

STCS Assignment 1

Determining suitable vector representations for words

Submitted by: Vitthal Bhandari 2017A7PS0136P

Submitted to: Dr Poonam Goyal (CSIS Department)

Dataset Used

The dataset used was **Reuters Corpus** of news articles from the text corpora of nltk. The Reuters Corpus contains 10,788 news documents totalling 1.3 million words. The documents have been classified into 90 topics. For this assignment, I listed down the top 15 categories of documents i.e. :

```
(3964, 'earn')
(2369, 'acq')
(717, 'money-fx')
(582, 'grain')
(578, 'crude')
(485, 'trade')
(478, 'interest')
(286, 'ship')
(283, 'wheat')
(237, 'corn')
(175, 'dlr')
(174,
'money-supply')
(171, 'oilseed')
(162, 'sugar')
(139, 'coffee')
```

The category 'coffee' contains a total of 36,710 words. By default the entire Reuters corpus is divided into test and train documents. For this project, I chose the test documents under the category 'coffee'

Of these, I chose the category 'coffee' to train the models. The test documents labelled 'coffee' were preprocessed to transform them into a form usable by the neural network.

Data Preprocessing

To make the data fit for consumption, various preprocessing techniques were employed. These were as follows:

1. Alphabetization

All characters in the corpus, except for the English alphabets (a-zA-Z), were removed by employing the regex functionality of python. Thus the corpus was left only with the English alphabets.

2. Tokenization

It is the process of splitting sentences into words. The entire corpus was split into words and then these individual words (called tokens) were further sent for preprocessing.

3. Lower casing

The individual tokens were converted to lowercase so that only the characters a-z appear in the corpus. This makes it easy for the Neural Network to learn the embeddings.

4. Stop words removal

Using a list of very commonly used stopwords (such as 'a', 'an', 'the' etc.) offered by nltk, the corpus was filtered to remove these words as they occur very frequently and do not contribute significantly to the embedding vector.

5. Lemmatization

It is the process of reducing the token to its base form. For e.g. cars → car. This way most word variations of a particular word can be avoided. Thus the network has to learn only one representation for a particular word and its variations.

All preprocessing tasks were performed by using predefined functions available in the nltk library such as nltk.stem, nltk.corpus and nltk.tokenize.

The clean corpus after preprocessing looked like this:

<pre>(['INDONESIAN COMMODITY EXCHANGE MAY EXPAND The Indonesian Commodity 'COLOMBIA BUSINESS ASKED TO DIVERSIFY FROM COFFEE A Colombia gov "COFFEE COULD DROP TO 70 / 80 CTS , CARDENAS SAYS International 'COLOMBIA COFFEE REGISTRATIONS REMAIN OPEN Colombia \' s coffee 'BRAZILIAN COFFEE RAINFALL THE FOLLOWING RAINFALL WAS RECORDED I ['indonesian commodity exchange may expand indonesian commodity e 'colombia business asked diversify coffee colombia government tr 'coffee could drop ct cardenas say international coffee price co 'colombia coffee registration remain open colombia coffee export 'brazilian coffee rainfall following rainfall recorded area past</pre>	<p>Raw corpus</p> <p>Clean corpus</p>
---	---

While cleaning, two additional parameters were computed on the corpus:

1. Vocabulary size (denoted by $|V|$): It is the total number of unique words that occur in the corpus. In this case, it was (1094) .
2. Word index (denoted by *word_index*) which is a dictionary mapping each word in the vocabulary to a corresponding integer in the range [0, 1093] .

Target-Context word generation

The last operation that was done on the clean corpus before sending it to the neural network was to generate a target word array and its corresponding context word array.

Different algorithms use different combinations of target and context words. In the continuous bag of words model, the target word is the one that comes in between the context words. In skip-gram, on the contrary, a typical "target" word is used to predict its context words. After generating the appropriate pairs, the model is ready to be initialized.

Model parameters initialization

Before going into the specifics, there are a few parameters that are common to all the three algorithms and pertain to the basic model (neural network architecture) that stays the same throughout the project.

The neural network model employed for learning word vectors is fairly simple. It consists of two layers - a hidden layer and an output layer. The weight matrix for the hidden layer (denoted by W_{hidden}) is a $|V| \times n$ sized matrix whereas the weight matrix for the output layer (denoted by W_{output}) has the size $n \times |V|$.

I have tested the models with embedding size, $n=50$ and 10 . The hyperparameter learning rate has also been varied and results have been noted.

An important parameter is the number of context words to choose for each input word. For the sake of easing the computations and maintaining generality, a total of 4 context words (2 before and 2 after) have been chosen for each input word.

Apart from the above, the model takes the generated context-target pairs as input.

A few features of the model implemented

- The model has been implemented from scratch imitating the working of a simple neural network with one hidden layer.
- The input given to the neural network is the one-hot-encoding of the input/context word. The input goes through the hidden layer matrix to produce hidden layer output and this output goes through the output layer matrix to produce final output.
- In CBOW and Skip-gram, the activation function used is softmax whereas in Skip-gram with Negative Sampling, the activation function used is sigmoid.
- Cross-entropy loss has been used as the choice of loss function since it pairs well with softmax and sigmoid activation functions.
- In skip-gram with negative sampling, only a fraction of the weights of the output layer are updated during backpropagation. This fraction, denoted by K (the number of negative samples per context words) has been chosen to be 10 for the purpose of this assignment. This is because K in the range of 10-25 is suitable for a small corpus.
- Window size remains 2 throughout. Thus a total of 4 context words surround a given target word.
- The hidden layer matrix W_{hidden} serves as the vector matrix for all words after training has ended. Thus for each word in vocabulary, the row in W_{hidden} corresponding to its index is its word embedding.
- To compare two vectors, cosine similarity has been used. Thus after computing the cosine similarity matrix for all words in the vocabulary we can compute most similar words to a given target word.

Code

The entire code has been written in Python 3.6.9 in Google Colab without using any external library other than NumPy. Nltk has been used only for data preprocessing.

