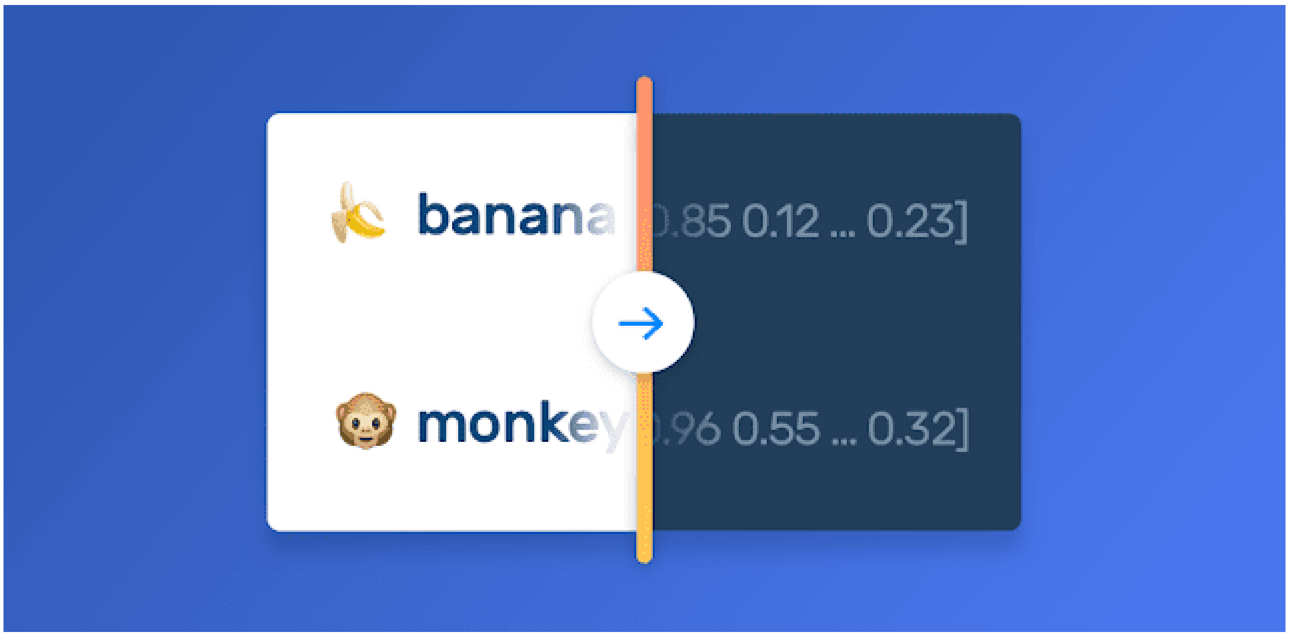
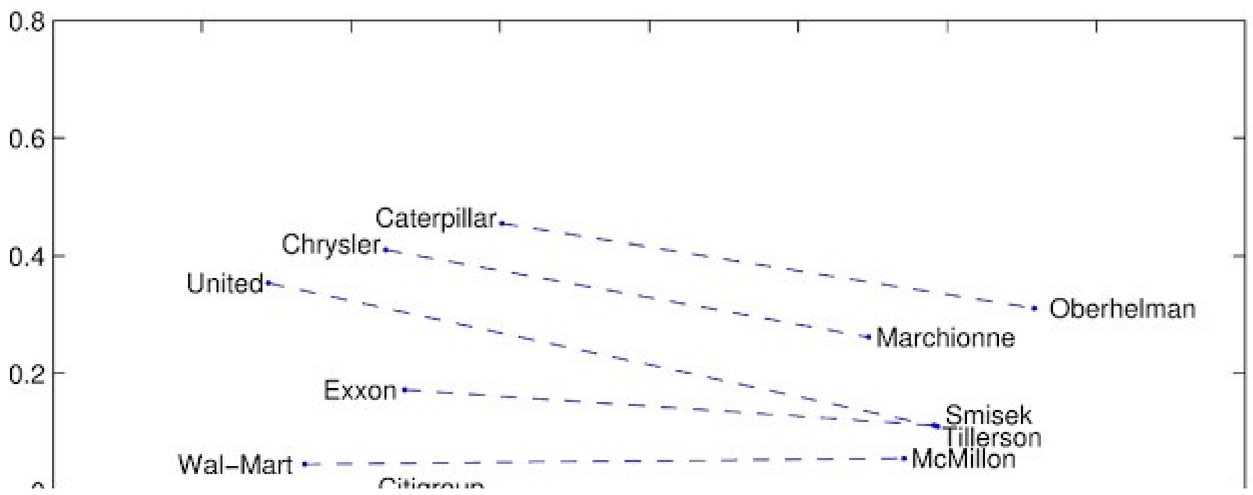


