**NLP project summary**

**Goal**

This project aims to build a model that predicts the sentiment of movie reviews. Humans can easily understand sentiment because we can perceive emotions. However, a machine does not have feelings. It will only predict ‘sentiment’ based on keywords contained in comments.

**Input**

A dataset called ‘movie\_review’ from the nltk package, which is a commonly used tool in the NLP field. ‘movie\_review’ contains two columns

* ‘review’: people’s comments on a movie; usually, it is 1 or 2 sentences.
* ‘sentiment’: positive or negative

**Method**

The algorithm consists of two parts: feature engineering and model fitting.

1. Feature engineering

Feature engineering is a vital step for data science because some noise in the raw data deteriorates prediction accuracy. Hence, I performed:

* *Removing punctuation*

Punctuation has almost no effect on the sentiment of movie reviews. One can argue that the exclamation mark may represent a strong emotion, but it is unclear whether this emotion is positive or negative. Thus, punctuation is removed.

* *Removing numbers*

Numbers do not reveal sentiments.

* *Removing stop words*

There is no exact definition for stop words. In my understanding, stop words are simple words that appear in most sentences. For example, ‘is,’ ‘a,’ ‘how,’ and ‘when’ are considered stop words. They do not change the sentiment of a sentence.

* *Stemming*

Stemming is a process that changes the verb to its original form. For example, ‘play,’ ‘plays,’ ‘played’ are all derived from the word ‘play.’ Hence, stemming will change all words to ‘play.’ After stemming, we want to remove duplicated ‘play’ because one word will be sufficient for the prediction.

1. Model fitting

This project adopts the Python Naïve Bayes classifier. The following diagram explains the Bayes rule clearly.

Text

Description automatically generated

As shown above, the LHS is the posterior probability we need to obtain. The RHS is an iterative process that updates the LHS. We can start with an inaccurate probability (i.e., prior) and then get a better and better result by multiplying the prior and likelihood. In practice, I used the naïve bayes classifier in sklearn package.

After the classifier is chosen, I built a dictionary, ‘labelsets,’ to record which words appear in the word\_feature array. A word\_feature array contains the 2000 most frequent words that exist in the set of all reviews. The intuition is that we want to differentiate reviews by counting the number of frequent words.

**Output**

The train-test ratio is 75%:25%. After training the model, the classifier has a 80.8% accuracy rate.