

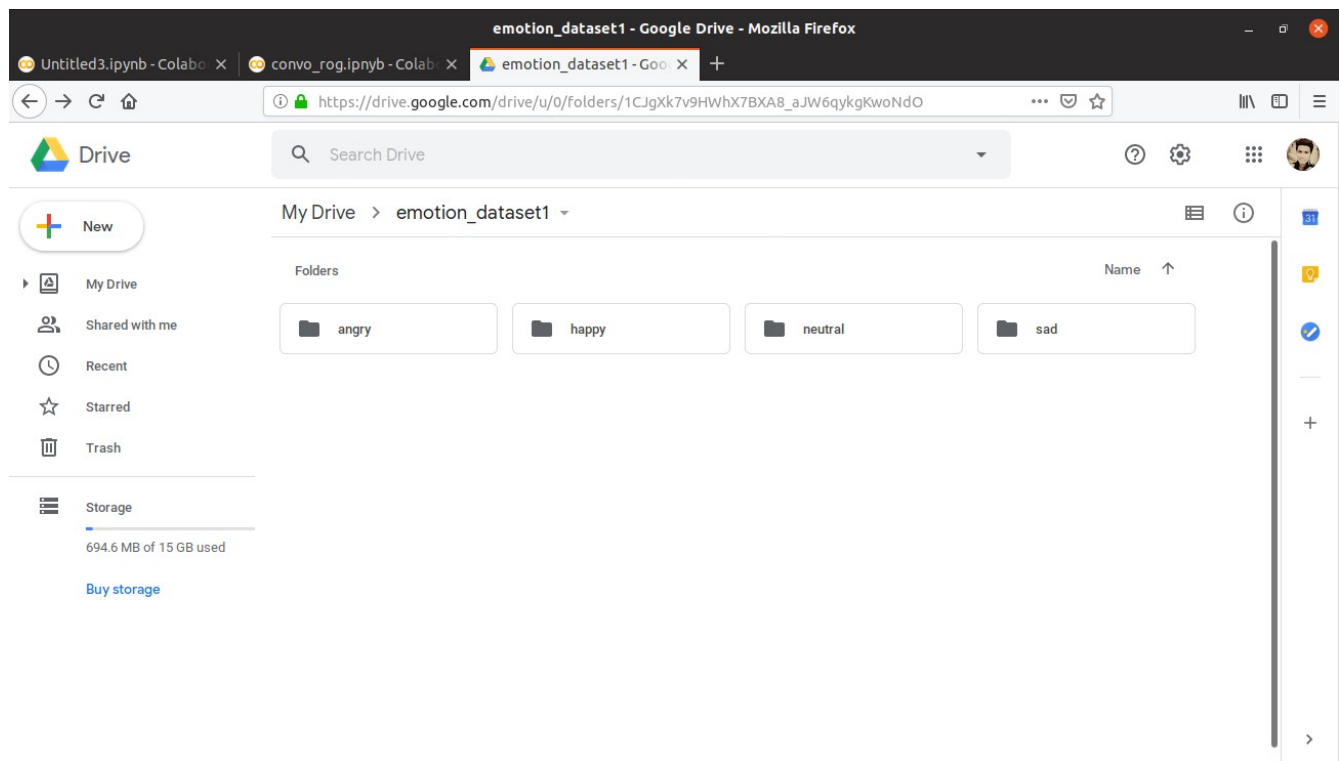
Emotion Classification on Youtube Comments

Abstract:

Youtube is one of the most popular video sharing platform. A person can react to a video by commenting on the video. A comment may contain an emotion that can be identified automatically. In this study, we conducted experiments on emotion classification on Youtube comments. Manually labelled using 3 basic emotion label (happy, sad, angry) and one neutral label. Word embedding is a popular technique in NLP, and have been used in many classification tasks. Here we are using Convolutional Neural Network (CNN) algorithm.

1. Dataset

We took Comments dataset from the oscar nominating trailer called “Three Billboards Outside Ebbing Missouri”. Dataset contains 1500 comments and we are labelling it with the emotions as Happy, Sad, Angry and Neutral.



