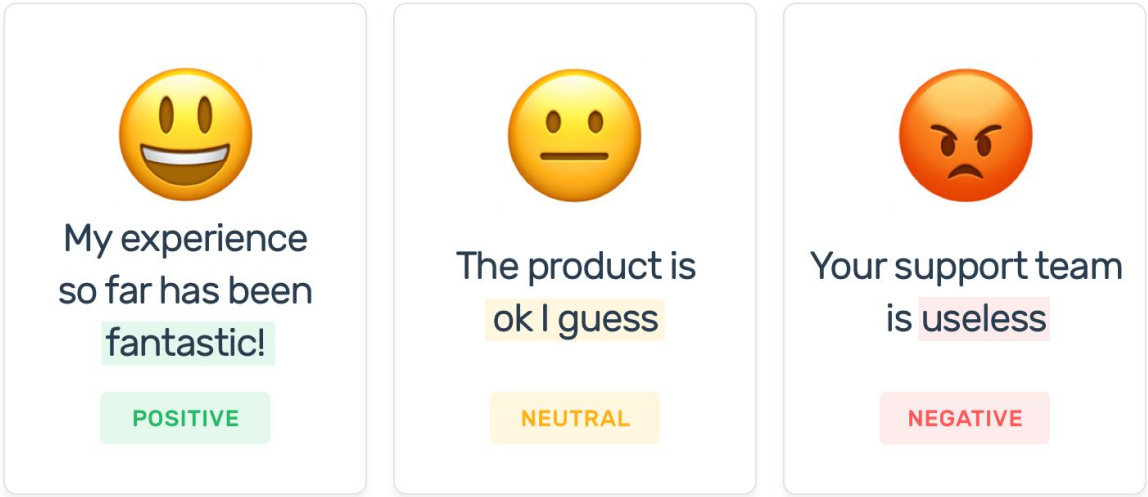
SENTIMENT ANALYSIS ON TWEETS!

BY:
BHARGAVI M 18PD06
MRUDHULA G.P. 18PD18
NITHILAA U. 18PD22

WHAT IS Sentiment analysis?

Sentiment analysis is the interpretation and classification of emotions (positive, negative and neutral) within text data using text analysis techniques.

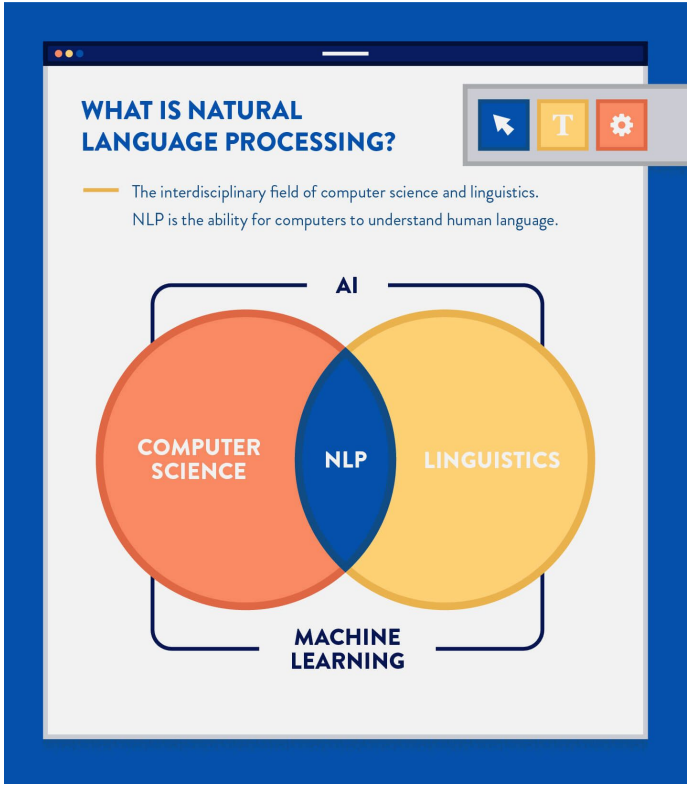


HOW DOES SENTIMENT ANALYSIS WORK?

Sentiment analysis uses various Natural Language Processing (NLP) methods and algorithms.

