CS 505 Report

Assignment 2: Vector Space Models

Name : Rajat Kumar Tripathi ID : U89360176

*I've discussed a couple of things with my classmate Shubhangi Jain

Interesting Finds

1)To talk about word similarity performance in the three methods, I've picked up sample words from the corpus to test them out. To see the contrast in the results I have picked three words ,two of them being similar and other being different. In our case I have picked up the words 'faith','religion','motorcycle'. I'm using cosine similarity in this case,the results are as follows :

- Word2Vec similarity:

```
[rajat@feanor code]$ python main.py
similarity of religion and faith: 0.5669254
similarity of religion and motorcycle 0.20745686
[rajat@feanor code]$
```

- Newsgroup similarity (without tf idf):

```
[rajat@feanor code]$ python main.py
similarity between religion and faith 0.7643610574375849
similarity between religion and motorcycle 0.00397787532828545
[rajat@feanor code]$
```

- Word Content similarity(without PPMI) :

```
similarity betweeen faith and religion 0.9478535955813493
similarity between faith and motorcycle 0.4665688735322332
[rajat@feanor code]$
```

Our results in all these three cases are as expected, words which are likely to occur together have higher similarity. Without tf-idf , the term document matrix has an unfair advantage as a lot of common occurring words skew the numbers. <u>But in the case of the words we picked I think Word2Vec gives us the best results.</u>

2)
With tf_idf_weighing we use inverse frequencies to eliminate the common words that occur in every document. We get the following result when we set tf_idf as True(using Jaccard):

```
[rajat@feanor code]$ python main.py
similarity between religion and faith 1.0
similarity between religion and motorcycle 0.25
```

We get similarity as 1, because I think religion and faith might be occurring in all the newsgroups together. How we interpret this result depends on our use case, in a certain semantic sense we can equate the words in many contexts but they definitely are not the same words in many cases so 1 would not be an accurate representation.

When we talk about ppmi_weighing , we do see the words maintaining the same relations , but on a smaller scale

```
similarity between faith and religion 0.18057557429069007
similarity between faith and bike 0.07169187901645874
[rajat@feanor code]$
```

But we have to be more careful when we use PPMI as it does not work well without a big corpus.

Different Similarity Measures :
Amongst cosine,Jaccard and Dice I found cosine similarity to work the best.It works well with all the three representations we use. Jaccard and Dice work with set wise operations while cosine is based on vectors. With word2vec the Jaccard/Dice similarity won't work because we can’t do that with vectors only sets.
If we want to use embeddings then we can use cosine similarity.

Biases:
I feel we cannot really identify biases just using frequencies and a bag of words like approach. Because it is possible the very opposing words might occur with good and bad adjectives and just frequencies here can’t reflect the biases. Still I came across a couple of examples , where I saw certain biases:

```
[rajat@feanor code]$ python main.py
similarity of guns and democrat: 0.17975242
similarity of guns and republican 0.31146035
```

Like here we see republicans and guns occur twice as much as with democrat, because of different stances in gun rights issues.