2. Result on sample set of data:

Dataset contains 1500 Comments and we are manually labelling it. We have trained the model on 48 comments and got accuracy of 37.5%.

3. Expected results:

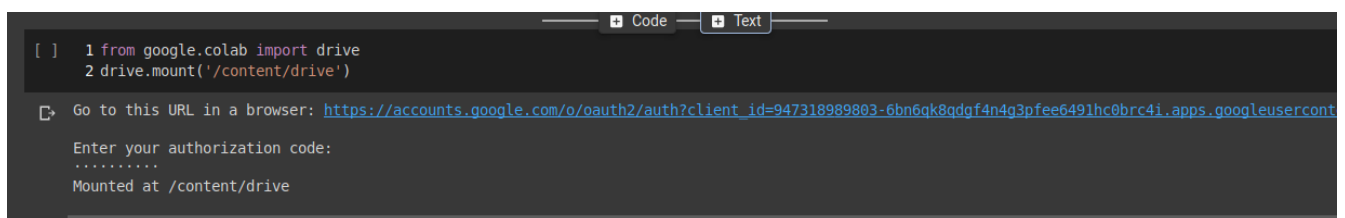
We are expecting the accuracy between 80 to 90%.

4. Algorithms and Techniques used:

We have used Convolution Neural Network(CNN) algorithm.

Snippets of Code

1. Mounting Google drive



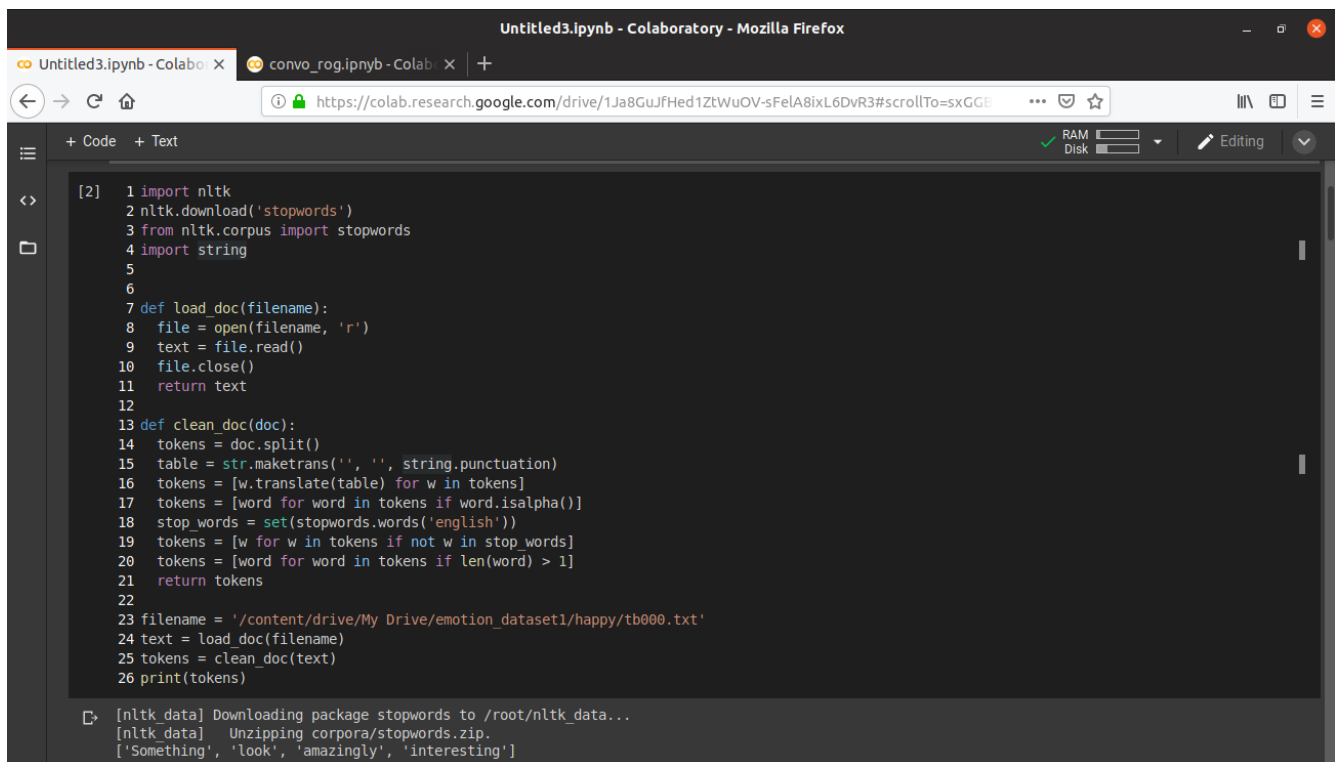
```
[ ] 1 from google.colab import drive
    2 drive.mount('/content/drive')
```

Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com

Enter your authorization code:
.....

Mounted at /content/drive

2. Loading the Stopwords, Files from google drive and removing the punctuations and non-alphabetic tokens



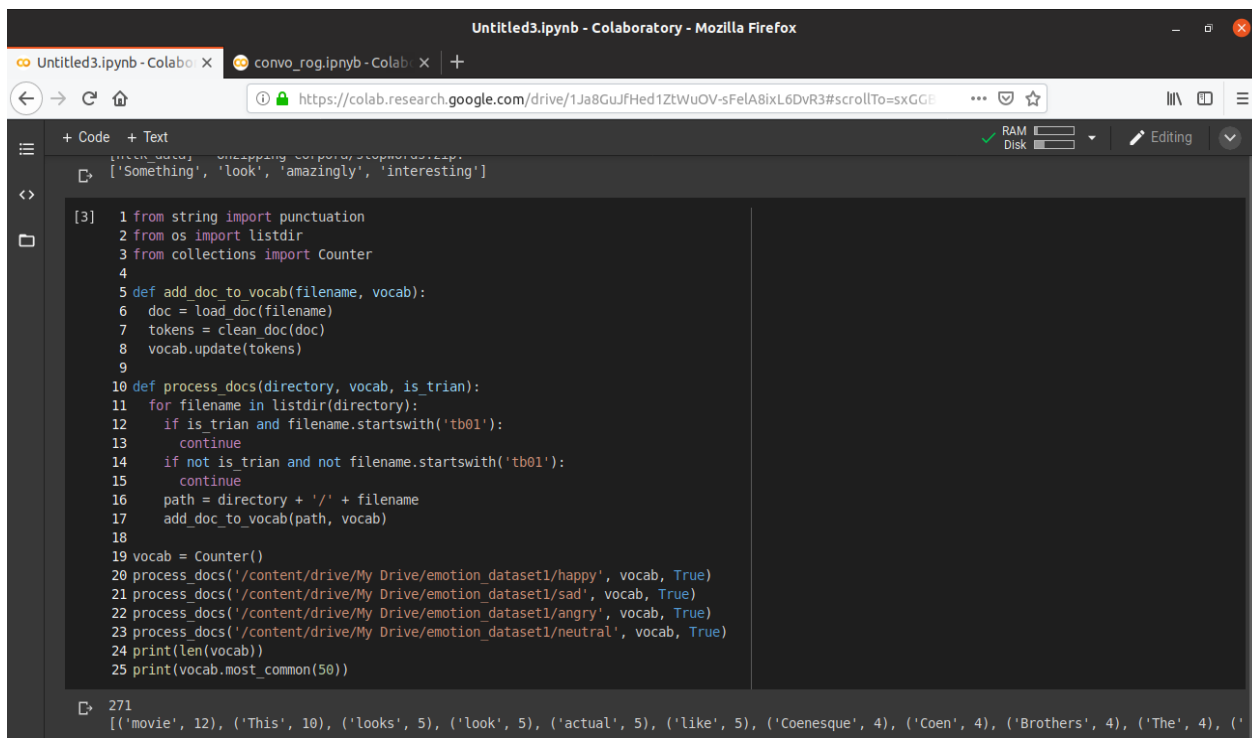
The screenshot shows a JupyterLab notebook interface in Mozilla Firefox. The notebook has two tabs: 'Untitled3.ipynb - Colab' and 'convo_rog.ipynb - Colab'. The active tab is 'Untitled3.ipynb - Colab'. The code in the notebook is as follows:

```
[2] 1 import nltk
    2 nltk.download('stopwords')
    3 from nltk.corpus import stopwords
    4 import string
    5
    6
    7 def load_doc(filename):
    8     file = open(filename, 'r')
    9     text = file.read()
   10     file.close()
   11     return text
   12
   13 def clean_doc(doc):
   14     tokens = doc.split()
   15     table = str.maketrans('', '', string.punctuation)
   16     tokens = [w.translate(table) for w in tokens]
   17     tokens = [word for word in tokens if word.isalpha()]
   18     stop_words = set(stopwords.words('english'))
   19     tokens = [w for w in tokens if not w in stop_words]
   20     tokens = [word for word in tokens if len(word) > 1]
   21     return tokens
   22
   23 filename = '/content/drive/My Drive/emotion_dataset1/happy/tb000.txt'
   24 text = load_doc(filename)
   25 tokens = clean_doc(text)
   26 print(tokens)
```

