Patrick Austin

CS 491: Social Networks: Lab 4

3 April, 2018

1. First I converted my methodology report, along with three of the main related words cited in it, into plaintext form (without references). These related works were [An Analysis of Friend Circles of Facebook Users](http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7365914) by Erdin et al., [SuperNova: Super-peers Based Architecture for Decentralized Online Social Networks](https://arxiv.org/pdf/1105.0074.pdf) by Sharma and Datta, and [The Growth of Diaspora – A Decentralized Online Social Network in the Wild](https://pdfs.semanticscholar.org/f58f/c6179286fd2aa5786a43d0f6879bfa0b62f2.pdf) by Bielenberg et al.

Then I followed the tutorial for measuring similarity between texts in Python located [here](https://sites.temple.edu/tudsc/2017/03/30/measuring-similarity-between-texts-in-python/). This allowed me to vectorize the documents and convert them into tf-idf scaled form. Then the cosine similarity measure was used to evaluate similarity between vectorized documents.

Below I show source code for the Python program omitting the full plaintext of the documents, due to the prohibitive length of the strings. Then I show a screenshot of the Python program with those strings included. Full source code is available upon request, but to keep this submission readable as a single .pdf it seemed prudent not to include dozens of pages of text from the papers the program operates on.

# -\*- coding: utf-8 -\*-

import nltk, string, numpy

d1= “My Methodology Report”

d2= “An Analysis Of Friend Circles of Facebook Users”

d3= “SuperNova: Super-peers Based Architecture for Decentralized Online Social Networks”

d4= “The Growth of Diaspora - A Decentralized Online Social Network in the Wild”

documents = [d1, d2, d3, d4]

lemmer = nltk.stem.WordNetLemmatizer()

def LemTokens(tokens):

return [lemmer.lemmatize(token) for token in tokens]

remove\_punct\_dict = dict((ord(punct), None) for punct in string.punctuation)

def LemNormalize(text):

return LemTokens(nltk.word\_tokenize(text.lower().translate(remove\_punct\_dict)))

from sklearn.feature\_extraction.text import CountVectorizer

LemVectorizer = CountVectorizer(tokenizer=LemNormalize, stop\_words='english')

LemVectorizer.fit\_transform(documents)

tf\_matrix = LemVectorizer.transform(documents).toarray()

from sklearn.feature\_extraction.text import TfidfTransformer

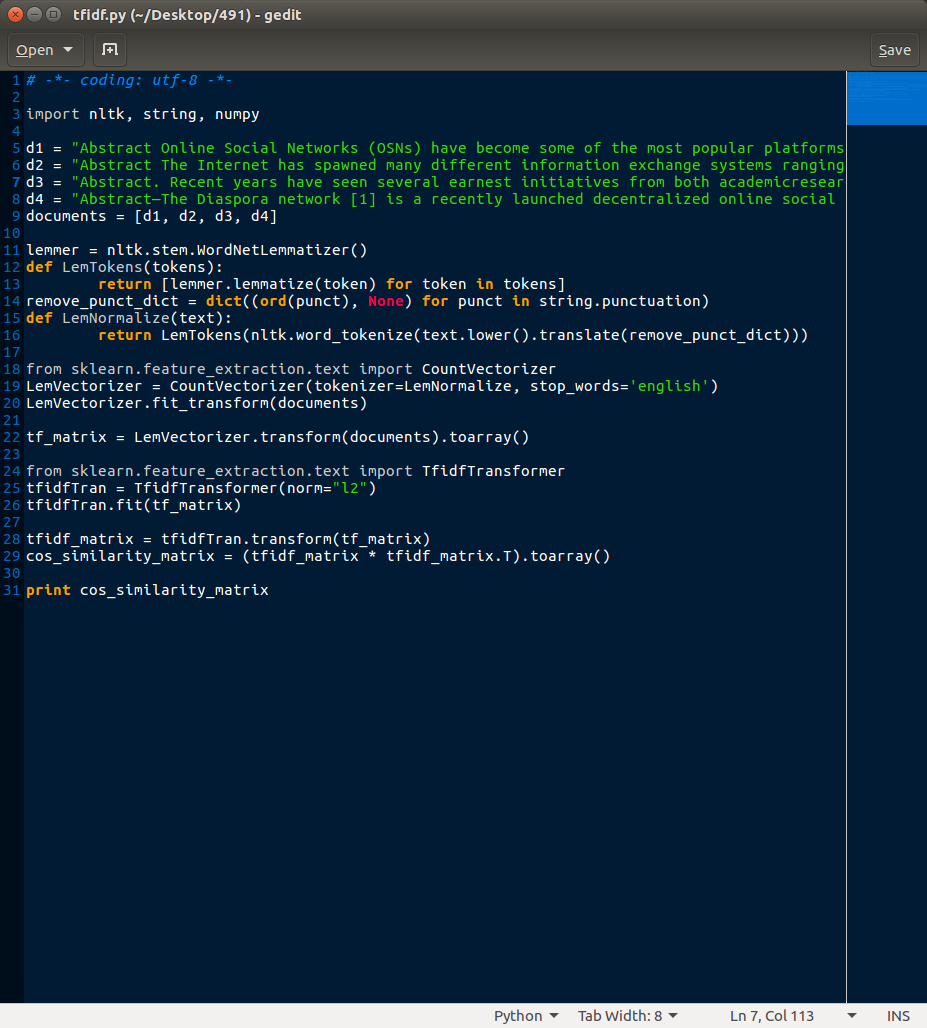
tfidfTran = TfidfTransformer(norm="l2")

