Stance Detection for the Fake News Challenge

Identifying Textual Relationships with Deep Neural Nets

Check the problem context here.

Download files required for the project from here.

Step1: Load the given dataset

1. Mount the google drive
2. Import Glove embeddings
3. Import the test and train datasets

Mount the google drive to access required project files

Run the below commands

```
from google.colab import drive

drive.mount('/content/drive/')
```

```
Drive already mounted at /content/drive/; to attempt to forcibly remount, call drive.mount("/content/drive/", force_rem
```

Path for Project files on google drive

Note: You need to change this path according where you have kept the files in google drive.

```
project_path = "/content/drive/My Drive/FakenewsDetection/"
project_path = '/content/drive/My Drive/FakeNewsDetection/'
```

Loading the Glove Embeddings

```
from zipfile import ZipFile
with ZipFile(project_path+'glove.6B.zip', 'r') as z:
  z.extractall()
```

Load the dataset [5 Marks]

1. Using read_csv() in pandas load the given train datasets files **`train_bodies.csv`** and **`train_stances.csv`**
2. Using merge command in pandas merge the two datasets based on the Body ID.

Note: Save the final merged dataset in a dataframe with name **`dataset`** .

```
import pandas as pd
```

```
train_bodies_df=pd.read_csv(project_path+'train_bodies.csv')
```

```
train_bodies_df.head(5)
```

	Body ID	articleBody
0	0	A small meteorite crashed into a wooded area i...
1	4	Last week we hinted at what was to come as Ebo...
2	5	(NEWSER) – Wonder how long a Quarter Pounder w...
3	6	Posting photos of a gun-toting child online, I...
4	7	At least 25 suspected Boko Haram insurgents we...

```
train_stances_df=pd.read_csv(project_path+'train_stances.csv')
```

```
train_stances_df.head(5)
```

	Headline	Body ID	Stance
0	Police find mass graves with at least '15 bodi...	712	unrelated
1	Hundreds of Palestinians flee floods in Gaza a...	158	agree
2	Christian Bale passes on role of Steve Jobs, a...	137	unrelated
3	HBO and Apple in Talks for $15/Month Apple TV ...	1034	unrelated
4	Spider burrowed through tourist's stomach and ...	1923	disagree

```
dataset=pd.merge(train_bodies_df, train_stances_df, how ='inner', on ='Body ID')
```

```
dataset.head(5)
```

	Body ID	articleBody	Headline	Stance
0	0	A small meteorite crashed into a wooded area i...	Soldier shot, Parliament locked down after gun...	unrelated
1	0	A small meteorite crashed into a wooded area i...	Tourist dubbed ‘Spider Man’ after spider burro...	unrelated
2	0	A small meteorite crashed into a wooded area i...	Luke Somers 'killed in failed rescue attempt i...	unrelated
3	0	A small meteorite crashed into a wooded area i...	BREAKING: Soldier shot at War Memorial in Ottawa	unrelated
4	0	A small meteorite crashed into a wooded area i...	Giant 8ft 9in catfish weighing 19 stone caught...	unrelated

Check1:

You should see the below output if you run `dataset.head()` command as given below

```
dataset.head()
```

	Body ID	articleBody	Headline	Stance
0	0	A small meteorite crashed into a wooded area i...	Soldier shot, Parliament locked down after gun...	unrelated
1	0	A small meteorite crashed into a wooded area i...	Tourist dubbed ‘Spider Man’ after spider burro...	unrelated
2	0	A small meteorite crashed into a wooded area i...	Luke Somers 'killed in failed rescue attempt i...	unrelated
3	0	A small meteorite crashed into a wooded area i...	BREAKING: Soldier shot at War Memorial in Ottawa	unrelated
4	0	A small meteorite crashed into a wooded area i...	Giant 8ft 9in catfish weighing 19 stone caught...	unrelated

Step2: Data Pre-processing and setting some hyper parameters needed for model

Run the code given below to set the required parameters.

1. `MAX_SENTS` = Maximum no.of sentences to consider in an article.
2. `MAX_SENT_LENGTH` = Maximum no.of words to consider in a sentence.
3. `MAX_NB_WORDS` = Maximum no.of words in the total vocabualry.
4. `MAX_SENTS_HEADING` = Maximum no.of sentences to consider in a heading of an article.

```
MAX_NB_WORDS = 20000
MAX_SENTS = 20
MAX_SENTS_HEADING = 1
MAX_SENT_LENGTH = 20
VALIDATION_SPLIT = 0.2
```

Download the `Punkt` from nltk using the commands given below. This is for sentence tokenization.