The output of the code is:

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Unzipping corpora/stopwords.zip.
['Something', 'look', 'amazingly', 'interesting']
```

3. Files are passed through the function `process_docs` in which we'll check that the files are in training or testing data and defining the Counter which is unordered collections with their count.



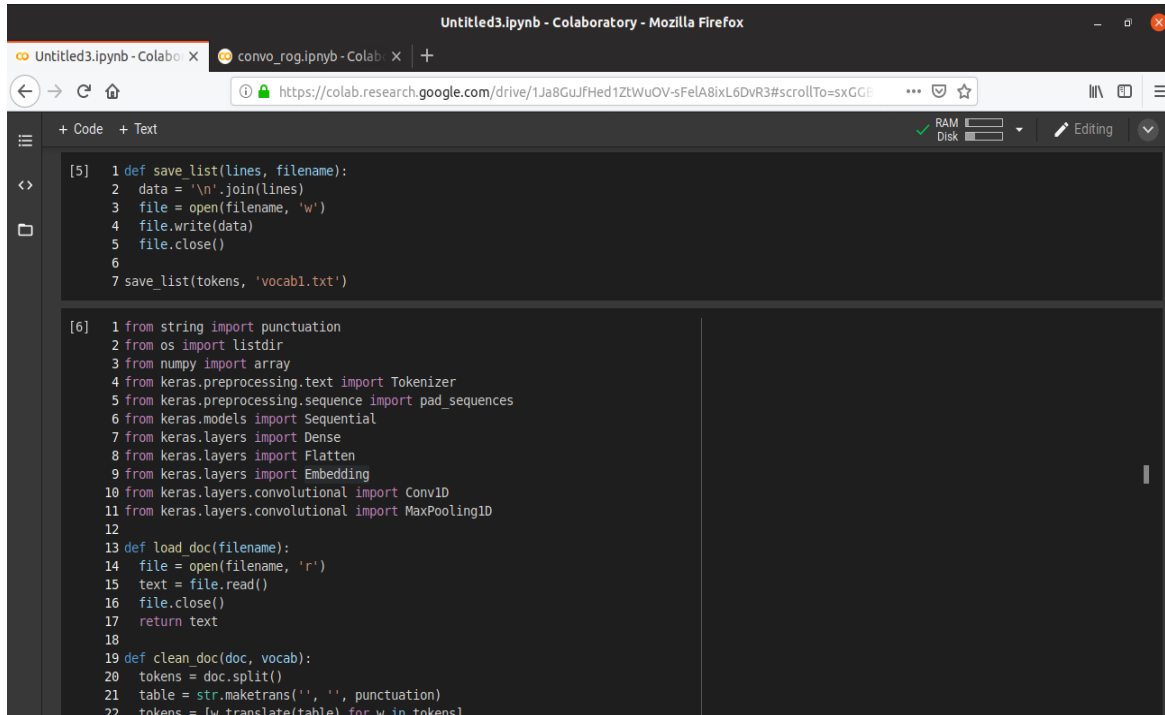
The screenshot shows a JupyterLab notebook interface in Mozilla Firefox. The notebook has two tabs: 'Untitled3.ipynb - Colab' and 'convo_rog.ipynb - Colab'. The active tab is 'Untitled3.ipynb - Colab'. The code in the notebook is as follows:

```
[3] 1 from string import punctuation
    2 from os import listdir
    3 from collections import Counter
    4
    5 def add_doc_to_vocab(filename, vocab):
    6     doc = load_doc(filename)
    7     tokens = clean_doc(doc)
    8     vocab.update(tokens)
    9
   10 def process_docs(directory, vocab, is_train):
   11     for filename in listdir(directory):
   12         if is_train and filename.startswith('tb01'):
   13             continue
   14         if not is_train and not filename.startswith('tb01'):
   15             continue
   16         path = directory + '/' + filename
   17         add_doc_to_vocab(path, vocab)
   18
   19 vocab = Counter()
   20 process_docs('/content/drive/My Drive/emotion_dataset1/happy', vocab, True)
   21 process_docs('/content/drive/My Drive/emotion_dataset1/sad', vocab, True)
   22 process_docs('/content/drive/My Drive/emotion_dataset1/angry', vocab, True)
   23 process_docs('/content/drive/My Drive/emotion_dataset1/neutral', vocab, True)
   24 print(len(vocab))
   25 print(vocab.most_common(50))
```

The output of the code is:

```
271
[('movie', 12), ('This', 10), ('looks', 5), ('look', 5), ('actual', 5), ('like', 5), ('Coenesque', 4), ('Coen', 4), ('Brothers', 4), ('The', 4), ('
```

4. Tokens are saved in vocab.txt file.

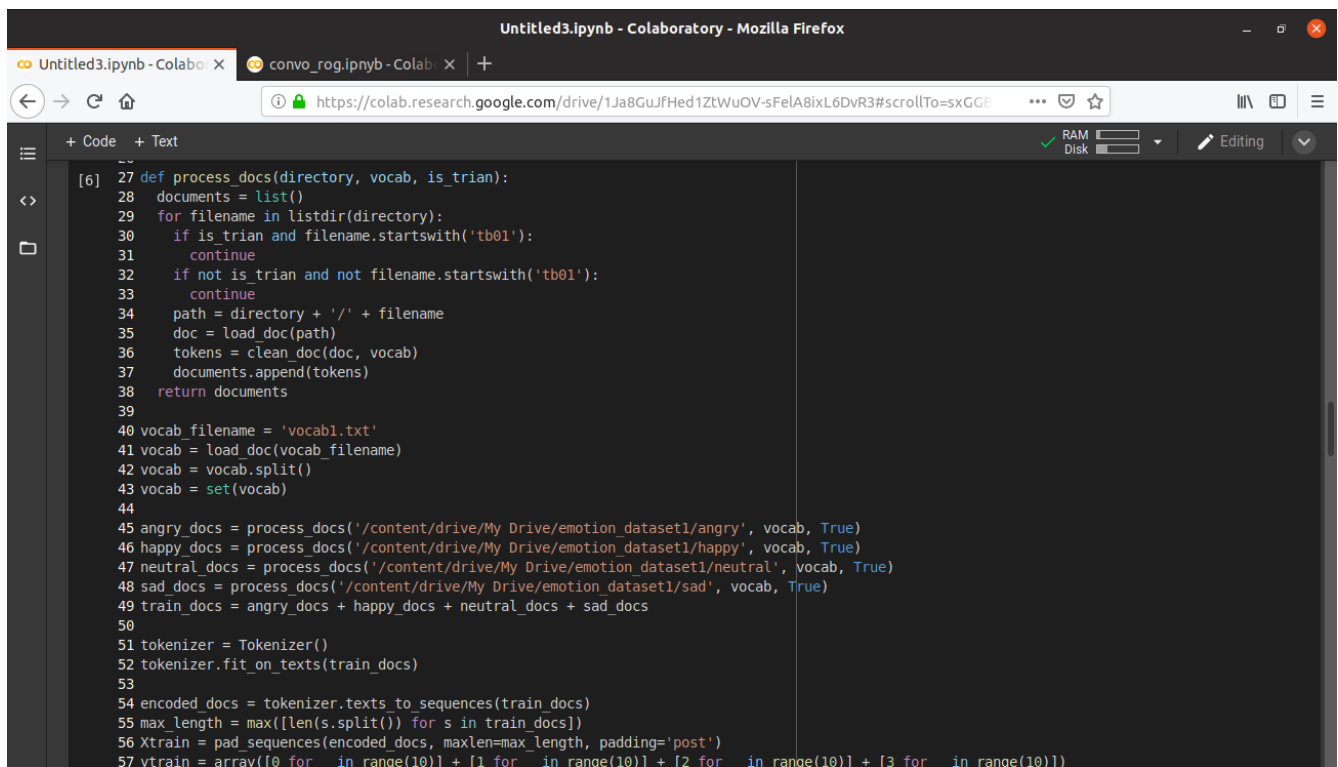


The screenshot shows a Jupyter Notebook interface in Mozilla Firefox. The browser address bar shows a Google Drive link. The notebook has two tabs: 'Untitled3.ipynb - Colab' and 'convo_rog.ipynb - Colab'. The active cell is [5] and contains the following Python code:

```
[5] 1 def save_list(lines, filename):
2     data = '\n'.join(lines)
3     file = open(filename, 'w')
4     file.write(data)
5     file.close()
6
7     save_list(tokens, 'vocab1.txt')
```

Below this cell, the start of cell [6] is visible, showing imports for string, os, numpy, keras preprocessing, and keras models, along with a function to load and clean documents.

5. Processing on training dataset



The screenshot shows the same Jupyter Notebook interface. The active cell is [6] and contains the following Python code:

```
[6] 27 def process_docs(directory, vocab, is_train):
28     documents = list()
29     for filename in listdir(directory):
30         if is_train and filename.startswith('tb01'):
31             continue
32         if not is_train and not filename.startswith('tb01'):
33             continue
34         path = directory + '/' + filename
35         doc = load_doc(path)
36         tokens = clean_doc(doc, vocab)
37         documents.append(tokens)
38     return documents
39
40 vocab_filename = 'vocab1.txt'
41 vocab = load_doc(vocab_filename)
42 vocab = vocab.split()
43 vocab = set(vocab)
44
45 angry_docs = process_docs('/content/drive/My Drive/emotion_dataset1/angry', vocab, True)
46 happy_docs = process_docs('/content/drive/My Drive/emotion_dataset1/happy', vocab, True)
47 neutral_docs = process_docs('/content/drive/My Drive/emotion_dataset1/neutral', vocab, True)
48 sad_docs = process_docs('/content/drive/My Drive/emotion_dataset1/sad', vocab, True)
49 train_docs = angry_docs + happy_docs + neutral_docs + sad_docs
50
51 tokenizer = Tokenizer()
52 tokenizer.fit_on_texts(train_docs)
53
54 encoded_docs = tokenizer.texts_to_sequences(train_docs)
55 max_length = max([len(s.split()) for s in train_docs])
56 xtrain = pad_sequences(encoded_docs, maxlen=max_length, padding='post')
57 ytrain = array([0 for _ in range(10)] + [1 for _ in range(10)] + [2 for _ in range(10)] + [3 for _ in range(10)])
```

6. Processing on testing dataset