tfidfTran.fit(tf\_matrix)

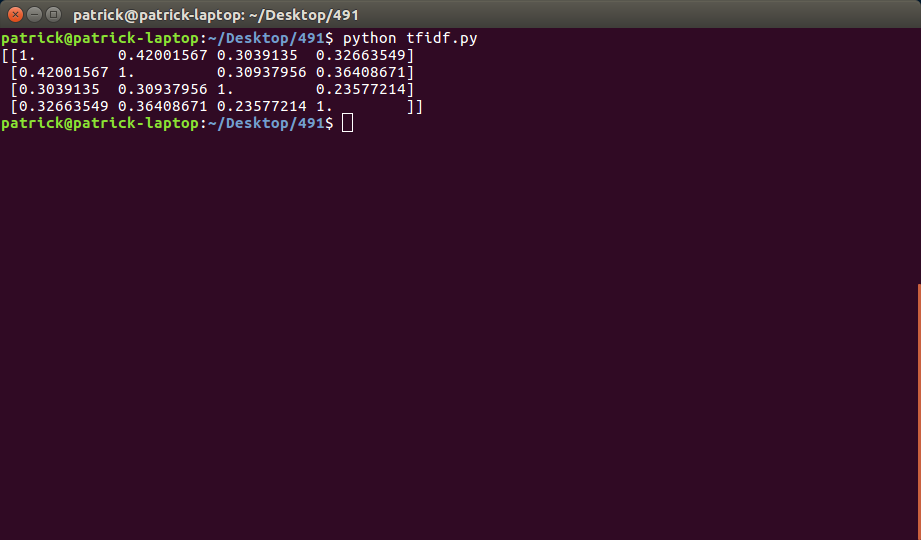
tfidf\_matrix = tfidfTran.transform(tf\_matrix)

cos\_similarity\_matrix = (tfidf\_matrix \* tfidf\_matrix.T).toarray()

print cos\_similarity\_matrix



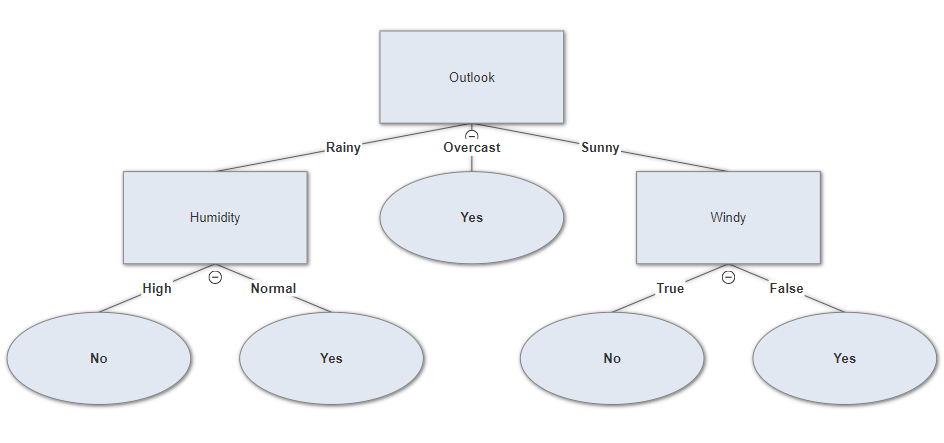
This program successfully generated cosine similarity between the four documents, with an output like so:



To briefly interpret these results, we first note that each document has a similarity of 1 with itself. In terms of my methodology report, it is most similar to the Erdin Facebook friend circles study, and least similar to the SuperNova study, with the Diaspora study somewhere in-between.

