myfasttext

November 9, 2019

0.1 Fasttext Model

In [1]:
```
# import necessary libraries
import pandas as pd
import fasttext
import re
import json
import multiprocessing
import csv
import nltk
from string import punctuation
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import sent_tokenize
from nltk import WordPunctTokenizer
```

In [2]:
```
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('stopwords')
en_stop = set(nltk.corpus.stopwords.words('english'))
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/anjaliverma/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/anjaliverma/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/anjaliverma/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [3]:
```
# read the first 500,000 yelp reviews
lines=open('yelp_dataset/review.json',encoding="utf8").readlines()[:500000]
```

In [4]:
```
# Define function that pre-processes yelp reviews by :
# Converting to lower case
```

```
        # Removing punctuation and non-alphanumeric characters
        # Removing stop words
        # Lemmatizing

        lem = WordNetLemmatizer()

        def preprocess_text(document):
            # Remove all the special characters
            document = re.sub(r'\W', ' ', str(document))

            # remove all single characters
            document = re.sub(r'\s+[a-zA-Z]\s+', ' ', document)

            # Remove single characters from the start
            document = re.sub(r'\^[a-zA-Z]\s+', ' ', document)

            # Substituting multiple spaces with single space
            document = re.sub(r'\s+', ' ', document, flags=re.I)

            # Removing prefixed 'b'
            document = re.sub(r'^b\s+', '', document)

            # Converting to Lowercase
            document = document.lower()

            # Lemmatization
            tokens = document.split()
            tokens = [lem.lemmatize(word) for word in tokens]
            tokens = [word for word in tokens if word not in en_stop]

            preprocessed_text = ' '.join(tokens)

            return preprocessed_text
```

In [5]:
```
        # Define function that extracts text review from json object
        def get_text(line):
            # convert the text line to a json object
            json_object = json.loads(line)
            # read in the text
            text=json_object['text']
            return text
```

In [15]:
```
        # Define function that extracts review label from json object
        def get_label(line):
            # convert the text line to a json object
            json_object = json.loads(line)
            # read the label and convert to an integer
            label=int(json_object['stars'])
            return label
```

```
In [17]: # distribute the processing across the machine cpus
         pool=multiprocessing.Pool(multiprocessing.cpu_count())
         result=pool.map(get_text, lines)
         stars = pool.map(get_label, lines)

In [14]: # result is a list of all text reviews, an example :
         result[0]

Out[14]: 'Total bill for this horrible service? Over $8Gs. These crooks actually had the nerve

In [12]: # Preprocess each text review
         reviews_clean = [preprocess_text(review_text) for review_text in result]

In [53]: # an example
         reviews_clean[0]

Out[53]: 'total bill horrible service 8gs crook actually nerve charge u 69 3 pill checked onli

In [30]: # Create dataset comprising of labels and corresponding text review
         df = pd.DataFrame({'label': stars, 'text':reviews_clean})

In [31]: df.head(5)

Out[31]:    label                                               text
         0      1  total bill horrible service 8gs crook actually...
         1      5  adore travis hard rock new kelly cardenas salo...
         2      5  say office really ha together organized friend...
         3      5  went lunch steak sandwich wa delicious caesar ...
         4      1  today wa second three session paid although fi...

In [32]: # Convert labels to required format by appending __label__ so that it can be recogniz
         df['label']=['__label__'+ str(s) for s in df['label']]
         df['text']= df['text'].replace('\n',' ', regex=True).replace('\t',' ', regex=True)
         df.to_csv(r'yelp_dataset/yelp_reviews_updated.txt', index=False, sep=' ', header=Fals

In [33]: df.head(5)

Out[33]:         label                                               text
         0  __label__1  total bill horrible service 8gs crook actually...
         1  __label__5  adore travis hard rock new kelly cardenas salo...
         2  __label__5  say office really ha together organized friend...
         3  __label__5  went lunch steak sandwich wa delicious caesar ...
         4  __label__1  today wa second three session paid although fi...

In [34]: # Split data into train and test
         !head -n 400000 "yelp_dataset/yelp_reviews_updated.txt" > "yelp_dataset/yelp_reviews_t
         !tail -n 100000 "yelp_dataset/yelp_reviews_updated.txt" > "yelp_dataset/yelp_reviews_t
```

0.2 Model 1 : Bag of word representation - Word level

```
In [35]: # train model
         model = fasttext.train_supervised(input="yelp_dataset/yelp_reviews_train.txt")
         model.save_model("model_word_level.bin")

In [37]: # test model
         model.test("yelp_dataset/yelp_reviews_test.txt")

Out[37]: (100000, 0.67493, 0.67493)

In [38]: model.test("yelp_dataset/yelp_reviews_test.txt", k=5)

Out[38]: (100000, 0.2, 1.0)

In [39]: # Precict a new review
         model.predict("like  leave  low  review  wa  terrible  sum  experience  server  wa  c

Out[39]: (('__label__2',), array([0.57297784]))
```

0.3 Model 2 : Bag of word representation - Ngram level 1-3 grams

```
In [40]: # train model
         model_ngram = fasttext.train_supervised(input="yelp_dataset/yelp_reviews_train.txt", w
         model_ngram.save_model("model_ngram_level.bin")

In [41]: # test model
         model_ngram.test("yelp_dataset/yelp_reviews_test.txt")

Out[41]: (100000, 0.68915, 0.68915)

In [42]: model_ngram.test("yelp_dataset/yelp_reviews_test.txt", k=5)

Out[42]: (100000, 0.2, 1.0)
```

0.4 Model 3 : Bag of word representation - Ngram level 1-3 grams : Change epoch

```
In [43]: # train model
         model_epoch = fasttext.train_supervised(input="yelp_dataset/yelp_reviews_train.txt", w
         model_epoch.save_model("model_epoch.bin")

In [44]: # test model
         model_epoch.test("yelp_dataset/yelp_reviews_test.txt")

Out[44]: (100000, 0.67562, 0.67562)

In [45]: model_epoch.test("yelp_dataset/yelp_reviews_test.txt",k=5)

Out[45]: (100000, 0.2, 1.0)
```

0.5 Model 4 : Bag of word representation - Ngram level 1-3 grams, Keep epochs at 50 (model 3 level) : Change learning rate

```
In [46]: # train model
         model_lr = fasttext.train_supervised(input="yelp_dataset/yelp_reviews_train.txt", word
         model_lr.save_model("model_lr.bin")

In [47]: # test model
         model_lr.test("yelp_dataset/yelp_reviews_test.txt")

Out[47]: (100000, 0.66296, 0.66296)

In [48]: # test model
         model_lr.test("yelp_dataset/yelp_reviews_test.txt", k=5)

Out[48]: (100000, 0.2, 1.0)
```

0.6 Model 5 : Bag of word representation - Ngram level 1-3 grams, Keep epochs at 50 (model 3 level) - Keep learning rate at 1 : Change loss function

```
In [49]: # train model
         model_loss = fasttext.train_supervised(input="yelp_dataset/yelp_reviews_train.txt", wo
         model_loss.save_model("model_loss.bin")

In [50]: # test model
         model_loss.test("yelp_dataset/yelp_reviews_test.txt")

Out[50]: (100000, 0.66882, 0.66882)

In [51]: # test model
         model_loss.test("yelp_dataset/yelp_reviews_test.txt", k=5)

Out[51]: (100000, 0.3173315135481284, 0.98117)

In [56]: model_loss.predict(df["text"][4])

Out[56]: (('__label__1',), array([0.99896502]))

In [ ]:
```