LING001_Final_Project

December 20, 2018

1 LING001 Final Project

The Vector Space Model for information retrieval is an interesting linguistic analysis technique. The paramount issue with language processing is representing words mathematically to do relevant computations. The Vector Space Model allows for words to be represented as vectors. Using the model, we can find the similarity between a document and a query, which will tell us the relative importance of the query to the document.

I think a good corpus to use is the Harry Potter books since they are a good length and I find them interesting.

Dataset: - SORCERER STONE - CHAMBER OF SECRETS - PRISONER OF AZKABAN
- GOBLET OF FIRE - ORDER OF THE PHOENIX
- HALF-BLOOD PRINCE - DEATHLY HALLOWS (PART 1 AND 2)

1.1 Question/Proposal

How does the cosine similarity of each of the main character's names in all of the Harry Potter books relate to the amount of screen time they have in the Harry Potter movies?

1.2 Hypothesis

If the name of the character has a higher cosine similarity, then the weight of the name in the corpus is higher, meaning the importance of the person to the plot is greater, thus, they should have more screen time in the movie.

1.3 Building Blocks

First, let's explore the building blocks that will allow us to determine the importance of a word/phrase to a corpus.

1.3.1 Term Frequency

Term frequency (TF) is the number of times a word appears in a given corpus.

For example, the term frequency of "dope" in "That was dope." is 1.

1.3.2 Inverse Document Frequency

Inverse document frequency (IDF) is a measure of the rareness of a word in a corpus. It is defined as:

2 $idf_t = log(\frac{N}{t_f})$

where N is the number of documents and t_f is the number of documents in the corpus which the term is present in.

The log function looks like:

```
In [21]: from IPython.display import HTML
         HTML("""
         <div align="middle">
         <video width="100%" autoplay loop>
               <source src="VSM.mp4" type="video/mp4">
         </video></div>""")
         #See attached video VSM.mp4 if unable to view

Out[21]: <IPython.core.display.HTML object>
```

From the above graph, we can see that when the number of documents the term is in is low, the input to the log function will be high, making the output higher.

2.0.1 TF-IDF

TF-IDF is the term freqency multiplied by the inverse document frequency. This is a measure of the uniqueness of a word in a document that exists in a corpus.

$tf - idf = tf * idf$

2.0.2 Cosine Similarity

The tf-idf values are then put into a vector that represents the word or document in a vector space. The cosine similarity is defined as:

$CosineSimilarity = \frac{a \cdot b}{\|a\|\|b\|}$ where a and b are two document or word/phrase vectors

```
In [22]: from IPython.display import HTML
         HTML("""
         <div align="middle">
         <video width="100%" autoplay loop>
               <source src="CS.mp4" type="video/mp4">
         </video></div>""")
         #See attached video CS.mp4 if unable to view

Out[22]: <IPython.core.display.HTML object>
```

The higher the cosine similarity, the more similar the vectors are because the dot product will be larger, since they are more in the same direction i.e. the projection of one vector onto the other is large. Each vector is divided by its magnitude to normalize the distance since we only care about the angle between them, so a unit vector is created in the direction of each vector. The cosine similarity is equal to $cos(\theta)$, which is shown above.

2.1 Data Exploration

Text for books was obtained from http://www.glozman.com/textpages.html.

```
In [85]: from sklearn.feature_extraction.text import TfidfVectorizer
         from sklearn.metrics.pairwise import cosine_similarity
         import nltk
         import pandas as pd

         file = open("B1.txt", 'rt')
         text = file.read()
         text=nltk.word_tokenize(text)
         print(text[0:10])
```

```
['Harry', 'Potter', 'and', 'the', 'Sorcerer', "'s", 'Stone', 'CHAPTER', 'ONE', 'THE']
```

The data looks like it is being tokenzied properly. Now, I will have to find the cosine similarity for individual queries, get the screen times of each character in each movie, and see if there is a correlation between the two.

2.2 Data Analysis

```
In [4]: corpus=[]
        books = [
            "SORCERER STONE",
            "CHAMBER OF SECRETS",
            "PRISONER OF AZKABAN",
            "GOBLET OF FIRE",
            "ORDER OF THE PHOENIX",
            "HALF-BLOOD PRINCE",
            "DEATHLY HALLOWS (PART 1 AND 2)"
        ]
        num_books = 7
        for i in range(num_books):
                file = open(f"B{i+1}.txt", 'rt')
                text = file.read()
                corpus.append(text)

        characters=[
            "HARRY POTTER",
            "RON WEASLEY",
            "HERMIONE GRANGER",
            "VOLDEMORT",
            "DUMBLEDORE",
            "HAGRID",
            "DRACO MALFOY",
            "SIRIUS BLACK",
            "NEVILLE LONGBOTTOM"
```

```
]

cosine_similarities={}
for character in characters:
    corpus.append(character)
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(corpus)
    character_sim=[]
    for i in range(num_books):
        similarity = float(cosine_similarity(tfidf_matrix[i],tfidf_matrix[-1]))*1000
        sim_rounded = round(similarity,3)
        character_sim.append(sim_rounded)
    cosine_similarities[character]=character_sim

pd.DataFrame(cosine_similarities, index=books)
```

