What portion of the code causes this to be printed? Find the documentation of the module that offers this functionality. Use this to describe what happens at each line of code that uses a method or object defined in this module.

Question 1: The *term frequency* (`TF`) of a word in a document F is defined as:

$$\texttt{TF}(w, F) = \text{ number of times the word } w \text{ appears in document } F.$$

1(a) Modify the code in `TextAnalyzer.py` so that, when executed as follows:

```
python TextAnalyzer.py TF input output
```

the program:

- reads a text file F from argument `input`,
- computes the $\texttt{TF}(w, F)$ of each word w in F, and
- saves the result through `saveAsTextFile` at `output`.

When computing the `TF`, every word should be (a) first converted to lowercase, and (b) have all non-alphabetic characters removed. I.e.,

`Ba,Na:N_a.123` and `banana`

should both count as the same term. The final RDD stored in `output` should be of the form `(WORD,VAL)` where `WORD` is a word and `VAL` is the `TF` value, and should **not** contain the frequency of the empty string (`""`). Include the code snippet you wrote in the report, explaining what each transform does.

1(b) Use your code to compute the `TF` of all words appearing in document

```
hotel-california.txt
```

by running:

```
python TextAnalyzer.py TF \
      masc_500k_texts/written/fiction/hotel-california.txt \
      hotel-california.tf
```

Executing this creates a directory called `hotel-california.tf`. What are the contents of this directory? Add the first 5 lines of file `part-00000` in your report.

Question 2: Find the words with highest frequency. In particular:

2(a) Modify the program `TextAnalyzer.py` so that, when executed as follows:

```
python TextAnalyzer.py TOP input output
```

then the program:

- reads file `input` comprising pairs of the form `(WORD,VAL)`, where `WORD` is a string and `VAL` is a numeric value,
- finds the pairs with the highest 20 values, and
- stores the result in file `output`.

Include the code snippet you wrote in the report, explaining what each transform or action does.

2(b) Use the program to find the 20 most frequent words in document

```
hotel-california.txt
```

Report the 20 most frequent words and their `TF`.

Question 3: The term frequency of a word in a *corpus* of documents

$$C = \{F_1, F_2, \ldots, F_k\}$$

is

$$\begin{aligned} \texttt{TF}(w, C) &= \text{ number of times the word } w \text{ appears in the corpus } C \\ &= \sum_{i=1}^{k} \texttt{TF}(w, F_k) \end{aligned}$$

Use `TextAnalyzer.py` to compute the `TF` of all words in the entire corpus. **No modification of your code is necessary!** You should be able to do this by typing[2]:

```
python TextAnalyzer.py TF "masc_500k_texts/*/*" all.tf
```

Use this and the `TOP` mode of your program to find the 20 most frequent words in the entire corpus; report these words as well as their frequency.

[2] Note the quotes.

Question 4. Given a corpus of documents $C = \{F_1, F_2, \dots, F_k\}$, the *inverse document frequency* (`IDF`) of a word w is given by

$$\begin{aligned}\texttt{IDF}(w, C) &= \log \frac{\text{Number of documents in corpus } C}{\text{Number of documents in } C \text{ containing word } w} \\ &= \log \frac{|C|}{|\{F \in C : w \in F\}|}\end{aligned}$$

where $|A|$ denotes the size of set A.

4(a) Modify the program `TextAnalyzer.py` so that, when executed as follows:

```
python TextAnalyzer.py IDF input output
```

then the program:

- reads a **list** of text files `input`, representing a corpus C,
- computes the `IDF`(w, C) of every word w in the corpus C, and
- stores the resulting RDD through `saveAsTextFile` in `output`.

Include the code snippet you wrote in the report.

4(b) Use your code to compute the `IDF` of all words in the entire ANC, by running

```
python TextAnalyzer.py IDF \
      "masc_500k_texts/*/*" \
      anc.idf
```

Executing this creates a directory called `anc.idf`. Add the first 5 lines of file `part-00000` in your report.

Question 5. Given a corpus C, the TFIDF score of a word w in a document $F \in C$ is given by

$$\texttt{TFIDF}(w, F, C) = \texttt{TF}(w, F) \cdot \texttt{IDF}(w, C).$$

5(a) Modify the program `TextAnalyzer.py` so that, when executed as follows:

```
python TextAnalyzer.py TFIDF input output --idfvalues idffile
```

then the program:

- reads TF scores of a document F from `input`,
- reads IDF scores from `idffile`,
- computes the $\texttt{TFIDF}(w, F, C)$ of every word w in the document F, and
- stores the resulting RDD in `output`.

Include the code snippet you wrote in the report.

5(b) Use your code to compute the words with the top 20 TFIDF scores of file

```
hotel-california.txt
```

Include these 20 words, along with their corresponding TFIDF scores, in your report. How do these compare to the terms you computed in Question 2(b)? Which ones are more representative of the document, and why?

Question 6. The **cosine similarity** between two documents F, F' in corpus C is defined as

$$\cos(F, F') = \frac{\sum_{w \in F \cap F'} \texttt{TFIDF}(w, F, C) \cdot \texttt{TFIDF}(w, F', C)}{\sqrt{\sum_{w \in F} (\texttt{TFIDF}(w, F, C))^2 \cdot \sum_{w \in F'} (\texttt{TFIDF}(w, F', C))^2}}$$

where $F \cap F'$ indicates the set words present in both files. Note that, by definition, $\cos(F, F') = \cos(F', F)$.

6(a) Modify the program `TextAnalyzer.py` so that, when executed as follows:

```
python TextAnalyzer.py SIM input output --other otherfile
```

then the program:

- reads the `TFIDF` scores of a file F from `input`,
- reads the `TFIDF` scores of a file F' from `otherfile`,
- computes the cosine similarity $\cos(F, F')$ between the two files, and
- stores the result in `output`.

Include the code snippet you wrote in the report.

6(b) Use the code that you wrote to compute the cosine similarities between any two of following categories

`face-to-face`, `fiction`, `spam`

and report them in a table like the one below:

	`face-to-face`	`fiction`	`spam`
`face-to-face`			
`fiction`			
`spam`			

Addendum. These are a few values to help you check the correctness of your code. Word round has:

- `TF` in `hotel-california.txt`: 3
- `TF` in entire corpus: 19
- `IDF`: 3.4011973816621555
- `TFIDF` in `hotel-california.txt`: 10.203592144986466

The cosine similarity between `hotel-california.txt` and `tweets1.txt` is: 0.109149153924