NAME : URMI DEDHIA
SAP ID : 60004200108
BRANCH : SE COMPS

Synapse Task 2

Natural Language Processing: Sentiment Analysis (SA)

▪ Preprocessing:

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model by cleaning it and putting it in a formatted way. We have a corpus of reviews of a restaurant on a food delivery app and we are given a labelled dataset here for a supervised learning model. Following are the pre-processing steps to be followed for our sentiment analysis model. (Note : More steps can be followed too for further accuracy and efficiency.)

1. **Tokenizing sentences** to break text down into sentences, words, or other units. But the captured tokens also include punctuation and non-word strings. So punctuation has to be removed further.

2. **Removing stop words** like ‘if’, ‘but’, ‘or’, ‘a’ and so on. They are the words which have little to no relevance due to their common occurrence and are removed to get a refined list of relevant words.

3. **Normalizing words** by condensing all forms of a word into a single form (‘laughed’, ‘laughing’ and ‘laugh’ will all get condensed to one word ‘laugh’).
   It can be done in two ways viz. stemming and lemmatization.
   1. Stemming : Stemming cuts off the word at its stem (root word) i.e. till the ancestor from which the other words can be created using suffixes. But stemming algorithms cannot normalize ‘better’ to its stem ‘good’ or ‘felt’ to its stem ‘feel’ hence we will use lemmatization over stemming for better accuracy.
   2. Lemmatization : In lemmatization, the algorithm actually knows the meaning of the word by referring to a dictionary and hence will condense even the above words correctly to their lemma (root word).

4. **Converting into lowercase** helps in maintaining simplicity and a consistent flow while working on the data.

5. **Vectorizing text** is a process that transforms a token into a vector, or a numeric array that is unique to and represents various features of a token.

6. **Splitting the data** into training and testing data sets after shuffling it to avoid any bias.

▪ Text analyzing techniques:

Usually, following are the steps followed for building a machine learning model :

1. Splitting the data into training and evaluation sets.
2. Selecting a model architecture.
3. Using training data to train the model.
4. Using test data to evaluate the performance of the model.
5. Using the trained model on new data to generate predictions, which in this case will be a number (0 or 1).

1. Splitting the data into training and evaluation sets: We've already split the dataset (words and their labels (p/n)) into two sets of each – training dataset and testing dataset. The training set, as the name implies, is used to train the model. The test set is a dataset that incorporates a wide variety of data to evaluate and accurately judge the performance of the model.

2. Selecting a model architecture: Here we select a machine learning classifier among the tons of options available depending upon our task and its requirements, size of our data, the complexity, the computing time, etc. I've selected the Naïve Bayes classifier.

Naïve Bayes algorithm :

Naïve Bayes is a probabilistic machine learning algorithm that can be used in a wide variety of classification tasks. Typical applications include filtering spam, classifying documents, **sentiment prediction** etc.

It is based on the works of Rev. Thomas Bayes (Bayes' theorem) and hence the name. The name naïve is used because it assumes that the features that go into the model are independent of each other. That is changing the value of one feature, does not directly influence or change the value of any of the other features used in the algorithm.

> **Bayes' Theorem** : Bayes' theorem, also known as Bayes' Rule or Bayes' law, is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability. The formula for Bayes' theorem is given as:
>
> $$P(A|B)= \frac{P(B|A)P(A)}{P(B)}$$
>
> Where, **P(A|B) is Posterior probability**: Probability of hypothesis A on the observed event B.
>
> **P(B|A) is Likelihood probability**: Probability of the evidence given that the probability of a hypothesis is true.
>
> **P(A) is Prior Probability**: Probability of hypothesis before observing the evidence.
>
> **P(B) is Marginal Probability**: Probability of Evidence.

Using Naïve Bayes classifier, we'll count up how many times each word occurs in the negative reviews, and how many times each word occurs in the positive reviews. This will allow us to eventually compute the probabilities of a new review belonging to each class.

e.g. : Let A = the label (positive/negative) and B = the word/feature.

Say we want to find the probability that the review 'didn't like it' expresses a negative sentiment. We would find the total number of times the word 'didn't' occurred in the

negative reviews, and divide it by the total number of words in the negative reviews to get **P(B|A)**. We would then do the same for 'like' and 'it'. We would multiply all three probabilities (since they are **independent** of each other) and then multiply by the probability of any document (review) expressing a negative sentiment i.e. **P(A)** to get our final probability that the sentence expresses negative sentiment i.e. **P(A|B)**. (**P(B)** is a normalization constant to make the probability distribution sum to 1). We would do the same for positive sentiment, and then whichever probability is greater would be the class that the review is assigned to. To do all this, we'll need to compute the probabilities of each class occurring in the data, and then make a function to compute the classification.

3. Using training data to train the model: After splitting the data sets and vectorizing the tokens, we fit a Naïve Bayes model to the training dataset. This will train the model using the word counts we computer and the existing classifications in the training set.

