CMSC 201 Section 40
Spring 2022
Project #3 - Implementing TF-IDF using Jupyter Notebooks

Due Date: Monday, May 16, 2022 before midnight

Value: 80 points

Overview:

You will build a movie review classifier that can input a movie review as a text file and determine whether the movie review is positive or negative. Your classifier will be based on a Term Frequency - Inverse Document Frequency (TF-IDF) model that you build yourself, from scratch. You will train your model on a set of movie reviews that you will be given. Once it is trained, you will test your model on a different set of movie reviews, and report on whether the reviews in question are positive or negative.

;

You will accomplish this project by building a Jupyter Notebook including both code cells and markdown cells. The code cells will be used to execute the Python code that implements your classifier. The markdown cells will be used to report on your results.

TF-IDF models

A major field of what is called “artificial intelligence” or “machine learning” is what is known as natural language processing, or NLP. In NLP, a program tries to make some sense of text written in “natural” - that is, used by humans - language such as English.

There are any number of NLP techniques that can be implemented. One powerful tool is known as “TF-IDF” which stands for “Term Frequency- Inverse Document Frequency.”

A good overview of TF-IDF can be found here:

(insert reference)

The basic idea is this: we can figure out the meaning of a text document by looking at the words that it uses most often. For any word in any document, the Term Frequency or “TF” is

the number of times the word appears in the document divided by the total number of words in the document.

Words that appear in the document a lot are important; they tell us a lot about what the document is about. If a review contains “fantastic,” “best,” “stupendous” and similar words, that tells us that the review is probably positive. If a review contains “terrible,” “disappointment,” “worst” and similar words, that tells us the review is most likely negative.

But English (and other natural languages) aren’t random. Because of grammar rules and the need for clarity, certain words appear a lot in any text, regardless of whether it’s a positive review, a negative review, or the menu at your local fast-food restaurant. Words like “a,” “the,” and “and” just naturally appear a lot in all documents. When analyzing Term Frequencies, we need to account for those words and don’t let them mess up our analysis.

If the only problems were words like “a” and “the” we could just drop those from our TF charts. But it’s not that easy. In every case, we have a document corpus - a set of relevant documents. Those documents have words that appear commonly in them. A set of movie reviews would be expected to contain a lot of occurrences of “actor” or “director,” so those words don’t tell us much about the review.

One way of addressing the problem is by using the “Inverse Document Frequency,”, or “IDF.” The document frequency, or DF, is

The number of documents that contain the word, at least once, divided by the total number of documents.

The Inverse Document Frequency is the log(1/DF).

The IDF is low for words that appear in almost every document; its highest values are achieved when a word only appears in one document in the corpus.

The TF-IDF of any given word is just the TF for a word in a document, times the IDF for that word in the set of documents.

The result of calculating TF-IDFs for a set of documents should be a two-dimensional array. The rows are the words that appear in any of the documents; the columns are the TF-IDF calculated for each word for each document; with one column per document.

(need to insert an example here)

Assignment:

Step 1:

Download Jupyter Notebook to your laptop, using either pip or Anaconda. Details are provided on a separate sheet.

Download the project 3 data files from the class Github repo to your laptop. Put them into a directory from which you can read them. There are 10 data files, and a “stopwords.txt” file.

Step 2:

Open a new Jupyter Notebook. Call it “project_3.ipynb”

Step 3:

Create a new markdown cell. In that cell, write:

- Your name
- Your section (41 or 42)
- The date
- Spring, 2022
- Project 3
- A summary of this project

Step 4:

Create a code cell. You will create a TF-IDF table as follows

Create a new empty 2D list called “tfidf” This list will ultimately have one row for each unique word you read in, in any data file. The rest of each row will be the number of times that word appeared in each file you read.

Read in one file at a time. Split each file into a list of words.Calculate the number of words in the file.

For each file read, you will create a new column in your tfidf 2D list that counts how many times each word appeared in that file. So add a new element to each row in tfidf, with an initial value of 0.

Go through this list of words, one at a time. Check to see if each word in already in your 2D list, tfidf.

If the word already appears in the list, add one to the cell that represents that row, for this file.

If the word is NOT in that list, add it. Append a new row to your tfidf list. Add a column to that new fow for each file you have already read, and set that [row][column] value to 0. That's because the word in question appeared 0 times in each file you had already read.

Then set the [row][column] entry for the current file to 1, because you have now read this word one time.

Now, go through JUST THIS COLUMN - the column for the current file - and divide each entry by the number of words in the file, which you calculated above. This gives you the actual Term Frequency - or TF - that you will need.

When you have read all data files, you have created the "tf" part of your assignment. Print out the first five rows of your table, one row per line. This should be the first five words in the first data file that you read.

Move on to Step 5.

Step 5:

Create a new markdown cell. Write a description of what you will don this next step - that is, eliminate frequently occurring "stopwords."

Create a code cell.

Eliminate stopwords. The file "stopwords.txt" contains a list of words that are very common in English text. We want to eliminate these words from our "tfidf" list because they provide no value. Read in the file "stopwords.txt" and split it into individual words. Now, one word at a time, check to see if the "stopword" is in your tfidf list; if it is, delete the entire row from your 2D list.

Again, print out the first five rows of your tfidf list; one row per line.

When finished with this, move on to Step 6.

Step 6:

Create another markdown cell to describe what you're doing in this step.

Then create a code cell. In that cell, create a two-dimension list that will contain the IDF value for each word in your tfidf list.The IDF is based on the number of files in which the word appeared. It doesn't matter how many times the word appeared; appearing one time is the same as appearing 85 times.

So: Go through each row of your tfidf list. For each row, count THE NUMBER OF CELLS with non-zero entries.

Import the math library. This is just the Python statement

```
import math
```

Then, calculate the idf of each word as

```
math.log(number_of_data_Files/number of non-zero cells you counted.)
```

Print this list. Then move on to Step 7.

Step 7

Start with another new markdown cell to describe what's happening.

You now have a 2D tfidf list (that actually contains the TF) and a separate IDF list. You need to now calculate the actual TF-IDF. To do this, move through your lists.When the words are the same, multiply each TF term by the IDF for the same word. You will now have a two dimensional list, with one row per word, and a separate column with the TF-IDF for that word in each data file.

Print out the first TEN rows of this 2D list, one row per line. Then submit the assignment.

Submitting:

The submission process for this file is as follows:

When you have your project working, upload it to gl.umbc.edu. Your Jupyter notebook will NOT run on gl because gl doesn't have the necessary graphical user interface. But you will submit on gl because that's the easiest way to to this.

After you have submitted, let Prof. Arsenault know that you are done. Prof. Arsenault will fetch all the .ipynb files from gl.umbc.edu and grade the assignments on a separate machine that exists for this purpose.

The submit command is:

```
  submit cmsc201 PROJECT3 project_3.ipynb
```

Note that the file you submit will NOT have a .py extension; it will be a .ipynb extension.

-