ASSESSMENT COVER SHEET

Assignment Details

Course: Applied Natural Language Processing

Semester/Academic Year: Semester 1/ 2022

Assignment title: Assignment 1: Building a sentiment analysis system with Naïve Bayes

Assessment Criteria

Assessment Criteria are included in the Assignment Descriptions that are published on each course's web site.

Plagiarism and Collusion

Plagiarism: using another person's ideas, designs, words or works without appropriate acknowledgement.

Collusion: another person assisting in the production of an assessment submission without the express requirement, or consent or knowledge of the assessor.

Consequences of Plagiarism and Collusion

The penalties associated with plagiarism and collusion are designed to impose sanctions on offenders that reflect the seriousness of the University's commitment to academic integrity. Penalties may include: the requirement to revise and resubmit assessment work, receiving a result of zero for the assessment work, failing the course, expulsion and/or receiving a financial penalty.

DECLARATION

I declare that all material in this assessment is my own work except where there is clear acknowledgement and reference to the work of others. I have read the University Policy Statement on Plagiarism, Collusion and Related Forms of Cheating:

http://www.adelaide.edu.au/policies/?230

I give permission for my assessment work to be reproduced and submitted to academic staff for the purposes of assessment and to be copied, submitted and retained in a form suitable for electronic checking of plagiarism.

HN 04-04-2020

SIGNATURE AND DATE

Applied Natural Language Processing (COMP SCI 7417)
Assignment 1 Report

Cheuk Hang Ng (a1821087)

University of Adelaide, a1821087@student.adelaide.edu.au

1 INTRODUCTION

This assignment aims to write a code in Python for building a sentiment analysis system based on the Naïve Bayes Classifier from scratch and evaluate the model by building a F1 measure from scratch. The code was written and run on Jupyter Notebook Python environment. The dataset provided is an IMDB movie review dataset provided by the course. The dataset was first read as a dataframe and investigated, and some manipulations were made before feeding into the model. For example, rows labelled as 'unsup' were dropped and the labels were mapped as 1 and 0. The dataset was then splitted into train set and test set according to the value under the 'Type' column. Some of the columns were then dropped such that the dataframe contains only the labels and the review text only. Impacts of different text normalization processes and two smoothing methods with a range of parameters to the performance of the model on the dataset were investigated.

2 DESCRIPTIONS OF DETAILS

2.1 Text Pre-processing Techniques

Numbers of text pre-processing techniques were used in the implementation, which include a general text pre-processing, stopwords removal and text normalization. In this assignment, stemming and lemmatization were applied in the text normalization process.

2.1.1 General Text pre-processing, tokenization and stopwords removal

The first variant of the dataset was generated by applying only the pre-processing described in this sub-section. The general text cleaning processes were lowercasing all characters, removing numbers, punctuations and extra spacing in the review text. Tokenization and stopwords removal were done immediately after the previous pre-processing, both with the methods from the nltk libraries (nltk.word_tokenize and nltk.corpus.stopwords). A small adjustment was made to the list of stopwords, which is the string '<br />' was added to the list. It was due to the observation of repeated occurrence of the string in the review text from the initial examination on the data.

2.1.2 Stemming and lemmatization

Two variants of the dataset were produced by applying the steps in previous section together with word stemming or lemmatization. Both process greatly reduces the pattern variance in the text by chopping off affixes (stemming) or reduce words to root words (lemmatization) despite the disadvantage on losing information. Snowballstemmer (nltk.stem.snowball.SnowballStemmer), which is a better version of the traditional Porter stemmer and wordNetLemmatizer fom nltk (nltk.stem.WordNetLemmatizer) were selected for stemming and lemmatization.

2.2 Training the Naïve Bayes Classifier

A Naïve Bayes Classifier was then built from the pre-processed review from the train set. The underlying principles of this classifier follow that introduced in the course, where the classifier first calculate the histogram of the word occurrence first, and then perform classification. This classifier also adapted the logarithm probability practice, which allows us to sum up the probabilities of the tokens instead of multiplying them altogether. Some parameters were defined before the data were fed into the model: (1) A dictionary with tokens as the key and occurrence as the value, regardless of the labels of the tokens, and the (2) total number of tokens in the dictionary. (3) & (4) The number of reviews from each label and the probability of a review being a specific label were also saved. (5) & (6) Two dictionaries with tokens from the 'pos' and 'neg' label respectively as the key and occurrence as the value, and the (7) & (8) total number of tokens in each dictionary were also recorded for later use. The classifier creates two columns named 'score' and 'prediction', writing the score from the classifier and the predicted label in the test set. The predicted label is either 1, representing 'pos' class, or 0 which represent the 'neg' class, depends on the score calculated from the classifier. The classifier calculates the probability of a token given the label and return the logarithm of the probability of the token given class 'pos' divided by the probability of the token given class 'neg' for next step. The score is calculated by summing up all the values in previous steps of all tokens in a particular review. The score is a real number and if it is greater than 0, the review is classified as 'pos' and vice versa. In the calculation of the probability, if the token does not exist in the vocabulary (i.e. (1)), it will lead to zero probability assignment which will fail the whole classifier. To handle this problem, the classifier built adapted two types of smoothing methods.

