**CSCI 544- HOMEWORK 3**

**1. Dataset Generation:**The dataset was generated using the same set of code I used in Homework   
  
I prepared 2 different datasets:  
1. amazon\_balanced\_df: This is a preprocessed data. We also applied data cleaning code as in Homework 1. This will be used to run Simple Models (TF-IDF)

2. amazon\_df: This dataset is not clean, and this will be used for different model testing including word embeddings.

**2. Word Embedding:**  
**a.** We pretrained the model. Most common functions that I chose to use to check similarities are:  
1. Cosine similarity  
2. Similarity  
3. Most\_similar  
  
The semantic similarity of all 3 examples along with their probabilities is:  
  
1. ([word2vec\_model['queen']], [word2vec\_model['king']])): 65%  
  
2. ("good"):  
  
[('great', 0.7291510105133057),

('bad', 0.7190051078796387),

('terrific', 0.6889115571975708),

('decent', 0.6837348341941833),

('nice', 0.6836092472076416),

('excellent', 0.644292950630188),

('fantastic', 0.6407778263092041),

('better', 0.6120728850364685),

('solid', 0.5806034803390503),  
 ('lousy', 0.576420247554779)]

3. (w1="daughter", w2="sister"): 78%

**b.** Check the semantic similarities for the same two examples in part (a). What do you conclude from comparing vectors generated by yourself and the pretrained model? Which of the Word2Vec models seems to encode semantic similarities between words better?

I created my own Word2Vec model and dataset by using the amazon\_balanced\_df(preprocessed).  
  
The semantic similarity of all 3 examples along with their probabilities in my model is:

1. ([word2vec\_model['queen']], [word2vec\_model['king']])): 40%  
  
2. ("good"):  
  
[('great', 0.7492009997367859),

('decent', 0.6979432106018066),

('nice', 0.6569323539733887),

('fantastic', 0.600950300693512),

('ok', 0.5753679275512695),

('bad', 0.5533730387687683),

('okay', 0.5282017588615417),

('alright', 0.5122868418693542),

('awesome', 0.5109883546829224),

('high', 0.4818119406700134)]

3. (w1="daughter", w2="sister"): 88%

1. 'queen’, 'king’ gave good similarity on pretrained model as they were dominant there but they were not at all common in our dataset

2. ‘good' produced almost similar similarity as they were common in both our and pretrained dataset

3. daughter and sister gave much higher similarity on our corpus as they are where common on dataset whereas same cannot be said for pretrained model(google)

3. **Simple Models:**  
  
**(a)Report your accuracy values:**

* 1. Perceptron

TFIDF: 0.62  
 W2V: 0.56

* 1. SVM  
      TFIDF: 0.68  
      W2V: 0.62

**(b)What do you conclude from comparing performances for the models trained using the two different feature types (TF-IDF and your trained Word2Vec Features)?**

The TFIDF model outperformed the W2V model in both simple models (Perceptron and SVM). This could be attributed to the fact that the W2V model was trained on the Google corpus (pre-trained model), whereas the TFIDF model was trained on our cleaned dataset.

**4. Feedforward Neural Network:  
  
 (a)Report accuracy values on the testing split for your MLP.**

The accuracy was 64.5.

**(b) What do you conclude by comparing accuracy values you obtain with those obtained in the “’Simple Models” section.**

The Feed Forward Network achieved an accuracy of 51.98% by concatenating the first ten vectors of each review with an 80-20% split. However, since the concatenation only considers the first ten words, the simple feed forward model performed better. Additionally, the MLP model outperformed both the Perceptron and SVM models in comparison to the simple models.

1. **Recurrent Neural Networks**   
   **(a)Train a simple RNN for sentiment analysis. What do you conclude by comparing accuracy values you obtain with those obtained with feedforward neural network models.**   
   The accuracy was 40.37.

**(b)Repeat part (a) by considering a gated recurrent unit cell.**   
The accuracy was 60.69%.

**(c)Repeat part (a) by considering an LSTM unit cell.**The accuracy was 61.14%.

**(d)What do you conclude by comparing accuracy values you obtain by GRU, LSTM, and simple RNN.**

Regarding RNNs, the GRU model and LSTM model almost gave similar accuracy. Additionally, for the less number of training epochs, the LSTM model performed better than the GRU model.