Classification of Sentiments in text

Akshaya A (CB.EN.U4CSE17401)
Divya Rathi (CB.EN.U4CSE17416)
Gayathri E (CB.EN.U4CSE17420)
Kaviya S (CB.EN.U4CSE17429)

Dept. of Computer Science & Engineering
Amrita School of Engineering
Amrita Vishwa Vidyapeetham

October 31, 2020

1 Introduction

Emotions are expressed in nuanced ways, which vary by collective or individual experiences, knowledge, and beliefs. Implicit emotion recognition is the most challenging problem to solve because such emotion is typically hidden within the text, and thus, its solution requires an understanding of the context. To understand and analyze texts and extract the underlying emotions, a robust mechanism capable of capturing and modeling different linguistic nuances and phenomena has to be designed. The use of semantic analysis draws the exact dictionary meaning from the text. This can be achieved by using a semantic analyzer that checks the text further for meaningfulness and for extracting useful semantic information from the content. We propose a model that primarily focuses on identifying sentiments by applying various Natural Language Processing (NLP) techniques. Thus, achieving classification of different emotions used in the text.

2 Objective

The aim is to classifying emotions present in the text documents by applying various NLP techniques such as feature selection, resolving lexical ambiguity using POS tagging and word sense disambiguation.

3 Problem Statement

To understand and classify polarity of sentences within text documents at sentence-level analysis using Natural Language Processing.

4 Architecture

The architecture basically comprises of following three stages:
Step 1 - Tokenization of data. In this process, we tokenize the sentences into words.
Step 2 - Stemming and lemmatization. In this process, the words are brought into present tense.
Step 3 - Removal of stop words. In this process, we remove the stop words which are not using the detection process example – Articles.

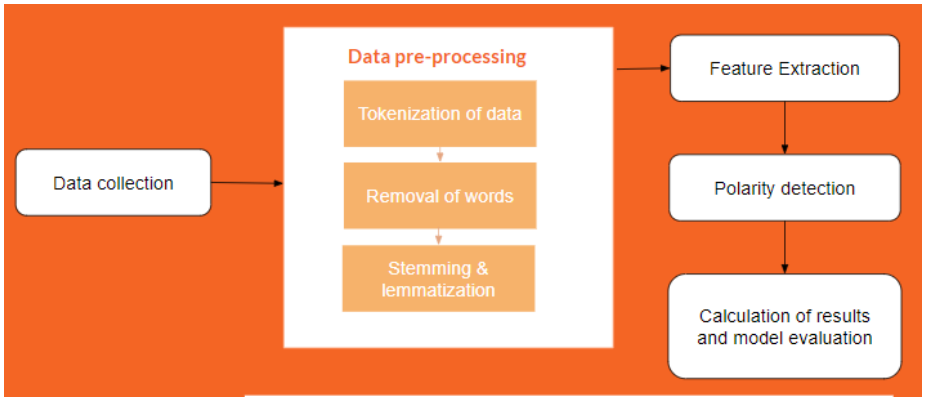

Figure 1: Architecture Diagram

5 Related Work

Although the area of sentiment analysis has recently enjoyed a huge burst of research activity, there has been a steady undercurrent of interest for quite a while. For example, determining whether a review is positive or negative [1]. In contrast, our experiments classify individual phrases and sentences present in text documents. A number of researchers have explored learning words and phrases with prior positive or negative polarity (another term is semantic orientation) [2]. In contrast, we begin with a lexicon of words with established prior polarities and identify the contextual polarity of phrases and sentences in which instances of those words appear in the corpus.

Nasukawa, Yi, and colleagues [3] classify the contextual polarity of sentiment expressions, as we do. Thus, their work is probably most closely related to ours. They classify expressions that are about specific items, and use manually developed patterns to classify polarity. These patterns are high-quality, yielding quite high precision, but very low recall. Their system classifies a much smaller proportion of the sentiment expressions in a corpus than ours does. In addition, our system assign one sentiment per sentence as well as it assigns contextual

polarity to individual expressions. And sentences often contain more than one sentiment expression.

6 Novelty of the Work

In this project, we are implementing sentence-level sentiment analysis. The task at this level goes to the sentences and determines whether each sentence expressed a positive, negative, or neutral opinion. Neutral usually means no opinion.This level of analysis is closely related to subjectivity classification (Wiebe, Bruce and O'Hara, 1999), which distinguishes sentences (called objective sentences) that express factual information from sentences (called subjective sentences) that express subjective views and opinions. Naturally the same document-level sentiment classification techniques can also be applied to individual sentences.The task of classifying a sentence as subjective or objective is often called subjectivity classification in the existing literature.

