IMPLEMENTATION OF WORD2VEC

Introduction:-

The Word2vec algorithm takes a text corpus as an input and produces the word vectors as output. The algorithm first creates a vocabulary from the training text data and then learns vector representations of the words. The vector space can include hundreds of dimensions, with each unique word in the sample corpus being assigned a corresponding vector in the space. In addition, words that share similar contexts in the corpus are placed in close proximity to one another in the space. The result is an H2O Word2vec model that can be exported as a binary model or as a MOJO.

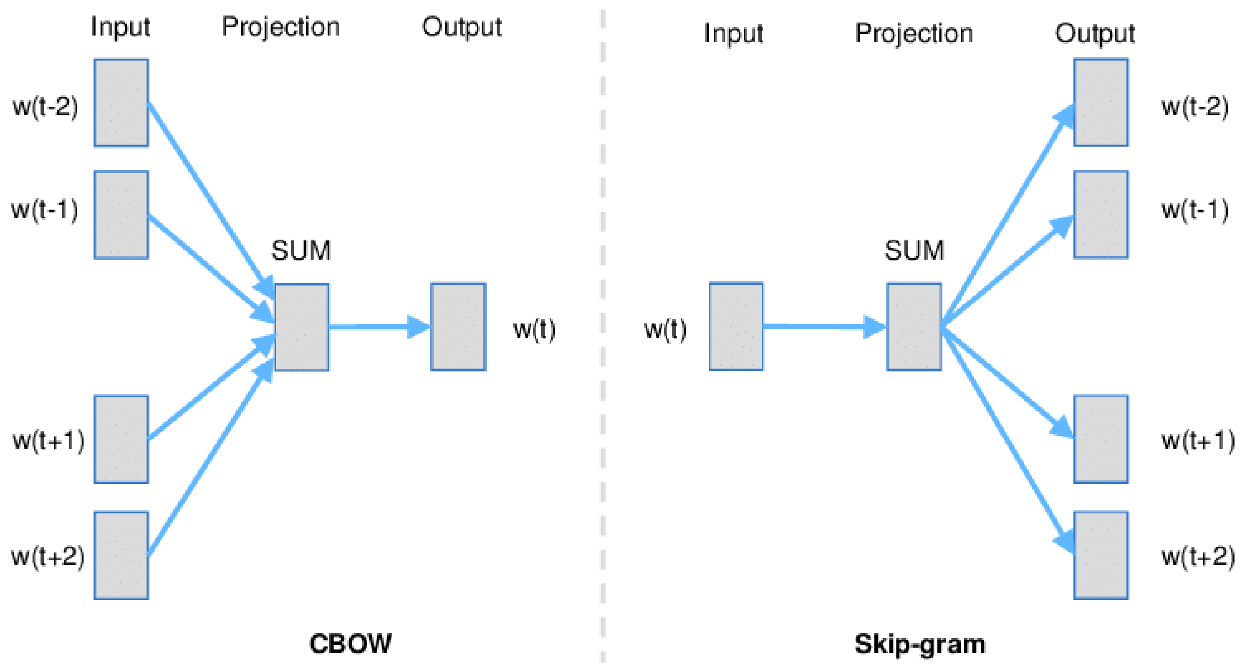
Word2vec can utilize either of two model architectures to produce a distributed representation of words:

- Continuous Bag of Words (CBOW)

- Continuous Skip-Gram

Continuous Bag of Words (CBOW): - In the continuous bag-of-words architecture, the model predicts the current word from a window of surrounding context words. The order of context words does not influence prediction (bag-of-words assumption).

Continuous Skip-Gram: - In the continuous skip-gram architecture, the model uses the current word to predict the surrounding window of context words. The skip-gram architecture weighs nearby context words more heavily than more distant context words.



<u>Gensim Python library and its installation:-</u>

Gensim is a Python library for topic modelling, document indexing and similarity retrieval with large corpora. Target audience is the natural language processing (NLP) and information retrieval (IR) community.

