ASSIGNMENT 4

AIM : Write a program that can give you the country by its capital .Write a function that takes in three words, and the embeddings dictionary .Your task is to find the capital cities .

THEORY :

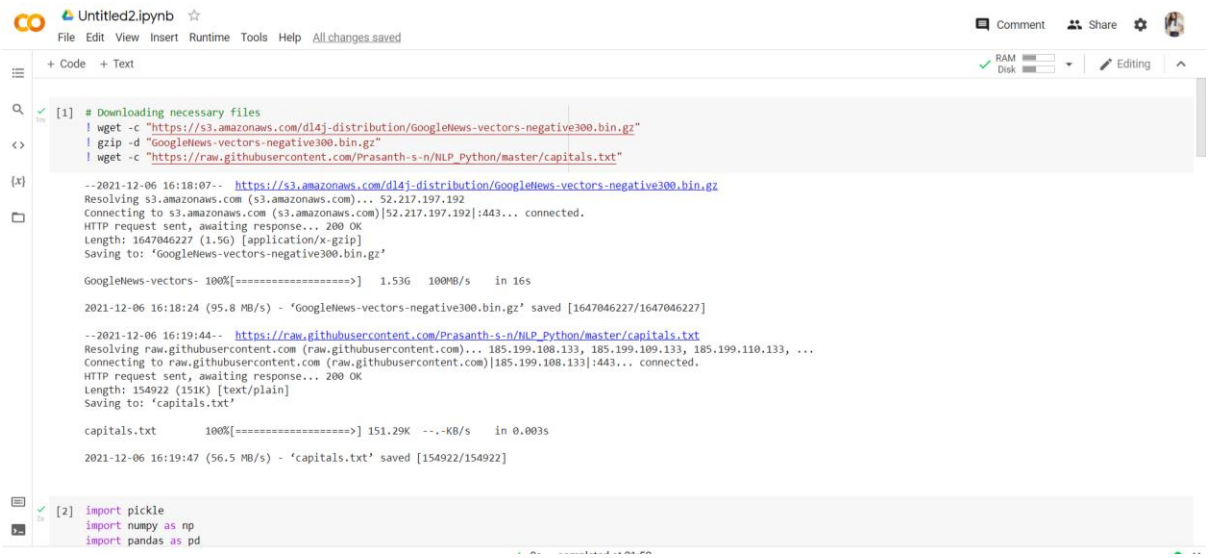
Word Embeddings:

It is the collective name for a set of language modeling and feature learning techniques in NLP where words or phrases from the vocabulary are mapped to vectors of real numbers.

Machine learning and deep learning algorithms generally deal with numeric data. So, for converting text into numbers, BagofWords technique has been developed to extract numeric features from text. It uses the concept of frequency distribution of words to find the number of times each word appeared in the text which is also known as the vectorization. It converts a text into features by creating a matrix of occurrences of words in a document within the corpus. Thus, each document is represented as a word-count vector of the size of length of vocabulary in the corpus. However, this model results in sparse matrices along with lack of capturing meaningful relationships in the text.

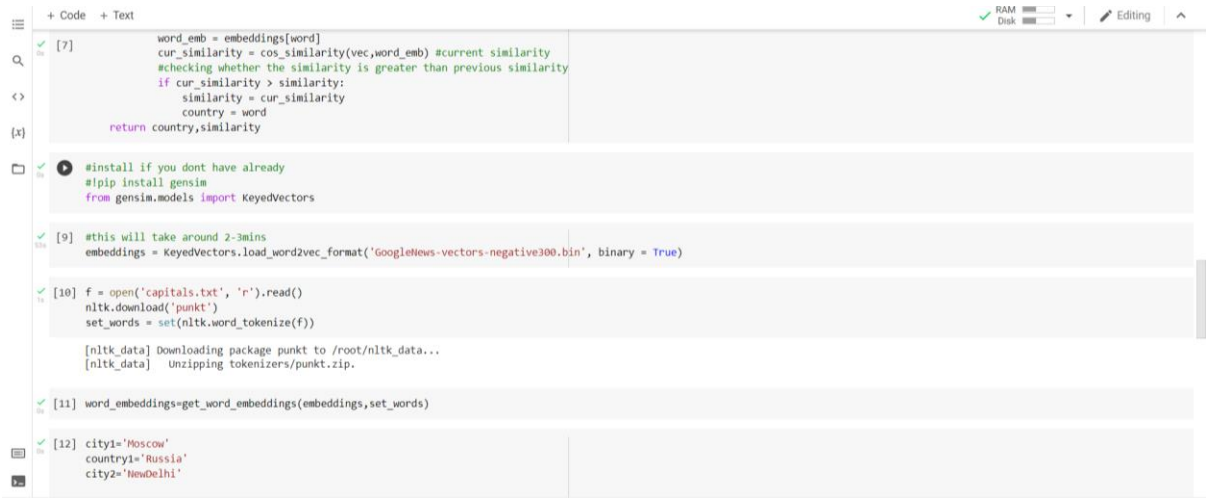
Word embedding is a technique which solves the above two problems. Using this method, each word in a language is represented as a real-valued vector in a lower-dimensional space such that semantically similar vectors are placed close to each other.