For more info on how to use it, read this.

```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

Tokenizing the text and loading the pre-trained Glove word embeddings for each tol [5 marks]

Keras provides Tokenizer API for preparing text. Read it before going any further.

Import the Tokenizer from keras preprocessing text

```
from keras.preprocessing import text
```

Initialize the Tokenizer class with maximum vocabulary count as `MAX_NB_WORDS` initialized at the start of step2.

```
texttokenizer=text.Tokenizer(num_words=MAX_NB_WORDS,lower=True, split=" ")
```

Now, using fit_on_texts() from Tokenizer class, lets encode the data

Note: We need to fit articleBody and Headline also to cover all the words.

```
texttokenizer.fit_on_texts(dataset['articleBody'])
texttokenizer.fit_on_texts(dataset['Headline'])
```

fit_on_texts() gives the following attributes in the output as given here.

- **word_counts:** dictionary mapping words (str) to the number of times they appeared on during fit. Only set after fit_on_texts was ca
- **word_docs:** dictionary mapping words (str) to the number of documents/texts they appeared on during fit. Only set after fit_on_tex called.
- **word_index:** dictionary mapping words (str) to their rank/index (int). Only set after fit_on_texts was called.
- **document_count:** int. Number of documents (texts/sequences) the tokenizer was trained on. Only set after fit_on_texts or fit_on_sequences was called.

Now, tokenize the sentences using nltk sent_tokenize() and encode the senteces with the ids we got form th above `t.word_index`

Initialise 2 lists with names `texts` and `articles`.

```
texts = [] to store text of article as it is.
```

```
articles = [] split the above text into a list of sentences.
```

```
texts=[]
articles=[]


texts=dataset['articleBody']


dataset['articles']=dataset['articleBody'].fillna("").map(nltk.sent_tokenize)


articles=dataset['articles']
```

Check 2:

first element of texts and articles should be as given below.

```
texts[0]
```

```
'A small meteorite crashed into a wooded area in Nicaragua\'s capital of Managua overnight, the government said Sunday.
```

```
articles[0]
```

```
["A small meteorite crashed into a wooded area in Nicaragua's capital of Managua overnight, the government said Sunday.
 "Residents reported hearing a mysterious boom that left a 16-foot deep crater near the city's airport, the Associated
 'Government spokeswoman Rosario Murillo said a committee formed by the government to study the event determined it was
 'House-sized asteroid 2014 RC, which measured 60 feet in diameter, skimmed the Earth this weekend, ABC News reports.',
 'Murillo said Nicaragua will ask international experts to help local scientists in understanding what happened.',
 'The crater left by the meteorite had a radius of 39 feet and a depth of 16 feet,  said Humberto Saballos, a volcanolo
 'He said it is still not clear if the meteorite disintegrated or was buried.',
 'Humberto Garcia, of the Astronomy Center at the National Autonomous University of Nicaragua, said the meteorite could
 '"We have to study it more because it could be ice or rock," he said.',
 'Wilfried Strauch, an adviser to the Institute of Territorial Studies, said it was "very strange that no one reported
 'We have to ask if anyone has a photo or something."',
 "Local residents reported hearing a loud boom Saturday night, but said they didn't see anything strange in the sky.",
 '"I was sitting on my porch and I saw nothing, then all of a sudden I heard a large blast.',
 'We thought it was a bomb because we felt an expansive wave," Jorge Santamaria told The Associated Press.',
 "The site of the crater is near Managua's international airport and an air force base.",
 'Only journalists from state media were allowed to visit it.']
```

Now iterate through each article and each sentence to encode the words into ids us t.word_index [5 marks]

Here, to get words from sentence you can use `text_to_word_sequence` from keras preprocessing text.

1. Import text_to_word_sequence
2. Initialize a variable of shape (no.of articles, MAX_SENTS, MAX_SENT_LENGTH) with name `data` with zeros first (you can use num np.zeros to initialize with all zeros)and then update it while iterating through the words and sentences in each article.

```
from keras.preprocessing.text import text_to_word_sequence
```

```
no_of_articles=articles.shape[0]

shape=(no_of_articles,MAX_SENTS,MAX_SENT_LENGTH)

import numpy as np

data=np.zeros(shape=shape)


for i, sentence in enumerate(articles):
            for j, sent in enumerate(sentence):
                if j < MAX_SENTS and sent.strip():
                    words = text_to_word_sequence(sent)
                    k = 0
                    for w in words:
                        if k < MAX_SENT_LENGTH:
                            if w in texttokenizer.word_index:
                                data[i][j][k] = texttokenizer.word_index[w]
                            k += 1

