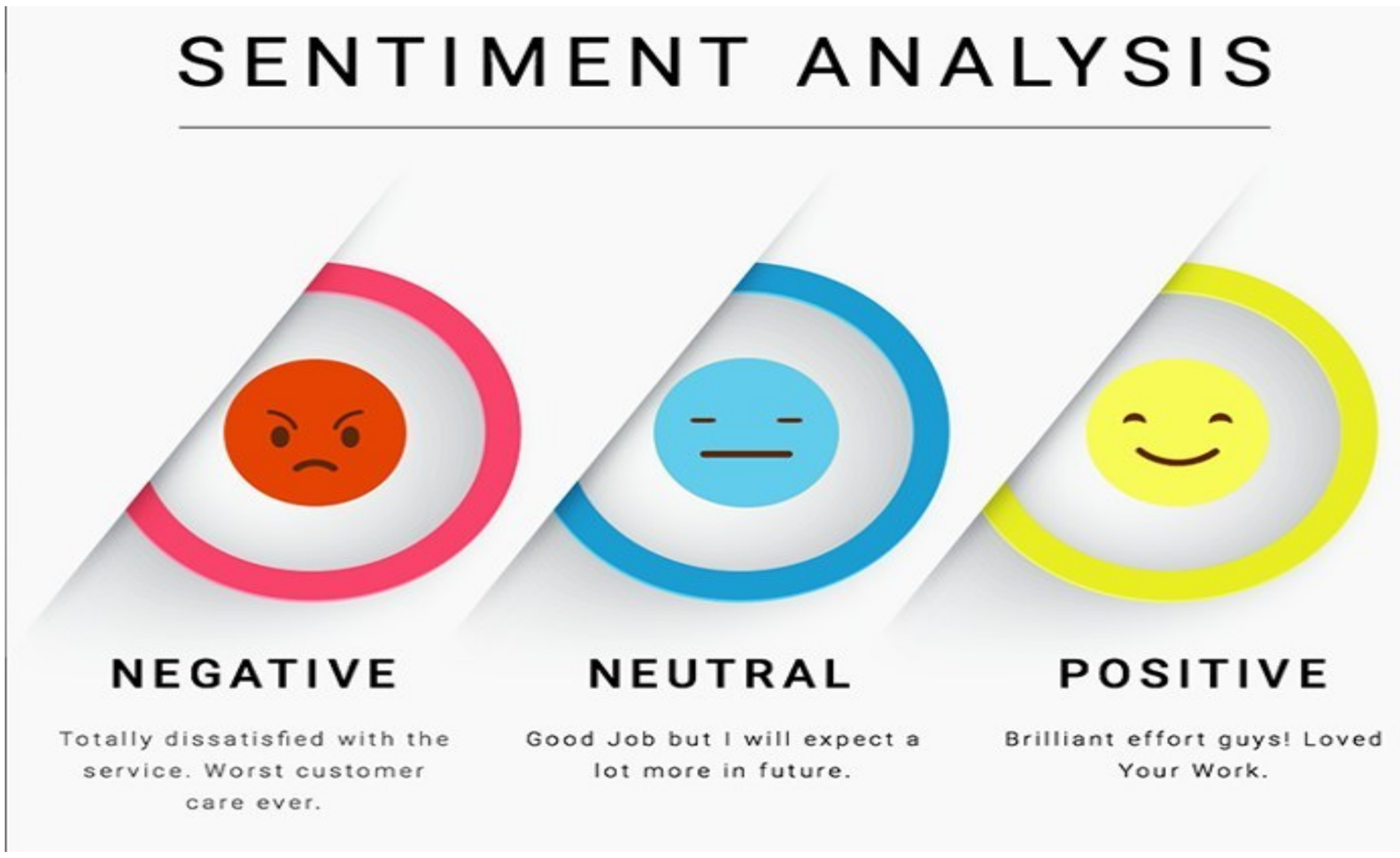
SENTIMENTAL ANALYSIS IN MARKETTING

PHASE 4 PROJECT SUBMISSION

410121104026:Meenakshi.K



TOPIC: Start building the sentiment analysis solution by Employing NLP techniques and generating insights

Introduction:

Sentiment analysis, also called opinion mining, is the field of study that analyzes people' s opinions, sentiments, appraisals, attitudes, and emotions toward entities and their attributes expressed in written text. The entities can be products, services, organizations, individuals, events, issues, or topics. The field represents a large problem space. Many related names and slightly different tasks – for example, sentiment analysis, opinion mining, opinion analysis, opinion extraction, sentiment mining, subjectivity analysis, affect analysis, emotion analysis, and review mining – are now all under the umbrella of sentiment analysis. The term *sentiment analysis* perhaps first appeared in Nasukawa and Yi (2003), and the term *opinion mining* first appeared in Dave et al. (2003). However, research on sentiment and opinion began earlier (Wiebe, 2000; Das and Chen, 2001; Tong, 2001; Morinaga et al., 2002; Pang et al., 2002; Turney, 2002). Even earlier related work includes interpretation of metaphors; extraction of sentiment adjectives; affective computing; and subjectivity

analysis, viewpoints, and affects (Wiebe, 1990, 1994; Hearst, 1992; Hatzivassiloglou and McKeown, 1997; Picard, 1997; Wiebe et al., 1999). An early patent on text classification included sentiment, appropriateness, humor, and many other concepts as possible class labels (Elkan, 2001).

