Sentiment Analysis

Social media posts can be on any topic. Sentiment analysis, analyze the different social media posts and get their feeling about different Topic. Users may feel positive , negative or neutral about it.

Using a Natural language processing , this texts are analyzed and people feelings are captured and Quantified like how many posts feel positive about this and How many people feel negative about this and How many people feel bad about this ?

Twitter is a widely used social media platform. It is one of widely used global platform for public self-expression. Twitter connects users to people, information, ideas, opinions, and news.

This project analyze the sentiment of each tweet. We can search for any Twitter hashtag. It can also help Data Analysts for getting information out of a particular hashtag.

Sentiment analysis used in all the Industries for how customers feel about their Product or service by market Research Team. It is heavily used in Media Industry. Media industry wish to know about Audience feeling about  their content like Shows , Movies , Sports Telecasts etc. This is highly connected to subscriptions ,  Advertisement revenue .

Python language used in this Project.

Libraries Used:

nltk: The Natural Language Toolkit (NLTK) work with human language data for applying in statistical natural language processing (NLP). It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning.

snscrape: Snscrape is a scraper for social networking services (SNS). It scrapes things like user profiles, hashtags, or searches and returns the discovered items.

**Note:**We faced Challenges using this. Twiiter has recently made this subscription faced.

googletrans: Googletrans is a free and unlimited python library that implemented Google Translate API. This uses the Google Translate Ajax API to make calls to such methods as detect and translate.

string: String module contains some constants, utility function, and classes for string manipulation.

re: A Regular Expressions (RegEx) is a special sequence of characters that uses a search pattern to find a string or set of strings.

wordcloud: Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance.

matplotib: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

tkinter: Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications.

What did you build?  
  
We built a solution for social media sentiment analysis using Google Cloud AI tools. Our solution focuses on using Google's Natural Language AI to analyze the sentiment of randomly generated positive and negative Twitter-like posts about popular characters. We were able to effectively use this tool to analyze each post. We collected each post's sentiment score and sentiment magnitude measures and created datasets that can be used for further analysis anf visualization later down the line.  
  
What problems/opportunity did your team seek to solve, and why is it important?  
  
Sentiment analysis provides the opportunity to understand customer opinions, provide valuable insights into market trends, and knowing customer preferences. These understandings are important for customer demand for one product versus another. Using Google Cloud Natural Language AI tools, you are able to efficiently build a solution for sentiment analysis, so you can focus more on gaining insights. Measures such as sentiment score and sentiment magnitude can be used for comparison between customer's emotional leaning towards one object versus another, as well the strength of the emotion towards the object

How did you build it?  
  
Our team ran into some issues with scrapping social media websites for posts. We have decided to use randomly generated positive and negative Twitter-like posts to showcase how the sentiment analysis solution can be built. After generated the data, we used Google Natural Language AI tool to analyze the sentiment of each post to gain sentiment score and magntitude score. Unfortunately, our team ran out of time, but we imagine data like this can be used for further visualization (on a dashboard) and analysis (reporting) for the client to better gain insights.

Appendix I

sentimentAnalysis.py

import snscrape.modules.twitter as sntwitter  
  
from googletrans import Translator  
  
import string  
  
import re  
  
from wordcloud import WordCloud  
  
import matplotlib.pyplot as plt  
  
from nltk.corpus import stopwords  
  
from nltk.tokenize import word\_tokenize  
  
from nltk.sentiment.vader import SentimentIntensityAnalyzer  
  
  
def stop\_word\_removal(sent):  
 stop\_words = set(stopwords.words("english"))  
  
 word\_tokens = word\_tokenize(sent)  
  
 swr = [word for word in word\_tokens if word.lower() not in stop\_words]  
  
 clear = ' '.join(swr)  
  
 return clear  
  
  
# Extracting Tweets and translating them in English  
  
def orig\_tweets(hashtag, num):  
 tweets = []  
  
 translator = Translator()  
  
 for tweet in sntwitter.TwitterSearchScraper(hashtag).get\_items():  
  
 if len(tweets) == num:  
  
 break  
  
 else:  
  
 translated = translator.translate(tweet.rawContent, dest="en")  
  
 tweets.append(translated.text)  
  
 return tweets  
  
  
# Cleaning the extracted tweets  
  
def clean\_tweets(tweets):  
 cleared = []  
  
 for i in range(len(tweets)):  
 sent = tweets[i]  
  
 lower\_case = sent.lower()  
  
 lower\_case = stop\_word\_removal(lower\_case)  
  
 cleaning = lower\_case.translate(str.maketrans('', '', string.punctuation))  
  
 cleaning = re.sub(r'@[A-Za-z0-9]+', '', cleaning)  
  
 cleaning = re.sub(r'#', '', cleaning)  
  
 cleaning = re.sub(r'RT[\s]+', '', cleaning)  
  
 cleaning = re.sub(r'https?:\/\/\S+', '', cleaning)  
  
 cleared.append(cleaning)  
  
 return cleared  
  
  
# Performing sentiment analysis on the cleaned Tweets  
  
def senti(ctweets):  
 sent\_analyzer = SentimentIntensityAnalyzer()  
  
 sentimentsList = []  
  
 for text in ctweets:  
 analysis = sent\_analyzer.polarity\_scores(text)  
  
 sentiment = analysis["compound"]  
  
 sentimentsList.append(sentiment)  
  
 return sentimentsList  
  
  
# Segregating tweets into Positive, Negative and Neutral  
  
def segregate\_tweets(sen, tweets):  
 positive = []  
  
 posiPol = []  
  
 negative = []  
  
 negPol = []  
  
 neutral = []  
  
 neutPol = []  
  
 for i in range(len(sen)):  
  
 if sen[i] >= 0.5:  
  
 posiPol.append(sen[i])  
  
 positive.append(tweets[i])  
  
 elif sen[i] <= -0.5:  
  
 negPol.append(sen[i])  
  
 negative.append(tweets[i])  
  
 else:  
  
 neutPol.append(sen[i])  
  
 neutral.append(tweets[i])  
  
 return positive, posiPol, negative, negPol, neutral, neutPol  
  
  
# For plotting a bar graph  
  
def bar\_graph(p, ne, n, h):  
 values = [p, ne, n]  
  
 attributes = ["Positive", "Neutral", "Negative"]  
  
 plt.figure(figsize=(10, 5))  
  
 plt.bar(attributes, values, color='maroon', width=0.4)  
  
 plt.title("Bar Graph of #" + h)  
  
 plt.xlabel("Emotions")  
  
 plt.ylabel("Number of Tweets")  
  
 plt.show()  
  
  
# For plotting a pie chart  
  
def pie\_chart(p, ne, n, h):  
 values = [p, ne, n]  
  
 attributes = ["Positive", "Neutral", "Negative"]  
  
 plt.title("Pie distribution of #" + h)  
  
 plt.pie(values, labels=attributes, autopct='%2.1f%%')  
  
 plt.show()  
  
  
# For creating a word cloud  
  
def word\_cloud(text):  
 wordcloud = WordCloud(width=800, height=400, random\_state=21, max\_font\_size=110).generate(text)  
  
 plt.figure(figsize=(10, 5))  
  
 plt.imshow(wordcloud, interpolation="bilinear")  
  
 plt.axis('off')  
  
 plt.show()

analysisWindow.py

import snscrape.modules.twitter as sntwitter  
  
from googletrans import Translator  
  
import string  
  
import re  
  
from wordcloud import WordCloud  
  
import matplotlib.pyplot as plt  
  
from nltk.corpus import stopwords  
  
from nltk.tokenize import word\_tokenize  
  
from nltk.sentiment.vader import SentimentIntensityAnalyzer  
  
  
def stop\_word\_removal(sent):  
 stop\_words = set(stopwords.words("english"))  
  
