Assignment – 1 Report

In this assignment, we focus on Word2Vec embeddings using the Gensim library, examining different training approaches and evaluating their performance. We preprocess a sizable text corpus, compare two sets of learned word embeddings, and visualize them using a custom set of 20 words. We also construct a dataset with similarity scores and assess the embeddings' quality using the Pearson correlation coefficient. Finally, we explore word similarity by finding the most similar words for a selected set, including those trained by us and the widely-used pre-trained Google News embeddings.

The corpus that is used in doing this assignment is the “Brown Corpus” which is a corpus compiled at the Brown University. The preprocessing steps that have been made use in doing this assignment are Lowercasing the words in the corpus i.e. Lowercasing involves converting all the characters in a text to their lowercase form and the Lemmatization of words in the corpus i.e. reducing words to their base or dictionary form. Since the corpus is readily available in a tokenized and segmented format, it does not have to be undergo the sentence segmentation and tokenization during the preprocessing stage. There are a total of 57340 sentences in the Brown Corpus which are a selection of current American English and the words are drawn from a wide variety of sources.

In this assignment, task 1, the two different ways that were used to obtain two different sets of word embeddings differ in terms of the model that was used i.e. one word embedding makes use of the SkipGram model and the other makes use of the CBOW model. The other difference is that the SkipGram model uses the epoch count of 10 and the CBOW model uses the epoch count of 5. This assignment makes use of these two methods as to demonstrate the approach of the SkipGram and CBOW models and the impact of different epoch values on these models.

Task-2 of this assignment results in the following visualization graphs:

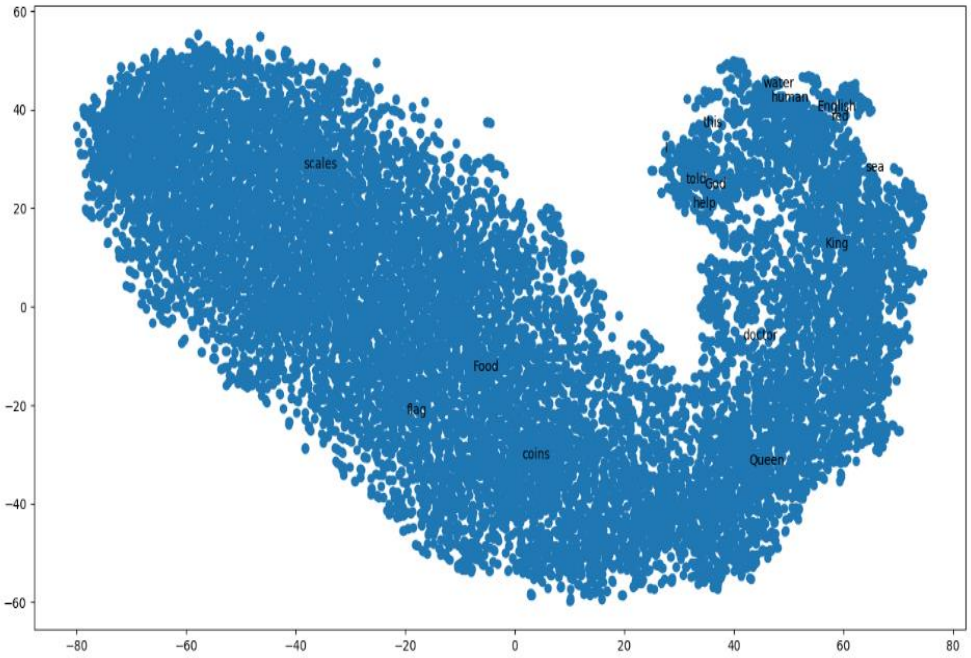

Fig 1: Graph visualization obtained after applying the CBOW model

The above graph shows the visualization of a set of 20 words, "I", "this", "red", "flag", "King", "Queen", "Math", "God", "English", "Food", "water", "eraser", "coins", "bookmark", "scales", "doctor", "help", "human", "told", "sea". This visualization involves representing words as nodes in a graph, where the edges between nodes represent semantic or contextual relationships between words based on the assigned dense vector representations to words in a continuous vector space. The words with the closest contextual relationship are: told, God and help. The word with the least contextual relationship with the words in the list is “scales”.

