Week8_1_BOW_doc2vec

May 31, 2021

Bag of words, Tokenizer, and Doc2Vec

- This is simple approach of bag of Words, Tokenizer, and Doc2Vec

[1]:
```
# Tenorflow
import tensorflow as tf
import numpy as np

# Tenorflow Padding Sequences
from keras.preprocessing.sequence import pad_sequences

from gensim.models.doc2vec import Doc2Vec, TaggedDocument
from nltk.tokenize import word_tokenize

import warnings
warnings.filterwarnings('ignore')
```

```
/home/jayanthikishore/anaconda3/lib/python3.8/site-
packages/gensim/similarities/__init__.py:15: UserWarning: The
gensim.similarities.levenshtein submodule is disabled, because the optional
Levenshtein package <https://pypi.org/project/python-Levenshtein/> is
unavailable. Install Levenhstein (e.g. `pip install python-Levenshtein`) to
suppress this warning.
  warnings.warn(msg)
```

[2]:
```
tweets = ['there is a snake in my boot', 'there is boot snake in my house', 'a␣
↪a a a']
labels = [75, 12, 50]
```

[3]:
```
tweets = [tweet.split(' ') for tweet in tweets]
unique_words = np.unique(tweets)
unique_words, tweets
```

[3]:
```
(array([list(['a', 'a', 'a', 'a']),
        list(['there', 'is', 'a', 'snake', 'in', 'my', 'boot']),
        list(['there', 'is', 'boot', 'snake', 'in', 'my', 'house'])],
       dtype=object),
 [['there', 'is', 'a', 'snake', 'in', 'my', 'boot'],
  ['there', 'is', 'boot', 'snake', 'in', 'my', 'house'],
```

```
       ['a', 'a', 'a', 'a']])
```

[4]:
```
tokenizer = {}

counter = 0
for tweet in tweets:
  for word in tweet:
    if word not in tokenizer:
      tokenizer[word] = counter
      counter += 1

tokenizer
```

[4]:
```
{'there': 0,
 'is': 1,
 'a': 2,
 'snake': 3,
 'in': 4,
 'my': 5,
 'boot': 6,
 'house': 7}
```

Bag of words

[5]:
```
bag_words = []
# count_count = [0]*len(unique_words)

for tweet in tweets:
  word_count = [0]*(counter)

  #Counts instence of every unique word that appears
  for word in tweet:
    locWord = tokenizer[word] # Get the index location of the words
    word_count[locWord] += 1 # Counts the number of times that word appears

  # Append after finnished counting
  bag_words.append(word_count)

bag_words
```

[5]:
```
[[1, 1, 1, 1, 1, 1, 1, 0], [1, 1, 0, 1, 1, 1, 1, 1], [0, 0, 4, 0, 0, 0, 0, 0]]
```

[6]:
```
word_count
```

[6]:
```
[0, 0, 4, 0, 0, 0, 0, 0]
```

[7]:
```
# Twitter length
token_len = 50

# Create Decorator
tokenizer = tf.keras.preprocessing.text.Tokenizer() # Sets up the Tikenizer␣
↪which we will feed

# Fitting the Tokenizer and building our Corpus
tokenizer.fit_on_texts(tweets)

# Create our sequence
X = tokenizer.texts_to_sequences(tweets)

# Padding the text
X = tf.keras.preprocessing.sequence.pad_sequences(X, maxlen=token_len, padding=␣
↪'post', truncating='post')

# Convert array to Tensor
X = tf.constant(X, dtype=tf.int64)
y = tf.constant(labels, dtype=tf.int64)
```

[8]:
```
X.shape
```

[8]:
```
TensorShape([3, 50])
```

[9]:
```
y.shape
```

[9]:
```
TensorShape([3])
```

Doc2Vector

[10]:
```
sampledata = ["Data scientists should know the Mathematics, statistics, and␣
↪programming",
              "Data scientist familiar in machine learning deep learning and␣
↪artificial intelligence",
              "Data Scientist should know python and R coding",
              "Data Scientist should know the how to train test and validation␣
↪of the model",
               "Metrics is more important for model validation"]

tag_sample = [TaggedDocument(words = word_tokenize(dat.lower()), tags␣
↪=[str(i)]) for i,dat in enumerate(sampledata)]
tag_sample
```

[10]:
```
[TaggedDocument(words=['data', 'scientists', 'should', 'know', 'the',
'mathematics', ',', 'statistics', ',', 'and', 'programming'], tags=['0']),
 TaggedDocument(words=['data', 'scientist', 'familiar', 'in', 'machine',
```

```
'learning', 'deep', 'learning', 'and', 'artificial', 'intelligence'],
tags=['1']),
 TaggedDocument(words=['data', 'scientist', 'should', 'know', 'python', 'and',
'r', 'coding'], tags=['2']),
 TaggedDocument(words=['data', 'scientist', 'should', 'know', 'the', 'how',
'to', 'train', 'test', 'and', 'validation', 'of', 'the', 'model'], tags=['3']),
 TaggedDocument(words=['metrics', 'is', 'more', 'important', 'for', 'model',
'validation'], tags=['4'])]
```

[11]:
```
# import nltk
# # nltk.download()
# from gensim.models.doc2vec import Doc2Vec, Tagdoc
# from nltk.tokenize import word_tokenize

# data = ["I love machine learning. Its awesome.",
#         "I love coding in python",
#         "I love building chatbots",
#         "they chat amagingly well"]

# tagged_data = [Tagdoc(words=word_tokenize(_d.lower()), tags=[str(i)]) for i,␣
 ↪_d in enumerate(data)]
# tagged_data
```

[12]:
```
import multiprocessing
from gensim.models.doc2vec import Doc2Vec
cores = multiprocessing.cpu_count()

doc2vecmodel = Doc2Vec(dm=1, vector_size=20, alpha=0.025,negative=5,␣
 ↪hs=0,min_count=1,min_alpha=0.00025,epochs=50)

doc2vecmodel.build_vocab(tag_sample)

for epoch in range(doc2vecmodel.epochs):
    print("Iteration number {0}".format(epoch))
    doc2vecmodel.train(tag_sample,total_examples=doc2vecmodel.corpus_count,
                       epochs=doc2vecmodel.epochs)
    #decrease the learning rate
    doc2vecmodel.alpha -=0.0002
    #fix the learning rate, no decay
    doc2vecmodel.min_alpha=doc2vecmodel.alpha

#save the model
doc2vecmodel.save("/home/jayanthikishore/Downloads/doc2vec.model")
print("Model Successfully Saved")
```

```
Iteration number 0
Iteration number 1
```

```
Iteration number 2
Iteration number 3
Iteration number 4
Iteration number 5
Iteration number 6
Iteration number 7
Iteration number 8
Iteration number 9
Iteration number 10
Iteration number 11
Iteration number 12
Iteration number 13
Iteration number 14
Iteration number 15
Iteration number 16
Iteration number 17
Iteration number 18
Iteration number 19
Iteration number 20
Iteration number 21
Iteration number 22
Iteration number 23
Iteration number 24
Iteration number 25
Iteration number 26
Iteration number 27
Iteration number 28
Iteration number 29
Iteration number 30
Iteration number 31
Iteration number 32
Iteration number 33
Iteration number 34
Iteration number 35
Iteration number 36
Iteration number 37
Iteration number 38
Iteration number 39
Iteration number 40
Iteration number 41
Iteration number 42
Iteration number 43
Iteration number 44
Iteration number 45
Iteration number 46
Iteration number 47
Iteration number 48
Iteration number 49
```

```
Model Successfully Saved
```

[13]:
```
print(doc2vecmodel)
```

```
Doc2Vec(dm/m,d20,n5,w5,s0.001,t3)
```

[15]:
```
#Access the saved model file
doc2vecmodel= Doc2Vec.load("/home/jayanthikishore/Downloads/doc2vec.model")

#to find the vector of a document which is not in the training data
test_line = word_tokenize("Data Scientist calculates metrics for every model")
vec = doc2vecmodel.infer_vector(test_line)
print("Infer: ",vec)

#to find most similar doc using tags
sim_doc = doc2vecmodel.docvecs.most_similar("1")
print(sim_doc)

# otherwise
print(doc2vecmodel.docvecs["1"])
```

```
Infer:  [ 0.22599038  0.03601338 -0.14775823 -0.07204668  0.2689113  -0.00141598
  0.10879105 -0.10913793  0.07871564 -0.08696382 -0.06835343  0.2569299
 -0.45772108 -0.30933982 -0.14404337  0.33680007  0.16130973 -0.07385437
  0.09844366 -0.0040016 ]
[('4', 0.2607925534248352), ('2', 0.23374393582344055), ('0',
0.11019726097583771), ('3', -0.009904157370328903)]
[-1.677322    0.9392231  -1.5109582  -0.26544833  2.6698425  -1.1711501
 -2.0134318  -2.188202    1.5914671  -1.6037436   1.5444556   0.6695683
 -3.3728755   0.05669191 -0.37545332 -0.52540493 -0.53095543 -3.1815312
 -2.2176416   0.25225586]
```

[ ]:

[ ]: