Document Vectors

Word vectors and word embeddings work or representing words but what if we wanted to represent whole documents. Document vectors is an extension of the same ideas a word vectors but for documents.

Note that when we refer to a document, we are referring to a collection of words that have some meaning to a user. A document can be a product review, a tweet or a line of movie dialogue and can be a few words or as many as thousands. What determines what we call a document is that we can identify it and use it in a machine learning project as an instance of something that the algorithm can learn from. More importantly, it depends on the objective of the project – so if the end goal is to read a tweet and predict something about it then that would consist a document.

Discussion 7: Discuss projects that students might want to do that involve analyzing text using document vectors. For example, could we use document vectors to predict events based on tweets, or news articles

Uses of Document Vectors

1. Similarity. We can use document vectors to compare texts for similarity. Legal AI software can use document vectors to find similar legal cases
2. Recommendations. For example, online magazines can recommend similar articles based on others that users have read
3. Predictions. Document vectors can be used as the input into machine learning algorithms in to build a predictive model

Exercise: From Movie Dialogue to Document Vectors

In this exercise we will convert movie dialogue into document vectors.

Each line of the movie will be converted to a vector.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Looks like things worked out tonight, huh? | to |  |  |  |  |  |  |  |  |  |  |

For this exercise we will use a part of the Cornell Movie Dialogue Dataset.

You can find the full dataset at <https://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html>

The dataset should be available locally in the lesson directory:

**vector-representations/data/cornell-movie-dialogs/**

1. Open the notebook called DocumentVectors.ipynb.
2. Add the import statements for the libraries we will use in this exercise. We will be using the **gensim** library.

The main **gensim** objects for Document Vectors are **Doc2Vec** and **TaggedDocument** and we will also need to import some utility and preprocessing code.

import pandas as pd

from gensim import utils

from gensim.models.doc2vec import TaggedDocument

from gensim.models import Doc2Vec

from gensim.parsing.preprocessing import preprocess\_string, remove\_stopwords

import random

1. Set the display column width to be as wide as it needs to be in order to display the movie lines

pd.set\_option('display.max\_colwidth', -1)

1. Add the following declaration for the location of the movie lines file

movie\_lines\_file = 'data/cornell-movie-dialogs/movie\_lines.txt'

1. Load the movie dialogs. You will need to iterate over the lines in the file and split the columns. The columns are delimited by '+++$+++'

Then you will create a dataframe containing the movie lines.

with open(movie\_lines\_file) as f:

movie\_lines = [line.strip().split('+++$+++') for line in f.readlines()];

lines\_df = pd.DataFrame([{'LineNumber': d[0].strip(),

'Person': d[3].strip(),

'Line': d[4].strip(),

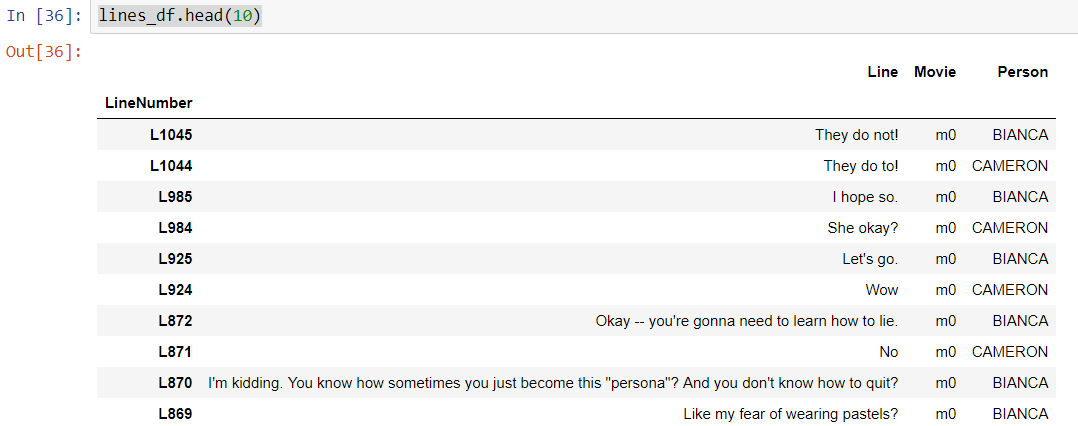
'Movie' : d[2].strip()}

for d in movie\_lines])

lines\_df = lines\_df.set\_index('LineNumber')