The code for all three notebooks can be viewed here :

1. [CBOW](#)
2. [Skip-Gram](#)
3. [SGNS](#)

1. CONTINUOUS BAG OF WORDS MODEL (CBOW)

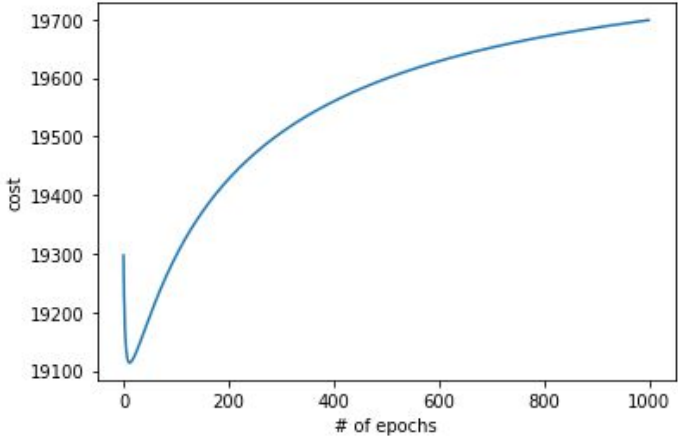
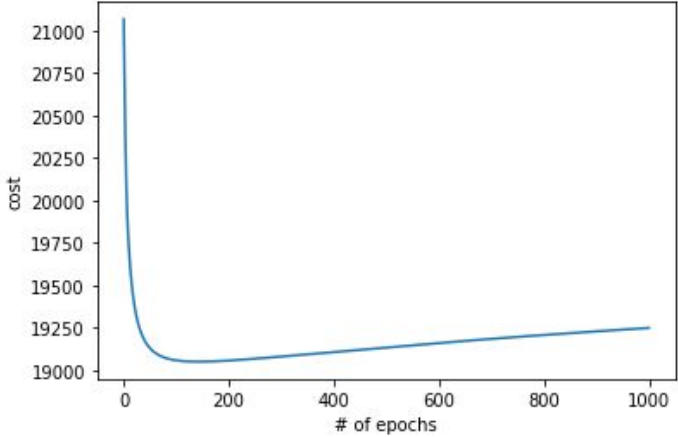
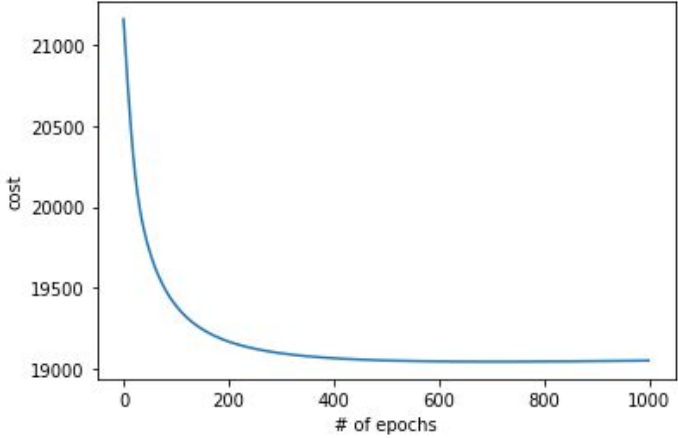
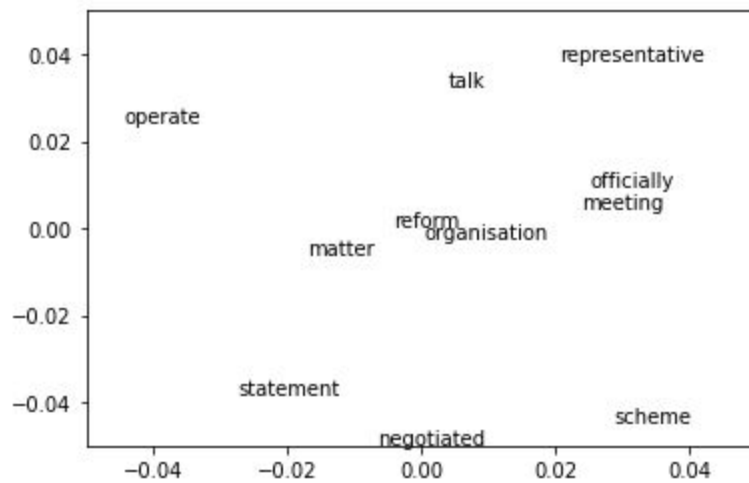
<p style="text-align: center;"><u>Model 1</u></p> <p>Leaning rate = 0.005 Word vector size = 50 No. of epochs = 1000 Training time = 00:51:13 hrs:min:sec</p> <p><u>Remarks</u> Very high learning rate. Thus the model diverged after converging because of overfitting. Hence the learning rate should be decreased.</p>	
<p style="text-align: center;"><u>Model 2</u></p> <p>Leaning rate = 0.0005 Word vector size = 50 No. of epochs = 1000 Training time = 00:53:46 hrs:min:sec</p> <p><u>Remarks</u> Learning rate decreased by a factor of 10. The model converges at around 100 epochs and slightly diverges afterwards due to overshooting. Thus the learning rate can be reduced even more.</p>	
<p style="text-align: center;"><u>Model 3</u></p> <p>Leaning rate = 0.0001 Word vector size = 50 No. of epochs = 1000 Training time = 00:50:07 hrs:min:sec</p> <p><u>Remarks</u> Learning rate decreased again by a factor of 5. Model converges to a minimum loss at around 600 epochs. Learning rate should not be increased more as the loss will start to increase due to overshooting. Hence this model is optimal.</p>	

Table 1: table depicting variations to model parameters and hyperparameters for the CBOW model

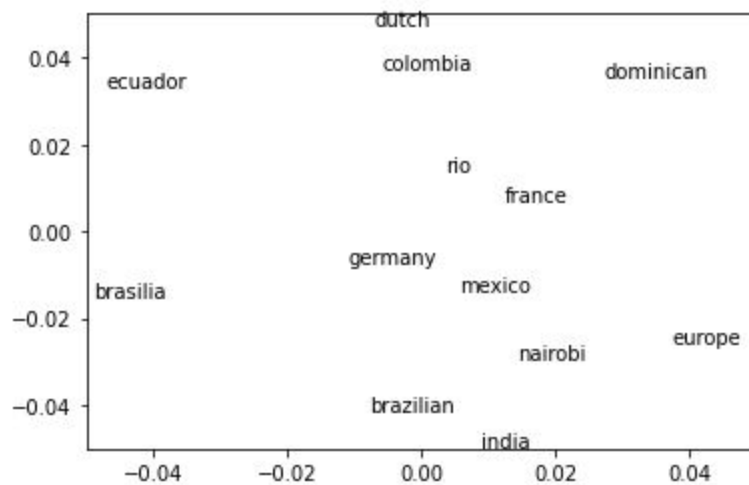
Table 1 depicts that after tweaking the hyperparameter α (the learning rate) we observed that the best performance was achieved with a value of $\alpha = 0.0001$ in model 3. The corresponding loss converged to a

minimum of 19005 under 1000 epochs and was chosen to be the one reproducing final word vectors.