Definition 6.1: Given a sentences, two sub-tasks are performed:

1. Subjectivity classification:Determine whether s is a subjective sentence or an objective sentence
2. Sentence-level sentiment classification:If s is subjective, determine whether it expresses a positive, negative or neutral opinion.

In most applications, one needs to know what entities or aspects of the entities are the targets of opinions.Knowing that some sentences have positive or negative opinions but not about what, is of limited use.However, the two sub-tasks are still useful because (1) it filters out those sentences which contain no opinions, and (2) after we know what entities and aspects of the entities are talked about in a sentence, this step can help us determine whether the opinions about the entities and their aspects are positive or negative

Much of the research on sentence-level sentiment classification makes the following assumption: Assumption:The sentence expresses a single opinion from a single opinion holder.

We assign a sentiment to a sentence by averaging the prior semantic orientations of instances of lexicon words in the sentence. Apart from this, we are assigning contextual polarity to individual phrases expressions.

7 Data Collection and Preparation

The data is collected from Kaggle. The data pre-processing is majorly done in three steps.

i. Tokenization of data:

In this process, after the data is retrieved from the dataset. The data which is taken is in the form of sentences and phrases. Now, these sentences and phrases are tokenized into words, so that it is easily understandable.

ii. Removal of stop words:
Stop words are the words that the search engine has programmed to ignore when both indexing and retrieving of entries. Articles are the best examples of stop words.
iii. Stemming and Lemmatization:
Stemming and lemmatization is the process of converting the words into their root words so that they can be analyzed as a single item.
iv. P-O-S tagging:
P-O-S tagging basically stands for parts-of-speech tagging. This is one of the most used in sentiment polarity detection. It helps us to classify the words into a verb, adverb, adjective, noun etc. These words are then mapped to sentiment dictionary.

8 Implementation

```
for l in y_val:

    if l == "joy":
        int_y_val.append(0)
    if l == "sadness":
        int_y_val.append(1)
    if l == "anger":
        int_y_val.append(2)
    if l == "fear":
        int_y_val.append(3)
    if l == "love":
        int_y_val.append(4)
    if l == "surprise":
        int_y_val.append(5)

int_y_train,int_y_test,int_y_val = np.array(int_y_train),np.array(int_y_test),np.array(int_y_val)

from sklearn import preprocessing
from keras.utils import np_utils

le = preprocessing.LabelEncoder()
le.fit(int_y_train)

encoded_y_train = le.transform(int_y_train)
encoded_y_test = le.transform(int_y_test)
encoded_y_val = le.transform(int_y_val)

encoded_y_train = np_utils.to_categorical(encoded_y_train)
encoded_y_test = np_utils.to_categorical(encoded_y_test)
encoded_y_val = np_utils.to_categorical(encoded_y_val)

print(encoded_y_train)

print(int_y_train[:10])
```

Figure 2: Data Preparation

```
import nltk
nltk.download('stopwords')
from tensorflow.python.keras.preprocessing.text import Tokenizer
from tensorflow.python.keras.preprocessing.sequence import pad_sequences
from nltk.corpus import stopwords
```

```
stopwords = stopwords.words('english')
stopwords[:5]
```

```
['i', 'me', 'my', 'myself', 'we']
```

```
x_train_cl = []
x_test_cl = []
x_val_cl = []


# Deleting stopwords
for text in x_train:

    text = text.split()
    text = [word for word in text if word not in stopwords]
    text = " ".join(text)
    x_train_cl.append(text)

for text in x_test:

    text = text.split()
    text = [word for word in text if word not in stopwords]
    text = " ".join(text)
    x_test_cl.append(text)
```

Figure 3: Tokenization I

```
# Tokenizing everything

x_train_token = tokenizer.texts_to_sequences(x_train)
x_test_token = tokenizer.texts_to_sequences(x_test)
x_val_token = tokenizer.texts_to_sequences(x_val)

total_token = np.concatenate((x_train_token,x_test_token,x_val_token),axis=0)
```

```
# Padding
print(np.mean([len(text) for text in total_token]))
```

```
7.6699
```

```
x_train_pad = pad_sequences(x_train_token,20)
x_test_pad = pad_sequences(x_test_token,20)
x_val_pad = pad_sequences(x_val_token,20)

print(x_train_pad[0],end="\n----------------------------------------------------------------------\n")
print(x_train_pad[1])
```

```
[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  50
   1 495]
----------------------------------------------------------------------
[   0    0    0    0    0    0    0    0    0    0    0    0   30    2
  464  425   44   54 1573 1304]
```