The main types of algorithms used include:

- **Rule-based systems** that perform sentiment analysis based on a set of manually crafted rules.
- **Automatic systems** that rely on machine learning techniques to learn from data.
- **Hybrid systems** that combine both rule-based and automatic approaches.



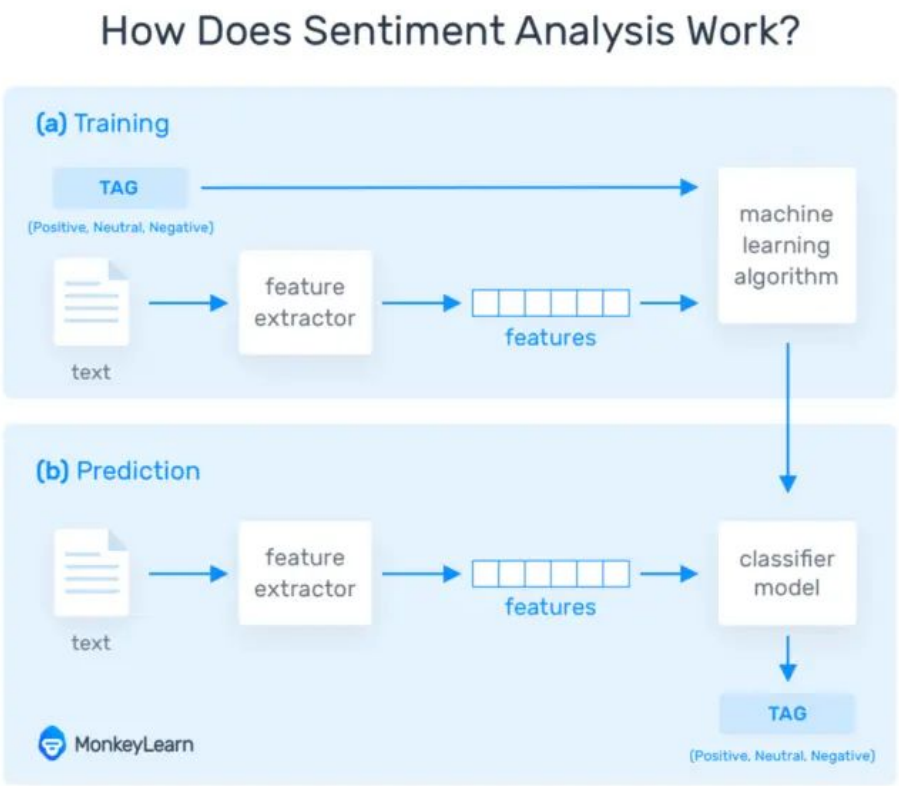
RULE-BASED SYSTEMS

- A rule-based system uses a set of human-crafted rules to help identify subjectivity, polarity, or the subject of an opinion.These rules may include various techniques developed in computational linguistics, such as:
    a. ***Stemming*, *tokenization*, *part-of-speech tagging* and *parsing*.**
    b. **Lexicons** (i.e. lists of words and expressions).
- Rule-based systems are very naive since they don't take into account how words are combined in a sequence. Of course, more advanced processing techniques can be used, and new rules added to support new expressions and vocabulary.

AUTOMATIC SYSTEMS

- Automatic methods, contrary to rule-based systems, don't rely on manually crafted rules, but on **machine learning** techniques.

- A sentiment analysis task is usually modeled as a **classification problem**, whereby a classifier is fed a text and returns a category, e.g. positive, negative, or neutral.



CLASSIFICATION ALGORITHMS

- **Naïve Bayes:** a family of probabilistic algorithms that uses Bayes Theorem to predict the category of a text.
- **Linear Regression:** a very well-known algorithm in statistics used to predict some value (Y) given a set of features (X).
- **Support Vector Machines:** a non-probabilistic model which uses a representation of text examples as points in a multidimensional space. Examples of different categories (sentiments) are mapped to distinct regions within that space. Then, new texts are assigned a category based on similarities with existing texts and the regions they're mapped to.
- **Deep Learning:** a diverse set of algorithms that attempt to mimic the human brain, by employing artificial neural networks to process data.