Take a quick look at the movie dialogue. You can use the following functions to look at the basic statistics of the **lines\_df** dataframe – **len, head, nunique**.

lines\_df.head(10)

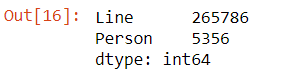


Look at the basic stats of the movie lines file. Use len() and nunique()

len(movie\_lines)



movie\_lines\_df.nunique()



1. Now because there are over 300,000 movie dialogue lines training might take a while. We can train on a subset of the movie lines. Let’s limit the training to 50000 rows

lines\_df\_small = lines\_df.head(50000)

1. We can now create the object that will create the training instances for the Doc2Vec model.

class DocumentDataset(object):

def \_\_init\_\_(self, data:pd.DataFrame, column):

document = data[column].apply(self.preprocess)

self.documents = [ TaggedDocument( text, [index])

for index, text in document.iteritems() ]

def preprocess(self, document):

return preprocess\_string(remove\_stopwords(document))

def \_\_iter\_\_(self):

for document in self.documents:

yield documents

def tagged\_documents(self, shuffle=False):

if shuffle:

random.shuffle(self.documents)

return self.documents

Doc2Vec requires each instance to be a TaggedDocument instance so internally we create a list of TaggedDocument for each movie line in the file.

1. Now create the document dataset object. Note that we specify which column contains the “document” – in this case it is the movie line

documents\_dataset = DocumentDataset(lines\_df\_small, 'Line')

1. Create the Doc2Vec model

model = Doc2Vec(min\_count=1, window=5, vector\_size=100, sample=1e-4, negative=5, workers=8)

model.build\_vocab(documents\_dataset.tagged\_documents())

1. Now train the model. It could take a while depending on how many records we train with. Note that we also set the number of epochs, or times through the training set to a reasonable number.

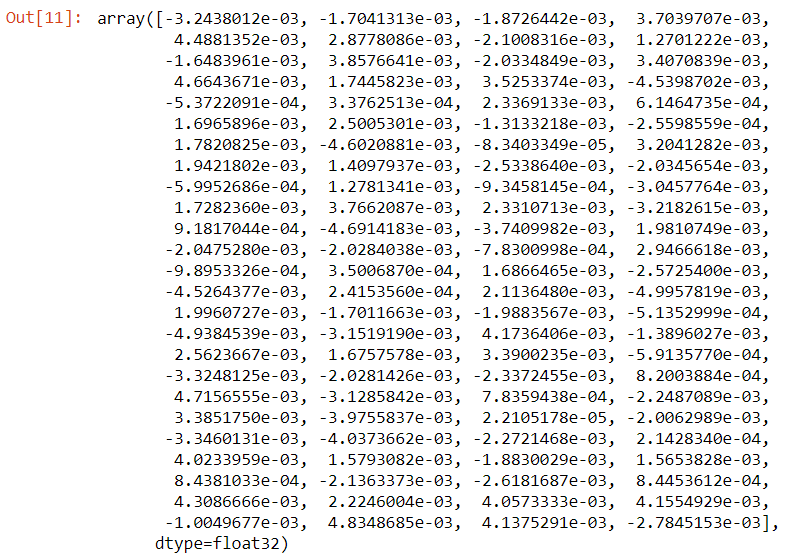
model.train(documents\_dataset.tagged\_documents(shuffle=True),

total\_examples = model.corpus\_count,

epochs=10)

1. Look at one of the vectors

model['L1045']



1. Add code to display each vector. We will define the following functions
   1. **show\_image** takes a vector and displays it as an image
   2. **show\_dialog** takes a line number e.g. L200 and returns the movie line and the vector for that line

import matplotlib.pyplot as plt

plt.style.use('fivethirtyeight')

def show\_image(vector, line):

fig, ax = plt.subplots(1,1, figsize=(10, 2))

ax.tick\_params(axis='both',

which='both',

left=False,

bottom=False,

top=False,

labelleft=False,

labelbottom=False)

ax.grid(False)

print(line)

ax.bar(range(len(vector)), vector, 0.5)

def show\_dialogue(line\_number):

line = lines\_df\_small.ix['L1045'].Line

doc\_vector = model[line\_number]

show\_image(doc\_vector, line)

1. Now all the method we implemented above to display the movie line

show\_dialogue('L1045')