 word\_tokens = word\_tokenize(sent)  
  
 swr = [word for word in word\_tokens if word.lower() not in stop\_words]  
  
 clear = ' '.join(swr)  
  
 return clear  
  
  
# Extracting Tweets and translating them in English  
  
def orig\_tweets(hashtag, num):  
 tweets = []  
  
 translator = Translator()  
  
 for tweet in sntwitter.TwitterSearchScraper(hashtag).get\_items():  
  
 if len(tweets) == num:  
  
 break  
  
 else:  
  
 translated = translator.translate(tweet.rawContent, dest="en")  
  
 tweets.append(translated.text)  
  
 return tweets  
  
  
# Cleaning the extracted tweets  
  
def clean\_tweets(tweets):  
 cleared = []  
  
 for i in range(len(tweets)):  
 sent = tweets[i]  
  
 lower\_case = sent.lower()  
  
 lower\_case = stop\_word\_removal(lower\_case)  
  
 cleaning = lower\_case.translate(str.maketrans('', '', string.punctuation))  
  
 cleaning = re.sub(r'@[A-Za-z0-9]+', '', cleaning)  
  
 cleaning = re.sub(r'#', '', cleaning)  
  
 cleaning = re.sub(r'RT[\s]+', '', cleaning)  
  
 cleaning = re.sub(r'https?:\/\/\S+', '', cleaning)  
  
 cleared.append(cleaning)  
  
 return cleared  
  
  
# Performing sentiment analysis on the cleaned Tweets  
  
def senti(ctweets):  
 sent\_analyzer = SentimentIntensityAnalyzer()  
  
 sentimentsList = []  
  
 for text in ctweets:  
 analysis = sent\_analyzer.polarity\_scores(text)  
  
 sentiment = analysis["compound"]  
  
 sentimentsList.append(sentiment)  
  
 return sentimentsList  
  
  
# Segregating tweets into Positive, Negative and Neutral  
  
def segregate\_tweets(sen, tweets):  
 positive = []  
  
 posiPol = []  
  
 negative = []  
  
 negPol = []  
  
 neutral = []  
  
 neutPol = []  
  
 for i in range(len(sen)):  
  
 if sen[i] >= 0.5:  
  
 posiPol.append(sen[i])  
  
 positive.append(tweets[i])  
  
 elif sen[i] <= -0.5:  
  
 negPol.append(sen[i])  
  
 negative.append(tweets[i])  
  
 else:  
  
 neutPol.append(sen[i])  
  
 neutral.append(tweets[i])  
  
 return positive, posiPol, negative, negPol, neutral, neutPol  
  
  
# For plotting a bar graph  
  
def bar\_graph(p, ne, n, h):  
 values = [p, ne, n]  
  
 attributes = ["Positive", "Neutral", "Negative"]  
  
 plt.figure(figsize=(10, 5))  
  
 plt.bar(attributes, values, color='maroon', width=0.4)  
  
 plt.title("Bar Graph of #" + h)  
  
 plt.xlabel("Emotions")  
  
 plt.ylabel("Number of Tweets")  
  
 plt.show()  
  
  
# For plotting a pie chart  
  
def pie\_chart(p, ne, n, h):  
 values = [p, ne, n]  
  
 attributes = ["Positive", "Neutral", "Negative"]  
  
 plt.title("Pie distribution of #" + h)  
  
 plt.pie(values, labels=attributes, autopct='%2.1f%%')  
  
 plt.show()  
  
  
# For creating a word cloud  
  
def word\_cloud(text):  
 wordcloud = WordCloud(width=800, height=400, random\_state=21, max\_font\_size=110).generate(text)  
  
 plt.figure(figsize=(10, 5))  
  
 plt.imshow(wordcloud, interpolation="bilinear")  
  
 plt.axis('off')  
  
 plt.show()