A Detailed Guide to understand the Word Embeddings and Embedding Layer in Keras.

[Don't forget to upvote ;)]

In this kernel I have explained the keras embedding layer. To do so I have created a sample corpus of just 3 documents and that should be sufficient to explain the working of the keras embedding layer.

Embeddings are useful in a variety of machine learning applications. Because of the fact I have attached many data sources to the kernel where I fell that embeddings and Keras embedding layer may prove to be useful.

Before diving in let us skim through some of the applications of the embeddings :

1) The first application that strikes me is in the Collaborative Filtering based Recommender Systems where we have to create the user embeddings and the movie embeddings by decomposing the utility matrix which contains the user-item ratings.

To see a complete tutorial on CF based recommender systems using embeddings in Keras you can follow this kernel of mine.

2) The second use is in the Natural Language Processing and its related applications where we have to create the word embeddings for all the words present in the documents of our corpus.

This is the terminology that I shall use in this kernel.

Thus the embedding layer in Keras can be used when we want to create the embeddings to embed higher dimensional data into lower dimensional vector space.

In [ ]:

IMPORTING MODULES

In [ ]:
```
# Ignore  the warnings
import warnings
warnings.filterwarnings('always')
warnings.filterwarnings('ignore')

# data visualisation and manipulation
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns
#configure
# sets matplotlib to inline and displays graphs below the corressponding cell.
%matplotlib inline
style.use('fivethirtyeight')
sns.set(style='whitegrid',color_codes=True)

#nltk
import nltk

#stop-words
from nltk.corpus import stopwords
stop_words=set(nltk.corpus.stopwords.words('english'))

# tokenizing
from nltk import word_tokenize,sent_tokenize

#keras
import keras
from keras.preprocessing.text import one_hot,Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense , Flatten ,Embedding,Input
from keras.models import Model
```

In [ ]:

CREATING SAMPLE CORPUS OF DOCUMENTS ie TEXTS

In [ ]:
```
sample_text_1="bitty bought a bit of butter"
sample_text_2="but the bit of butter was a bit bitter"
sample_text_3="so she bought some better butter to make the bitter butter better"

corp=[sample_text_1,sample_text_2,sample_text_3]
no_docs=len(corp)
```

INTEGER ENCODING ALL THE DOCUMENTS

After this all the unique words will be reprsented by an integer. For this we are using **one_hot** function from the Keras. Note that the **vocab_size** is specified large enough so as to ensure **unique integer encoding** for each and every word.

Note one important thing that the integer encoding for the word remains same in different docs. eg 'butter' is denoted by 31 in each and every document.

In [ ]:
```
vocab_size=50
encod_corp=[]
for i,doc in enumerate(corp):
    encod_corp.append(one_hot(doc,50))
    print("The encoding for document",i+1," is : ",one_hot(doc,50))
```

PADDING THE DOCS (to make very doc of same length)

The Keras Embedding layer requires all individual documents to be of same length. Hence we wil pad the shorter documents with 0 for now. Therefore now in Keras Embedding layer the '**input_length**' will be equal to the length (ie no of words) of the document with maximum length or maximum number of words.

To pad the shorter documents I am using **pad_sequences** functon from the Keras library.

In [ ]:
```
# length of maximum document. will be nedded whenever create embeddings for the words
maxlen=-1
for doc in corp:
    tokens=nltk.word_tokenize(doc)
    if(maxlen<len(tokens)):
        maxlen=len(tokens)
print("The maximum number of words in any document is : ",maxlen)
```

In [ ]:
```
# now to create embeddings all of our docs need to be of same length. hence we can pad the docs with zeros.
pad_corp=pad_sequences(encod_corp,maxlen=maxlen,padding='post',value=0.0)
print("No of padded documents: ",len(pad_corp))
```

In [ ]:
```
for i,doc in enumerate(pad_corp):
    print("The padded encoding for document",i+1," is : ",doc)
```

ACTUALLY CREATING THE EMBEDDINGS using KERAS EMBEDDING LAYER

Now all the documents are of same length (after padding). And so now we are ready to create and use the embeddings.

I will embed the words into vectors of 8 dimensions.

In [ ]:
```
# specifying the input shape
input=Input(shape=(no_docs,maxlen),dtype='float64')
```

In [ ]:
```
'''
shape of input.
each document has 12 element or words which is the value of our maxlen variable.

'''
word_input=Input(shape=(maxlen,),dtype='float64')

# creating the embedding
word_embedding=Embedding(input_dim=vocab_size,output_dim=8,input_length=maxlen)(word_input)

word_vec=Flatten()(word_embedding) # flatten
embed_model =Model([word_input],word_vec) # combining all into a Keras model
```

PARAMETERS OF THE EMBEDDING LAYER ---

'input_dim' = the vocab size that we will choose. In other words it is the number of unique words in the vocab.

'output_dim' = the number of dimensions we wish to embed into. Each word will be represented by a vector of this much dimensions.

'input_length' = lenght of the maximum document. which is stored in maxlen variable in our case.

In [ ]:
```
embed_model.compile(optimizer=keras.optimizers.Adam(lr=1e-3),loss='binary_crossentropy',metrics=['acc'])
# compiling the model. parameters can be tuned as always.
```

In [ ]:
```
print(type(word_embedding))
print(word_embedding)
```

In [ ]:
```
print(embed_model.summary()) # summary of the model
```

In [ ]:
```
embeddings=embed_model.predict(pad_corp) # finally getting the embeddings.
```

In [ ]:
```
print("Shape of embeddings : ",embeddings.shape)
print(embeddings)
```

In [ ]:
```
embeddings=embeddings.reshape(-1,maxlen,8)
print("Shape of embeddings : ",embeddings.shape)
print(embeddings)
```

In [ ]:

The resulting shape is (3,12,8).

3---> no of documents

12---> each document is made of 12 words which was our maximum length of any document.

& 8---> each word is 8 dimensional.

In [ ]:

GETTING ENCODING FOR A PARTICULAR WORD IN A SPECIFIC DOCUMENT

In [ ]:
```
for i,doc in enumerate(embeddings):
    for j,word in enumerate(doc):
        print("The encoding for ",j+1,"th word","in",i+1,"th document is : \n\n",word)
```

Now this makes it easier to visualize that we have 3(size of corp) documents with each consisting of 12(maxlen) words and each word mapped to a 8-dimensional vector.

In [ ]:

HOW TO WORK WITH A REAL PIECE OF TEXT

Just like above we can now use any other document. We can sent_tokenize the doc into sentences.

Each sentence has a list of words which we will integer encode using the 'one_hot' function as below.

Now each sentence will be having different number of words. So we will need to pad the sequences to the sentence with maximum words.

At this point we are ready to feed the input to Keras Embedding layer as shown above.

'input_dim' = the vocab size that we will choose

'output_dim' = the number of dimensions we wish to embed into

'input_length' = lenght of the maximum document

In [ ]:

THE END !!!

If you want to see the application of Keras embedding layer on a real task eg text classification then please check out my this repo on Github in which I have used the embeddings to perform sentiment analysis on IMdb movie review dataset.

[Please Do upvote the kernel;)]

In [ ]: