Multiclass_TextClassification_Keras

January 14, 2020

[1]:
```
!pip install gensim
```

```
Requirement already satisfied: gensim in ./anaconda3/lib/python3.7/site-packages
(3.7.3)
Requirement already satisfied: numpy>=1.11.3 in ./anaconda3/lib/python3.7/site-
packages (from gensim) (1.16.4)
Requirement already satisfied: scipy>=0.18.1 in ./anaconda3/lib/python3.7/site-
packages (from gensim) (1.3.1)
Requirement already satisfied: six>=1.5.0 in ./anaconda3/lib/python3.7/site-
packages (from gensim) (1.12.0)
Requirement already satisfied: smart-open>=1.7.0 in
./anaconda3/lib/python3.7/site-packages (from gensim) (1.8.3)
Requirement already satisfied: requests in ./anaconda3/lib/python3.7/site-
packages (from smart-open>=1.7.0->gensim) (2.22.0)
Requirement already satisfied: boto3 in ./anaconda3/lib/python3.7/site-packages
(from smart-open>=1.7.0->gensim) (1.9.146)
Requirement already satisfied: boto>=2.32 in ./anaconda3/lib/python3.7/site-
packages (from smart-open>=1.7.0->gensim) (2.49.0)
Requirement already satisfied: certifi>=2017.4.17 in
./anaconda3/lib/python3.7/site-packages (from requests->smart-
open>=1.7.0->gensim) (2019.6.16)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in
./anaconda3/lib/python3.7/site-packages (from requests->smart-
open>=1.7.0->gensim) (3.0.4)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
./anaconda3/lib/python3.7/site-packages (from requests->smart-
open>=1.7.0->gensim) (1.24.2)
Requirement already satisfied: idna<2.9,>=2.5 in ./anaconda3/lib/python3.7/site-
packages (from requests->smart-open>=1.7.0->gensim) (2.8)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in
./anaconda3/lib/python3.7/site-packages (from boto3->smart-open>=1.7.0->gensim)
(0.9.4)
Requirement already satisfied: s3transfer<0.3.0,>=0.2.0 in
./anaconda3/lib/python3.7/site-packages (from boto3->smart-open>=1.7.0->gensim)
(0.2.0)
Requirement already satisfied: botocore<1.13.0,>=1.12.146 in
./anaconda3/lib/python3.7/site-packages (from boto3->smart-open>=1.7.0->gensim)
```

```
(1.12.146)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1; python_version >=
"2.7" in ./anaconda3/lib/python3.7/site-packages (from
botocore<1.13.0,>=1.12.146->boto3->smart-open>=1.7.0->gensim) (2.8.0)
Requirement already satisfied: docutils>=0.10 in ./anaconda3/lib/python3.7/site-
packages (from botocore<1.13.0,>=1.12.146->boto3->smart-open>=1.7.0->gensim)
(0.15.2)
```

[2]:
```
import logging
import pandas as pd
import numpy as np
from numpy import random
import gensim
import nltk
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
import re
from bs4 import BeautifulSoup
%matplotlib inline

df = pd.read_csv('/Users/spillai/Downloads/stack-overflow-data.csv')
df = df[pd.notnull(df['tags'])]
print(df.head(10))
print(df['post'].apply(lambda x: len(x.split(' '))).sum())
```

```
                                                post           tags
0  what is causing this behavior  in our c# datet...             c#
1  have dynamic html load as if it was in an ifra...        asp.net
2  how to convert a float value in to min:sec  i ...    objective-c
3  .net framework 4 redistributable  just wonderi...           .net
4  trying to calculate and print the mean and its...         python
5  how to give alias name for my website  i have ...        asp.net
6  window.open() returns null in angularjs  it wo...      angularjs
7  identifying server timeout quickly in iphone  ...         iphone
8  unknown method key  error in rails 2.3.8 unit ...  ruby-on-rails
9  from the include  how to show and hide the con...      angularjs
10286120
```

1 Word2vec and Logistic Regression

Word2vec, like doc2vec, belongs to the text preprocessing phase. Specifically, to the part that transforms a text into a row of numbers. Word2vec is a type of mapping that allows words with similar meaning to have similar vector representation.

