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/')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=9473189

Enter your authorization code:
··········
Mounted at /content/drive/
```

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/Data Science/"
```

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_sta**
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 = pd.read_csv('/content/drive/My Drive/Data Science/train_bodies.csv')
train_stances = pd.read_csv('/content/drive/My Drive/Data Science/train_stances.csv')


dataset = pd.merge(train_bodies , train_stances , on = 'Body ID')

dataset.tail()
```

	Body ID	articleBody	
49967	2532	ANN ARBOR, Mich. – A pizza delivery man in Mic...	Pizza delivery man gets tipped mo
49968	2532	ANN ARBOR, Mich. – A pizza delivery man in Mic...	Pizza delivery man
49969	2532	ANN ARBOR, Mich. – A pizza delivery man in Mic...	Luckiest Pizza Delivery Guy Ever (
49970	2532	ANN ARBOR, Mich. – A pizza delivery man in Mic...	Ann Arbor pizza delivery driver s
49971	2532	ANN ARBOR, Mich. – A pizza delivery man in Mic...	Ann Arbor pizza delivery driver s

Check1:

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

```
dataset.head()
```

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

Step2: Data Pre-processing and setting some hyper parameters neede

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 sente

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]   Unzipping tokenizers/punkt.zip.
True
```

Tokenizing the text and loading the pre-trained Glove word eml [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.text import Tokenizer
```

```
Using TensorFlow backend.
```

Initialize the Tokenizer class with maximum vocabulary count as `MAX_NB_WORDS` initialized

```
tk = Tokenizer(MAX_NB_WORDS)
```

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.

```
codetext = dataset['articleBody'] + dataset['Headline']
tk.fit_on_texts(codetext)
```

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

```
tk.word_counts
```

```
('unpublished', 90),
('anecdotal', 334),
('findings', 248),
('testing', 2330),
('baboons', 90),
('bottom', 1030),
('great', 6190),
('someone’s', 224),
('working', 12988),
('male', 4420),
('control', 15084),
('highly', 2736),
('unlikely', 2380),
('particular', 2112),
('product', 4726),
('reach', 4600),
('market', 5918),
('5', 11944),
('shooting', 22738),
('california', 4110),
('marine', 3003),
('corps', 528),
('anonymous', 2206),
('sparked', 2050),
('panic', 938),
('north', 40458),
('san', 3424),
('diego', 404),
('shooter', 3146),
('active', 2930),
('installation', 106),
('“all', 1366),
('marines', 1070),
('instructed', 262),
('lock', 1106),
('inside', 16244),
('rooms', 490),
('stupidly', 90),
('followed', 5594),
('prank', 1658),
('post', 24360),
('“updates”', 90),
('situation', 7224),
('shared', 5938),
('hundreds', 4938),
('police', 46945),
('insider', 1882),
('evidence', 7656),
('“saw', 90),
('networks', 1198),
('telling', 4548),
('dispatcher', 144),
('“it’s', 2962),
('rumor', 6989),
('hearsay', 322),
('possible', 9686),
('camp', 1472),
```

```
...]/
```