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

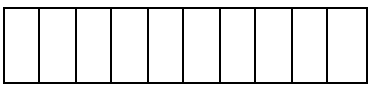
1. Similarity. We can use document vectors to compare texts for similarity. Legal AI software can use document vectors to find similar legal cases
2. Recommendations. For example, online magazines can recommend similar articles based on others that users have read
3. Predictions. Document vectors can 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
```

3. 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)
```

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

```
movie_lines_file = 'data/cornell-movie-dialogs/movie_lines.txt'
```

5. 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)
```

```
In [36]: lines_df.head(10)
```

Out[36]:

LineNumber	Line	Movie	Person
L1045	They do not!	m0	BIANCA
L1044	They do to!	m0	CAMERON
L985	I hope so.	m0	BIANCA
L984	She okay?	m0	CAMERON
L925	Let's go.	m0	BIANCA
L924	Wow	m0	CAMERON
L872	Okay -- you're gonna need to learn how to lie.	m0	BIANCA
L871	No	m0	CAMERON
L870	I'm kidding. You know how sometimes you just become this "persona"? And you don't know how to quit?	m0	BIANCA
L869	Like my fear of wearing pastels?	m0	BIANCA

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

```
len(movie_lines)
```

```
Out[17]: 304713
```

```
movie_lines_df.nunique()
```

```
Out[16]: Line      265786
         Person    5356
         dtype: int64
```

6. 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)
```

7. 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.

8. 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')
```

9. 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())
```

10. 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)
```

11. Look at one of the vectors

```
model['L1045']
```

```
Out[11]: array([-3.2438012e-03, -1.7041313e-03, -1.8726442e-03,  3.7039707e-03,
                 4.4881352e-03,  2.8778086e-03, -2.1008316e-03,  1.2701222e-03,
                -1.6483961e-03,  3.8576641e-03, -2.0334849e-03,  3.4070839e-03,
                 4.6643671e-03,  1.7445823e-03,  3.5253374e-03, -4.5398702e-03,
                -5.3722091e-04,  3.3762513e-04,  2.3369133e-03,  6.1464735e-04,
                 1.6965896e-03,  2.5005301e-03, -1.3133218e-03, -2.5598559e-04,
                 1.7820825e-03, -4.6020881e-03, -8.3403349e-05,  3.2041282e-03,
                 1.9421802e-03,  1.4097937e-03, -2.5338640e-03, -2.0345654e-03,
                -5.9952686e-04,  1.2781341e-03, -9.3458145e-04, -3.0457764e-03,
                 1.7282360e-03,  3.7662087e-03,  2.3310713e-03, -3.2182615e-03,
                 9.1817044e-04, -4.6914183e-03, -3.7409982e-03,  1.9810749e-03,
                -2.0475280e-03, -2.0284038e-03, -7.8300998e-04,  2.9466618e-03,
                -9.8953326e-04,  3.5006870e-04,  1.6866465e-03, -2.5725400e-03,
                -4.5264377e-03,  2.4153560e-04,  2.1136480e-03, -4.9957819e-03,
                 1.9960727e-03, -1.7011663e-03, -1.9883567e-03, -5.1352999e-04,
                -4.9384539e-03, -3.1519190e-03,  4.1736406e-03, -1.3896027e-03,
                 2.5623667e-03,  1.6757578e-03,  3.3900235e-03, -5.9135770e-04,
                -3.3248125e-03, -2.0281426e-03, -2.3372455e-03,  8.2003884e-04,
                 4.7156555e-03, -3.1285842e-03,  7.8359438e-04, -2.2487089e-03,
                 3.3851750e-03, -3.9755837e-03,  2.2105178e-05, -2.0062989e-03,
                -3.3460131e-03, -4.0373662e-03, -2.2721468e-03,  2.1428340e-04,
                 4.0233959e-03,  1.5793082e-03, -1.8830029e-03,  1.5653828e-03,
                 8.4381033e-04, -2.1363373e-03, -2.6181687e-03,  8.4453612e-04,
                 4.3086666e-03,  2.2246004e-03,  4.0573333e-03,  4.1554929e-03,
                -1.0049677e-03,  4.8348685e-03,  4.1375291e-03, -2.7845153e-03],
               dtype=float32)
```

12. Add code to display each vector. We will define the following functions

    a. **show_image** takes a vector and displays it as an image

    b. **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)
```

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

```
show_dialogue('L1045')
```

```
In [71]: show_dialogue('L1045')
They do not!
```