The idea behind Word2vec is rather simple: we want to use the surrounding words to represent the target words with a Neural Network whose hidden layer encodes the word representation.

First we load a word2vec model. It has been pre-trained by Google on a 100 billion word Google News corpus.

[3]:
```
from gensim.models import Word2Vec

wv = gensim.models.KeyedVectors.load_word2vec_format("/Users/spillai/Downloads/
 ↪GoogleNews-vectors-negative300.bin.gz", binary=True)
wv.init_sims(replace=True)
```

[4]:
```
#Exploring Some Vacabularies

from itertools import islice
list(islice(wv.vocab, 13030, 13050))
```

[4]:
```
['Memorial_Hospital',
 'Seniors',
 'memorandum',
 'elephant',
 'Trump',
 'Census',
 'pilgrims',
 'De',
 'Dogs',
 '###-####_ext',
 'chaotic',
 'forgive',
 'scholar',
 'Lottery',
 'decreasing',
 'Supervisor',
 'fundamentally',
 'Fitness',
 'abundance',
 'Hold']
```

[5]:
```
#BOW based approaches that includes averaging, summation, weighted addition.␣
 ↪The common way is to average
#the two word vectors. Therefore, we will follow the most common way

def word_averaging(wv, words):
    all_words, mean = set(), []

    for word in words:
        if isinstance(word, np.ndarray):
            mean.append(word)
        elif word in wv.vocab:
            mean.append(wv.syn0norm[wv.vocab[word].index])
```

```
            all_words.add(wv.vocab[word].index)

    if not mean:
        logging.warning("cannot compute similarity with no input %s", words)
        # FIXME: remove these examples in pre-processing
        return np.zeros(wv.vector_size,)

    mean = gensim.matutils.unitvec(np.array(mean).mean(axis=0)).astype(np.
 ↪float32)
    return mean

def  word_averaging_list(wv, text_list):
    return np.vstack([word_averaging(wv, post) for post in text_list ])
```

We will tokenize the text and apply the tokenization to "post" column, and apply word vector averaging to tokenized text.

[6]:
```
def w2v_tokenize_text(text):
    tokens = []
    for sent in nltk.sent_tokenize(text, language='english'):
        for word in nltk.word_tokenize(sent, language='english'):
            if len(word) < 2:
                continue
            tokens.append(word)
    return tokens

train, test = train_test_split(df, test_size=0.3, random_state = 42)

test_tokenized = test.apply(lambda r: w2v_tokenize_text(r['post']), axis=1).
 ↪values
train_tokenized = train.apply(lambda r: w2v_tokenize_text(r['post']), axis=1).
 ↪values

X_train_word_average = word_averaging_list(wv,train_tokenized)
X_test_word_average = word_averaging_list(wv,test_tokenized)
```

```
/Users/spillai/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:11:
DeprecationWarning: Call to deprecated `syn0norm` (Attribute will be removed in
4.0.0, use self.vectors_norm instead).
  # This is added back by InteractiveShellApp.init_path()
```

[8]:
```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(n_jobs=1, C=1e5)
logreg = logreg.fit(X_train_word_average, train['tags'])
y_pred = logreg.predict(X_test_word_average)
```

```
/Users/spillai/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
```

```
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/spillai/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:469: FutureWarning: Default
multi_class will be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
```

[9]:
```
from sklearn.metrics import classification_report
my_tags = ['java','html','asp.
 ↪net','c#','ruby-on-rails','jquery','mysql','php','ios','javascript','python','c','css','and
 ↪net']
print('accuracy %s' % accuracy_score(y_pred, test.tags))
print(classification_report(test.tags, y_pred,target_names=my_tags))
```

```
accuracy 0.64375
              precision    recall  f1-score   support

        java       0.62      0.58      0.60       613
        html       0.74      0.76      0.75       620
     asp.net       0.64      0.66      0.65       587
          c#       0.62      0.66      0.64       586
ruby-on-rails       0.70      0.78      0.74       599
      jquery       0.44      0.37      0.40       589
       mysql       0.68      0.63      0.66       594
         php       0.73      0.84      0.78       610
         ios       0.57      0.58      0.57       617
  javascript       0.54      0.52      0.53       587
      python       0.57      0.52      0.54       611
           c       0.63      0.60      0.62       594
         css       0.62      0.60      0.61       619
     android       0.59      0.55      0.57       574
      iphone       0.70      0.73      0.71       584
         sql       0.43      0.44      0.44       578
 objective-c       0.69      0.71      0.70       591
         c++       0.75      0.79      0.77       608
   angularjs       0.82      0.83      0.82       638
        .net       0.70      0.71      0.70       601

    accuracy                           0.64     12000
   macro avg       0.64      0.64      0.64     12000
weighted avg       0.64      0.64      0.64     12000
```