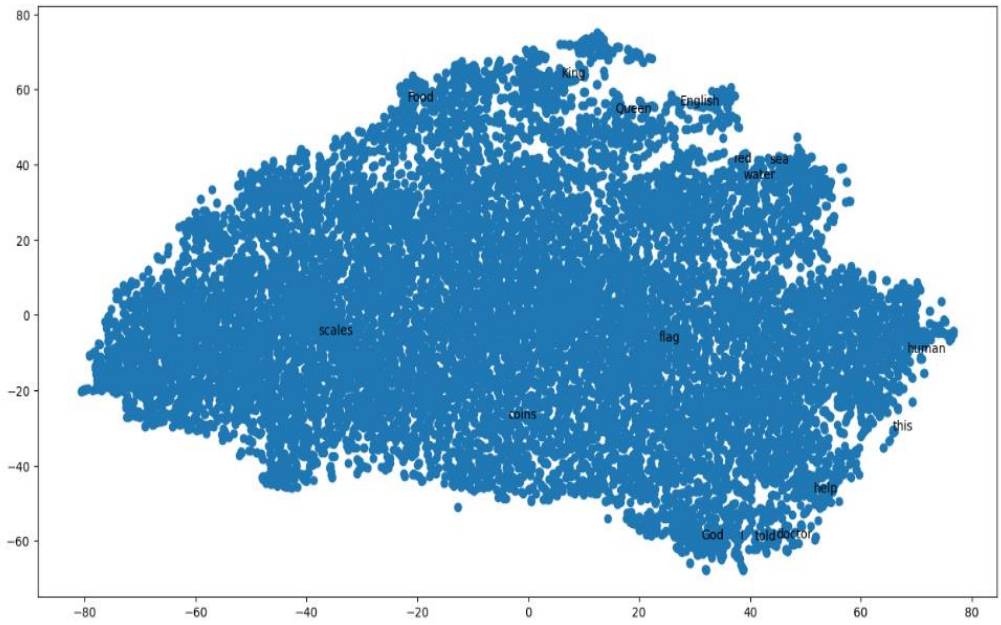

Fig 2: Graph visualization obtained after applying the SkipGram model

The above graph shows the visualization of a set of 20 words, "I", "this", "red", "flag", "King", "Queen", "Math", "God", "English", "Food", "water", "eraser", "coins", "bookmark", "scales", "doctor", "help", "human", "told", "sea". This visualization also involves representing words as nodes in a graph, where the edges between nodes represent semantic or contextual relationships between words. But here the surrounding context of a word is visualized. The words with the closest contextual relationship are: red, sea, water. The word with the least contextual relationship with the words in the list is “scales”.

Task-3 of this assignment has the following results:

Word Embedding	Pearson correlation coefficient
SkipGram embedding	-0.34190613616445764
CBOW embedding	-0.43207089694664996
Pre-trained Google News embeddings	0.35056250576058884

Table 1: Table showing the pearson correlation coefficient with respect to the word embeddings evaluated.

The results provide insights into Pearson correlation coefficients for various types of word embeddings, which measure how linearly related these embeddings are to some reference data. The coefficients range from -1 (strong negative correlation) to 1 (strong positive correlation), with 0 indicating no linear relationship.

In summary, for SkipGram embeddings, the Pearson correlation coefficient is approximately -0.342, suggesting a weak negative linear relationship with the reference data. CBOW embeddings have a Pearson correlation coefficient of about -0.432, indicating a similar weak negative linear relationship with the reference data.

In contrast, pre-trained Google News embeddings show a Pearson correlation coefficient of approximately 0.351, implying a weak positive linear relationship with the reference data.

