Sentiment analysis

Sentiment analysis was initially formulated as the NLP task of retrieval of sentiments expressed in texts. A simple keyword finding will not be appropriate for mining all kinds of opinions. Hence the need for use of sophisticated opinion extraction methods. Sentiment analysis is a natural language processing technique, helps to find out subjective information in source materials. Sentiment analysis aims to establish the approach of the writer with respect to the entire contextual polarity of a document.

A fundamental process in sentiment analysis is classification of opinion polarity of a specified context at the document; whether the given opinion in an article, a context or an entity feature is positive, negative or neutral.

Subtask A: Contextual Polarity Disambiguation: Given a message containing a marked instance of a word or phrase, determine whether that instance is positive, negative or neutral in that context.

Subtask B: Message Polarity Classification: Given a message, classify whether the message is of positive, negative, or neutral sentiment. For messages conveying both a positive and negative sentiment, whichever is the stronger sentiment should be chosen.

Subtask C: Topic-Based Message Polarity Classification: Given a message and a topic, classify whether the message is of positive, negative, or neutral sentiment towards the given topic. For messages conveying both a positive and negative sentiment towards the topic, whichever is the stronger sentiment should be chosen.

Subtask D: Detecting Trends Towards a Topic: Given a set of messages on a given topic from the same period of time, determine whether the dominant sentiment towards the target topic in these messages is (a) strongly positive, (b) weakly positive, (c) neutral, (d) weakly negative, or (e) strongly negative.

Sentiment Analysis Use Cases

Sentiment Analysis for Brand Monitoring

To get a full view of how a brand, product, or company is analysed by your customers and stakeholders. Widely available media, like product reviews, news and social, can give key informations about what your business is doing right or wrong. Companies can also use sentiment analysis to measure the impact of a new product, by advertisement campaign or consumer's response to recent company news on social media.

Sentiment Analysis for Customer Service

To automatically sort incoming user email into "urgent" or "not urgent" buckets based on the sentiment of the email, proactively identifying frustrated users. The agent then directs their time toward resolving the users with the most urgent needs first. As customer service becomes more and more automated through Machine Learning, understanding the sentiment of a given case becomes increasingly important.

Sentiment Analysis for Market Research and Analysis

It is used in business intelligence(BI) to understand the subjective reasons why consumers are or are not responding to something.

e.g. why are consumers buying a product? What do they think of the user experience? Did customer service support meet their expectations?.

Real-Time Politics Analysis

Data-driven media and journalism. PR(Public relations) management for political figures and parties.

Financial Analysis

Intelligent tools for aiding decision-making for financial traders and analysts

Challenges of Sentiment Analysis

1. Sentiment Analysis runs into a similar set of problems as emotion recognition does – before deciding what the sentiment of a given sentence is, **we need to figure out what "sentiment" is in the first place.** Is it categorical, and sentiment can be split into clear buckets like *happy, sad, angry, or bored* ? Or is it dimensional, and sentiment needs to be evaluated on some sort of bi-directional spectrum LIKE Good or Bad, Profit or Loss, Happy or Sad etc.

2. In sentiment analysis categorical framework: sentiment is analysed as belonging to a certain bucket, to a certain degree. For example, a given sentence may be 45% happy, 23% sad, 89% excited, and 55% hopeful. These numbers don't add up to 100.

3. In addition to the definition problem, there are multiple layers of meaning in any human generated sentence. People express opinions in complex ways; rhetorical devices like sarcasm, irony, and implied meaning can mislead sentiment analysis.

Implementing Sentimental Analysis

Read the original dataset which is collected from [**https://www.figure-eight.com/data-for-everyone/**]
Then split them into two part for training and testing.
Here we divided 20% of original data for testing purpose.

```
from sklearn.model_selection import train_test_split # function for splitting data to train and test sets
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Read Sentiment.csv
df = pd.read_csv('Sentiment.csv')

# Keeping only the neccessary columns
data = df[['text','sentiment']]

senti_data = data[ data['sentiment'] != 'Neutral']

# Splitting the dataset into train and test set
train, test = train_test_split(senti_data,test_size = 0.2)
print('Size of Train Data:',len(train),'\nSize of Test Data:',len(test),'\nTOTAL:',len(train)+len(test))
```

```
Size of Train Data: 8583
Size of Test Data: 2146
TOTAL: 10729
```

And also discard all the features except text and target ie sentiment. Because sentiment only comes from the text data.

Text Cleaning:

Removal of Stop-words: When data analysis needs to be data driven at the word level, the commonly occurring words (stop-words) should be removed. One can either create a long list of stop-words or one can use predefined language specific libraries.

Removal of Punctuations: The commonly occurring Punctuations like '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~' should be removed while we are analysing text. This Punctuations not more descriptive.

Stemming: A stemming algorithm reduces the words "working", "worked" to the root word, "work".

```
import string
from nltk.corpus import stopwords
import nltk
from nltk.stem import PorterStemmer
def text_Cleaning(mess):
    non_punc =" ".join([char for char in mess.split() if char not in string.punctuation])  # REMOVING PUNCTUATION
    non_punc =re.sub('\. ', ' ', non_punc)
    non_punc =re.sub('\? ', ' ', non_punc)
    non_punc =re.sub('\! ', ' ', non_punc)
    non_punc =" ".join(filter(lambda x:x[0]!='#' and x[0]!='@' and x!= 'RT', non_punc.split()))
    tokens = nltk.word_tokenize(non_punc)
    non_stop = [word for word in tokens if word.lower() not in stopwords.words('english')] # REMOVING STOP-WORDS
    st = PorterStemmer()
    clean_txt = " ".join([st.stem(w) for w in non_stop])  #STEMMING
    clean_txt = clean_txt.split()
    return clean_txt
```

After removing of stop-words , punctuations and then stemming of text we get the clean text.

Removing few more things like #tags, @xxxx , urls,

```
train['text'][0]
```

```
'RT @NancyLeeGrahn: How did everyone feel about the Climate Change question last night? Exactly. #GOPDebate'
```

```
text_Cleaning(train['text'][0])
```

```
['everyon', 'feel', 'climat', 'chang', 'question', 'last', 'night', 'exactli']
```

Feature Extraction

Vectorisation of Words:

Now we'll convert each message, represented as a list of tokens above, into a vector that machine learning models can understand.

We'll do that in three steps using the **bag-of-words model:**The bag-of-words model is commonly used in methods of document classification where the (frequency of) occurrence of each word is used as a feature for training a classifier.

```
from sklearn.feature_extraction.text import CountVectorizer
word_vector = CountVectorizer(analyzer=text_Cleaning).fit(senti_data['text'])
text_bow = word_vector.transform(senti_data['text'])
```

Training And Testing of model

Here we are using multinomial Naive Bayes for sentimental analysis.

```
from sklearn.naive_bayes import MultinomialNB
spam_detect_model = MultinomialNB().fit(text_bow, senti_data['sentiment'])
all_predictions = spam_detect_model.predict(text_bow)


from sklearn.metrics import classification_report
print (classification_report(senti_data['sentiment'], all_predictions))

from sklearn.metrics import accuracy_score
print("ACCURACY")
accuracy_score(senti_data['sentiment'],all_predictions)
```

```
             precision    recall  f1-score   support

   Negative       0.93      0.95      0.94      8493
   Positive       0.79      0.71      0.75      2236

avg / total       0.90      0.90      0.90     10729

ACCURACY

0.899058626153416
```