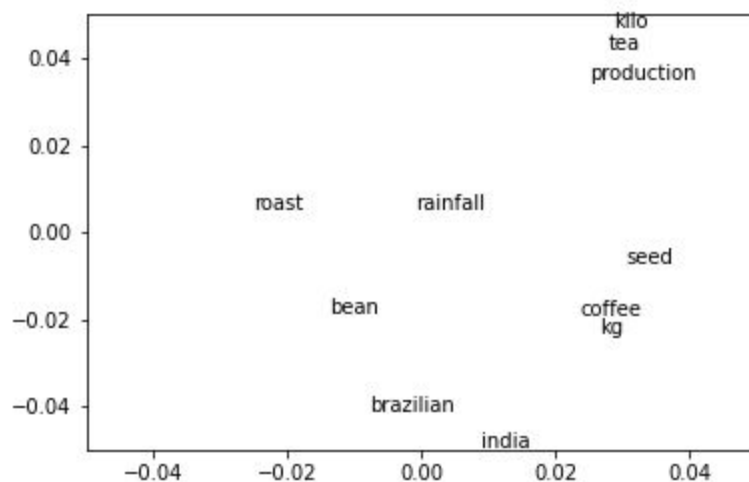
Plotting five groups of similar words



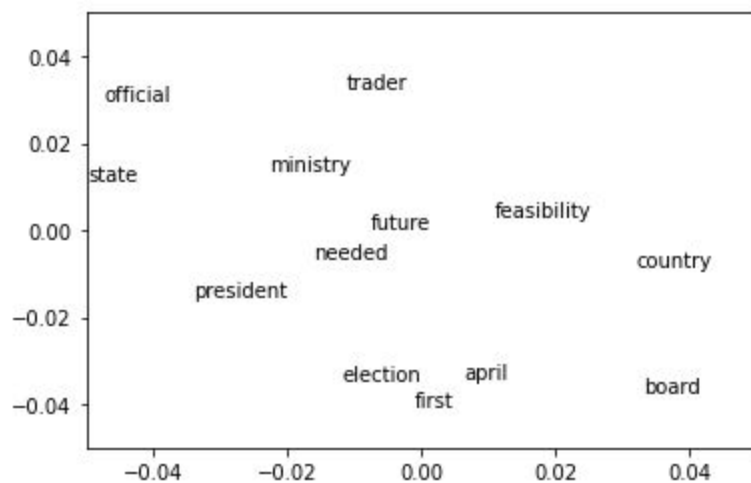
Plot depicting relationship between words related to 'organisation'



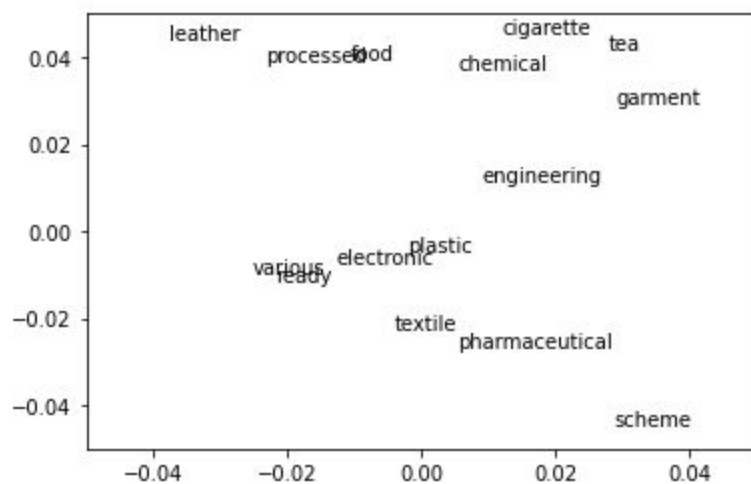
Plot depicting relationship between various country words



Plot depicting relationship between words related to 'tea' and 'coffee'



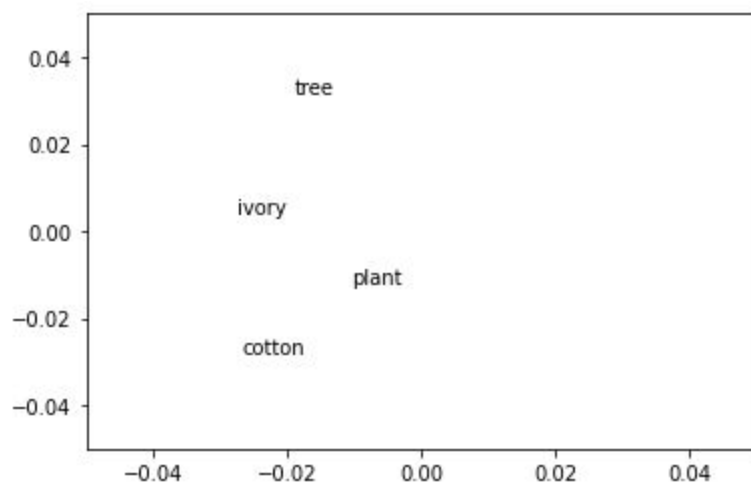
Plot depicting relationship between words related to 'ministry'



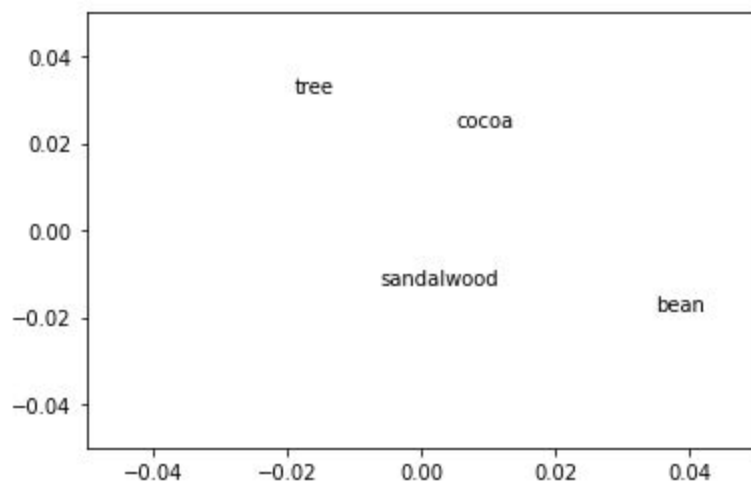
Plot depicting relationship between various industries

Plotting two Semantic analogies

Semantic relations between words refers to the similarity or dissimilarity of their meanings. Thus these relationships are dependent upon the actual meaning of the words



Tree : Ivory :: Plant : Cotton



Sandalwood : Tree :: Bean : Cocoa

Depicting two Syntactic analogies

Syntactic relations refer to relationships between words that would normally occur together (i.e. syntactically, these words are used together in language). Shorter context windows tend to lead to representations that are a bit more syntactic, since the information is coming from immediately nearby words. Since a context window of 2 is used in this project, we expect most relationships to be syntactic in nature as seen below

```
obtain_similar_words('coffee', 10)
```

The 10 most similar words to coffee are :

coffee, said, quota, export, year, bag, trade, new, mln, producer

Here coffee is syntactically similar to the highlighted words as shown in its usage with them such as '*the quota of coffee ...*', '*coffee export...*' and '*trade coffee bags ...*'

```
obtain_similar_words('brazilian', 10)
```

The 10 most similar words to brazilian are :

brazilian, crop, exchange, trade, export, producer, official, formula, country, coffee

Here brazilian is syntactically similar to the highlighted words as shown in its usage with them such as '*the brazilian crops ...*', '*brazilian trade ...*' and '*brazilian coffee ...*'

2. SKIP-GRAM MODEL

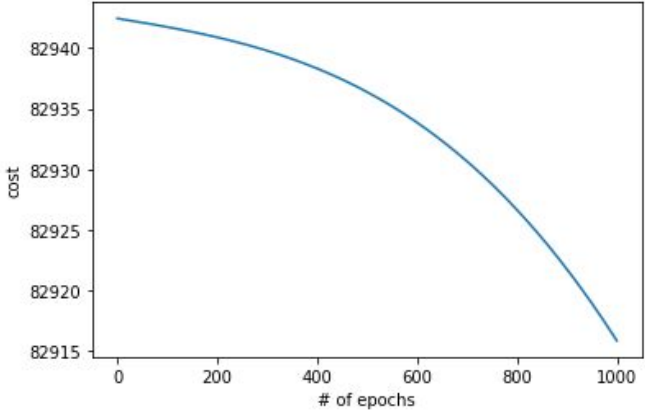
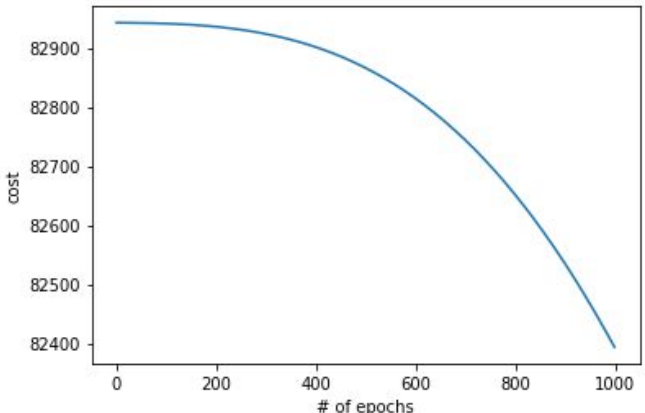
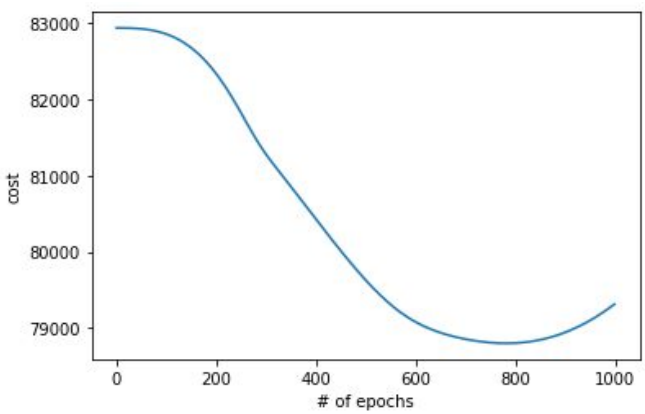
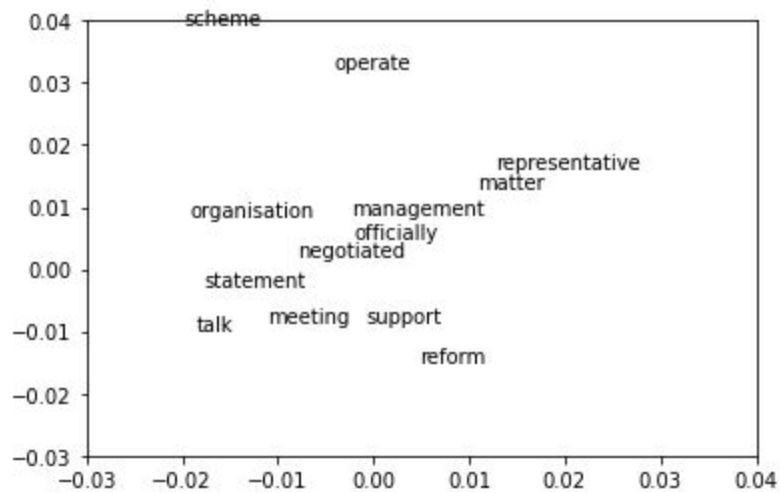
<p style="text-align: center;"><u>Model 1</u></p> <p>Leaning rate = 0.0001 Word vector size = 50 No. of epochs = 1000 Training time = 01:25:00 hrs:min:sec</p> <p><u>Remarks</u> Very low learning rate. Thus the model will take more than 1000 iterations to converge. Hence the learning rate should be increased. Since the training corpus is small, embedding size (=50) can also be decreased to reduce training time for 1000 epochs.</p>	 <table border="1"> <caption>Data for Model 1 Cost vs Epochs</caption> <thead> <tr> <th># of epochs</th> <th>cost</th> </tr> </thead> <tbody> <tr><td>0</td><td>82940</td></tr> <tr><td>200</td><td>82938</td></tr> <tr><td>400</td><td>82935</td></tr> <tr><td>600</td><td>82930</td></tr> <tr><td>800</td><td>82925</td></tr> <tr><td>1000</td><td>82915</td></tr> </tbody> </table>	# of epochs	cost	0	82940	200	82938	400	82935	600	82930	800	82925	1000	82915
# of epochs	cost														
0	82940														
200	82938														
400	82935														
600	82930														
800	82925														
1000	82915														
<p style="text-align: center;"><u>Model 2</u></p> <p>Leaning rate = 0.0005 Word vector size = 10 No. of epochs = 1000 Training time = 00:53:30 hrs:min:sec</p> <p><u>Remarks</u> Learning rate increased by a factor of 5 and embedding size decreased by a factor of 5. Considerable reduction in training time. Also, the loss curve begins to decrease more rapidly. Thus the learning rate can be increased more to allow the loss to decrease faster.</p>	 <table border="1"> <caption>Data for Model 2 Cost vs Epochs</caption> <thead> <tr> <th># of epochs</th> <th>cost</th> </tr> </thead> <tbody> <tr><td>0</td><td>82900</td></tr> <tr><td>200</td><td>82895</td></tr> <tr><td>400</td><td>82885</td></tr> <tr><td>600</td><td>82850</td></tr> <tr><td>800</td><td>82750</td></tr> <tr><td>1000</td><td>82400</td></tr> </tbody> </table>	# of epochs	cost	0	82900	200	82895	400	82885	600	82850	800	82750	1000	82400
# of epochs	cost														
0	82900														
200	82895														
400	82885														
600	82850														
800	82750														
1000	82400														
<p style="text-align: center;"><u>Model 3</u></p> <p>Leaning rate = 0.0025 Word vector size = 10 No. of epochs = 1000 Training time = 00:46:33 hrs:min:sec</p> <p><u>Remarks</u> Learning rate increased again by a factor of 5. Model converges to a minimum loss at around 800 epochs. Training time has also reduced as compared to Model 2. Learning rate should not be increased more as the loss will start to increase due to overshooting.</p>	 <table border="1"> <caption>Data for Model 3 Cost vs Epochs</caption> <thead> <tr> <th># of epochs</th> <th>cost</th> </tr> </thead> <tbody> <tr><td>0</td><td>83000</td></tr> <tr><td>200</td><td>82500</td></tr> <tr><td>400</td><td>81500</td></tr> <tr><td>600</td><td>80500</td></tr> <tr><td>800</td><td>78802</td></tr> <tr><td>1000</td><td>79000</td></tr> </tbody> </table>	# of epochs	cost	0	83000	200	82500	400	81500	600	80500	800	78802	1000	79000
# of epochs	cost														
0	83000														
200	82500														
400	81500														
600	80500														
800	78802														
1000	79000														

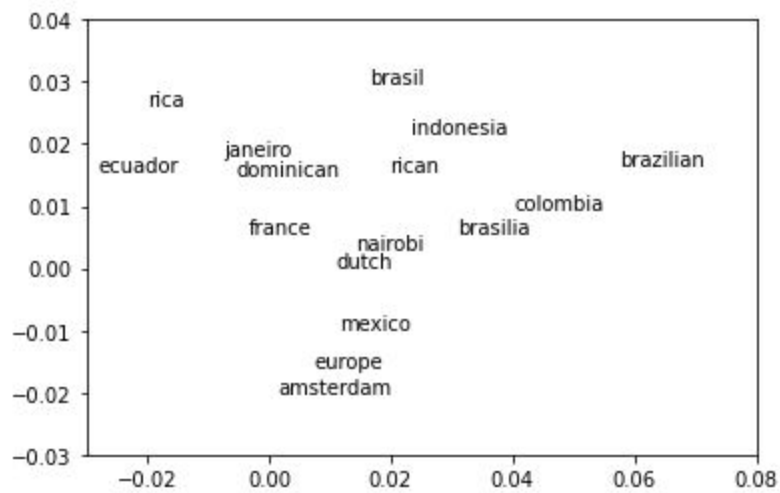
Table 2: table depicting variations to model parameters and hyperparameters for the skip-gram model

Table 2 depicts that after tweaking the hyperparameter α (the learning rate) we observed that the best performance was achieved with a value of $\alpha = 0.0025$ in model 3. The corresponding loss converged to a minimum of 78802 under 1000 epochs and was chosen to be the one reproducing final word vectors. Further to decrease the training time, the parameter 'embedding size' was reduced from 50 to 10 and it provided a significant reduction in training time. This modification was logical since a vector of length 10 will be sufficient to capture the inter-word complexities within a corpus of limited size.

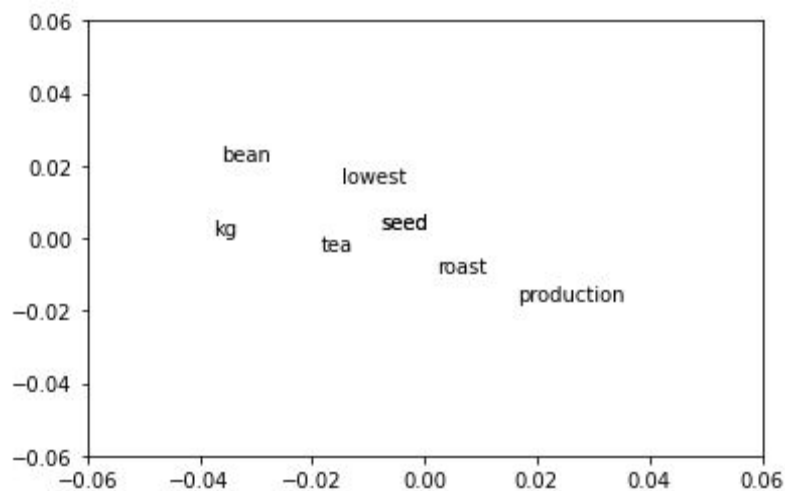
Plotting five groups of similar words



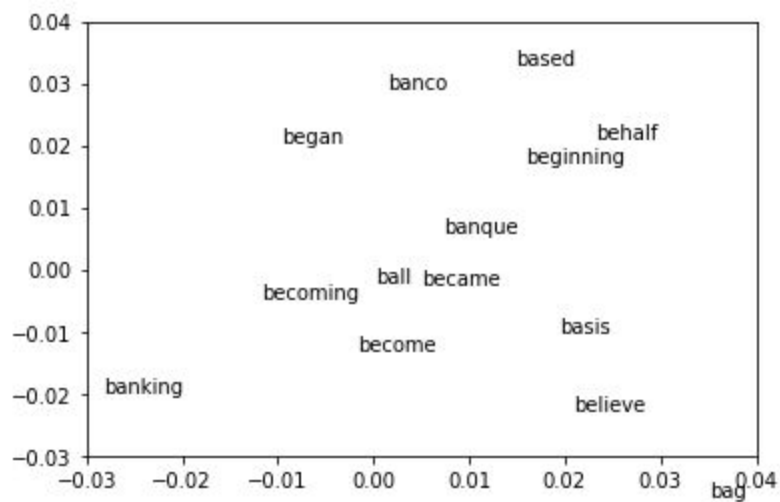
Plot depicting relationship between words related to 'organisation'



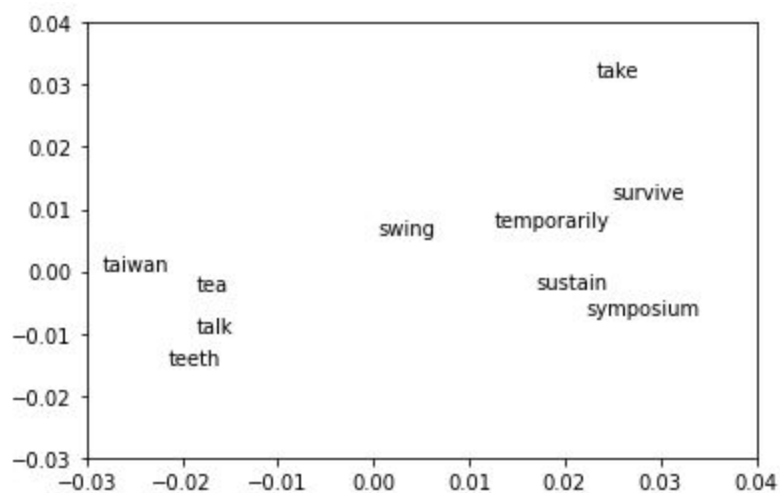
Plot depicting relationship between various country words



