Intro to NLP

Assignment 3 Report

:- Hitesh Goel
2020115003

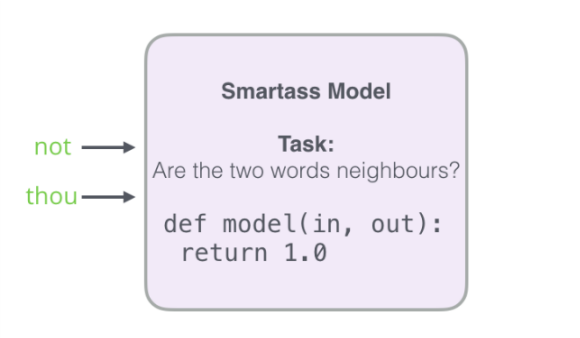
2.1 Theory

Ans 1. The concept of negative sampling was introduced to improve the skip-gram version of training the word2vec model in terms of both accuracy and efficiency. The traditional skip-gram method involves predicting the context words using the centre word for a given window size (usually taken to be two). This task is usually modelled as inputting the centre word and the context word and outputting the probability that the two words are neighbours. The output should be one if they are neighbours and zero if they are not. However, while creating the dataset, they only input positive samples, that is, words which were neighbours, to train the model. The dataset was of the form:

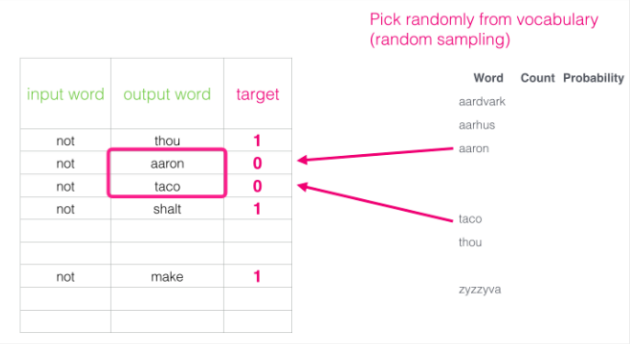
input word	target word
not	thou
not	shalt
not	make
not	a
make	shalt
make	not
make	a
make	machine

input word	output word	target
not	thou	1
not	shalt	1
not	make	1
not	a	1
make	shalt	1
make	not	1
make	a	1
make	machine	1

However, the problem with feeding a model such inputs is that it could start outputting one for every input to maximise its training accuracy without actually learning any parameters.



This is obviously not ideal. To improve on this, it was decided to generate some amount of negative samples, which will be sampled from a noise distribution for each positive sample that is input to the model. The noise distribution is usually defined as a unigram distribution raised to the power of 3/4, which biases the sampling towards less frequent words. The probability for a positive sample should be one, and for a negative sample, it should be zero. Such an input would help the model learn the parameters for the embedding better.



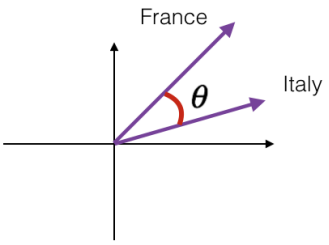
By using negative sampling instead of the normal skip-gram algorithm, word2vec training can be approximated more efficiently. Instead of computing probabilities for the entire vocabulary for each training sample, the model can be trained only for a subset of the positive and negative samples.

Ans 2. Semantic similarity refers to the degree to which two words have similar meanings or are used in similar contexts. It is a fundamental concept in natural language processing and is used in a variety of tasks, such as information retrieval, text classification, and machine translation. Word embeddings, which represent words as dense vectors in a high-dimensional space, have become a popular method for measuring semantic similarity.

As I have mentioned, word embeddings are vector representations of words. So, to calculate semantic similarity between the words, we can look to find the relative distance between two word embeddings using various distance measures, and that will give us an approximate of the similarity between the two words.