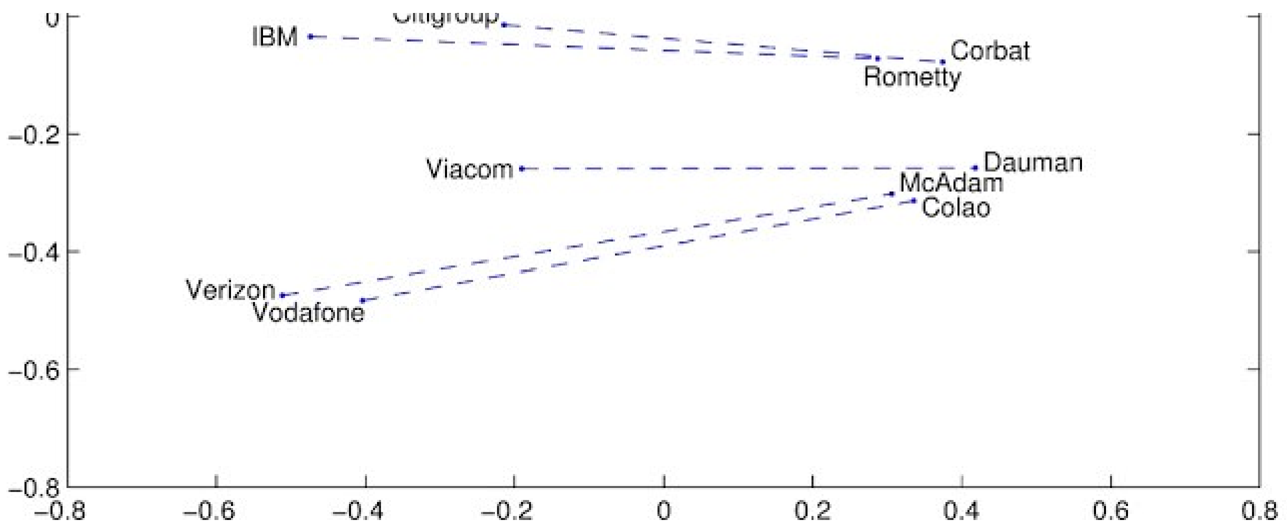
What is GloVe?

