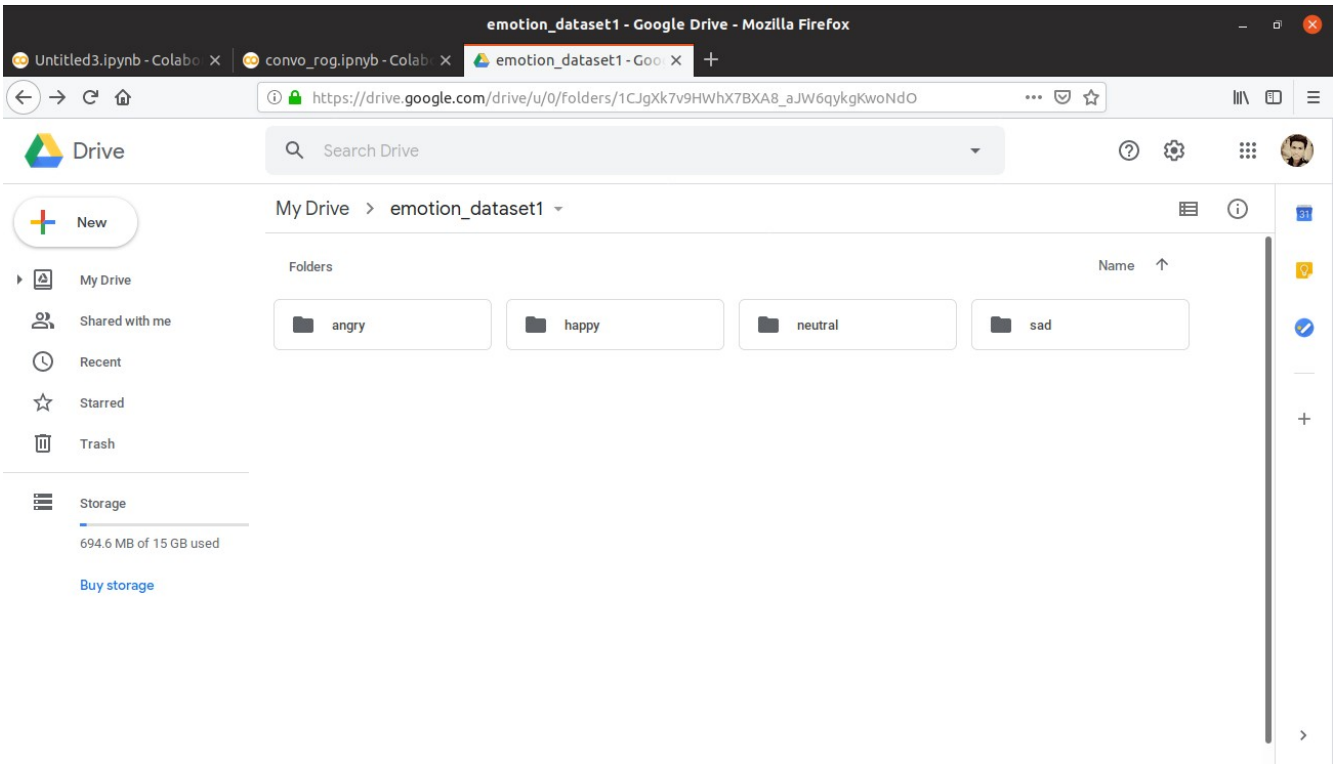
Emotion Classification on Youtube Comments

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.

<u>Snippets of Code</u>

1. Mounting Google drive

```
Code  Text
[ ]  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.googleusercont

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

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

Untitled3.ipynb - Colaboratory - Mozilla Firefox

Untitled3.ipynb - Colabo | convo_rog.ipnyb - Colab

https://colab.research.google.com/drive/1Ja8GuJfHed1ZtWuOV-sFelA8ixL6DvR3#scrollTo=sxGGE

+ Code + Text | RAM Disk | Editing

[2]

```python
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
import string


def load_doc(filename):
  file = open(filename, 'r')
  text = file.read()
  file.close()
  return text

def clean_doc(doc):
  tokens = doc.split()
  table = str.maketrans('', '', string.punctuation)
  tokens = [w.translate(table) for w in tokens]
  tokens = [word for word in tokens if word.isalpha()]
  stop_words = set(stopwords.words('english'))
  tokens = [w for w in tokens if not w in stop_words]
  tokens = [word for word in tokens if len(word) > 1]
  return tokens

filename = '/content/drive/My Drive/emotion_dataset1/happy/tb000.txt'
text = load_doc(filename)
tokens = clean_doc(text)
print(tokens)
```

```
[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.

Untitled3.ipynb - Colaboratory - Mozilla Firefox

Untitled3.ipynb - Colabo | convo_rog.ipnyb - Colab

https://colab.research.google.com/drive/1Ja8GuJfHed1ZtWuOV-sFelA8ixL6DvR3#scrollTo=sxGGE

+ Code + Text | RAM Disk | Editing

```
['Something', 'look', 'amazingly', 'interesting']
```