HYBRID SYSTEMS

- Hybrid systems combine the desirable elements of rule-based and automatic techniques into one system.
- One huge benefit of these systems is that results are often **more accurate.**

AUTOMATIC SYSTEMS

NAIVE BAYES CLASSIFIER

NAIVE BAYES CLASSIFIER

- A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task.
- The crux of the classifier is **based on the Bayes theorem**, given as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

FUNDAMENTAL NAIVE BAYES ASSUMPTION

- We assume that **no pair of features are dependent**. The assumption made here is that the predictors/features are independent.
- Secondly, **each feature is given the same weight**(or importance). That is presence of one particular feature does not affect the other. None of the attributes is irrelevant and assumed to be contributing equally to the outcome.

Hence it is called naive.

Note: The assumptions made by Naive Bayes are not generally correct in real-world situations. In-fact, the independence assumption is never correct but often works well in practice.

CALCULATION

- Bayes theorem can be rewritten as:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

- The variable y is the class variable and variable X represent the parameters/features where X is given as,

$$X = (x_1, x_2, x_3, ..., x_n)$$

where x_1,x_2....x_n represent the features.

NAIVE ASSUMPTION

- Now, it's time to put a naive assumption to the Bayes theorem, which is, independence among the features. So now, we split evidence into the independent parts.
- Now, if any two events A and B are independent, then,

P(A^B) = P(A)P(B)

- By substituting for X and expanding using the chain rule we get,

$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

- Or in short ,it can be expressed as

$$P(y|x_1,...,x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1)P(x_2)...P(x_n)}$$

- Now, as the denominator remains constant for a given input, we can remove that term:

$$P(y|x_1,...,x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y)$$

- Now, we need to create a classifier model. For this, we find the probability of given set of inputs for all possible values of the class variable y and pick up the output with maximum probability. This can be expressed mathematically as:

$$y = argmax_y P(y)\prod_{i=1}^{n} P(x_i|y)$$

TYPES OF NAIVE BAYESIAN CLASSIFIERS

TYPES OF NAIVE BAYESIAN CLASSIFIERS

1.Gaussian Naive Bayes classifier:

In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution(Normal distribution).

2.Multinomial Naive Bayes classifier:

Feature vectors represent the frequencies with which certain events have been generated by a multinomial distribution. This is the event model typically used for document classification.

3.Bernoulli Naive Bayes classifier:

In the multivariate Bernoulli event model, features are independent booleans (binary variables) describing inputs. Like the multinomial model, this model is popular for document classification tasks, where binary term occurrence(i.e. a word occurs in a document or not) features are used.

MULTINOMIAL NAIVE BAYES CLASSIFIER

MULTINOMIAL NAIVE BAYES CLASSIFIER

- The multinomial Naive Bayes classifier is suitable for **classification with discrete features** (e.g., word counts for text classification).
- The multinomial distribution normally requires integer feature counts.
- In this case, we have text. We need to convert this text into numbers that we can do calculations on.
- We use **word frequencies**. That is treating every document as a set of the words it contains. Our features will be the counts of each of these words.

EXAMPLE

DATASET

DOCUMENT TYPE	ID	WORDS	CLASS
Training	D1	Chinese Beijing Chinese	China
Training	D2	Chinese Chinese Shanghai	China
Training	D3	Chinese Macao	China
Training	D4	Tokyo Japanese Chinese	Japan
Test	D5	Chinese Chinese Chinese Tokyo Japanese	?

SOLUTION

$$P(Class) = \frac{N_{class}}{N}$$

$$P\ (word|class) = \frac{count(word,class)+1}{count(c) + |V|}$$

P(China)	3/4
P(Japan)	1/4

P(Chinese|China)	(5+1)/(8+6)	6/14
P(Tokyo|China)	(0+1)/(8+6)	1/14
P(Japanese|China)	(0+1)/(8+6)	1/14
P(Chinese|Japan)	(1+1)/(3+6)	2/9
P(Tokyo|Japan)	(1+1)/(3+6)	2/9
P(Japanese|Japan)	(1+1)/(3+6)	2/9

SOLUTION CONTINUED...

P(Class|D5) = P(Class)*P(Word|Class)

P(China|D5) = ¾ * 6/14 * 6/14 * 6/14 * 1/14 * 1/14
= 0.0003

P(Japan|D5) = ¼ * 2/9 * 2/9 * 2/9 * 2/9 * 2/9
= 0.0001

As P(China|D5) > P(Japan|D5),
the document D5 belongs to the class China.