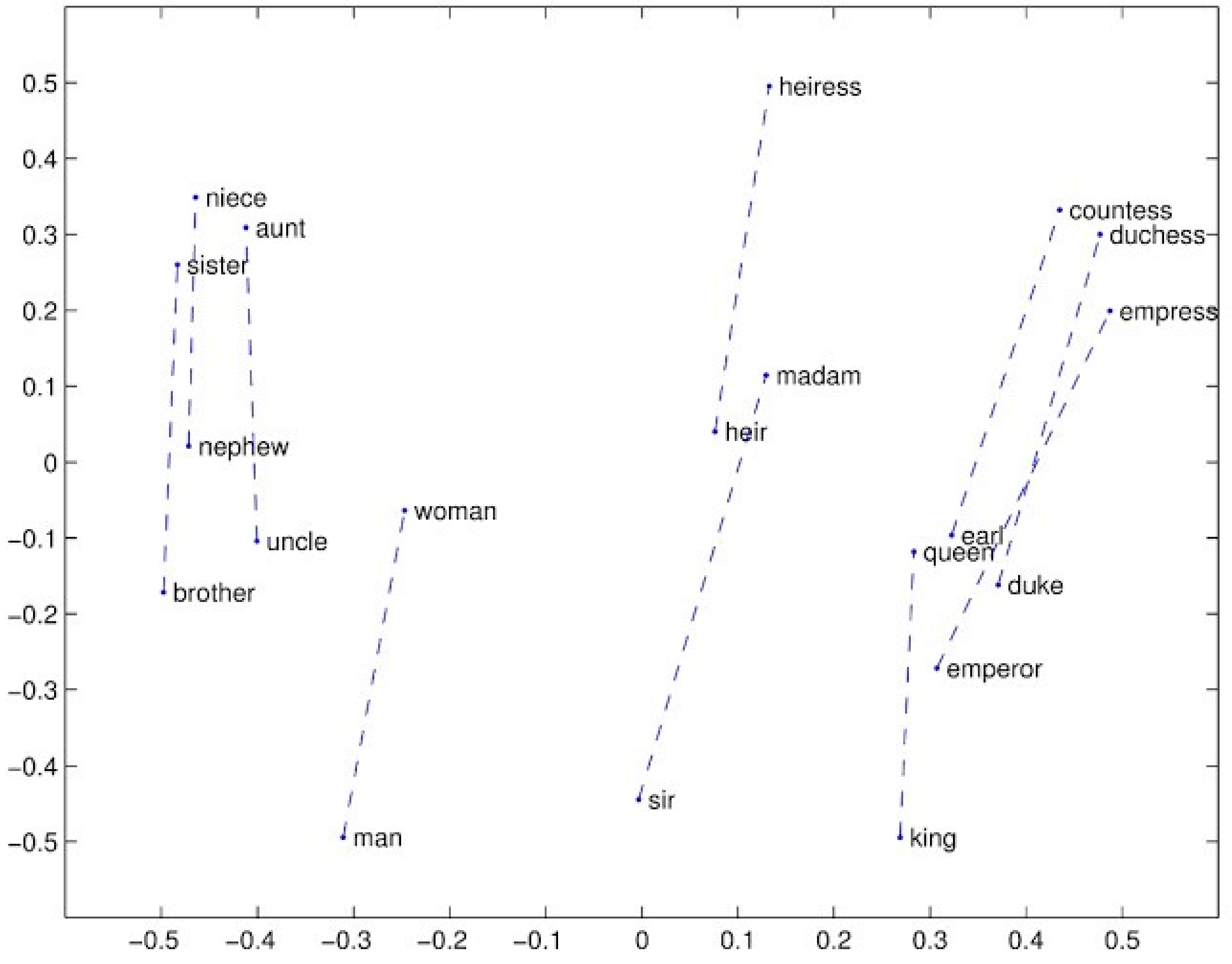
Japneet Singh Chawla
Apr 24, 2018 · 4 min read

GloVe stands for global vectors for word representation. It is an unsupervised learning algorithm developed by Stanford for generating word embeddings by aggregating global word-word co-occurrence matrix from a corpus. The resulting embeddings show interesting linear substructures of the word in vector space.

Examples for linear substructures are:







These results are pretty impressive. This type of representation can be very useful in many machine learning algorithms.

To read more about Word Vectorization you can read my other article.

In this blog post, we will be learning about GloVe implementation in python. So, let’s get started.

Let’s create the Embeddings

Installing Glove-Python

The GloVe is implementation in python is available in library **glove-python.**

```
pip install glove_python
```

Text Preprocessing

In this step, we will pre-process the text like removing the stop words, lemmatize the words etc.

> *You can perform different steps based on your requirements.*

I will use nltk stopword corpus for stop word removal and nltk word lemmatization for finding lemmas.

I order to use nltk corpus you will need to download it using the following commands.