Task-4 of this assignment has the following results:

```
>>> #the five words I am using are the computer,pizza,ocean,rain,hat
>>> #first finding the most similar five words using pre-trained Google News embeddings
>>> import gensim.downloader as api
>>> wv = api.load("word2vec-google-news-300")
>>> list = wv.most_similar("computer", topn=5)
>>> print(list)
[('computers', 0.7979379892349243), ('laptop', 0.6640493273735046), ('laptop_computer', 0.6548868417739868), ('Computer', 0.647333562374115), ('com_puter', 0.60820800065994
>>> list = wv.most_similar("pizza", topn=5)
>>> print(list)
[('pizzas', 0.7863470315933228), ('Domino_pizza', 0.7342829704284668), ('Pizza', 0.6988078355789185), ('pepperoni_pizza', 0.6902607083320618), ('sandwich', 0.68404018878936
>>> list = wv.most_similar("ocean", topn=5)
>>> print(list)
[('sea', 0.7643541693687439), ('oceans', 0.7482994198799133), ('Pacific_Ocean', 0.7037094831466675), ('Atlantic_Ocean', 0.6659377217292786), ('oceanic', 0.6610181927680969)
>>> list = wv.most_similar("rain", topn=5)
>>> print(list)
[('heavy_rain', 0.8421464562416077), ('downpour', 0.796761691570282), ('rains', 0.7827130556106567), ('torrential_rain', 0.7578904628753662), ('Rain', 0.7476006150245667)]
>>> list = wv.most_similar("hat", topn=5)
>>> print(list)
[('hats', 0.7639798521995544), ('straw_hat', 0.6072703003883362), ('cowboy_hat', 0.5864576101303101), ('Wearing_spangled', 0.5848298072814941), ('fedora', 0.583099484443664
```

Fig 3: Results showing the top 5 most similar words using the pre-trained Google News embeddings.

Here's what the results indicate for each target word:

"computer": The top 5 similar words include "computers," "laptop," "laptop_computer," "Computer," and "com_puter." The similarity scores are relatively high, suggesting that these words are closely related to "computer" in the context of the training data.

"pizza": The top 5 similar words include "pizzas," "Domino_pizza," "Pizza," "pepperoni_pizza," and "sandwich." Again, the similarity scores are high, indicating that these words are strongly associated with "pizza."

"ocean": The top 5 similar words include "sea," "oceans," "Pacific_Ocean," "Atlantic_Ocean," and "oceanic." These words are related to "ocean," and their similarity scores suggest a high degree of association.

"rain": The top 5 similar words include "heavy_rain," "downpour," "rains," "torrential_rain," and "Rain." These words are strongly related to "rain," with high similarity scores indicating close association.

"hat":

The top 5 similar words include "hats," "straw_hat," "cowboy_hat," "Wearing_spangled," and "fedora." These words are related to "hat," with "hats" being the most similar. The similarity scores are relatively high.

```
>>> #second finding the most similar five words using skipgram embeddings
>>> from gensim.models import KeyedVectors
>>> wv = KeyedVectors.load_word2vec_format("skipgram_embeddings.txt", binary=False)
>>> list = wv.most_similar("computer", topn=5)
>>> print(list)
[('digital', 0.8765760660171509), ('automation', 0.870654821395874), ('generate', 0.8688483238220215), ('packaged', 0.8658013939857483), ('injecting', 0.8652753233909607)]
>>> list = wv.most_similar("pizza", topn=5)
>>> print(list)
[('dine', 0.9492061734199524), ('mauch', 0.9457874894142151), ("probl'y", 0.9438025951385498), ('hating', 0.942836403846740 7), ('hotdog', 0.9427337646484375)]
>>> list = wv.most_similar("ocean", topn=5)
>>> print(list)
[('basement', 0.8379862308502197), ('drainage', 0.8359599113464355), ('keel', 0.8224723935127258), ('stretching', 0.8180164098739624), ('masonry', 0.8179476857185364)]
>>> list = wv.most_similar("rain", topn=5)
>>> print(list)
[('mud', 0.8179415464401245), ('squall', 0.8155068159103394), ('wind', 0.8146893978118896), ('mist', 0.8075737953186035), ('storm', 0.8071972727775574)]
>>> list = wv.most_similar("hat", topn=5)
>>> print(list)
[('shirt', 0.8361958265304565), ('cap', 0.8165348172187805), ('pistol', 0.8153178095817566), ('boot', 0.811457097530365), ('forehead', 0.8098595142364502)]
```

Fig 4: Results showing the top 5 most similar words using the SkipGram embeddings.