Out[4]:

	HARRY POTTER	RON WEASLEY	HERMIONE GRANGER	\
SORCERER STONE	141.351	47.602	29.664	
CHAMBER OF SECRETS	159.369	78.557	31.224	
PRISONER OF AZKABAN	149.567	68.196	44.165	
GOBLET OF FIRE	103.896	91.440	29.137	
ORDER OF THE PHOENIX	128.195	63.199	39.236	
HALF-BLOOD PRINCE	130.245	49.585	32.546	
DEATHLY HALLOWS (PART 1 AND 2)	124.315	60.461	54.717	

	VOLDEMORT	DUMBLEDORE	HAGRID	DRACO MALFOY	\
SORCERER STONE	5.436	22.853	53.403	13.619	
CHAMBER OF SECRETS	3.247	20.357	21.093	23.424	
PRISONER OF AZKABAN	3.532	11.281	27.362	16.343	
GOBLET OF FIRE	6.507	3.149	1.459	8.282	
ORDER OF THE PHOENIX	5.911	28.657	16.230	6.455	
HALF-BLOOD PRINCE	15.609	66.981	15.092	20.805	
DEATHLY HALLOWS (PART 1 AND 2)	16.536	33.336	11.902	1.443	

	SIRIUS BLACK	NEVILLE LONGBOTTOM
SORCERER STONE	5.867	12.972
CHAMBER OF SECRETS	5.478	3.831
PRISONER OF AZKABAN	31.253	11.076
GOBLET OF FIRE	10.485	1.214
ORDER OF THE PHOENIX	24.405	7.192
HALF-BLOOD PRINCE	7.490	3.825
DEATHLY HALLOWS (PART 1 AND 2)	7.959	0.435

*the cosine similarity was multiplied by 1000 for easier viewing and distinction between the values. All calculations past this point will be based off the multiplied cosine similarity values.

2.3 Get screen times

Using IMDB's movie database (https://www.imdb.com/list/ls027460372/), we can get the screen times for each of the actors in the films.

```
In [5]: #screen times are in order of the books as shown above
        screen_time={"HARRY POTTER":[72.75,83.75,74.5,63.25,61.75, 67.00,round(58+8/60,2)],
                     "RON WEASLEY":[28.25,38.25, 21.25,20.5,21.0,21.75,round(30+23/60,2)],
                     "HERMIONE GRANGER":[23.25,15.5,34.75,16.5,23,20,36],
                     "VOLDEMORT":[2,6.75,0,6.5,2.25,4.25,7.75],
                     "DUMBLEDORE":[9.75,10.75,6.25,14.25,7.5,22.25,3.25],
                     "HAGRID":[16.5,8.25,5.25,3.75,2.75,4,round(2+38/60,2)],
                     "DRACO MALFOY":[4.25,7,4,2.25,1.25,8.25,2.33],
                     "SIRIUS BLACK":[0,0,11.5,1.25,7.5,0,0],
                     "NEVILLE LONGBOTTOM":[3.25,1.25,3,4.25,9,1.5,round(7/60,2)]
                    }
        pd.DataFrame(screen_time, index=books)
```

Out[5]:

	HARRY POTTER	RON WEASLEY	HERMIONE GRANGER	\
SORCERER STONE	72.75	28.25	23.25	
CHAMBER OF SECRETS	83.75	38.25	15.50	
PRISONER OF AZKABAN	74.50	21.25	34.75	
GOBLET OF FIRE	63.25	20.50	16.50	
ORDER OF THE PHOENIX	61.75	21.00	23.00	
HALF-BLOOD PRINCE	67.00	21.75	20.00	
DEATHLY HALLOWS (PART 1 AND 2)	58.13	30.38	36.00	

	VOLDEMORT	DUMBLEDORE	HAGRID	DRACO MALFOY	\
SORCERER STONE	2.00	9.75	16.50	4.25	
CHAMBER OF SECRETS	6.75	10.75	8.25	7.00	
PRISONER OF AZKABAN	0.00	6.25	5.25	4.00	
GOBLET OF FIRE	6.50	14.25	3.75	2.25	
ORDER OF THE PHOENIX	2.25	7.50	2.75	1.25	
HALF-BLOOD PRINCE	4.25	22.25	4.00	8.25	
DEATHLY HALLOWS (PART 1 AND 2)	7.75	3.25	2.63	2.33	

	SIRIUS BLACK	NEVILLE LONGBOTTOM
SORCERER STONE	0.00	3.25
CHAMBER OF SECRETS	0.00	1.25
PRISONER OF AZKABAN	11.50	3.00
GOBLET OF FIRE	1.25	4.25
ORDER OF THE PHOENIX	7.50	9.00
HALF-BLOOD PRINCE	0.00	1.50
DEATHLY HALLOWS (PART 1 AND 2)	0.00	0.12

2.4 Get average cosine similarity and screen time for each character

```
In [84]: average_similarities = {}
         average_screen_times = {}
```