Sentiment Analysis using Naive Bayes Classifier

What is Sentiment Analysis?

- Sentiment analysis (also known as opinion mining) refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials.

- Sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document.

- Mainly built for commercial use.

Concept

- The basic task in sentiment analysis is classifying the *polarity* of a given text at the document, sentence, or feature/aspect level.

- Whether the expressed opinion in a document, a sentence or an entity feature/aspect is positive, negative, or neutral.

- We use various Data Mining and Machine Learning techniques to achieve this.

- **Aggregating :**
  - Storing the text data by Scraping different Websites.

- **Training classifier :**
  - Training the classifier (Naive Bayes in this case) using the stored data.

- **Filtering the Live data :**
  - Keep only the relevant sentences useful for finding sentiments.

- **Classifying :**
  - We will classify the filtered data into respective classes using the trained classifier.

Relation to Research Paper

- The research paper specifies about filtering the text data before tagging words into a particular class.
- This will help to reduce the erroneous results hence increasing accuracy of the prediction.
- We will score the classified words and accordingly present the output.

Applications

- Consumers can use sentiment analysis to research products or services before making a purchase. E.g. Kindle

- Marketers can use this to research public opinion of their company and products, or to analyze customer satisfaction. E.g. Election Polls

- Organizations can also use this to gather critical feedback about problems in newly released products. E.g. Brand Management (Nike, Adidas)

