FIT5149 Assessment 2: Sentiment Classification for Product Reviews

Oct 2019

Marks	35% of all marks for the unit
Due Date	23:55 Friday 25 Oct 2019
Extension	An extension could be granted for circumstances. A special consideration application form must be submitted. Please refer to the university webpage on special consideration.
Lateness	For all assessment items handed in after the official due date, and without an agreed extension, a 10% penalty applies to the student's mark for each day after the due date (including weekends, and public holidays) for up to 5 days. Assessment items handed in after 5 days will not be considered.
Authorship	This assignment is **a group assignment** and the final submission must be identifiable your group's own work. Breaches of this requirement will result in an assignment not being accepted for assessment and many result in disciplinary action.
Submission	Each group is required to submit two files, one PDF file contains the report, and another is a ZIP file containing the implementation and the grouped assignment coversheet. The two files must be submitted via Moodle. All the group members are required to accept the terms and conditions in the Moodle submission page. A draft submission won't be marked.
Programming language	Either R or Python

Note: **Please read the description from the start to the end carefully before you start your work!** Given that it is a group assessment, each group should evenly distribute the work among all the group members.

1 Introduction

Sentiment analysis also known as opinion mining is a subfield within Natural Language Processing (NLP) that builds machine learning algorithms to classify a text according to the sentimental polarities of opinions it contains, e.g., positive or negative. In recent year, sentiment analysis has become a topic of great interest and development in both academics and industry. Analysing the sentiment of texts could benefit, for example, customer services, product analytics, market research etc. Take Ebay as an example: Customers on Ebay choose their preferred products based on the reviews from other users. an automatic sentiment classification system can not only help companies grasp the satisfaction level of the products, but also significantly assist new customers to locate their online shopping shelves.

In this data analysis challenge, we are interested in developing such an automatic sentiment classification system that relies on machine learning techniques to learn from a large set of product reviews provided by Yelp. The levels of polarity of opinion we consider include strong negative, weak negative, neutral, weak positive, and strong positive. For example, "*Website says open, Google says open, Yelp says open on Sundays. Our delivery was cancelled suddenly and no one is answering the phone. Shame*" gives us a strong negative sentiment, whereas the sentiment of "*They have great food & definitely excellent service. Tried their mochi mango flavored and it s definitely delis*" is likely to be strong positive.

The sentiment analysis task is often formulated as a classification problem, where a classifier is fed with a text and returns the corresponding sentiment label, e.g., positive, negative, or neutral. In other words, the problem of learning the sentimental polarities of opinions is reduced to a classification problem. There are many machine learning methods that can be used in the classification task. They can be categorised into supervised method (like SVM), unsupervised method (like clustering), and semi-supervised method (like self training).

In a supervised learning setting, there are in general three major steps, including generating features, developing a proper classifier, and applying the classifier to the unseen data. The feature extractor is shared by both training and prediction, which tells us that data used in training and prediction should share the same feature space.

In the real word scenario, labeled data is rare and expensive while unlabelled data is abundant. **The aim of this challenge** is to develop a sentiment classifier that can assign a large set of product reviews to the five levels of polarity of opinion as accurately as possible, given a small amount of labeled reviews and a large amount of unlabelled reviews. It is a multi-class classification task, where each product review is labeled with one of the five sentiment labels, which are strong negative, weak negative, neutral, weak positive, and strong positive.

2 Dataset

Dataset	Type	classes	num of labeled docs	num of unlabelled docs	num of testing docs
Yelp	Sentiment	5	50k	600k	50k

Table 1: Sentiment Classification data set.

We provide the following data sets (Table 1):

- labeled_data.csv: review text and sentiment labels. It contains 50,000 product reviews, each of which is annotated with the corresponding sentiment label. The label set (1,2,3,4,5) refer to five polarity levels (strong negative, weak negative, neutral, weak positive, strong and positive)
- unlabelled_data.csv: review text only. It contains 600,000 product review without labels. This data set can be used in training.
- test_data.csv: test_id and review text. It contains 50,000 product reviews which acts as the testing data. Please note that the testing data set will be released about one week before the assessment due day.

Warning:

- Reverse engineering on the provided datasets (i.e., both the unlabelled data set and the test data set) is **not allowed**! In other words, using reverse engineering to recover the labels for the reviews in both unlabelled_data.csv and test_data.csv is not allowed.
- Any information about the test data cannot be used in training the classifiers.

3 Data Preparation & Feature Extraction

Selecting relevant features and deciding how to encode them for a classification algorithm is crucial for learning a good model. Free language text cannot be used directly as input to classification algorithms. It must be pre-processed and transformed into a set of features represented in a numerical form. In this section, we will discuss the basic text pre-processing steps and the common features used in text classification.

The basic text pre-processing steps include

- *Case normalization*: Text can contain upper- or lowercase letters. It is a good idea to just allow either uppercase or lowercase.
- *Tokenization* is the process of splitting a stream of text into individual words.
- *Stopwords* are words that are extremely common and carry little lexical content. The list of English stop words can be downloaded from the Internet. For example, a comprehensive stop-word list can be found from Kevin Bouge's website[1].
- *Removing the most/least frequent word*: Besides the stopwords, we usually remove words appearing in more than 95% of the documents and less than 5% of the documents as well. The percentages can be varied for corpus to corpus.

Those are the common steps used in the text preprocessing that you **could** consider while preprocessing the product reviews. Besides those common steps listed above, there is no limitation on the pre-processing steps you can use in the task.