• All algorithms are memory-independent w.r.t. the corpus size (can process input larger than RAM, streamed, out-of-core),

• Intuitive interface

Easy to plug in your own input corpus/datastream (trivial streaming API)

Easy to extend with other Vector Space algorithms (trivial transformation API)

The simple way to install gensim is:

pip install -U genism

If you have instead downloaded and unzipped the source tar.gz package,

python setup.py test

python setup.py install

The Gensim library and other necessary libraries are included as follows –

import os

import pandas as pd

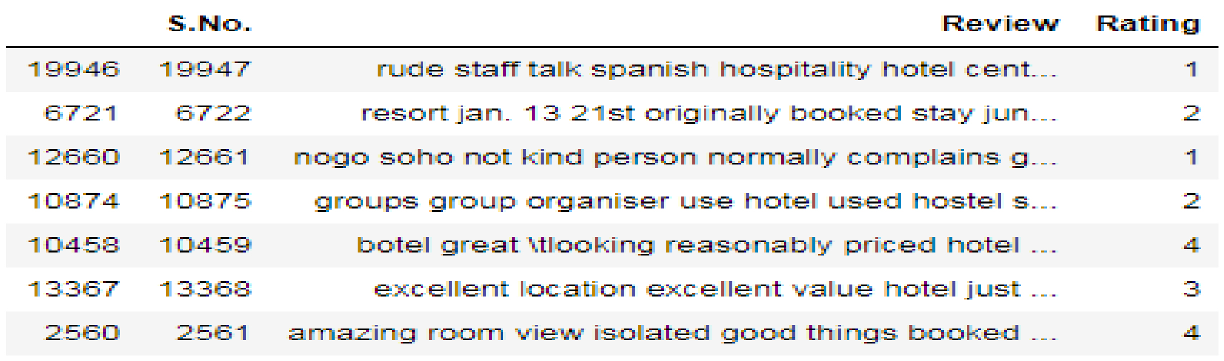
import nltk

import gensim

from gensim.models.word2vec import Word2Vec

from nltk.tokenize import RegexpTokenizer

The dataset which we are going to use for our model is TripAdvisor Hotel Review Dataset. This review dataset contains features such as 'Review' and 'Rating.



	S.No.	Review	Rating
19946	19947	rude staff talk spanish hospitality hotel cent...	1
6721	6722	resort jan. 13 21st originally booked stay jun...	2
12660	12661	nogo soho not kind person normally complains g...	1
10874	10875	groups group organiser use hotel used hostel s...	2
10458	10459	botel great \tlooking reasonably priced hotel ...	4
13367	13368	excellent location excellent value hotel just ...	3
2560	2561	amazing room view isolated good things booked ...	4

Data Preprocessing:

Genism word2vec requires that a format of ‘list of lists’ for training where every document is contained in a list and every list contains lists of tokens of that document. At first, we need to generate a format of ‘list of lists’ for training the make model word embedding.

To be more specific, each make model is contained in a list and every list contains lists of features of that make model.

df = pd.read_csv('testdata.csv', encoding = "ISO-8859-1");

train = df["Review"].values

tokenizer = RegexpTokenizer(r'\w+')

trainvec = [nltk.word_tokenize(Review) for Review in train[:5000]]

Genism word2vec Model Training:

Now we will train our genism model with processed data.

model = Word2Vec(trainvec,min_count=1, size=30, seed=123)

The hyperparameters of this model are as follows:

- size: The number of dimensions of the embeddings and the default is 100.

- window: The maximum distance between a target word and words around the target word. The default window is 5.

- min_count: The minimum count of words to consider when training the model; words with occurrence less than this count will be ignored. The default for min_count is 5.

- workers: The number of partitions during training and the default workers is 3.

- sg: The training algorithm, either CBOW(0) or skip gram(1). The default training algorithm is CBOW.