DATA SE LINK:

https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment

EMPLOYING NLP TECHNIQUES:

As the range of methods and purposes for sentiment analysis is so broad, for the purposes of this introduction we will be focusing on the most common form of sentiment analysis system: assigning a positive, neutral, or negative sentiment to text data drawn from any source - this could be in the form of social media comments, emails, online conversations, or even the automatically generated transcription from phone conversations.

There are two main approaches to perform this kind of sentiment analysis: classical and deep learning. Both of these methods are available with python.

Using a dictionary of manually defined keywords :

It is based on the assumption that we know what words are typically associated with positive and negative emotions. For example, if we are going to classify movie reviews, we expect to find words such as “ great” ,“ super” , and “ love” in positive comments and words like “ hate” ,“ bad” , and “ awful” in negative comments. We can count the number of occurrences of every selected word to define feature vectors. Then, we can train a sentiment analysis classifier on each comment.

BAG OF WORDS:

Instead of calculating only words selected by domain experts, we can calculate the occurrences of every word that we have in our language (or every word that occurs at least once in all of our data). This will cause our vectors to be much longer, but we can be sure that we will not miss any word that is important for prediction of sentiment.

TF-IDF STRATERGY:

We can think about **TF-IDF** as a modified version of the bag of words. Instead of treating every word equally, we normalize the number of occurrences of specific words by the number of its occurrences in our whole data set and the number of

words in our document (comments, reviews, etc.). This means that our model will be less sensitive to occurrences of common words like “ and” ,“ or” ,“ the” ,“ opinion” etc., and focus on the words that are valuable for analysis.

PROGRAM:

```
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
```

```
def expression_check(prediction_input):
    if prediction_input == 0:
        print("Input statement has Negative Sentiment.")
    elif prediction_input == 1:
        print("Input statement has Positive Sentiment.")
    else:
        print("Invalid Statement.")
# function to take the input statement and perform the same transformations we did earlier
def sentiment_predictor(input):
    input = text_transformation(input)
    transformed_input = cv.transform(input)
    prediction = rfc.predict(transformed_input)
    expression_check(prediction)
input1 = ["Sometimes I just want to punch someone in the face."]
input2 = ["I bought a new phone and it's so good."]
sentiment_predictor(input1)
sentiment_predictor(input2)
```

Output:

```
Input statement has Negative Sentiment.
Input statement has Positive Sentiment.
```

GENERATING INSIGHTS :

- **Topics:** Sentiment analysis can identify the main themes that are coming up in your data – like product quality, shipping, customer service, pricing, and so on. You can get pretty granular by analyzing sentiment for each topic to prioritize where you should focus your efforts. Often times, a single customer comment will cover many topics at once.

- **Intent:** Going beyond simple polarity (positive or negative), sentiment analysis can uncover where customers are asking questions or even giving suggestions. Looking at intent can provide more qualitative context behind customer comments.

- **Emotion:** Of course, this is the bread and butter of sentiment analysis. What are your customers feeling in their interactions with your brand? Delight, frustration, excitement, anger? NLP does the heavy lifting here to identify and categorize customer sentiment.

- **Root Cause:** When customers are having issues, sentiment analysis can get to the root cause, a powerful outcome for any company. Details like at which stage of the customer journey a problem occurs enable you to take action to resolve it quickly and efficiently.

- **Sentiment Score:** This is a pretty common metric designed to benchmark the general sentiment for analyzed text data. Drawing correlations between trends in the sentiment score and business decisions, new product offerings, or other milestones can indicate how customers felt about a specific product, service, or experience.

PROGRAM:

```
        import pandas as pd
import matplotlib.pyplot as plt

# Sample sentiment data
data = [
    {"text": "I love this product!", "sentiment": "positive"},
    {"text": "It's an okay movie.", "sentiment": "neutral"},
    {"text": "This is awful.", "sentiment": "negative"},
    # Add more data as needed
]

# Create a DataFrame from the sample data
df = pd.DataFrame(data)

# Overall sentiment distribution
sentiment_counts = df["sentiment"].value_counts()

# Plot the sentiment distribution
sentiment_counts.plot(kind="bar", title="Sentiment Distribution")
plt.xlabel("Sentiment")
plt.ylabel("Count")
plt.show()

# Calculate and display the percentage of positive, negative, and neutral sentiments
total_count = len(df)
positive_percentage = (sentiment_counts.get("positive", 0) / total_count) * 100
negative_percentage = (sentiment_counts.get("negative", 0) / total_count) * 100
neutral_percentage = (sentiment_counts.get("neutral", 0) / total_count) * 100

print(f"Positive Sentiment: {positive_percentage:.2f}%")
print(f"Negative Sentiment: {negative_percentage:.2f}%")
print(f"Neutral Sentiment: {neutral_percentage:.2f}%")
```

OUTPUT:

Positive Sentiment: 33.33%
Negative Sentiment: 33.33%
Neutral Sentiment: 33.33%

CONCLUSION:

A sentiment analysis system helps businesses improve their product offerings by learning what works and what doesn't. Marketers can analyze comments on online review sites, survey responses, and social media posts to gain deeper insights into specific product features.