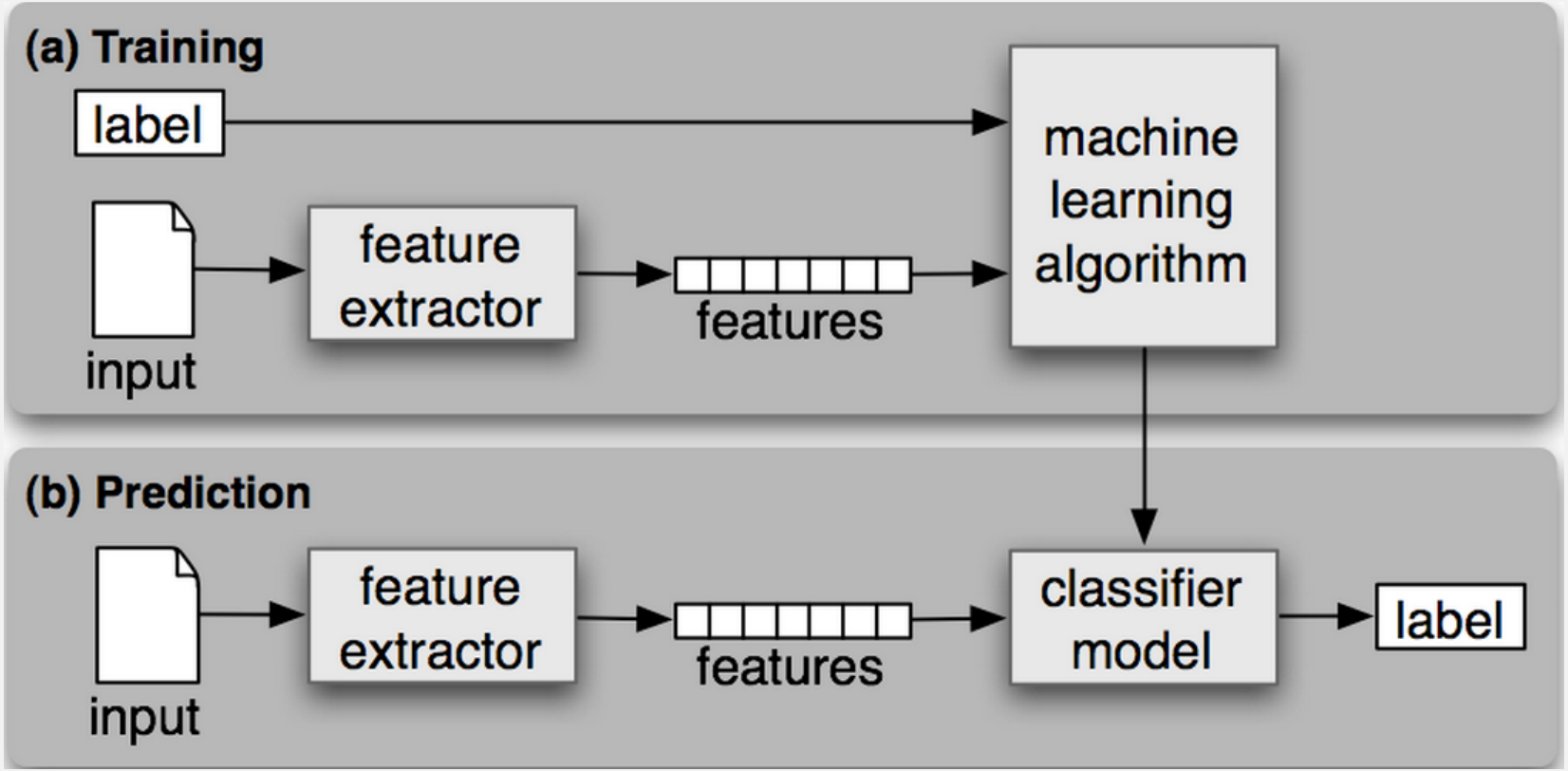
Advantages

- With positive and negative sentiments, we also focus on neutral sentiments without which we may have an incorrect prediction of sentiments.

- We will filter out the “Stop Words”, these words do not have an impact on the sentiments, hence increasing the accuracy of prediction.

- We use the concept of Feature Vector which will create a model for more accurately training the Classifier.

BLOCK DIAGRAM



Implementation

- The text data is stored and parsed for the relevant information like **nouns**(names), **adjectives**(great),etc. We will filter the data first.

- Filtering involves removing **emoticons**(:P),**extra alphabets**(hungryyy), **punctuation marks**(, . ! #),stop words (is,a,the).

- The data is then fed into the classifier algorithm like **Naive Bayes**, SVM, Maximum Entropy,etc which will understand the data and give **weight** to the data.

- In this way the classifier algorithm will keep **learning** new data every time it encounters a new data.

- Also it becomes more and more **accurate** over the number of times new data it learns.

- For the purpose of this project and **feasibility** concerns we will use a data-set of **10,000** which the algorithm will learn.

- At the end the algorithm will be able to accurately **classify the data as positive, negative or neutral**.

- The result can be show visually using **graph** or **chart** by means of an web application.

Hardware Requirements

- Intel **i3** Processor
- **2 GB** RAM
- **500GB** HDD
- Server for hosting Web Application like **GoDaddy**.com

Software Requirements

- Python 3 or above
- Natural Language Toolkit 3.0 (NLTK)
- Representational State Transfer (REST) API
- HTML ,CSS , JavaScript ,etc. for Web Application

Learning Requirements

- Learning **basics** of Python and importing libraries.
- Learning to parse **regular expression** and extracting data.
- Implementation of **NLTK library** in Python.
- Learning the NLTK library and understanding the **fundamentals** of it.
- Understanding the **corpus** to use and implement.
- Implementation of **REST API** for integration with Twitter.

PLAN

INITIAL PHASE:

- Learning the basics required for scraping the data from various sources.
- Building a basic scraping tool in Python.
- Make the necessary changes required as per the need.
- Choosing the source for scraping the data.

MIDDLE PHASE:

- Understanding the classifier algorithm i.e. Naive Bayes.
- Coding the algorithm.
- Testing the scraped data on the algorithm and doing initial tests.
- Train the algorithm for the dataset scraped.
- Confirm the correctness of the result.

LAST PHASE:

- Making the UI of Web Application.
- Using various tools to create Web Application.
- Integrating the results of the classifier algorithm to the application.
- Displaying the results using charts or graphs.

THANK YOU

GAURAV CHAVAN - 11
PARIKSHIT HEGDE - 30
SAGAR MANJARE - 44