[3]

```python
from string import punctuation
from os import listdir
from collections import Counter

def add_doc_to_vocab(filename, vocab):
 doc = load_doc(filename)
 tokens = clean_doc(doc)
 vocab.update(tokens)

def process_docs(directory, vocab, is_trian):
  for filename in listdir(directory):
    if is_trian and filename.startswith('tb01'):
      continue
    if not is_trian and not filename.startswith('tb01'):
      continue
    path = directory + '/' + filename
    add_doc_to_vocab(path, vocab)

vocab = Counter()
process_docs('/content/drive/My Drive/emotion_dataset1/happy', vocab, True)
process_docs('/content/drive/My Drive/emotion_dataset1/sad', vocab, True)
process_docs('/content/drive/My Drive/emotion_dataset1/angry', vocab, True)
process_docs('/content/drive/My Drive/emotion_dataset1/neutral', vocab, True)
print(len(vocab))
print(vocab.most_common(50))
```

```
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.

Untitled3.ipynb - Colaboratory - Mozilla Firefox

Untitled3.ipynb - Colabo | convo_rog.ipnyb - Colab

https://colab.research.google.com/drive/1Ja8GuJfHed1ZtWuOV-sFelA8ixL6DvR3#scrollTo=sxGG

+ Code + Text

RAM Disk Editing

[5]
```python
def save_list(lines, filename):
  data = '\n'.join(lines)
  file = open(filename, 'w')
  file.write(data)
  file.close()

save_list(tokens, 'vocab1.txt')
```

[6]
```python
from string import punctuation
from os import listdir
from numpy import array
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Embedding
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D

def load_doc(filename):
  file = open(filename, 'r')
  text = file.read()
  file.close()
  return text

def clean_doc(doc, vocab):
  tokens = doc.split()
  table = str.maketrans('', '', punctuation)
  tokens = [w.translate(table) for w in tokens]
```

5. Processing on training dataset

Untitled3.ipynb - Colaboratory - Mozilla Firefox

Untitled3.ipynb - Colabo | convo_rog.ipnyb - Colab

https://colab.research.google.com/drive/1Ja8GuJfHed1ZtWuOV-sFelA8ixL6DvR3#scrollTo=sxGG

+ Code + Text

RAM Disk Editing

[6]
```python
def process_docs(directory, vocab, is_trian):
  documents = list()
  for filename in listdir(directory):
    if is_trian and filename.startswith('tb01'):
      continue
    if not is_trian and not filename.startswith('tb01'):
      continue
    path = directory + '/' + filename
    doc = load_doc(path)
    tokens = clean_doc(doc, vocab)
    documents.append(tokens)
  return documents

vocab_filename = 'vocab1.txt'
vocab = load_doc(vocab_filename)
vocab = vocab.split()
vocab = set(vocab)

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

tokenizer = Tokenizer()
tokenizer.fit_on_texts(train_docs)

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)])
```

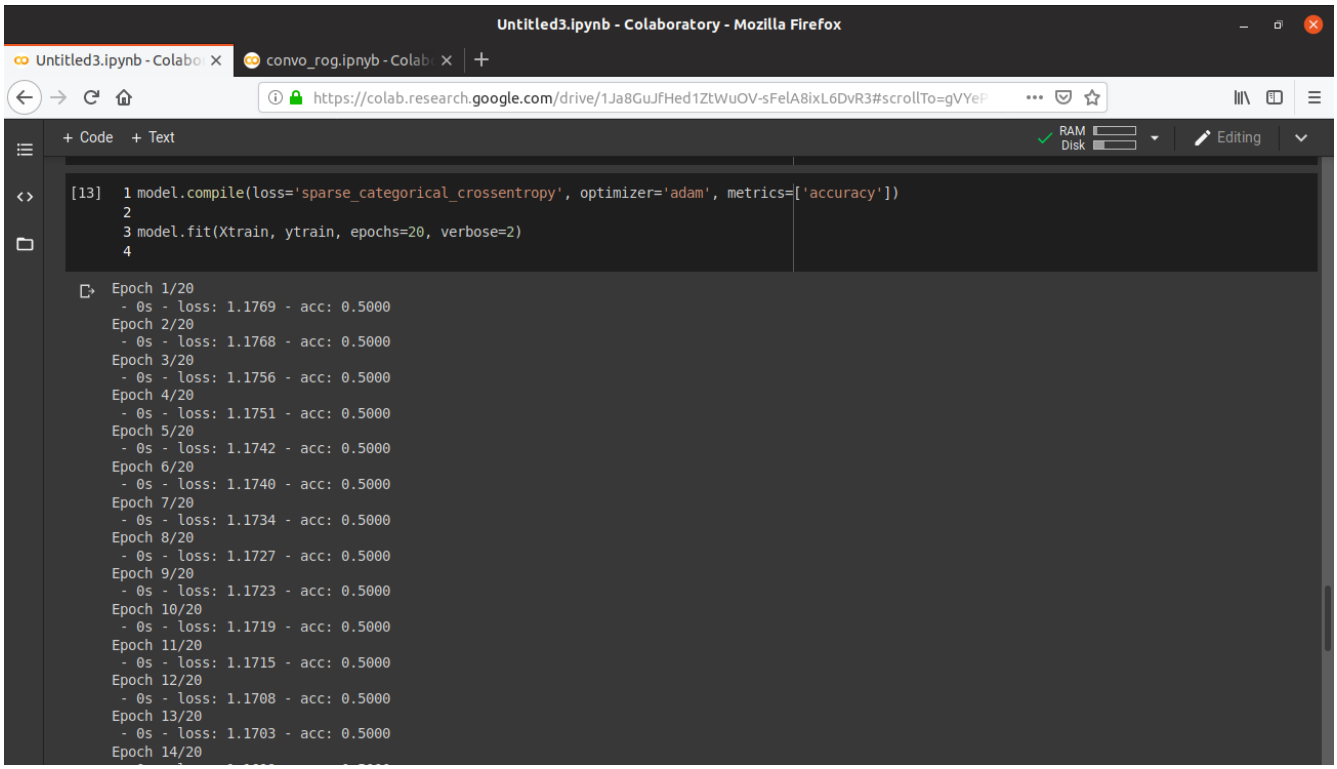
6. Processing on testing dataset

```
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



9. Accuracy of model