FIT5149 S1 2020 Assessment 2: Authorship Profiling

May 2020

Marks	35% of all marks for the unit
Due Date	23:55 Sunday 14 June 2020
Extension	An extension could be granted for circumstances. Please refer to the university webpage on special consideration. A special consideration application form must be submitted. Please note that aLL special consideration, including within the semester, is now to be submitted centrally. All students MUST submit an online special consideration form via Monash Connect.
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/marked.
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. 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

Authorship analysis deals with the classification of texts into classes based on the stylistic choices of their authors. Beyond the author identification and author verification tasks where the style of individual authors is examined, author profiling distinguishes between classes of authors studying their sociology aspect, that is, how language is shared by people. This helps in identifying profiling aspects such as gender, age, native language, or personality type. Author profiling is a problem of growing importance in applications in forensics, security, and marketing. E.g., from a forensic linguistics perspective one would like being able to know the linguistic profile of the author of a harassing text message (language used by a certain type of people) and identify certain characteristics (language as evidence). Similarly, from a marketing viewpoint, companies may be interested in knowing, on the basis of the analysis of blogs and online product reviews, the demographics of people that like or dislike their products. The focus is on author profiling in social media since we are mainly interested in everyday language and how it reflects basic social and personality processes.

The authorship profiling task is often formulated as a classification problem, where a classifier is fed with a text and returns the corresponding gender or native language label. There are many machine learning methods that can be used in the classification task. They can be categorised into supervised method (like SVM) and unsupervised method (like clustering). Figure 1 shows a typical framework used in the supervised classification.[1]

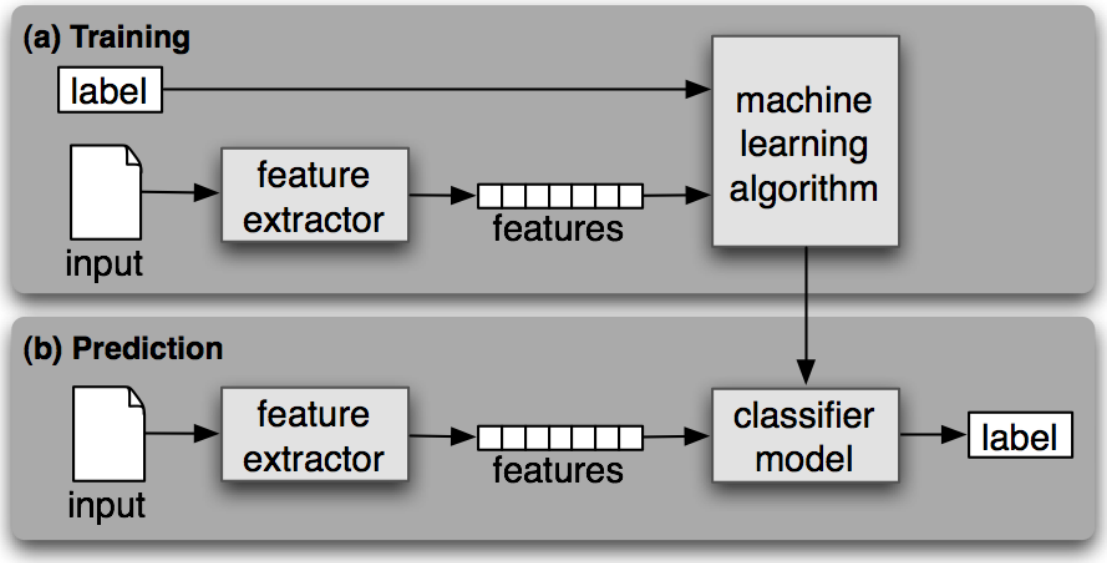

Figure 1: A general framework for the supervised classification.

As shown in the figure, there are 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.

The aim of this challenge is to develop a classifier that can assign a set of twitter texts to their corresponding labels. In this assessment, you will focus on the gender classification task, where you are required to develop a classifier that can identify the gender of the tweet's author as accurate as possible.

2 Dataset

We provide the following data sets (Table 1):

[1] The figure is download from https://www.nltk.org/book/ch06.html

Data Source	Type	classes	training examples	testing examples
Twitter	Gender	2	3100	500

Table 1: Authorship Profiling data set.

- train_labels.csv contains training ids and gender. It contains twitter posts from 3,100 authors and acts as the training data.
- test.csv: only testing ids are available. It contains the twitter posts from 500 authors. The gender is left blank for your model to predict. Please note that the testing label will be released about one week before the assessment due day.
- data.zip contains 3,600 twitter texts for those authors, which acts as the training and testing data.

Warning: Reverse engineering on the provided dataset is not allowed! Any information about the test data cannot be used in training the classifiers.

3 Data Preparation & Feature Extration

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 most common and basic 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[2].
- *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 only the common steps used in pre-processing text. Please note that the steps are of your choice and **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[3]: 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".

[2] https://sites.google.com/site/kevinbouge/stopwords-lists
[3] https://www.tidytextmining.com/ngrams.html

- POS tags[4]: part-of-speech annotation.
- TF-IDF[5] (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,

- Feature extraction in Scikit-learn[6]
- Working with text data[7]
- R code: reading, pre-processing and counting text[8]
- "Text Mining with R"[9], 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 classifier that can give you the most accurate prediction in the gender classification task. The algorithm that you can use are not limited to the algorithms covered in the lectures/tutorials. The goal is to find the most accurate classifier.

In order to find the most accurate classifier, each group should empirically compare at least 3 different types of classification methods in the context of gender classification, and then submit the one perform 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. We will use the accuracy for both gender and language classification in this case.

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

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

- Accuracy score in Python[10]
- Accuracy score in R[11]

6 Submission

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

- **"pred_labels.csv"**, where the label prediction on the testing documents is stored.

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

  - The format of "pred_labels.csv" must be the same as "train_labels.csv", i.e., in your "pred_labels.csv", there must be two columns: the first one is the id column, and the second one is the *gender* column. Remember the first row of your "pred_labels.csv" file should be "id" and "gender".
  - The "pred_labels.csv" must be reproducible by the assessor with your submitted R/Python code.

- The **R/Python implementation** of your **final** classifier. The implementation must include
  - The code for text preprocessing and feature extraction.
  - The code for training and testing your final classifier.
  - A README file that tells the assessor how to set up and run your code.

  The output of your implementation must include the label prediction for all the testing documents. The use of Jupyter notebook or R Markdown is **not required**. All the files that are required for running your implementation must be compressed into a **zip** file, named as "**groupName_ass2_impl.zip**". 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 the submission. **The discussion of the comparison should be included in your report.** *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 of the features and 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 must 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 cover sheet, which will also be included in your zip file.
  Warning: typing name is not counted as a signature in the cover sheet.

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.**