This seems to me like a good result, considering that my report and the friend circles study have the most overlap by far in terms of material being covered. The SuperNova and Diaspora studies, while related to my general topic, cover some substantially different material.

2. Here is the required decision tree:



When the outlook is rainy, the temperature is mild, humidity is normal and it is not windy, this decision tree predicts that the decision should be “Yes”.

3. When the outlook is rainy, the temperature is mild, humidity is normal and it is not windy, we can use a naive Bayes classifier to predict the decision by comparing

P( PG=Y | Outlook=R, Temp=M, Humidity=N, Windy=F ) and

P( PG=N | Outlook=R, Temp=M, Humidity=N, Windy=F ) to see which probability is higher.

P( PG=Y | Outlook=R, Temp=M, Humidity=N, Windy=F ) =

P( Outlook=R, Temp=M, Humidity=N, Windy=F | PG=Y ) \* P(PG=Y) /

P( Outlook=R, Temp=M, Humidity=N, Windy=F ) =

P( Outlook=R | PG=Y ) \* P( Temp=M | PG=Y ) \* P( Humidity=N | PG=Y ) \* P( Windy=F | PG=Y ) \* P( PG=Y ) / P( Outlook=R, Temp=M, Humidity=N, Windy=F ) =

(2/9) \* (4/9) \* (6/9) \* (6/9) \* (9/14) / P( Outlook=R, Temp=M, Humidity=N, Windy=F ) =

0.0282 / P( Outlook=R, Temp=M, Humidity=N, Windy=F )

Likewise, P( PG=N | Outlook=R, Temp=M, Humidity=N, Windy=F ) =

P( Outlook=R | PG=N ) \* P( Temp=M | PG=N ) \* P( Humidity=N | PG=N ) \* P( Windy=F | PG=N ) \* P( PG=N ) / P( Outlook=R, Temp=M, Humidity=N, Windy=F ) =

(3/5) \* (2/5) \* (1/5) \* (2/5) \* (5/14) / P( Outlook=R, Temp=M, Humidity=N, Windy=F ) =

0.00686 / P( Outlook=R, Temp=M, Humidity=N, Windy=F )

Since .0282 > .00686, the naive Bayes classifier predicts the decision should be “Yes”.

4. See the table below. Where data instances had equal similarity, I ranked them by data instance value, with lower instance values being considered closer neighbors to tie-break.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Data Instance | Outlook | Temp. | Humidity | Windy | Similarity | Label | K | Prediction |
| 8 | 1 | 1 | 0 | 1 | 3 | N | 1 | N |
| 9 | 1 | 0 | 1 | 1 | 3 | Y | 2 | ? |
| 10 | 0 | 1 | 1 | 1 | 3 | Y | 3 | Y |
| 11 | 1 | 1 | 1 | 0 | 3 | Y | 4 | Y |
| 1 | 1 | 0 | 0 | 1 | 2 | N | 5 | Y |
| 4 | 0 | 1 | 0 | 1 | 2 | Y | 6 | Y |
| 5 | 0 | 0 | 1 | 1 | 2 | Y | 7 | Y |
| 13 | 0 | 0 | 1 | 1 | 2 | Y | 8 | Y |
| 2 | 1 | 0 | 0 | 0 | 1 | N | 9 | Y |
| 3 | 0 | 0 | 0 | 1 | 1 | Y | 10 | Y |

5.8.6. Given two documents that have been vectorized into the numerical incidence of terms in each document, we can create a matrix where if a given term is incident in both documents, we assign a positive weight, and if the word does not occur in both documents, we assign a weight of zero.

In such a scheme a variety of similarity measures could be used. If weights are binary, we could use simple matching, Dice’s Coefficient, Jaccard’s Coefficient, the Cosine Coefficient, or the Overlap Coefficient as potential measures of similarity (among others). We could also experiment with a weighting scheme such as tf-idf to attempt to correct the matrix values of highly co-incident and highly infrequently incident terms to more meaningful or appropriate values.

5.8.9. There are 2 ^ (2n) possible distinct decision trees given n Boolean attributes for a binary class. This is the number of distinct truth tables with 2n rows. Each decision tree then represents a potential function of the attributes given distinct attribute values.

5.8.10. Zero entropy means maximum homogeneity and minimum heterogeneity. For example, if a split at a node in a decision tree results in 10 instances of the + class and 0 instances of the - class, zero entropy was achieved. In terms of information theory, this reflects the maximum possible information gain from the split.