Github Repo : https://github.com/anjaliverma96/avw4127_msia414_2019/tree/homework_two

Question 1. I experimented with the following model parameters :

- '**sg**' which indicates whether the model is Skip-gram or CBOW - changed from 0 to 1
- '**window'** which indicates the maximum distance between the target word and its neighboring word - changed from 3 to 5

The 10 words compared on the basis of similarity outputted by the model are
"Popular", "technology", "question", "information", "letter", "digit", "data", "prohibition", "faith"

1. I then compared the euclidean similarity of these words to a particular word 'computer' to see how the similarity changes as the words change from more relevant to less relevant. As expected, across all variations of the model, words, "technology", "information", "digit", "data" are more similar to the word 'computer' than the words "prohibition", "faith","question", "letter" (as seen from the euclidean and cosine similarities)

2. I then found the top 5 most similar words to each of the 10 words mentioned above according to a) Euclidean similarity b) Cosine similarity
    - The CBOW model gave more accurate results in terms of the top 5 similar words outputted for each word
      Example: for the word technology, CBOW gave words like 'telecommunication', 'development', 'marketable' whereas Skip-gram gave the words 'telecommunication', 'accommodate' and 'spur'.
    - As the window size increased from 3 to 5, the model produced less relevant words i.e. the smaller window size gave top 5 results that were more similar to the given word
      Example : for the word faith, model with window 3 resulted in the words 'salvation','rule','surely','truth','obey' whereas the model with window 5 resulted in the words 'salvation', 'evil', 'share', 'lie','truth'

Question 2 :

Criterion of Comparison	**Distributed representations of words and phrases and their compositionality** **'Word2vec'**	**BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding** **'BERT'**
Ability to capture polysemy differences i.e. context dependency Example: The sentences *"The man was accused of robbing a bank."* and *"The man went fishing by the bank of the river." refer to two different meanings of the word 'bank'*	Word embeddings generated by Word2vec models are **context independent** i.e. the model outputs just one vector representation of a word **regardless** of the different meanings the word may have depending on its position in a sentence.	BERT **produces context informed** word representations that are dynamically informed by the words around them to capture the meanings relative to their position in a sentence.
	Under word2vec, "bank" has a single, fixed representation.	Under BERT the word embedding for "bank" would be different for each sentence
Input/Output representations : **Ability to deal with 'out of vocabulary' words**	The model takes words as input and outputs word embeddings. It assigns 'out of vocabulary' words to an overloaded unknown vocabulary token	The model represents input as subwords and learns embeddings for subwords. Input representation for a token is constructed by summing the corresponding token, segment, and position embeddings.It splits 'out of vocabulary' words into subword tokens that retain some of the contextual meaning of the original word and hence generates an approximate vector for the original word by averaging subword embedding vectors
Learning Model Details Model Architecture; Transfer Learning	Word2vec is a two-layer neural net that outputs a vocabulary in which each item has a vector attached to it, which can be fed into a deep-learning net or simply queried to detect relationships between words, for downstream tasks. It too can benefit from transfer learning.	BERT makes use of Transformer, an attention mechanism that includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Instead of starting from scratch to train and build a model, BERT deploys transfer learning, **pre-trains** an unsupervised language model on a large corpus of data such as Wikipedia articles and then **fine-tunes** the pre-trained model on downstream tasks like entity recognition, classification, question answering, etc.
Training Corpus size	Word2vec does not necessarily require a large training corpus. However, accuracy and model performance increases overall as the number of words used increases. At the same time, training embeddings over huge corpora is computationally expensive because the input is typically sequentially processed and weights are synchronously updated.	Fine-tuning pre-trained weights allows the use of a much smaller training dataset than would be required for a model trained from scratch. BERT uses a minimal vocabulary. Once the model is pre-trained, fine tuning is computationally inexpensive.

Number of Google scholar citations	15358	1894
Year Published	2013	2018