Plot depicting relationship between words related to 'tea' and 'coffee'

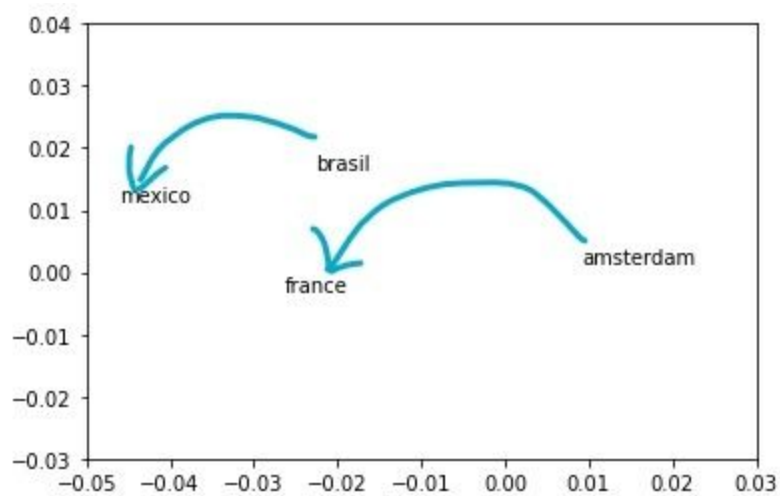


Plot depicting relationship between words related to 'become'

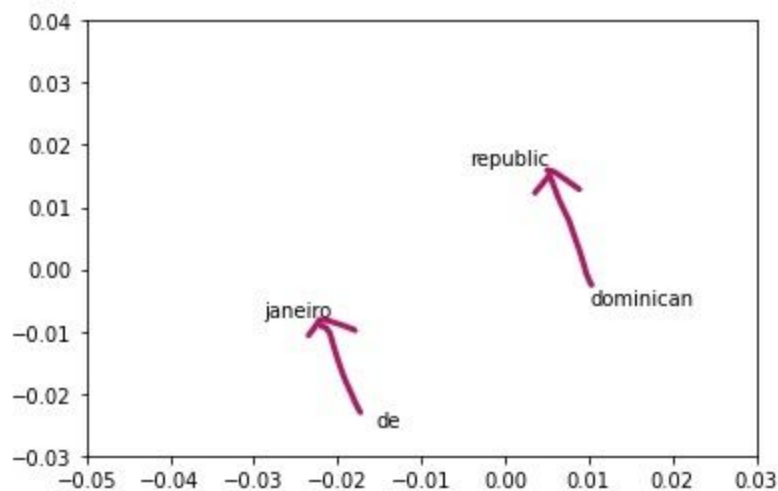


Plot depicting relationship between a random set of words

Plotting two Semantic analogies.



Brazil : Mexico :: Amsterdam : France



De : Janeiro :: Dominican : Republic

Depicting two Syntactic analogies

```
obtain_similar_words('product', 10)
```

The 10 most similar words to product are :-

product , whole , necessary , fair , possible , tied , august , shipment , producing , pessimistic

Here product is syntactically similar to the highlighted words as shown in its usage with them such as '*the whole product ...*', '*the necessary product ...*' and '*shipment of the product ...*'

```
obtain_similar_words('state', 10)
```

The 10 most similar words to state are :-

state , undermine , indonesia , physical , earlier , apparently , membership , lt , usda , moving

Here state is syntactically similar to the highlighted words as shown in its immediate usage with them such as '*undermine the state ...*', '*the state of indonesia ...*' and '*physical state ...*'. Moreover, '*usda*' stands for the United States Department of Agriculture.

3. SKIP-GRAM WITH NEGATIVE SAMPLING (SGNS)

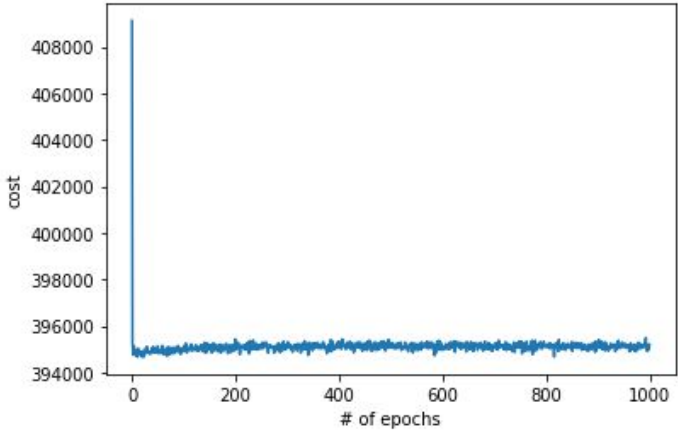
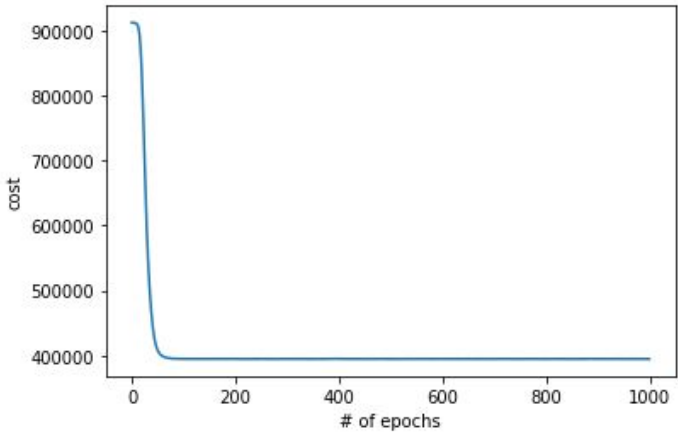
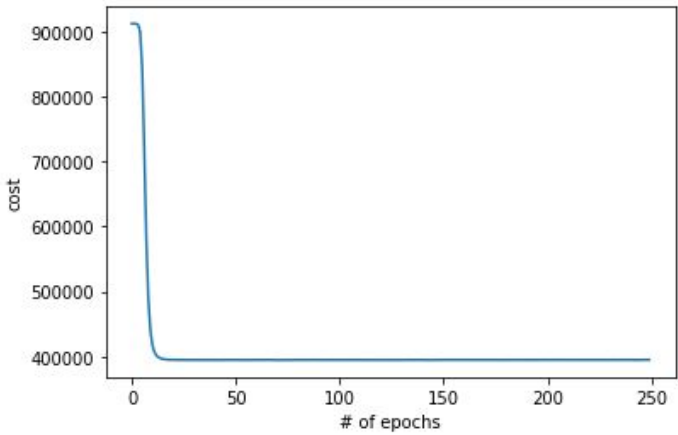
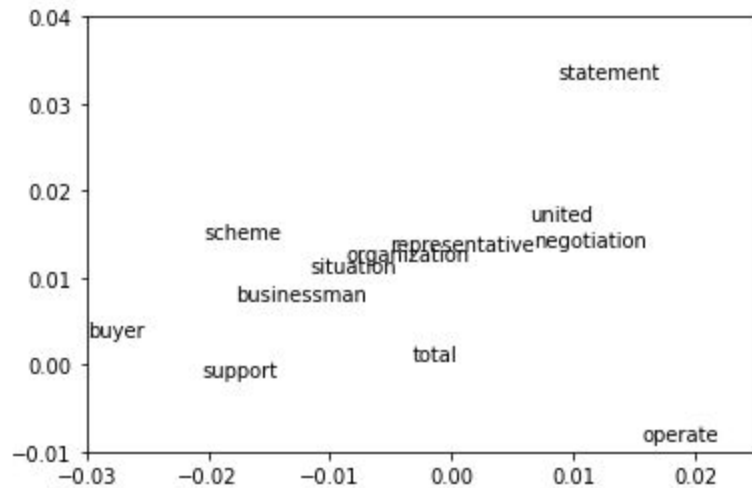
<p style="text-align: center;"><u>Model 1</u></p> <p>Leaning rate = 0.0025 Word vector size = 50 No. of epochs = 1000 Training time = 01:45:40 hrs:min:sec</p> <p><u>Remarks</u> Very high learning rate. Thus the model took less than 50 epochs to converge and the gradient was too steep. Hence the learning rate should be decreased.</p>	
<p style="text-align: center;"><u>Model 2</u></p> <p>Leaning rate = 0.0001 Word vector size = 10 No. of epochs = 1000 Training time = 01:34:38 hrs:min:sec</p> <p><u>Remarks</u> Learning rate decreased by a factor of 25. The loss curve begins to decrease more smoothly and takes around 150 epochs to completely converge. This shows that the number of epochs can be reduced since our corpus is not very large.</p>	
<p style="text-align: center;"><u>Model 3</u></p> <p>Leaning rate = 0.0005 Word vector size = 10 No. of epochs = 250 Training time = 00:24:33 hrs:min:sec</p> <p><u>Remarks</u> Learning rate increased again by a factor of 5. Model converges to a minimum loss at around 25 epochs. Training time has also reduced as compared to Model 2 since 1000 epochs were unnecessary. Model converges faster as compared to model 2 since the learning rate is larger.</p>	

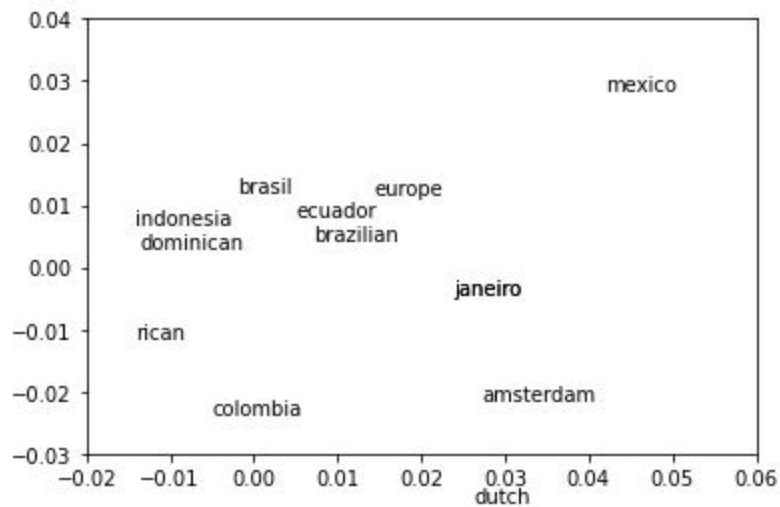
Table 3: table depicting variations to model parameters and hyperparameters for the SGNS model

Table 3 depicts that after tweaking the hyperparameter α (the learning rate) we observed that the best performance was achieved with a value of $\alpha = 0.0005$ in model 3. The corresponding loss converged to a minimum of 78802 under 25 epochs and was chosen to be the one reproducing final word vectors. Further to decrease the training time, the number of epochs was reduced from 1000 to 250 and it provided a significant reduction (upto 4 times) in training time.

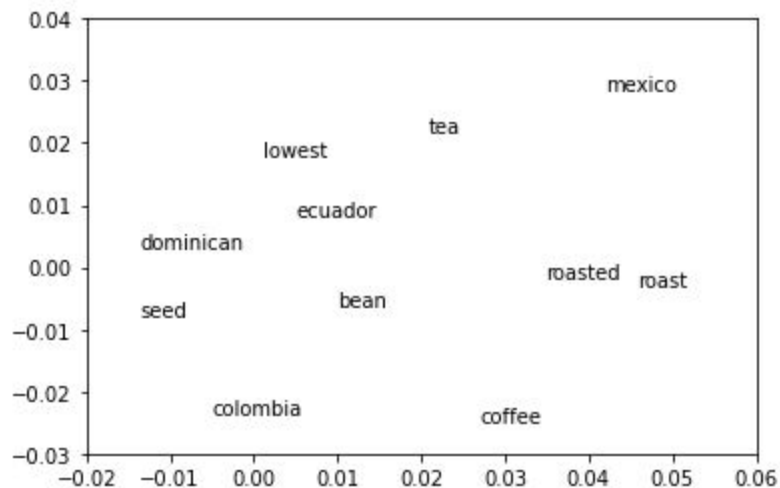
Plotting five groups of similar words



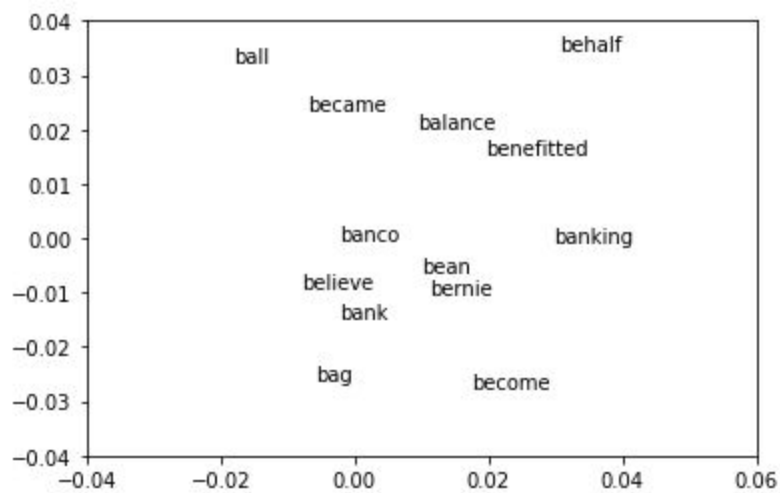
Plot depicting relationship between words related to 'organisation'