2.2.1 Add-k smoothing

The add-k smoothing method assumes every token occurred k times more than the actual occurrence in the training data. Under such assumption, even an unseen token appears in the test data, it would be assumed that it occurred in the training data k times, which avoid the zero probability assignment problem. In this assignment, the performances of add-1 and add-10 smoothing were compared. In particular, the probability of the token given class 'pos' ('neg') is calculated by accessing the occurrence of that token from (5) ((6)) plus the parameter k, divided by the sum of the total number of tokens in 'pos' (7) ('neg' (8)) and k times the total number of tokens regardless of class (2).

2.2.2 Linear interpolation

The linear interpolation method interprets the probability of a token given the label by a weighted sum of two parts with a parameter $\lambda \in (0, 1)$. The first part being the $(1 - \lambda)$ times the occurrence of a particular token from (5) ((6)) divided by the sum of the total number of tokens in 'pos' (7) ('neg' (8)), while the second part is λ times the occurrence of that token in all review divided by the total number of tokens regardless of class (2). It avoids the zero probability assignment problem by returning the logarithm ratio of the numbers of tokens in each dictionary ((7) & (8)).

3 DESCRIPTION OF EVALUATION METHOD

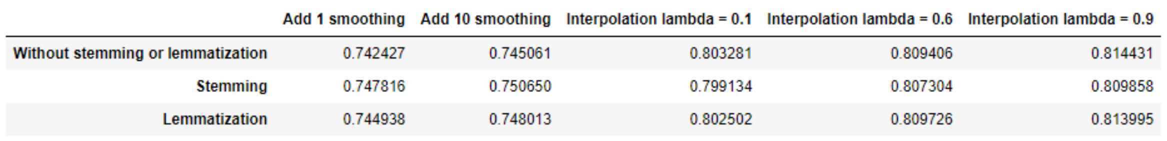
The model was evaluated by implementing the F1 measure from scratch. The F1 measure is calculated by the following formula:

$$F_1 = \frac{(1^2 + 1)PR}{1^2P + R} = \frac{2PR}{P + R}$$

where P refers to the precision of the model and R refers to the recall of the model. Precision is calculated by the ratio between true positive and the sum of true positive and false negative. Recall is calculated by the ratio between true positive and the sum of true positive and false positive. This assignment took the label 1 as positive case. To be specific, the true positives are the rows where both label and prediction equals 1, the false negative are the rows where label equals 1 while prediction equals 0 and the false positive are the rows where label equals 0 while prediction equals 1.

4 RESULT AND DISCUSSION

The results are shown in the table below. The first row being the dataset pre-processed only with the general steps, while the second and third rows correspond to the same dataset but cleaned and stemmed or lemmatized. The columns are the two smoothing methods described in section 2.2 with varied parameters. The cell values disclose the F1 measures for each pair of data and method.



	Add 1 smoothing	Add 10 smoothing	Interpolation lambda = 0.1	Interpolation lambda = 0.6	Interpolation lambda = 0.9
Without stemming or lemmatization	0.742427	0.745061	0.803281	0.809406	0.814431
Stemming	0.747816	0.750650	0.799134	0.807304	0.809858
Lemmatization	0.744938	0.748013	0.802502	0.809726	0.813995

In general, the F1 scores with the interpolation smoothing are higher than those from the add-k smoothing. This might be due to the nature of the add-k smoothing method only assume k more occurrence of an instance while the interpolation method interprets the probability by a weighted sum of the occurrence ratio in the class dictionary and the universal dictionary. The performance of add-10 smoothing is slightly better than the add-1 smoothing, which might be relevant to the symmetry in the dataset, i.e. the same number of reviews from each class, as the outcome of add-k smoothing tends to 0.5 as k increases, which favors the number of correct guesses from the model under larger k. The same happens to the choices of λ in the interpolation smoothing. Normally, the performance of the interpolation smoothing with small λ would be better, as it weights more on the class dictionary of the token rather than the general dictionary. In the case of this assignment, the interpolation smoothing with large λ performed slight better because of the numbers of reviews from each label are same, and therefore the model made more correct 'blind' guesses (given that it weighted more on the universal dictionary, it omitted the class dictionary in some sense).

For text pre-processing methods-wise, we only see insignificant difference between the performances of the model on the three variants of the dataset with different pre-processing steps. If we ignore the insignificance in the F1 scores, we would find that the performance from the lemmatized dataset was the best out of the three variants, outperforming the other two candidates in 3 smoothing methods out of 5. To some degree, the results prove the general impression of lemmatization has the greater ability to reduce token size but keeping the word meaning than stemming, while it gathers more tokens with same original form that doing nothing on the text, which enhances the performance of a sentiment analysis model.