Doc2vec and Logistic Regression

[10]:
```
!pip install tqdm
```

```
Requirement already satisfied: tqdm in ./anaconda3/lib/python3.7/site-packages
(4.41.1)
```

[11]:
```
pip install tqdm --upgrade
```

```
Requirement already up-to-date: tqdm in ./anaconda3/lib/python3.7/site-packages
(4.41.1)
Note: you may need to restart the kernel to use updated packages.
```

[12]:
```
from tqdm import tqdm
tqdm.pandas(desc="progress-bar")
from gensim.models import Doc2Vec
```

```
/Users/spillai/anaconda3/lib/python3.7/site-packages/tqdm/std.py:658:
FutureWarning: The Panel class is removed from pandas. Accessing it from the
top-level namespace will also be removed in the next version
  from pandas import Panel
```

[13]:
```
from sklearn import utils
import gensim
from gensim.models.doc2vec import TaggedDocument
import re
```

[14]:
```
def label_sentences(corpus, label_type):
    """
    Gensim's Doc2Vec implementation requires each document/paragraph to have a␣
 ↪label associated with it.
    We do this by using the TaggedDocument method. The format will be "TRAIN_i"␣
 ↪or "TEST_i" where "i" is
    a dummy index of the post.
    """
    labeled = []
    for i, v in enumerate(corpus):
        label = label_type + '_' + str(i)
        labeled.append(TaggedDocument(v.split(), [label])) ## doc2vec.
 ↪TaggedDocument
    return labeled
```

[15]:
```
X_train, X_test, y_train, y_test = train_test_split(df.post, df.tags,␣
 ↪random_state=0, test_size=0.3)
X_train = label_sentences(X_train, 'Train')
X_test = label_sentences(X_test, 'Test')
all_data = X_train + X_test
```

[16]:
```
all_data[:2]
```

[16]:
```
[TaggedDocument(words=['full-text', 'search', 'with', 'php', 'and', 'pdo',
'not', 'returning', 'any', 'result', 'i', 've', 'searched', 'a', 'lot', 'on',
```