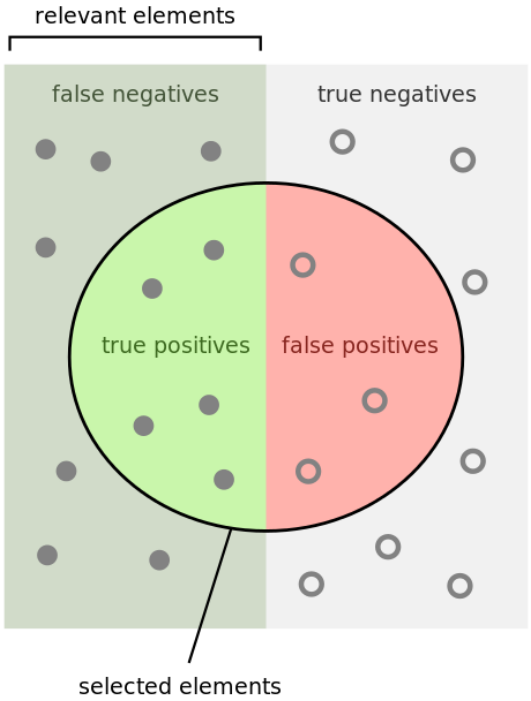
e.g. MultinomialNB() from scikit learn is a suitable Naïve Bayes classifier for discrete features.

4. Using test data to evaluate the performance of the model: After training our model with the training dataset, we have to use another fresh dataset to check and evaluate the working of our model because we will get misleadingly good results if we predict on the same reviews from the training dataset, as the probabilities were generated from it and the algorithm has prior knowledge about the data it's predicting on.

After testing our model with the testing dataset, to know the level of accuracy of our model, several evaluation metrics are used. They give different levels of accuracy scores which help us determine our model's efficiency.

Evaluation metrics used :

(Here, TP = True Positives, TN = True Negatives, FP = False Positives, FN = False Negatives)



1. **Accuracy :** Classification Accuracy is the ratio of number of correct predictions to the total number of input samples.

   Accuracy = (TP + TN) / (TP + TN + FP + FN)

   But classification accuracy scores give a false sense of high accuracy of the model hence we resort to other metrics for better evaluation.

2. **Precision :** Precision quantifies the number of positive class predictions that actually belong to the positive class i.e. true positives upon the selected elements.

   Precision = TP / (TP + FP)

3. **Recall :** Recall quantifies the number of positive class predictions made out of all positive examples in the dataset i.e. true positives upon the relevant elements.

   Recall = TP / (TP + FN)

4. **F-Score :** F-Measure or F-Score provides a way to combine both precision and recall into a single measure that captures both properties, giving each the same weighting. It is the harmonic mean of precision and recall.

   F-Score = (2 * Precision * Recall) / (Precision + Recall)

5. Using the trained model on new data to generate predictions: Finally, we use our model to predict the class for the required data i.e. the reviews for the restaurant on the food delivery app.

Features which can be used for improving accuracy of our model :

1. **Laplace Correction :** Laplace smoothing is a smoothing technique that helps tackle the problem of zero probability that occurs in our Naïve Bayes algorithm. when a word in a review we test was not present in the training dataset.

2. **TF-IDF :** It is a method of extracting the features from the text data. TF stands for Term Frequency and IDF stands for Inverse Document Frequency.
   - Term Frequency : Number of times word occurs in a review.
   - Inverse Document Frequency : It is computed as,

     idf(t) = log [number of documents / number of documents containing the term] + 1

   This helps in extracting only the most relevant words and hence helps in saving computing time and makes our model more efficient.

3. **Using n-grams :** In our model, we used only single words as features while predicting. But single word classification can make errors while computing a phrase like “not bad” or “not good”. Using frequent n-grams as features in addition to single words can overcome this problem.

- **Rating system:**

Let us consider two cases depending upon the type of output our model predicts :

1. Our model returns '1' for a positive review and '0' for a negative review.
2. Our model returns a number (can be a float value) between -1 to +1 with -1 being the most negative and +1 being the most positive. (If another suitable algorithm is used)

We will be devising a five-star rating system for both the cases.

1. Consider N reviews labelled either '1' for positive review and '0' for negative review.
   Let PR be the number of positive reviews.
   Let X be the amount of stars out of five given to the restaurant.
   $$\therefore X = \frac{5 \times PR}{N}$$

2. Consider N reviews labelled any float value between the range -1 to +1
   First for each review, we will calculate its separate rating of x where x is the amount of stars out of five.
   Let y be the predicted float value of that review.
   $$\therefore x = \frac{5(y+1)}{2}$$
   Let X be the amount of stars out of five given to the restaurant.
   $$\therefore X = \frac{\sum\{[5(y+1)]/2\}}{N}$$

▪ Conclusion (Algorithm):

Hence we've built a model for sentiment analysis using Naïve Bayes algorithm.

However, some algorithms which could've been used for text classification except Naïve Bayes :

1. Support vector machines (SVM) - SVM assigns a hyperplane that best separates the tags (in our case - positive/negative) i.e. anything on one side of the plane is positive and on the other, negative.
2. Linear regression - Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x).
3. Random Forest - It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

Deep learning algorithms :

1. Recurrent Neural Network (RNN)
2. Convolutional Neural Network (CNN)

Some **Advantages** of working with Naïve Bayes are :

1. It requires a small amount of training data to learn the parameters.
2. It can be trained relatively fast compared to sophisticated models.
3. It uses less computing power.
4. It is the simplest and fastest classification algorithm for a large chunk of data.

Although there are some disadvantages to it too :

1. It assumes that all features are independent, which rarely happens in real life.
2. It brings the problem of zero probability however it can be dealt with smoothing methods like Laplace's correction.
3. It is a poorer estimator than the other algorithms.

Based on the above advantages and disadvantages collectively, I decided to use Naïve Bayes algorithm since the advantages overshadow the disadvantage.