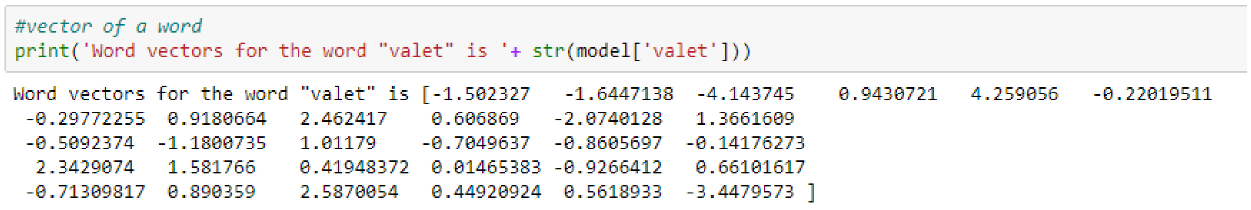
The trained model can be saved into a file which we can use it without training the model again and again.

model.save('reviews.model')

The saved model can be loaded into our project as follows:

model = Word2Vec.load('reviews.model')

After training the word2vec model, we can obtain the word embedding directly from the training model as follows –



```
#vector of a word
print('Word vectors for the word "valet" is '+ str(model['valet']))
```

```
Word vectors for the word "valet" is [-1.502327   -1.6447138  -4.143745    0.9430721   4.259056   -0.22019511
 -0.29772255  0.9180664   2.462417    0.606869   -2.0740128   1.3661609
 -0.5092374  -1.1800735   1.01179    -0.7049637  -0.8605697  -0.14176273
  2.3429074   1.581766    0.41948372  0.01465383 -0.9266412   0.66101617
 -0.71309817  0.890359    2.5870054   0.44920924  0.5618933  -3.4479573 ]
```

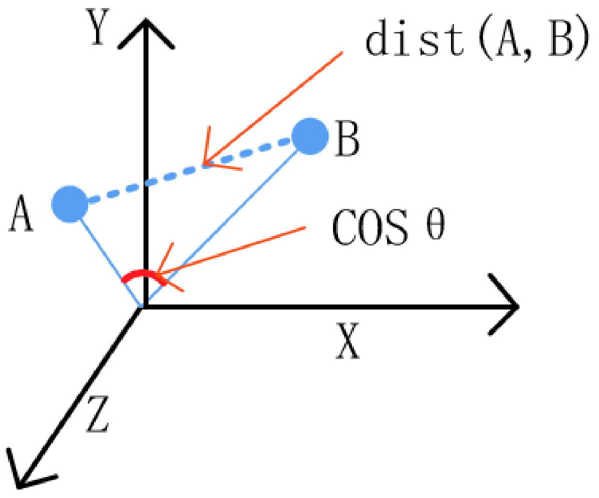
Cosine Similarity: -

Cosine similarity is used to get similarity between two words. Cosine distance is nothing but getting distance between two vectors in n dimension space. Distance represent how words are related to each other.

Cosine similarity = 1 - cosine distance.

```
#Cosine Similarity
print('Cosine similarity between "desk" and "suite" is ' + str(model.wv.similarity('desk','suite')))
```

```
Cosine similarity between "desk" and "suite" is 0.31599447
```



Euclidean Distance:

In 'n'-Dimensional Space, Euclidean distance is a measure of the true straight line distance between two points in Euclidean space

```
#Euclidean Distance
print('Euclidean Distance between "suite","room" is '+ str(model.similarity('suite', 'room')))
```

```
Euclidean Distance between "suite","room" is 0.7690108
```

N closest words of a given word:

```
# 'n' closest words for a given word
n=input('Enter number here: ')
print('Similar words for "suite" is '+ str(model.similar_by_word('suite', topn=int(n), restrict_vocab=None)))
```

```
Enter number here: 4
Similar words for "suite" is [('junior', 0.8522325754165649), ('deluxe', 0.8464635610580444), ('upgraded', 0.8410863876342773),
('apartment', 0.8095237612724304)]
```