'this', 'matter', 'but', 'i', 'can', 't', 'find', 'out', 'what', 's', 'wrong',
'with', 'my', 'setup.', 'i', 'm', 'trying', 'to', 'do', 'a', 'full-text',
'search', 'using', 'pdo', 'and', 'php', 'but', 'i', 'don', 't', 'get', 'any',
'results', 'or', 'error', 'messages', 'at', 'all.', 'my', 'table', 'contains',
'customer', 'details:', '<pre><code>id', 'int(11)', 'auto_increment', 'name',
'varchar(150)', 'lastname', 'varchar(150)', 'company', 'varchar(250)', 'adress',
'varchar(150)', 'postcode', 'int(5)', 'city', 'varchar(150)', 'email',
'varchar(250)', 'phone', 'varchar(20)', 'orgnr', 'varchar(15)', 'timestamp',
'timestamp', 'current_timestamp', '</code></pre>', 'i', 'did', 'run', 'the',
'sql-query:', '<pre><code>alter', 'table', 'system_customer', 'add',
'fulltext(name', 'lastname', '...', '...)', '</code></pre>', 'except', 'for',
'the', 'columns', 'id', 'postcode', 'and', 'timestamp', '.', 'no', 'signs',
'of', 'any', 'trouble', 'so', 'far.', 'i', 'have', 'no', 'idea', 'if', 'the',
'problem', 'lies', 'in', 'my', 'db', 'configuration', 'or', 'my', 'php', 'code',
'so', 'here', 'goes', 'the', 'php:', '<pre><code>$sth', '=',
'$dbh->prepare(', 'select', 'name', 'lastname', 'company', 'adress', 'city',
'phone', 'email', 'orgnr', 'from', '.$db_pre.', 'customer', 'where',
'match(name', 'lastname', 'company', 'adress', 'city', 'phone', 'email',
'orgnr)', 'against(:search', 'in', 'boolean', 'mode)', ');', '//bind',
'placeholders', '$sth->bindparam(', ':search', '$data);',
'$sth->execute();', '$rows', '=', '$sth->fetchall();', '//just', 'for',
'testing', 'print_r($dbh->errorinfo());', 'if(empty($rows))', '{', 'echo',
'[...]', ';', '}', 'else', '{', 'echo', '[...]', ';', 'foreach', '($rows',
'as', '$row)', '{', 'echo', '<tr', 'data-href=', 'new_order.php', 'cid=',
'.$row[', 'id', '].', '>', ';', 'echo', '<td>', '.$row[', 'name', '].',
'</td>', ';', 'echo', '<td>', '.$row[', 'lastname', '].',
'</td>', ';', 'echo', '<td>', '.$row[', 'company', '].',
'</td>', ';', 'echo', '<td>', '.$row[', 'phone', '].',
'</td>', ';', 'echo', '<td>', '.$row[', 'email', '].',
'</td>', ';', 'echo', '<td>', '.date(', 'y-m-d', 'strtotime($row[',
'timestamp', '])).', '</td>', ';', 'echo', '</tr>', ';', '}',
'echo', '[...]', ';', '}', '</code></pre>', 'i', 'tried', 'to', 'change',
'the', 'parameter', 'in', 'the', 'searchquery', 'to', 'a', 'string', 'like',
'<pre><code>against(', 'testcompany', 'somename', 'in', 'boolean', 'mode)',
'</code></pre>', 'i', 'also', 'read', 'about', 'that', 'if', 'a', 'word', 'is',
'found', 'in', '50%', 'or', 'more', 'of', 'the', 'rows', 'it', 'counts', 'as',
'a', 'common', 'word.', 'i', 'm', 'pretty', 'sure', 'that', 's', 'not', 'the',
'case', 'here', '(uses', 'very', 'specific', 'words)', 'my', 'table', 'uses',
'myisam', 'engine', 'i', 'don', 't', 'get', 'any', 'results', 'or', 'any',
'error', 'messages.', 'please', 'help', 'my', 'point', 'out', 'what', 's',
'wrong', 'thank', 'you'], tags=['Train_0']),
 TaggedDocument(words=['select', 'everything', 'from', '1', 'table', 'and',
'only', 'x', 'rows', 'from', 'another', 'im', 'making', 'a', 'join', 'query',
'like:', '<pre><code>select', '*', 'from', 'clothes', 'as', 'c', 'join',
'style', 'as', 's', 'on', 'c.styleid', '=', 's.sylelid', 'where', 'clothesid',
'=', '19', '</code></pre>', 'but', 'i', 'dont', 'want', 'to', 'select',
'everything', 'from', 'style', 'i', 'want', 'to', 'select', 'everything',

```
'from', 'clothes', '(20', 'rows)', 'and', 'only', 'select', '1', 'row', '(from',
'10)', 'from', 'style', 'what', 'is', 'the', 'easyest', 'way', 'to', 'do',
'this', 'without', 'having', 'to', 'select', 'every', 'row', 'from', 'clothes',
'(with', '20', 'things', 'to', 'select)', 'like:', '<pre><code>select', 'c.id',
'c.description', 'c.name', 'c.size', 'c.brand', 's.name', 'from', 'clothes',
'as', 'c', 'join', 'style', 'as', 's', 'on', 'c.styleid', '=', 'st.sylelid',
'where', 'clothesid', '=', '19', '</code></pre>', 'what', 'would', 'be', 'the',
'fastest', 'way', 'or', 'is', 'this', 'the', 'only', 'possibillity'],
tags=['Train_1'])]
```

When training the doc2vec, we will vary the following parameters:

dm=0 , distributed bag of words (DBOW) is used. vector_size=300 , 300 vector dimensional feature vectors. negative=5 , specifies how many "noise words" should be drawn. min_count=1, ignores all words with total frequency lower than this. alpha=0.065 , the initial learning rate. We initialize the model and train for 30 epochs.