PROS OF NAIVE BAYES CLASSIFIER

- Very fast, low storage requirements
- Robust to irrelevant features
    - Irrelevant features cancel each other without affecting results
- Very good in domains with equally important
    - Decision trees suffer from fragmentation in such cases-especially if little data
- Optimal if the independence assumptions hold

IMPLEMENTATION USING PYTHON

TOKENIZATION

Tokenization is the process by which big quantity of text is divided into smaller parts called tokens.

These tokens are very useful for finding patterns as well as is considered as a base step for stemming and lemmatization.

nltk library allows two type of tokenization

1. Tokenization of words

   "God is Great! I won a lottery."⇢ ['God', 'is', 'Great', '!', 'I', 'won', 'a', 'lottery', '.']

2. Tokenization of sentence

   "God is Great! I won a lottery." ⇢['God is Great!', 'I won a lottery ']

NORMALIZING THE DATA

Normalization in NLP is the process of converting a word to its canonical form.Normalization helps group together words with the same meaning but different forms.

Lemmatization

- Lemmatization is the algorithmic process of finding the lemma of a word depending on their meaning.
- Lemmatization usually refers to the morphological analysis of words, which aims to remove inflectional endings.
- It helps in returning the base or dictionary form of a word, which is known as the lemma.

Stemming and lemmatization can be done using *PorterStemmer* and *WordNetLemmatizer* in *nltk.stem* respectively

REMOVING NOISE FROM THE DATA

Noise is any part of the text that does not add meaning or information to data. So we have to remove them manually.

Hyperlinks:

> All hyperlinks in Twitter are converted to the URL shortener t.co. Therefore, keeping them in the text processing would not add any value to the analysis.

Twitter handles in replies:

> These Twitter usernames are preceded by a @ symbol, which does not convey any meaning.

Punctuation and special characters:

While these often provide context to textual data, this context is often difficult to process. For simplicity, you will remove all punctuation and special characters from tweets.

Stop words:

The most common words in a language are called stop words. Eg :“is”, “the”, and “a”. They are generally irrelevant when processing language, unless a specific use case warrants their inclusion.*nltk* has a stopwords library and they can be used to remove the unwanted data from our dataset.

DETERMINING WORD DENSITY

- The most basic form of analysis on textual data is to take out the word frequency.
- A single tweet is too small of an entity to find out the distribution of words, hence, the analysis of the frequency of words would be done on all positive tweets(and negative tweets).
- So we combine all the tokens from positive tweets first and then find their frequency using *FreqDist* class of *nltk.*
- We find the word frequency of negative tweets similarly.

APPLYING THE ALGORITHM

- Here, we define a function, (here the model for our data), as in the way discussed in the above example, which hence acts as the predictor for classifying a given custom tweet.
- The model is made to fit for the training data and tested using the test data and any randomly accepted input tweet
- The accuracy is then found to rate the model built, and is found to be 99.85%.

NOTE: Python also offers an inbuilt function **MultinomialNB()** which can be imported as **from sklearn.naive_bayes import MultinomialNB,** to implement the same

APPLICATIONS OF SENTIMENT ANALYSIS

- Social media monitoring
- Brand monitoring
- Voice of customer (VoC)
- Customer service
- Market research

BENEFITS OF SENTIMENT ANALYSIS

1. Sentiment analysis helps businesses process huge amounts of data in an efficient and cost-effective way.

2. Real-Time Analysis Sentiment analysis can identify critical issues in real-time, so you can take action right away.

3. By using a centralized sentiment analysis system, companies can apply the same criteria to all of their data, helping them improve accuracy and gain better insights.

CHALLENGES IN SENTIMENT ANALYSIS

1. **Subjectivity and Tone**

   The detection of subjective and objective texts is just as important as analyzing their tone. In fact, so called *objective* texts do not contain explicit sentiments.

2. **Irony and Sarcasm**

   When it comes to *irony* and *sarcasm*, people express their negative sentiments using positive words, which can be difficult for machines to detect without having a thorough understanding of the context of the situation in which a feeling was expressed.

3. **Comparisons**

   How to treat comparisons in sentiment analysis is another challenge worth tackling.

4. **Defining Neutral**

   Defining what we mean by *neutral* is another challenge to tackle in order to perform accurate sentiment analysis.

THANK YOU!