Next, what kind of features one can extract from the free language text for document classification? There are some common features often considered in document classification, which include

- N-gram feature[2]: N-grams are basically a set of co-occurring words within a given window. For example, for the sentence "The cow jumps over the moon", if $N = 2$ (known as bigrams), then the n-grams would be "the cow", "cow jumps", "jumps over", "over the", "the moon". If $N = 3$ (known as trigram), the n-grams would be "the cow jumps", "cow jumps over", "jumps over the", "over the moon".
- Unigram feature: a case of N-grams, if N = 1. Given the above sentence, the unigrams are "The", "cow", "jumps", "over", "the", "moon".
- POS tags[3]: part-of-speech annotation.
- TF-IDF[4] (Term Frequency-Inverse Document Frequency): It is a measure of how important a word/n-gram is to a document in a collection.

You can choose to use either an individual feature or the combination of multiple features. The features listed above are **candidate features** that you **could** consider in the task. However, you can go beyond those features and try to find the set of features that can give you the best possible classification accuracy.

There are many useful online tutorials on text preprocessing in either R or Python, for example,

[1] https://sites.google.com/site/kevinbouge/stopwords-lists
[2] https://www.tidytextmining.com/ngrams.html
[3] martinschweinberger.de/docs/articles/PosTagR.pdf
[4] https://www.tidytextmining.com/tfidf.html

- Feature extraction in Scikit-learn[5]
- Working with text data[6]
- R code: reading, pre-processing and counting text[7]
- "Text Mining with R"[8], a tutorial that discusses how the deal with text in R. It provides compelling examples of real text mining problems

4 Classifier

Now, you should develop a sentiment classifier that can give you the most accurate prediction. Meanwhile, it is necessary to think about how to make good use of the unlabelled data. The algorithm that you can use is **not limited to the algorithms covered in the lectures**. The goal is to find the most accurate classifier.

In order to find the most accurate classifiers, each group should empirically compare **at least** 3 different types of classification methods in the context of sentiment classification, and then submit the one performs the best in your comparison.

5 Evaluation

The evaluation method used in testing is the accuracy score, which is defined as the proportion of correct predictions among all of the predictions.

$$\text{Accuracy} \quad = \quad \frac{\textbf{number of correct predictions}}{\textbf{number of all predictions}}$$

You can use the existing python/R code to compute the Accuracy score, for example

- Accuracy score in Python[9]
- Accuracy score in R[10]

6 Submission

To finish this data analysis challenge, all the groups are required to submit the following files:

- **"predict_label.csv"**, where the label prediction on the testing documents is stored.
  - Please note that the first row must be the column names (i.e., "test_id" and "label").
  - The "predict_label.csv" must be reproducible by the assessor with the submitted R/Python code.
- The **R/Python implementation** of your **final** classifier. The implementation must include
  - The code for text preprocessing and feature extraction (if).
  - The code for training and testing your final classifier.
  - A readme file that tells the assessor how to set up and run your code.

[5] https://scikit-learn.org/stable/modules/feature_extraction.html
[6] https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html
[7] http://www.katrinerk.com/courses/words-in-a-haystack-an-introductory-statistics-course/schedule-words-in-a-haystack/r-code-the-text-mining-package
[8] https://www.tidytextmining.com/index.html
[9] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html
[10] https://www.rdocumentation.org/packages/rfUtilities/versions/2.1-4/topics/accuracy

The output of your implementation must include the label prediction for all the testing documents, i.e., "predict_label.csv". The use of Jupyter notebook or R Markdown is **not required**. Please note that the unnecessary code must be excluded in your final submission. For example, if you tried three different types of models, say multinomial regression, LDA and classification tree, and your group decides to submit LDA as the final model, you should remove the code for the other two models from your submission. The discussion of the comparison must be included in your report instead. *However, you should keep a copy of the implementation used for comparison for the purpose of the interview.*

- **A Turnitin report**, where you should document in details the development the submitted model. **The maximum number of pages allowed is 8**. The report must be in the PDF format, named a **"groupdName_ass2_report.pdf"**. The report should include (but not limited to)
  - The discussion of how the data preprocessing/features selection has been done.
  - The development of the submitted classifier: To choose an optimal classifier for a task, we often carry out empirical comparisons of multiple candidate models with different feature sets. In your report, you should include a comprehensive analysis of how the comparisons are done. For example, the report can include (but not limited to)
    - A description of the classifier(s) considered in your comparison.
    - The detailed experimental settings, which can include the discussion of how the cross-validation is set up, how the parameters for the model considered (if applicable) are chosen, or the setting of semi-supervised learning (if applicable).
    - Classification accuracy with comprehensive discussion.
    - The justification of the final model submitted.

  Warning: If a report exceeds the page limit, the assessment will only be based on the first 8 pages.
- A signed group assignment coversheet. **Warning**: typing name is not counted as a signature in the coversheet.

7 How to submit the files?

The Moodle setup allows you to upload only two files

- "**groupdName_ass2_report.pdf**": A pdf report file, which will be submitted to Turnitin.
- "**groupName_ass2_impl.zip**"': a **zip** file includes
  - the implementation of the final submitted model
  - "predict_label.csv", where the label prediction on the testing documents is stored.
  - the signed grouped assignment coversheet

Please note that

- **Only one group member need to upload the two files. But all the group members have to click the submit button in order to make the final submission.** If anyone member does not click the submit button, the uploaded files will remain as a draft submission. A draft submission won't be marked!
- **The two files must be uploaded separately.**