[17]:
```
model_dbow = Doc2Vec(dm=0, vector_size=300, negative=5, min_count=1, alpha=0.
↪065, min_alpha=0.065)
model_dbow.build_vocab([x for x in tqdm(all_data)])

for epoch in range(30):
    model_dbow.train(utils.shuffle([x for x in tqdm(all_data)]),␣
↪total_examples=len(all_data), epochs=1)
    model_dbow.alpha -= 0.002
    model_dbow.min_alpha = model_dbow.alpha
```

```
100%|| 40000/40000 [00:00<00:00, 2604187.26it/s]
100%|| 40000/40000 [00:00<00:00, 3516498.85it/s]
100%|| 40000/40000 [00:00<00:00, 2875026.30it/s]
100%|| 40000/40000 [00:00<00:00, 3243603.74it/s]
100%|| 40000/40000 [00:00<00:00, 2925051.17it/s]
100%|| 40000/40000 [00:00<00:00, 2481249.41it/s]
100%|| 40000/40000 [00:00<00:00, 3017973.41it/s]
100%|| 40000/40000 [00:00<00:00, 2414161.59it/s]
100%|| 40000/40000 [00:00<00:00, 2879961.55it/s]
100%|| 40000/40000 [00:00<00:00, 2985269.75it/s]
100%|| 40000/40000 [00:00<00:00, 2555749.26it/s]
100%|| 40000/40000 [00:00<00:00, 2732223.11it/s]
100%|| 40000/40000 [00:00<00:00, 2718454.86it/s]
100%|| 40000/40000 [00:00<00:00, 2852879.88it/s]
100%|| 40000/40000 [00:00<00:00, 2874779.99it/s]
100%|| 40000/40000 [00:00<00:00, 2698125.79it/s]
100%|| 40000/40000 [00:00<00:00, 2714979.53it/s]
100%|| 40000/40000 [00:00<00:00, 2577184.90it/s]
100%|| 40000/40000 [00:00<00:00, 2694139.68it/s]
100%|| 40000/40000 [00:00<00:00, 2947197.41it/s]
100%|| 40000/40000 [00:00<00:00, 3004085.37it/s]
100%|| 40000/40000 [00:00<00:00, 2937444.80it/s]
100%|| 40000/40000 [00:00<00:00, 2650052.28it/s]
```

```
100%|| 40000/40000 [00:00<00:00, 2962079.10it/s]
100%|| 40000/40000 [00:00<00:00, 2896868.86it/s]
100%|| 40000/40000 [00:00<00:00, 2865744.74it/s]
100%|| 40000/40000 [00:00<00:00, 2853996.09it/s]
100%|| 40000/40000 [00:00<00:00, 2745951.75it/s]
100%|| 40000/40000 [00:00<00:00, 2738377.27it/s]
100%|| 40000/40000 [00:00<00:00, 2753297.12it/s]
100%|| 40000/40000 [00:00<00:00, 2729289.58it/s]
```

[18]:
```python
def get_vectors(model, corpus_size, vectors_size, vectors_type):
    """
    Get vectors from trained doc2vec model
    :param doc2vec_model: Trained Doc2Vec model
    :param corpus_size: Size of the data
    :param vectors_size: Size of the embedding vectors
    :param vectors_type: Training or Testing vectors
    :return: list of vectors
    """
    vectors = np.zeros((corpus_size, vectors_size))
    for i in range(0, corpus_size):
        prefix = vectors_type + '_' + str(i)
        vectors[i] = model.docvecs[prefix]
    return vectors

train_vectors_dbow = get_vectors(model_dbow, len(X_train), 300, 'Train')
test_vectors_dbow = get_vectors(model_dbow, len(X_test), 300, 'Test')
```