```
encoded_docs = tokenizer.texts_to_sequences(train_docs)
max_length = max([len(s.split()) for s in train_docs])
Xtrain = pad_sequences(encoded_docs, maxlen=max_length, padding='post')
ytrain = array([0 for _ in range(10)] + [1 for _ in range(10)] + [2 for _ in range(10)] + [3 for _ in range(10)])

angry_docs = process_docs('/content/drive/My Drive/emotion_dataset1/angry', vocab, False)
happy_docs = process_docs('/content/drive/My Drive/emotion_dataset1/happy', vocab, False)
neutral_docs = process_docs('/content/drive/My Drive/emotion_dataset1/neutral', vocab, False)
sad_docs = process_docs('/content/drive/My Drive/emotion_dataset1/sad', vocab, False)
test_docs = angry_docs + happy_docs + neutral_docs + sad_docs

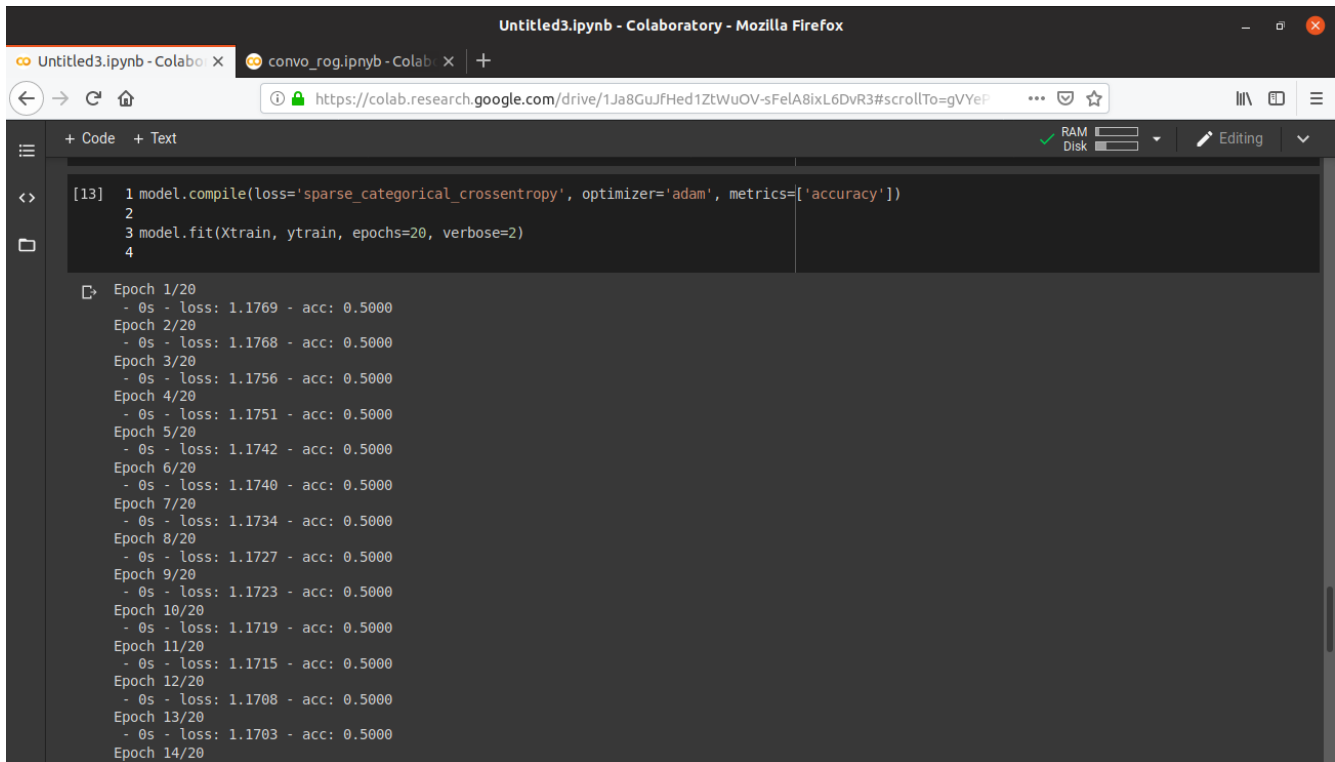
encoded_docs = tokenizer.texts_to_sequences(test_docs)