data[0,:,:]
```

```
array([[3.0000e+00, 4.8100e+02, 4.2700e+02, 7.2110e+03, 8.1000e+01,
         3.0000e+00, 3.7330e+03, 3.3100e+02, 5.0000e+00, 3.8910e+03,
         3.5000e+02, 4.0000e+00, 1.4310e+03, 2.9580e+03, 1.0000e+00,
         8.9000e+01, 1.2000e+01, 4.6400e+02, 0.0000e+00, 0.0000e+00],
        [7.5800e+02, 9.5000e+01, 1.0470e+03, 3.0000e+00, 2.6790e+03,
         1.7520e+03, 7.0000e+00, 1.8900e+02, 3.0000e+00, 1.2170e+03,
         1.0750e+03, 2.0300e+03, 7.0000e+02, 1.5900e+02, 1.0000e+00,
         3.0320e+03, 4.4800e+02, 1.0000e+00, 5.5500e+02, 2.3500e+02],
        [8.9000e+01, 1.0670e+03, 4.1150e+03, 2.3490e+03, 1.2000e+01,
         3.0000e+00, 1.0920e+03, 3.3060e+03, 1.9000e+01, 1.0000e+00,
         8.9000e+01, 2.0000e+00, 1.7930e+03, 1.0000e+00, 5.2100e+02,
         2.0090e+03, 1.5000e+01, 9.0000e+00, 3.0000e+00, 3.1110e+03],
        [1.8100e+02, 3.6400e+03, 9.7200e+02, 2.0000e+02, 2.5560e+03,
         4.4000e+01, 6.7750e+03, 1.7220e+03, 1.2520e+03, 5.0000e+00,
         1.3317e+04, 1.7936e+04, 1.0000e+00, 7.7800e+02, 3.1000e+01,
         7.4000e+02, 3.9900e+03, 6.7000e+01, 8.5000e+01, 0.0000e+00],
        [2.3490e+03, 1.2000e+01, 1.5570e+03, 3.8000e+01, 1.0940e+03,
         3.5100e+02, 7.7500e+02, 2.0000e+00, 3.6700e+02, 2.6000e+02,
         1.7700e+03, 5.0000e+00, 4.4500e+03, 7.0000e+01, 4.9400e+02,
         0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00],
        [1.0000e+00, 7.0000e+02, 1.8900e+02, 1.9000e+01, 1.0000e+00,
         4.2700e+02, 3.2000e+01, 3.0000e+00, 7.4170e+03, 4.0000e+00,
         2.1590e+03, 1.2520e+03, 6.0000e+00, 3.0000e+00, 5.2700e+03,
         4.0000e+00, 1.2170e+03, 1.2520e+03, 1.2000e+01, 3.3630e+03],
        [1.3000e+01, 1.2000e+01, 1.5000e+01, 8.0000e+00, 1.4900e+02,
         2.5000e+01, 5.4300e+02, 6.4000e+01, 1.0000e+00, 4.2700e+02,
         3.7270e+03, 4.1000e+01, 9.0000e+00, 1.8500e+03, 0.0000e+00,
         0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00],
        [3.3630e+03, 5.7330e+03, 4.0000e+00, 1.0000e+00, 5.8750e+03,
         6.1400e+02, 2.1000e+01, 1.0000e+00, 3.1100e+02, 3.4380e+03,
         7.9400e+02, 4.0000e+00, 1.5570e+03, 1.2000e+01, 1.0000e+00,
         4.2700e+02, 6.9000e+01, 2.3000e+01, 7.8700e+02, 2.0000e+00],
        [3.7000e+01, 1.7000e+01, 2.0000e+00, 1.7930e+03, 1.5000e+01,
         5.2000e+01, 1.2000e+02, 1.5000e+01, 6.9000e+01, 2.3000e+01,
         4.9210e+03, 4.1000e+01, 1.9630e+03, 1.3000e+01, 1.2000e+01,
         0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00],
        [4.7360e+03, 3.3380e+03, 2.4000e+01, 3.9690e+03, 2.0000e+00,
         1.0000e+00, 1.3160e+03, 4.0000e+00, 3.0720e+03, 1.6530e+03,
         1.2000e+01, 1.5000e+01, 9.0000e+00, 1.9500e+02, 1.4200e+03,
         7.0000e+00, 5.8000e+01, 4.0000e+01, 9.5000e+01, 3.0000e+00],
        [3.7000e+01, 1.7000e+01, 2.0000e+00, 1.0940e+03, 6.4000e+01,
         5.1000e+02, 2.0000e+01, 3.0000e+00, 2.5000e+02, 4.1000e+01,
         2.6400e+02, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
         0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00],
        [2.6000e+02, 7.5800e+02, 9.5000e+01, 1.0470e+03, 3.0000e+00,
         1.8060e+03, 1.7520e+03, 5.3100e+02, 2.7600e+02, 2.9000e+01,
         1.2000e+01, 3.3000e+01, 7.0300e+02, 1.6300e+02, 8.9200e+02,
         1.4200e+03, 5.0000e+00, 1.0000e+00, 2.0810e+03, 0.0000e+00],
        [3.5000e+01, 9.0000e+00, 2.0570e+03, 1.0000e+01, 1.1600e+02,
         5.8250e+03, 6.0000e+00, 3.5000e+01, 5.7600e+02, 6.5600e+02,
         1.0400e+02, 5.9000e+01, 4.0000e+00, 3.0000e+00, 2.4100e+03,
         3.5000e+01, 2.4100e+02, 3.0000e+00, 5.1200e+02, 1.9110e+03],
        [3.7000e+01, 3.4100e+02, 1.5000e+01, 9.0000e+00, 3.0000e+00,
         2.0820e+03, 1.2000e+02, 3.7000e+01, 8.8100e+02, 2.4000e+01,
         4.4510e+03, 2.5840e+03, 4.3150e+03, 4.9220e+03, 5.5000e+01,
         1.0000e+00, 5.5500e+02, 2.3500e+02, 0.0000e+00, 0.0000e+00],
        [1.0000e+00, 2.5500e+02, 4.0000e+00, 1.0000e+00, 7.0000e+02,
         8.0000e+00, 1.5900e+02, 3.9610e+03, 3.5100e+02, 4.4800e+02,
         6.0000e+00, 2.4000e+01, 1.5500e+02, 4.6500e+02, 1.9290e+03,
         0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00],
        [1.2600e+02, 9.2100e+02, 2.2000e+01, 4.7000e+01, 1.0000e+02,
         3.6000e+01, 1.8330e+03, 2.0000e+00, 1.2120e+03, 1.5000e+01,
         0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
         0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00],
        [0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
         0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
         0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
         0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00],
        [0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
         0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
         0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
         0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00],
        [0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
         0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
         0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
         0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00],
        [0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
         0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
         0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
         0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00]])
```

Check 3:

Accessing first element in data should give something like given below.