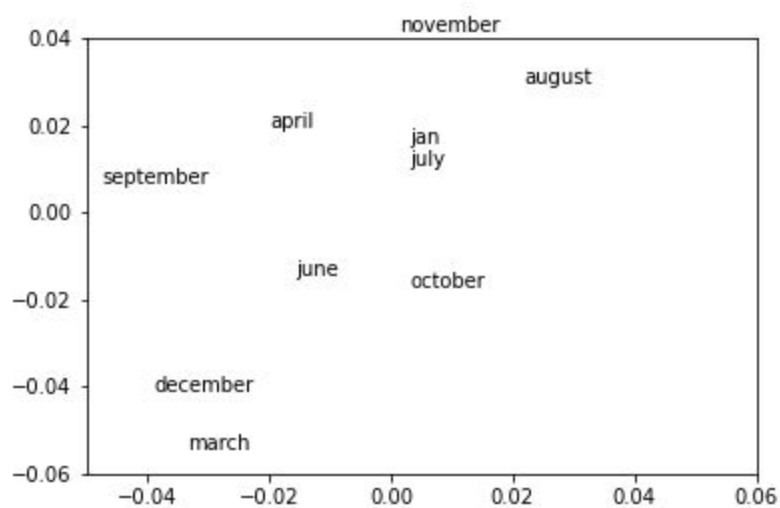
Plot depicting relationship between various country words



Plot depicting relationship between words related to 'tea' and 'coffee'

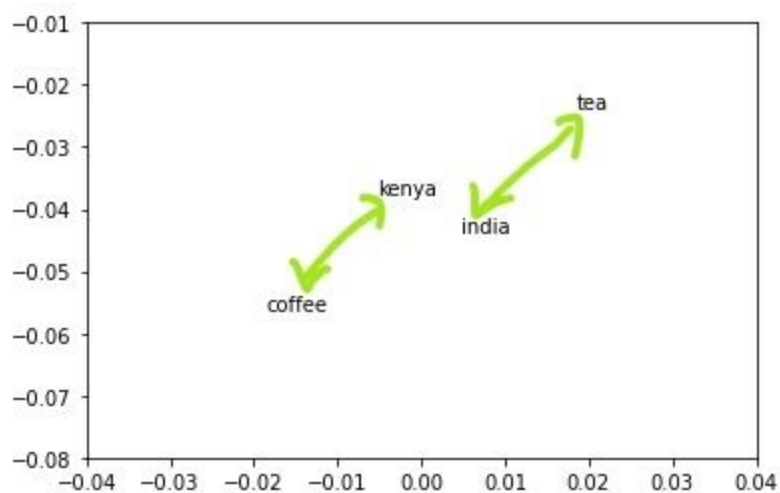


Plot depicting relationship between words related to 'become'

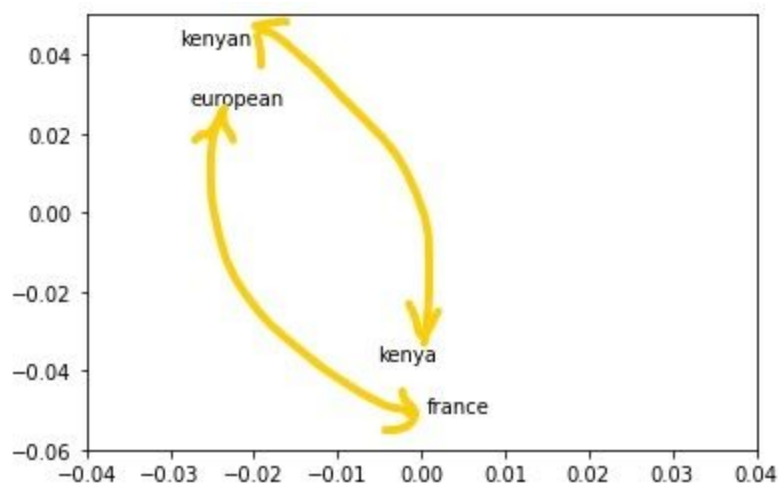


Plot depicting relationship between various months

Plotting two Semantic analogies



Tea : India :: Coffee : Kenya



Kenya : Kenyan :: France : European

Depicting two Syntactic analogies

```
obtain_similar_words('production', 10)
```

The 10 most similar words to production are :-

production , germany , stockpiled , become , include , frequently , traditionally , banco , reason , mexican

Here production is syntactically similar to the highlighted words as shown in its usage such as '*the production stockpiled ...*', '*the production became ...*' and '*traditionally, the production of ...*'

```
obtain_similar_words('agriculture', 10)
```

The 10 most similar words to agriculture are :-

agriculture , american , paine , building , united , recently , reform , roast , office , pose

Here agriculture is used with the highlighted words frequently. Examples being '*american agriculture ...*', '*agricultural buildings*' and '*agricultural office ...*'.

Conclusion

In this report we first analyzed the performance of CBOW on our preprocessed corpus. By training with different values of learning rate α , we observed that lower the value of α , better the performance of the model. The training time remained almost unaffected by α , suggesting that decreasing the embedding size, n , would reduce computation and hence, the training time.

Next we implemented the skip-gram algorithm and varied both α and n . It was observed that a higher value of α favoured the training process. Due to increased computations, the model took more time to train as compared to CBOW with the same parameters. Hence by decreasing n , and consequently decreasing the number of trainable parameters, the training time was significantly reduced.

Finally we interpreted the performance of our model by implementing the skip-gram model with negative sampling (SGNS). It was observed that the model converged faster than any of the above two implementations. The speed of convergence was directly proportional to the value of α . Thus, a carefully chosen value of α would result in an appropriate rate of convergence. Nevertheless, it became clear that SGNS was the fastest of the three algorithms and rightfully so, since only a tiny fraction of the weights were updated.

A quick comparison of the word vectors obtained from these models revealed that accuracy wise, both skip-gram and SGNS algorithms produced similar vectors whereas CBOW produced vectors with better syntactic relationships. Efficiency-wise, SGNS was the fastest of the three.

Given the limitation of time and hardware constraints, more accurate word vectors can be trained by choosing a larger corpus, increasing n and experimenting with different values of α .