Xtest = pad_sequences(encoded_docs, maxlen=max_length, padding='post')
ytest = array([0 for _ in range(2)] + [1 for _ in range(2)] + [2 for _ in range(2)] + [3 for _ in range(2)])
```

7. Defining CNN Model

```
model = Sequential()
model.add(Embedding(vocab_size, 100, input_length=max_length))
model.add(Conv1D(filters=32, kernel_size=8, activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.add(Dense(4, activation='softmax'))
```

8. Compiling model with 20 epochs



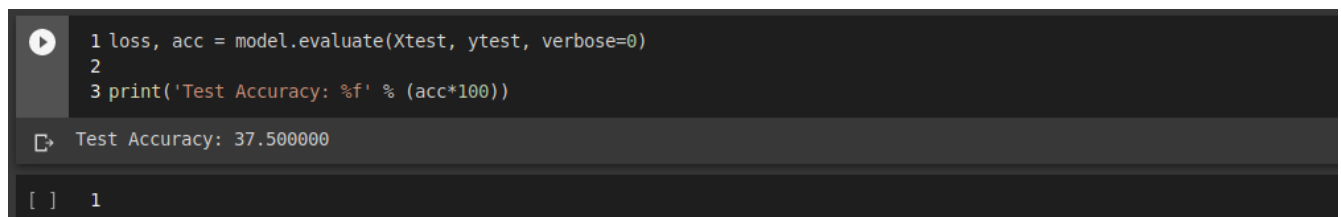
The screenshot shows a Google Colaboratory notebook interface. The browser tab is titled "Untitled3.ipynb - Colaboratory - Mozilla Firefox". The address bar shows the URL: <https://colab.research.google.com/drive/1Ja8GuJfHed1ZtWuOV-sFeIA8ixL6DvR3#scrollTo=gVYeP>. The notebook has two tabs: "Untitled3.ipynb - Colab" and "convo_rog.ipynb - Colab". The code cell [13] contains the following Python code:

```
[13] 1 model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
      2
      3 model.fit(Xtrain, ytrain, epochs=20, verbose=2)
      4
```

The output of the code cell shows the training progress for 20 epochs. The output is as follows:

```
Epoch 1/20
- 0s - loss: 1.1769 - acc: 0.5000
Epoch 2/20
- 0s - loss: 1.1768 - acc: 0.5000
Epoch 3/20
- 0s - loss: 1.1756 - acc: 0.5000
Epoch 4/20
- 0s - loss: 1.1751 - acc: 0.5000
Epoch 5/20
- 0s - loss: 1.1742 - acc: 0.5000
Epoch 6/20
- 0s - loss: 1.1740 - acc: 0.5000
Epoch 7/20
- 0s - loss: 1.1734 - acc: 0.5000
Epoch 8/20
- 0s - loss: 1.1727 - acc: 0.5000
Epoch 9/20
- 0s - loss: 1.1723 - acc: 0.5000
Epoch 10/20
- 0s - loss: 1.1719 - acc: 0.5000
Epoch 11/20
- 0s - loss: 1.1715 - acc: 0.5000
Epoch 12/20
- 0s - loss: 1.1708 - acc: 0.5000
Epoch 13/20
- 0s - loss: 1.1703 - acc: 0.5000
Epoch 14/20
```

9. Accuracy of model



The screenshot shows a Google Colaboratory notebook interface. The code cell contains the following Python code:

```
1 loss, acc = model.evaluate(Xtest, ytest, verbose=0)
2
3 print('Test Accuracy: %f' % (acc*100))
```

The output of the code cell shows the test accuracy:

```
Test Accuracy: 37.500000
```

The output is displayed in a text box with a play button icon. Below the text box, there is a small table with one row and one column:

1