[19]:
```python
logreg = LogisticRegression(n_jobs=1, C=1e5)
logreg.fit(train_vectors_dbow, y_train)
logreg = logreg.fit(train_vectors_dbow, y_train)
y_pred = logreg.predict(test_vectors_dbow)
print('accuracy %s' % accuracy_score(y_pred, y_test))
print(classification_report(y_test, y_pred,target_names=my_tags))
```

```
/Users/spillai/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/spillai/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:469: FutureWarning: Default
multi_class will be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)
/Users/spillai/anaconda3/lib/python3.7/site-
packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver
will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/Users/spillai/anaconda3/lib/python3.7/site-
```

```
packages/sklearn/linear_model/logistic.py:469: FutureWarning: Default
multi_class will be changed to 'auto' in 0.22. Specify the multi_class option to
silence this warning.
  "this warning.", FutureWarning)

accuracy 0.79925
              precision    recall  f1-score   support

          java       0.69      0.67      0.68       589
          html       0.87      0.89      0.88       661
       asp.net       0.92      0.93      0.92       606
            c#       0.78      0.75      0.76       613
ruby-on-rails       0.82      0.88      0.85       601
        jquery       0.70      0.70      0.70       585
         mysql       0.86      0.79      0.82       621
           php       0.83      0.86      0.84       587
           ios       0.72      0.69      0.70       560
    javascript       0.67      0.62      0.64       611
        python       0.63      0.65      0.64       593
             c       0.78      0.80      0.79       581
           css       0.81      0.77      0.79       608
       android       0.84      0.88      0.86       593
        iphone       0.84      0.81      0.83       592
           sql       0.68      0.63      0.66       597
  objective-c       0.86      0.88      0.87       604
           c++       0.91      0.94      0.92       610
     angularjs       0.93      0.94      0.94       595
          .net       0.80      0.86      0.83       593

      accuracy                           0.80     12000
     macro avg       0.80      0.80      0.80     12000
  weighted avg       0.80      0.80      0.80     12000
```

BOW with Keras

Finally, we are going to do a text classification with Keras which is a Python Deep Learning library.

The following code were largely taken from a Google workshop. The process is like this:

Separate the data into training and test sets. Use tokenizer methods to count the unique words in our vocabulary and assign each of those words to indices. Calling fit_on_texts() automatically creates a word index lookup of our vocabulary. We limit our vocabulary to the top words by passing a num_words param to the tokenizer. With our tokenizer, we can now use the texts_to_matrix method to create the training data that we'll pass our model. We feed a one-hot vector to our model. After we transform our features and labels in a format Keras can read, we are ready to build our text classification model. When we build our model, all we need to do is tell Keras the shape of our input data, output data, and the type of each layer. keras will look after the rest. When training the model, we'll call the fit() method, pass it our training data and labels, batch size and epochs.

[ ]:
```
### KERAS BAG OF WORDS
#https://github.com/tensorflow/workshops/blob/master/extras/keras-bag-of-words/
 ↪keras-bow-model.ipynb
```

[20]:
```
import itertools
import os

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import tensorflow as tf

from sklearn.preprocessing import LabelBinarizer, LabelEncoder
from sklearn.metrics import confusion_matrix

from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.preprocessing import text, sequence
from keras import utils
```

```
Using TensorFlow backend.
```

[21]:
```
train_size = int(len(df) * .7)
train_posts = df['post'][:train_size]
train_tags = df['tags'][:train_size]

test_posts = df['post'][train_size:]
test_tags = df['tags'][train_size:]
```

[30]:
```
print ("Train size: %d" % train_size)
print ("Test size: %d" % (len(df) - train_size))
```

```
Train size: 28000
Test size: 12000
```

Text tokenization utility class.

This class allows to vectorize a text corpus, by turning each text into either a sequence of integers (each integer being the index of a token in a dictionary) or into a vector where the coefficient for each token could be binary, based on word count, based on tf-idf...

[22]:
```
max_words = 1000
# Only the top 1000 words by word count are considered
tokenize = text.Tokenizer(num_words=max_words, char_level=False)
#Tokenizer is a class #
tokenize.fit_on_texts(train_posts) # only fit on train
```

Once fit, the Tokenizer provides 4 attributes that you can use to query what has been learned about your documents:

word_counts: A dictionary of words and their counts. word_docs: A dictionary of words and how many documents each appeared in. word_index: A dictionary of words and their uniquely assigned integers. document_count:An integer count of the total number of documents that were used to fit the Tokenizer.

[36]:
```
#-----The Tokenizer API has been well explained in the API below ----
#https://machinelearningmastery.com/prepare-text-data-deep-learning-keras/
#print(tokenize.word_counts)
#print(tokenize.document_count)
#print(tokenize.word_index)
#print(tokenize.word_docs)
```

28000

[23]:
```
x_train = tokenize.texts_to_matrix(train_posts) # There are 4 modes available␣
 ↪here. It is Binary by default
x_test = tokenize.texts_to_matrix(test_posts)
```

Label Encoder

```
Encode target labels with value between 0 and n_class-1
This transformer should be used to encode target values, i.e. y, and not the input X
```

[39]:
```
#Example - Not Required in the model
#https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.
 ↪LabelEncoder.html
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
le.fit([1, 2, 2, 6])
le.classes_
```

[39]:
```
array([1, 2, 6])
```

[24]:
```
encoder = LabelEncoder() # To convert target labels in numerical categories
encoder.fit(train_tags)
y_train = encoder.transform(train_tags)
y_test = encoder.transform(test_tags)
```

[40]:
```
np.max(y_train)
```

[40]:
```
1.0
```

[45]:
```
#Example to_Categorical
#Converts a class vector (integers) to binary class matrix.
# Consider an array of 5 labels out of a set of 3 classes {0, 1, 2}:
labels=np.array([0, 2, 1, 2, 0])
utils.to_categorical(labels)
```

[45]:
```
array([[1., 0., 0.],
       [0., 0., 1.],
       [0., 1., 0.],
       [0., 0., 1.],
```

```
       [1., 0., 0.]], dtype=float32)
```

[25]:
```
num_classes = np.max(y_train) + 1
y_train = utils.to_categorical(y_train, num_classes) # C
y_test = utils.to_categorical(y_test, num_classes)
```

[26]:
```
batch_size = 32
epochs = 2

# Build the model
model = Sequential()
model.add(Dense(512, input_shape=(max_words,)))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes))
model.add(Activation('softmax'))
```

```
WARNING:tensorflow:From /Users/spillai/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:66: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /Users/spillai/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:541: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /Users/spillai/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:4432: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /Users/spillai/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:148: The name
tf.placeholder_with_default is deprecated. Please use
tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /Users/spillai/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:3733: calling dropout (from
tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed
in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 -
keep_prob`.
```

[27]:
```
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

history = model.fit(x_train, y_train,
                    batch_size=batch_size,
```

```
                    epochs=epochs,
                    verbose=1,
                    validation_split=0.1)
```

```
WARNING:tensorflow:From /Users/spillai/anaconda3/lib/python3.7/site-
packages/keras/optimizers.py:793: The name tf.train.Optimizer is deprecated.
Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From /Users/spillai/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:3576: The name tf.log is
deprecated. Please use tf.math.log instead.

WARNING:tensorflow:From /Users/spillai/anaconda3/lib/python3.7/site-
packages/tensorflow/python/ops/math_grad.py:1250:
add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Train on 25200 samples, validate on 2800 samples
Epoch 1/2
25200/25200 [==============================] - 5s 182us/step - loss: 1.0199 -
acc: 0.7092 - val_loss: 0.6477 - val_acc: 0.8061
Epoch 2/2
25200/25200 [==============================] - 4s 167us/step - loss: 0.5564 -
acc: 0.8209 - val_loss: 0.6181 - val_acc: 0.8032
```

[28]:
```
score = model.evaluate(x_test, y_test,
                       batch_size=batch_size, verbose=1)
print('Test accuracy:', score[1])
```

```
12000/12000 [==============================] - 0s 23us/step
Test accuracy: 0.79975
```

[ ]:
```

```