```
data[0, :, :]
```

```
array([[    3,   487,   474,  7113,    79,     3,  3687,   325,     5,
         4200,   361,     4,  1525,  2913,     1,    89,    12,   451,
            0,     0],
       [  743,    96,  1044,     3,  2814,  1759,     7,   186,     3,
         1219,  1070,  1987,   736,   154,     1,  2990,   458,     1,
          543,   232],
       [   89,  1052,  4057,  2314,    12,     3,  1073,  3248,    19,
            1,    89,     2,  1751,     1,   518,  1980,    15,     9,
            3,  2879],
       [  182,  3691,   976,   196,  2515,    42,  6688,  1691,  1227,
            5, 13011, 17379,     1,   762,    30,   722,  3931,    66,
           87,     0],
       [ 2314,    12,  1882,    38,  1076,   346,   793,     2,   356,
          261,  1782,     5,  4396,    67,   486,     0,     0,     0,
            0,     0],
       [    1,   736,   186,    19,     1,   474,    32,     3,  7307,
            4,  2122,  1227,     6,     3,  5195,     4,  1219,  1227,
           12,  3308],
       [   13,    12,    15,     8,   143,    25,   531,    63,     1,
          474,  3679,    41,     9,  1825,     0,     0,     0,     0,
            0,     0],
       [ 3308,  5643,     4,     1,  5788,   620,    22,     1,   302,
         3125,   786,     4,  1882,    12,     1,   474,    70,    23,
          801,     2],
       [   35,    17,     2,  1751,    15,    54,   119,    15,    70,
           23,  4850,    41,  1885,    13,    12,     0,     0,     0,
            0,     0],
       [ 4664,  3279,    24,  3915,     2,     1,  1298,     4,  3028,
         1630,    12,    15,     9,   187,  1423,     7,    56,    40,
           96,     3],
       [   35,    17,     2,  1076,    63,   497,    20,     3,   252,
           41,   260,     0,     0,     0,     0,     0,     0,     0,
            0,     0],
       [  261,   743,    96,  1044,     3,  1765,  1759,   520,   273,
           29,    12,    33,   702,   160,   818,  1423,     5,     1,
         2068,     0],
       [   34,     9,  2035,    10,   112,  5741,     6,    34,   562,
          644,   104,    57,     4,     3,  2382,    34,   238,     3,
          504,  1922],
       [   35,   341,    15,     9,     3,  2053,   119,    35,   872,
           24,  4397,  2541,  4258,  4851,    55,     1,   543,   232,
            0,     0],
       [    1,   254,     4,     1,   736,     8,   154,  4116,   346,
          458,     6,    24,   152,   460,  1908,     0,     0,     0,
            0,     0],
       [ 124,   896,    21,    48,   102,    37,  1803,     2,  1195,
          15,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0],
       [   0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0],
       [   0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0],
       [   0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0],
       [   0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0]], dtype=int32)
```

Repeat the same process for the `Headings` as well. Use variables with names `texts_heading` and `articles_heading` accordingly. [5 marks]