The distance measures that can be used are:

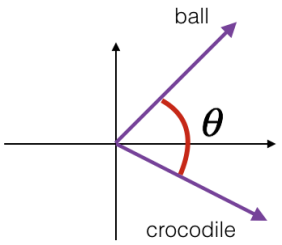
1. Cosine Similarity: Cosine similarity measures the cosine of the angle between two vectors in the embedding space. It is a widely used metric for measuring semantic similarity and can be easily computed using the dot product of the two word vectors. Given two word vectors, a and b, their cosine similarity is defined as: (a.b)/(||a|| ||b||). The lower the angle, the greater the cosine similarity and the greater the semantic similarity.



France and Italy are quite similar

θ is close to 0°

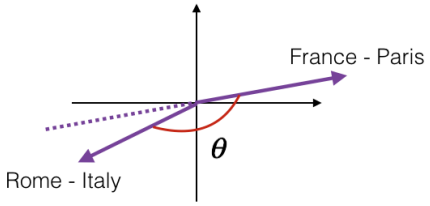
$\cos(\theta) \approx 1$



ball and crocodile are not similar

θ is close to 90°

$\cos(\theta) \approx 0$

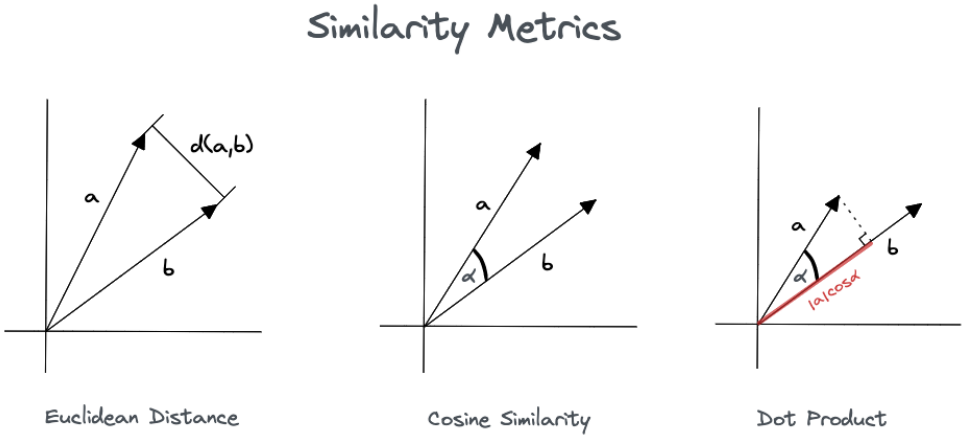


the two vectors are similar but opposite
the first one encodes (city - country)
while the second one encodes (country - city)

θ is close to 180°

$\cos(\theta) \approx -1$

2. Euclidean distance: Euclidean distance measures the straight-line distance between two vectors in the embedding space. The lower the distance, the greater the semantic similarity.

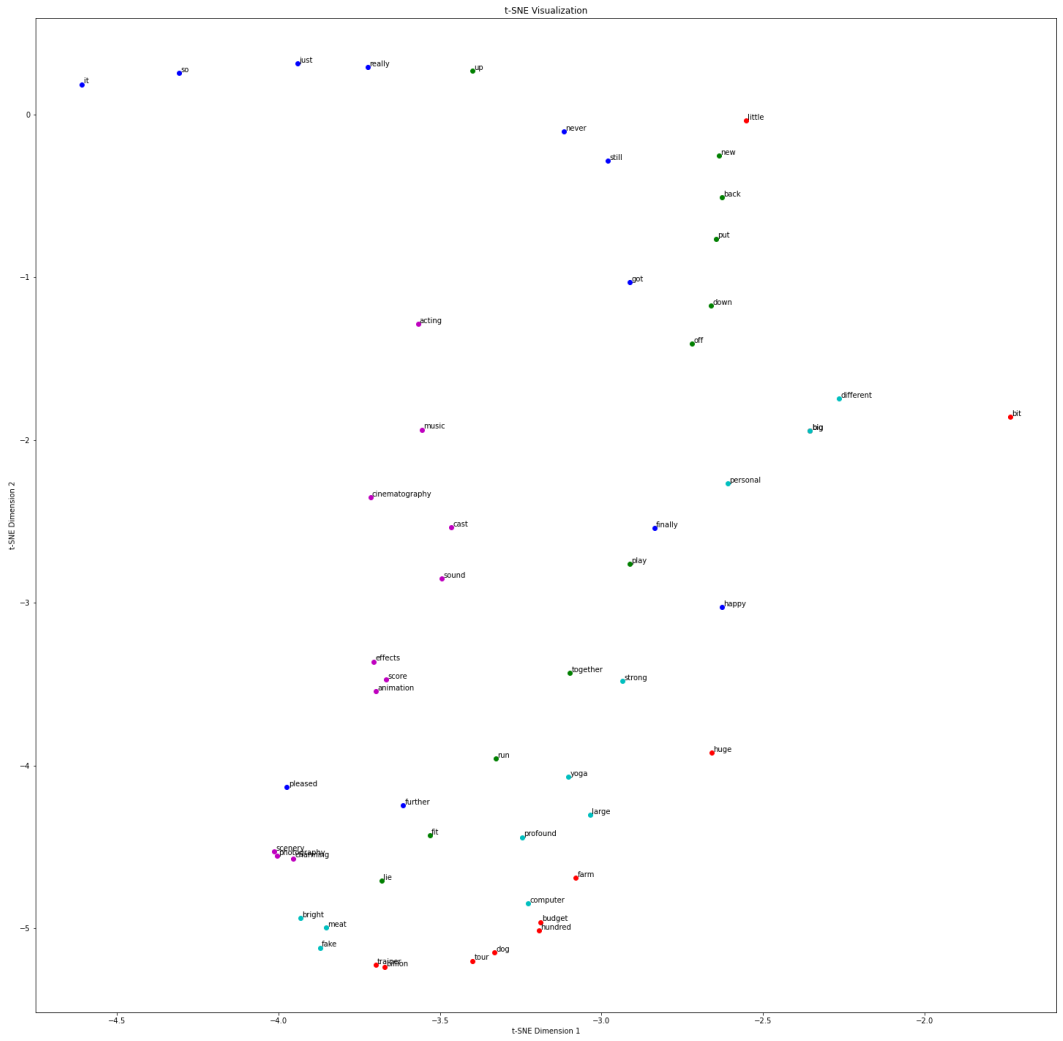


2.2 Implementation

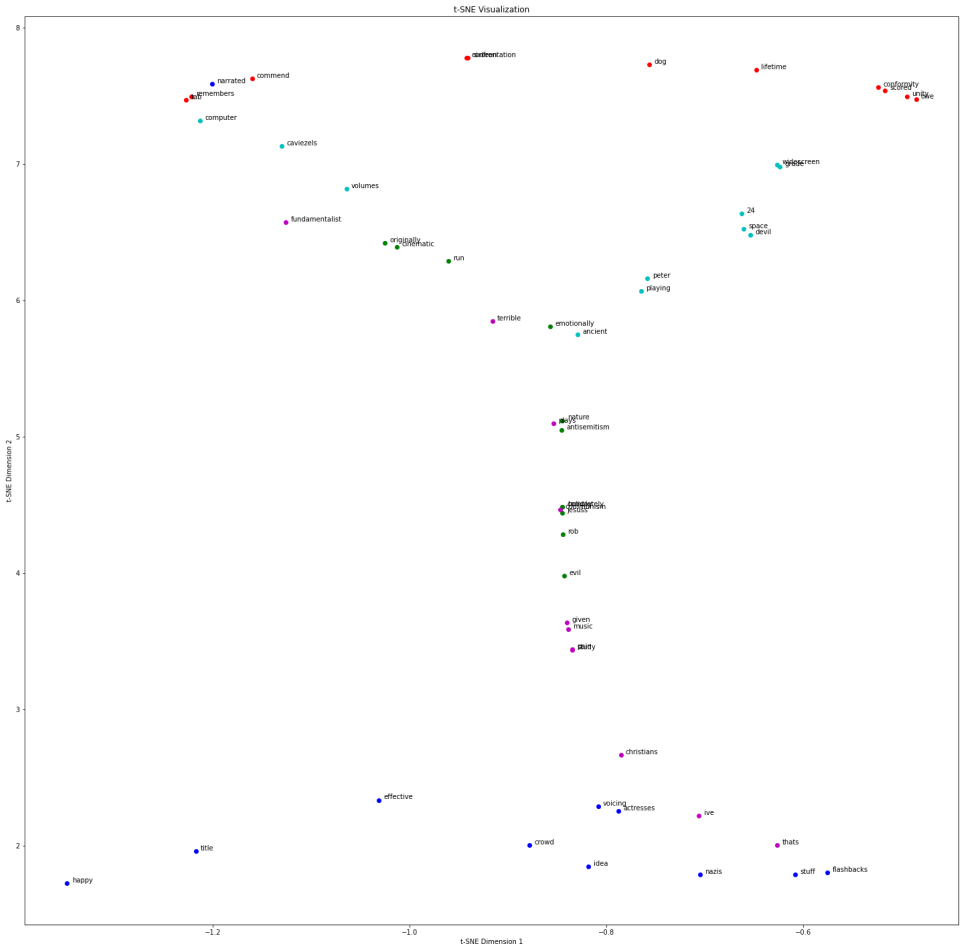
For the neural model using the algorithm CBOW with negative sampling, I have tried two approaches. One involves using the method provided in the resource document, where we calculate the embeddings for the context words and the centre word, take their dot product and pass it through a sigmoid function to get probabilities. This probability should be zero for negative samples and one for positive samples.
Another approach I have tried is by training my model for a classification task. I input a probability from the last linear layer, which gives the measure of two words being in the same neighbourhood. This is then passed through a binary cross entropy loss function along with the actual probability (0/1).

2.3 Analysis

Ans 1.
t-SNE visualisation for the embeddings generated using SVD:



t-SNE visualisation for word embeddings generated using CBOW with negative sampling:



Since the outputs are not very clear, I have also uploaded the two images on OneDrive:
[A3_tsne_images]

Ans 2.

Top 10 words similar to “titanic” for the SVD embeddings:

```
Closest words to titanic accroding to SVD:
titanic
persecuting
auto
ventures
boiling
playback
graham
opinionated
jeffrey
godless
claymations
```

Top 10 words similar to “titanic for the CBOW with negative sampling embeddings (method 1):

```
['titanic',
 'uncomfortable',
 'opened',
 'mayhem',