Here are the results for each target word:

"computer": The top 5 similar words include "digital," "automation," "generate," "packaged," and "injecting." These words are considered similar to "computer" based on the embeddings learned from your specific training data. However, the results seem to include words that might not be semantically related to "computer" in a general sense.

"pizza": The top 5 similar words include "dine," "mauch," "probl'y," "hating," and "hotdog." These results are somewhat unusual, as they do not appear to be strongly related to "pizza."

"ocean": The top 5 similar words include "basement," "drainage," "keel," "stretching," and "masonry." The words in the results seem unrelated to "ocean."

"rain": The top 5 similar words include "mud," "squall," "wind," "mist," and "storm." These words are related to weather conditions and natural phenomena, which is somewhat relevant to "rain."

"hat": The top 5 similar words include "shirt," "cap," "pistol," "boot," and "forehead."The results are quite different from what you might expect when looking for words similar to "hat."

```
>>> #third finding the most similar five words using cbow embeddings
>>> wv = KeyedVectors.load_word2vec_format("cbow_embeddings.txt", binary=False)
>>> list = wv.most_similar("computer", topn=5)
>>> print(list)
[('generating', 0.9132937788963318), ('installation', 0.9041382074356079), ('irregular', 0.896868109703064), ('reinforced', 0.8963931202888489), ('cleaning', 0.88852155208
>>> list = wv.most_similar("pizza", topn=5)
>>> print(list)
[('pro-trujillo', 0.9089798927307129), ('rag', 0.8928787708282471), ('soberly', 0.8883790373802185), ('vivian', 0.8805361390113831), ('get-together', 0.879514932632446 3)]
>>> list = wv.most_similar("ocean", topn=5)
>>> print(list)
[('plug', 0.9035034775733948), ('chain', 0.8944990634918213), ('lamp', 0.8909759521484375), ('slope', 0.8890172243118286), ('blowing', 0.8852494359016418)]
>>> list = wv.most_similar("rain", topn=5)
>>> print(list)
[('storm', 0.8726685047149658), ('cool', 0.8587278127670288), ('bottle', 0.8579422831535339), ('grass', 0.8534609079360962), ('sun', 0.8533768653869629)]
>>> list = wv.most_similar("hat", topn=5)
>>> print(list)
[('lip', 0.9334967136383057), ('mouth', 0.899549126625061), ('hair', 0.8939570784568787), ('knee', 0.8880735635757446), ('beard', 0.8876423239707947)]
>>>
```

Fig 5: Results showing the top 5 most similar words using the CBOW embeddings.

Here are the results for each target word:

"computer": The top 5 similar words include "generating," "installation," "irregular," "reinforced," and "cleaning." These words do not seem semantically related to "computer" in the typical sense, and the results may indicate that the word embeddings are not capturing the expected semantic relationships effectively.

"pizza": The top 5 similar words include "pro-trujillo," "rag," "soberly," "vivian," and "get-together." These results do not appear to be strongly related to "pizza."

"ocean": The top 5 similar words include "plug," "chain," "lamp," "slope," and "blowing." The words in the results seem unrelated to "ocean."

"rain": The top 5 similar words include "storm," "cool," "bottle," "grass," and "sun." These words are not strongly related to "rain" in the sense of weather-related precipitation.

"hat": The top 5 similar words include "lip," "mouth," "hair," "knee," and "beard." The results are different from what you might expect when looking for words similar to "hat."