```
texts_heading=[]
articles_heading=[]
```

```
texts_heading=dataset['Headline']
```

```
dataset['Headlines']=dataset['Headline'].fillna("").map(nltk.sent_tokenize)
```

```
articles_heading=dataset['Headlines']
```

```
texts_heading[0]
```

```
'Soldier shot, Parliament locked down after gunfire erupts at war memorial'
```

```
articles_heading[0]
```

```
['Soldier shot, Parliament locked down after gunfire erupts at war memorial']
```

```
no_of_articlesheading=articles_heading.shape[0]
```

```
shape=(no_of_articlesheading)
```

```
Headings=np.zeros(shape=shape)
```

```
for i, sentence in enumerate(texts_heading):
            for j, sent in enumerate(sentence):
                if j < MAX_SENTS and sent.strip():
                    words = text_to_word_sequence(sent)
                    k = 0
                    for w in words:
                        if k < MAX_SENT_LENGTH:
                            if w in texttokenizer.word_index:
                                #Headings[i][j][k] = texttokenizer.word_index[w]
                                np.append(Headings, texttokenizer.word_index[w])
                            k += 1
```

```
Headings=[]
for sentence in texts_heading:
  words=text_to_word_sequence(sentence)
  for w in words:
    if w in texttokenizer.word_index:
      #print(texttokenizer.word_index[w])
      Headings.append(texttokenizer.word_index[w])
```

```
Headings=np.array(Headings)
```

```
Headings=Headings[:49972]
```

```
Headings.shape
```

```
(49972,)
```

```
Headings[0]
```

```
718
```

Now the features are ready, lets make the labels ready for the model to process.

Convert labels into one-hot vectors

You can use get_dummies in pandas to create one-hot vectors.

```
labels=pd.get_dummies(Headings)
```

Check 4:

The shape of data and labels shoould match the given below numbers.

```
print('Shape of data tensor:', data.shape)
print('Shape of label tensor:', labels.shape)
```

```
Shape of data tensor: (49972, 20, 20)
Shape of label tensor: (49972,)
```

Shuffle the data

```
## get numbers upto no.of articles
indices = np.arange(data.shape[0])
## shuffle the numbers
np.random.shuffle(indices)


## shuffle the data
```

```
data = data[indices]
Headings = Headings[indices]
## shuffle the labels according to data
labels = labels[indices]
```

Split into train and validation sets. Split the train set 80:20 ratio to get the train and validation sets.

Use the variable names as given below:

x_train, x_val - for body of articles.

x-heading_train, x_heading_val - for heading of articles.

y_train - for training labels.

y_val - for validation labels.

Check 5:

The shape of x_train, x_val, y_train and y_val should match the below numbers.

```
print(x_train.shape)
print(y_train.shape)

print(x_val.shape)
print(y_val.shape)
```

```
(39978, 20, 20)
(39978, 4)
(9994, 20, 20)
(9994, 4)
```

Create embedding matrix with the glove embeddings

Run the below code to create embedding matrix which has all the words and their glove embedding if present in glove word list.

```
# load the whole embedding into memory
embeddings_index = dict()
f = open('./glove.6B.100d.txt')
for line in f:
  values = line.split()
  word = values[0]
  coefs = np.asarray(values[1:], dtype='float32')
  embeddings_index[word] = coefs
f.close()
print('Loaded %s word vectors.' % len(embeddings_index))

# create a weight matrix for words in training docs
embedding_matrix = np.zeros((vocab_size, 100))


for word, i in t.word_index.items():
  embedding_vector = embeddings_index.get(word)
  if embedding_vector is not None:
    embedding_matrix[i] = embedding_vector
```

```
Loaded 400000 word vectors.
```

Try the sequential model approach and report the accuracy score. [10 marks]

Import layers from Keras to build the model

Model

▾ Compile and fit the model

▾ Build the same model with attention layers included for better performance (Optional)

▾ Fit the model and report the accuracy score for the model with attention layer (Optional)