 'courage',
 'clark',
 'imminent',
 'throws',
 'happen',
 'sea',
 'fritz']
```

Top 10 words similar to “titanic for the CBOW with negative sampling embeddings (method 2):

```
['titanic',
 'troy',
 '22',
 'smashing',
 'towards',
 'serve',
 'wellread',
 'amongst',
 'our',
 'skinning',
 '34a']
```

Top 10 words similar to “titanic for the pre-trained word2vec embeddings:

```
Closest words to titanic according to word2vec:
('epic', 0.600616455078125)
('colossal', 0.5896502733230591)
('gargantuan', 0.5718227624893188)
('titanic_proportions', 0.5610266327857971)
('titantic', 0.5592557191848755)
('monumental', 0.5530510544776917)
('monstrous', 0.5457674860954285)
('epic_proportions', 0.5437003970146179)
('gigantic', 0.5176912546157837)
('mighty', 0.5088781118392944)
```

We can see that the outputs are not as good for the trained models, it is mainly because of the nature of our dataset and the fact that we could only train on 40,000 sentences due to insufficient compute. This is why we are unable to capture the meaning of words which are rather rare. On the other hand, if we look at some common words, our model performs much better.

Here is an example of the word “great”:

```
['great',
 'excellent',
 'wonderful',
 'craig',
 'mustsee',
 'good',
 'predictable',
 'heavy',
 'est',
 'ethnically',
 'quotfilm']
```

We see that some of the words achieved are still semantically similar to the original word.

OneDrive link to the models:
[A3_models]