CODE:


Untitled2.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
[1] # Downloading necessary files
! wget -c "https://s3.amazonaws.com/dl4j-distribution/GoogleNews-vectors-negative300.bin.gz"
! gzip -d "GoogleNews-vectors-negative300.bin.gz"
! wget -c "https://raw.githubusercontent.com/Prasanth-s-n/NLP_Python/master/capitals.txt"
--2021-12-06 16:18:07-- https://s3.amazonaws.com/dl4j-distribution/GoogleNews-vectors-negative300.bin.gz
Resolving s3.amazonaws.com (s3.amazonaws.com)... 52.217.197.192
Connecting to s3.amazonaws.com (s3.amazonaws.com)|52.217.197.192|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1647046227 (1.5G) [application/x-gzip]
Saving to: 'GoogleNews-vectors-negative300.bin.gz'
GoogleNews-vectors- 100%[==========>] 1.53G 100MB/s in 16s
2021-12-06 16:18:24 (95.8 MB/s) - 'GoogleNews-vectors-negative300.bin.gz' saved [1647046227/1647046227]
--2021-12-06 16:19:44-- https://raw.githubusercontent.com/Prasanth-s-n/NLP_Python/master/capitals.txt
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 154922 (151K) [text/plain]
Saving to: 'capitals.txt'
capitals.txt 100%[==========>] 151.29K --.-KB/s in 0.003s
2021-12-06 16:19:47 (56.5 MB/s) - 'capitals.txt' saved [154922/154922]
[2] import pickle
import numpy as np
import pandas as pd


+ Code + Text
RAM
Disk
Editing
[2] import pickle
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import nltk
#install if you dont have already
#!pip install gensim
from gensim.models import KeyedVectors
[3] def get_vectors(embeddings,words):
X = np.zeros((1, 300))
for word in words:
english = word
eng_emb = embeddings[english]
X = np.row_stack((X, eng_emb))
X = X[1:,:]
return X
[4] def get_word_embeddings(embeddings,set_words,complete=False):
word_embeddings = {}
for word in embeddings.vocab:
if word in set_words:
word_embeddings[word] = embeddings[word]
if complete:
word_embeddings[word] = embeddings[word]
return word_embeddings


[5] def cos_similarity(W1,W2):
num = np.dot(W1,W2)
det = np.linalg.norm(W1)*np.linalg.norm(W2)
return (num/det)
[6] def euclidean(W1,W2):
return np.linalg.norm(np.subtract(W1,W2))
we can use this to find 4th word related to 3rd word similar to how 1 and 2 related.
#for now since my subset mostly hav countries and cities, i have named this function as predict country
def predict_country(city1, country1, city2, embeddings):
group = set((city1, country1, city2))
city1_emb = embeddings[city1]
country1_emb = embeddings[country1]
city2_emb = embeddings[city2]
vec = city2_emb-city1_emb+country1_emb
# Initialize the similarity to -1 (it will be replaced by a similarities that are closer to +1)
similarity = -1
# initialize country to an empty string
country = ''
# loop through all words in the embeddings dictionary
for word in embeddings.keys():
if word not in group:



```
            word_emb = embeddings[word]
            cur_similarity = cos_similarity(vec,word_emb) #current similarity
            #checking whether the similarity is greater than previous similarity
            if cur_similarity > similarity:
                similarity = cur_similarity
                country = word
    return country,similarity
```

```
#install if you dont have already
#!pip install gensim
from gensim.models import KeyedVectors
```

```
#this will take around 2-3mins
embeddings = KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin', binary = True)
```

```
f = open('capitals.txt', 'r').read()
nltk.download('punkt')
set_words = set(nltk.word_tokenize(f))
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

```
word_embeddings=get_word_embeddings(embeddings,set_words)
```

```
city1='Moscow'
country1='Russia'
city2='NewDelhi'
```

```
# Predicting country from city
country2,similarity=predict_country(city1,country1,city2,word_embeddings)
print("Country is {} with cosine similarity being {}".format(country2,similarity))


# Predicting city from country
country3='England'
city3,similarity = predict_country(country1,city1,country3,word_embeddings)
print("City is {} with cosine similarity being {}".format(city3,similarity))
```

```
Country is India with cosine similarity being 0.5113393664360046
City is London with cosine similarity being 0.6056334972381592
```

```
def get_accuracy(word_embeddings, data):

    num_correct = 0
    for i, row in data.iterrows():
        city1 = row[0]
        country1 = row[1]
        city2 =  row[2]
        country2 = row[3]
        predicted_country2, _ = predict_country(city1,country1,city2,word_embeddings)

        if predicted_country2 == country2:
            num_correct += 1

    m = len(data)
    accuracy = num_correct/m
    return accuracy
```

```
    m = len(data)
    accuracy = num_correct/m
    return accuracy
```

```
data = pd.read_csv('capitals.txt', delimiter=' ')
data.columns = ['city1', 'country1', 'city2', 'country2']
acc=get_accuracy(word_embeddings,data)
```

```
#print({acc:.2f})
print("Accuracy is {:.2f}".format(acc))
```

```
Accuracy is 0.92
```

CONCLUSION: Successfully implemented country prediction by its capital using word embedding