Figure 4: Tokenization II

```
from tensorflow.python.keras.models import Sequential
from tensorflow.python.keras.layers import Dense,CuDNNGRU,Embedding,Bidirectional


model = Sequential()

model.add(Embedding(input_dim=2000
                    ,output_dim=100
                    ,input_length=20))

model.add(Bidirectional(CuDNNGRU(units=16,return_sequences=True)))

model.add(Bidirectional(CuDNNGRU(units=8)))

model.add(Dense(6,activation="softmax"))

model.compile(loss="categorical_crossentropy",optimizer="rmsprop",metrics=["accuracy"])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 20, 100)           200000
_________________________________________________________________
bidirectional (Bidirectional (None, 20, 32)            11328
_________________________________________________________________
bidirectional_1 (Bidirection (None, 16)                2016
_________________________________________________________________
dense (Dense)                (None, 6)                 102
=================================================================
Total params: 213,446
Trainable params: 213,446
Non-trainable params: 0
_________________________________________________________________
```

Figure 5: Modeling and Predicting I

```
print(x_train_pad.shape)
print(y_train.shape)
model.fit(x_train_pad, encoded_y_train, epochs=10, batch_size=20)
```

```
(16000, 20)
(16000,)
Epoch 1/10
800/800 [==============================] - 6s 8ms/step - loss: 1.0660 - accuracy: 0.6082
Epoch 2/10
800/800 [==============================] - 6s 7ms/step - loss: 0.3933 - accuracy: 0.8703
Epoch 3/10
800/800 [==============================] - 6s 7ms/step - loss: 0.2252 - accuracy: 0.9221
Epoch 4/10
800/800 [==============================] - 6s 8ms/step - loss: 0.1721 - accuracy: 0.9346
Epoch 5/10
800/800 [==============================] - 6s 8ms/step - loss: 0.1450 - accuracy: 0.9419
Epoch 6/10
800/800 [==============================] - 6s 8ms/step - loss: 0.1322 - accuracy: 0.9469
Epoch 7/10
800/800 [==============================] - 6s 8ms/step - loss: 0.1212 - accuracy: 0.9516
Epoch 8/10
800/800 [==============================] - 6s 8ms/step - loss: 0.1099 - accuracy: 0.9563
Epoch 9/10
800/800 [==============================] - 6s 8ms/step - loss: 0.1021 - accuracy: 0.9595
Epoch 10/10
800/800 [==============================] - 6s 7ms/step - loss: 0.0967 - accuracy: 0.9615
<tensorflow.python.keras.callbacks.History at 0x7f3210399710>
```

```
preds = model.predict_classes(x_test_pad)
from sklearn.metrics import accuracy_score
```

```
accuracy_score(preds,int_y_test)
```

```
0.9265
```

Figure 6: Modeling and Predicting II

9 Conclusion

Sentiment analysis can be applied to countless aspects of business, from brand monitoring and product analytics, to customer service and market research. By incorporating it into their existing systems and analytics, leading brands (not to mention entire cities) are able to work faster, with more accuracy, toward more useful ends.

Sentiment analysis has moved beyond merely an interesting, high-tech whim, and will soon become an indispensable tool for all companies of the modern age. Ultimately, sentiment analysis enables us to glean new insights, better understand our customers, and empower our own teams more effectively so that they do better and more productive work.

References

[1] Riloff and J. Wiebe. 2003. Learning extraction patterns for subjective expressions. In *EMNLP-2003*.

[2] Takamura, Hiroya Inui, Takashi Okumura, Manabu. (2005). *Extracting emotional polarity of words using spin model.*

[3] Yi, Jeonghee Nasukawa, Tetsuya Bunescu, Razvan Niblack, Wayne. (2003). Sentiment Analyzer: Extracting sentiments about a given topic using natural language processing techniques. Proceedings - IEEE International Conference on Data Mining, ICDM. 427- 434. 10.1109/ICDM.2003.1250949.