LING001 Final Project Data Exploration

December 1, 2018

1 LING001 Final Project Data Exploration

Sadly, I couldn't find a good dataset for sarcasm, and the models needed seemed a little out of my scope, so I have changed my focus.

However, I think using the Vector Space Model for information retrieval would also be an interesting analysis technique that will be more in my wheel house. A main 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 cosine similarity between a corpus and a query that will tell us the relative importance of the query in the corpus.

I think a good corpus to use is the Harry Potter books.

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)

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?

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.

1.1 Building Blocks

First, let's explore the building blocks that will allow us to calculate cosine similarity

1.1.1 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.

1.1.2 Inverse Document Frequency

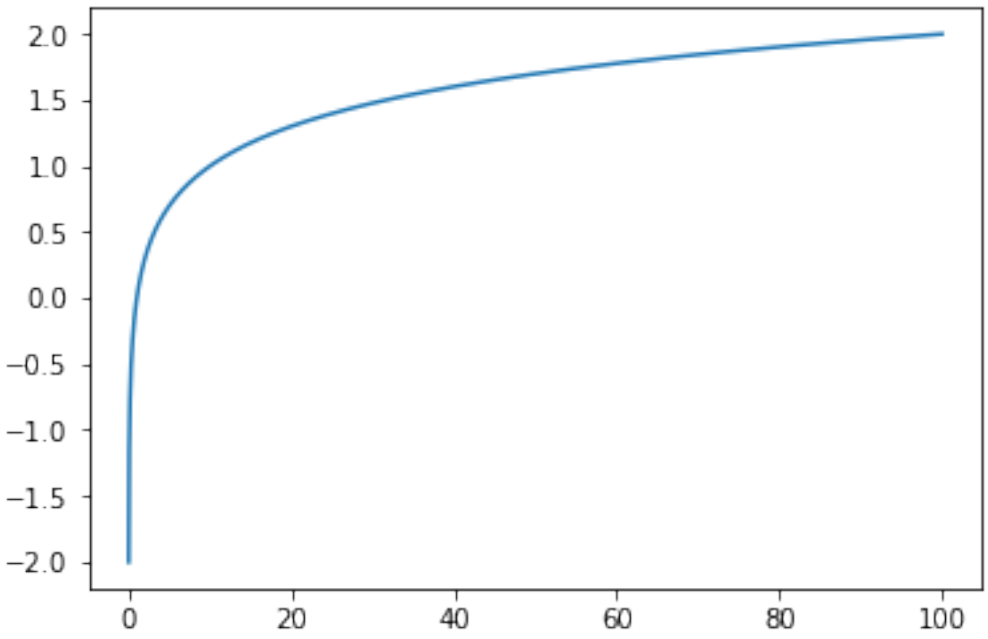
Inverse document frequency (IDF) is a measure of the rareness of a word in a corpus. It is defined as:

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 it is present in.

The log function looks like:

```
In [10]: import warnings
         warnings.filterwarnings('ignore')
         import matplotlib.pyplot as plt
         import numpy as np
         x = np.arange(0.0, 100, 0.01)
         y = np.log10(x)
         plt.plot(x,y)
         plt.show()
```



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 corpus.

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:

$CosineSimilarity = \frac{a \cdot b}{\|a\|\|b\|}$

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

2.1 Preliminary Data Analysis

```
In [36]: 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])
         tfidf_vectorizer = TfidfVectorizer()
         tfidf_matrix = tfidf_vectorizer.fit_transform(text)
         #text=nltk.word_tokenize("Harry Potter")
         #potter = tfidf_vectorizer.fit_transform(text)
         cosine_similarity(tfidf_matrix[0:2], tfidf_matrix)
```

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

```
Out[36]: array([[1., 0., 0., ..., 0., 0., 0.],
                [0., 1., 0., ..., 0., 0., 0.]])
```

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 chracter in each movie, and see if there is a correlation between the two.

2.2 To be continued...