Downloading the corpus

> *import nltk nltk.download() #this will open a GUI from which you can download the corpus*

Input initialization

> *#list of sentences to be vectorized lines=[“Hello this is a tutorial on how to convert the word in an integer format”, “this is a beautiful day”,”Jack is going to office”]*

Removing the Stop Words

> *from nltk.corpus import stopwords stop_words=set(stopwords.words(‘english’)) lines_without_stopwords=[] #stop words contain the set of stop words for line in lines: temp_line=[] for word in lines: if word not in stop_words: temp_line.append (word) string=’ ‘ lines_without_stopwords.append(string.join(temp_line)) lines=lines_without_stopwords*

Lemmatization

> *#import WordNet Lemmatizer from nltk from nltk.stem import WordNetLemmatizer wordnet_lemmatizer = WordNetLemmatizer() lines_with_lemmas=[] #stop words contain the set of stop words for line in lines: temp_line=[] for word in lines: temp_line.append (wordnet_lemmatizer.lemmatize(word)) string=' ' lines_with_lemmas.append(string.join(temp_line)) lines=lines_with_lemmas*

Now we have done the basic preprocessing of the text. Any other preprocessing stuff can be achieved similarly.

Preparing Input

We have out input in the form of an array of lines. In order for the model to process the data, we need covert our input to an array of array of words (:\ ).

Our Input

> *lines=["Hello this is a tutorial on how to convert the word in an integer format","this is a beautiful day","Jack is going to office"]*

New Input

> *lines=[['Hello', 'this','tutorial', 'on', 'how','convert' ,'word',' integer','format'],['this' ,'beautiful', 'day'],['Jack','going' , 'office']*
>
> *new_lines=[] for line in lines: new_lines=line.split('') #new lines has the new format lines=new_lines*

Building a Glove model

```
#importing the glove library
from glove import Corpus, Glove

# creating a corpus object
corpus = Corpus()

#training the corpus to generate the co occurence matrix which is
used in GloVe
corpus.fit(lines, window=10)
```

```
#creating a Glove object which will use the matrix created in the
above lines to create embeddings
#We can set the learning rate as it uses Gradient Descent and number
of components

glove = Glove(no_components=5, learning_rate=0.05)

glove.fit(corpus.matrix, epochs=30, no_threads=4, verbose=True)
glove.add_dictionary(corpus.dictionary)
glove.save('glove.model')
```

Creating a glove model uses the co-occurrence matrix generated by the Corpus object to create the embeddings.

The **corpus.fit** takes two arguments:

- lines — this is the 2D array we created after the pre-processing
- window — this is the distance between two words algo should consider to find some relationship between them

Parameters of Glove:

- no_of_components — This is the dimension of the output vector generated by the GloVe
- learning_rate — Algo uses gradient descent so learning rate defines the rate at which the algo reaches towards the minima (lower the rate more time it takes to learn but reaches the minimum value)

Parameters of glove.fit :

- cooccurence_matrix: the matrix of word-word co-occurrences
- epochs: this defines the number of passes algo makes through the data set
- no_of_threads: number of threads used by the algo to run

After the training glove object has the word vectors for the lines we have provided. But the dictionary still resides in the corpus object. We need to add the dictionary to the glove object to make it complete.

```
glove.add_dictionary(corpus.dictionary)
```

This line does the dictionary addition in the glove object. After this, the object is ready to provide you with the embeddings.

```
print glove.word_vectors[glove.dictionary['samsung']]
OUTPUT:

[ 0.04521741  0.02455266 -0.06374787 -0.07107575  0.04608054]
```

This will print the embeddings for the word “samsung”.

End Notes

We have learned how to generate the vectors for the text data which is very useful for creating many Machine Learning models and techniques like SVM, KNN, K-Means, Logistic Classifiers, Sentiment Analysis, Document Classification etc.

More can be learned about the GloVe on Stanford’s website.

Machine Learning Glove Word Embeddings Embedding Vectors