Data Analysis Challenge: Classifying News Articles

- Due date: Sunday 21-October-2018, 23:55
- Assessment type: Group with maximum 3 members
- Weight: 30 out of 100

Text documents are one of the richest information sources where we can learn valuable information for various purposes, like business. As one of the important and typical analytic tasks in NLP/ML, document classification is to automatically label a document with one or more classes or categories. A document can be a scientific paper, a medical report, a news article, an email, a product review, a blog, and so on. It has many applications in, for example, spam filtering, email routing, medical triage, sentiment analysis.

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]

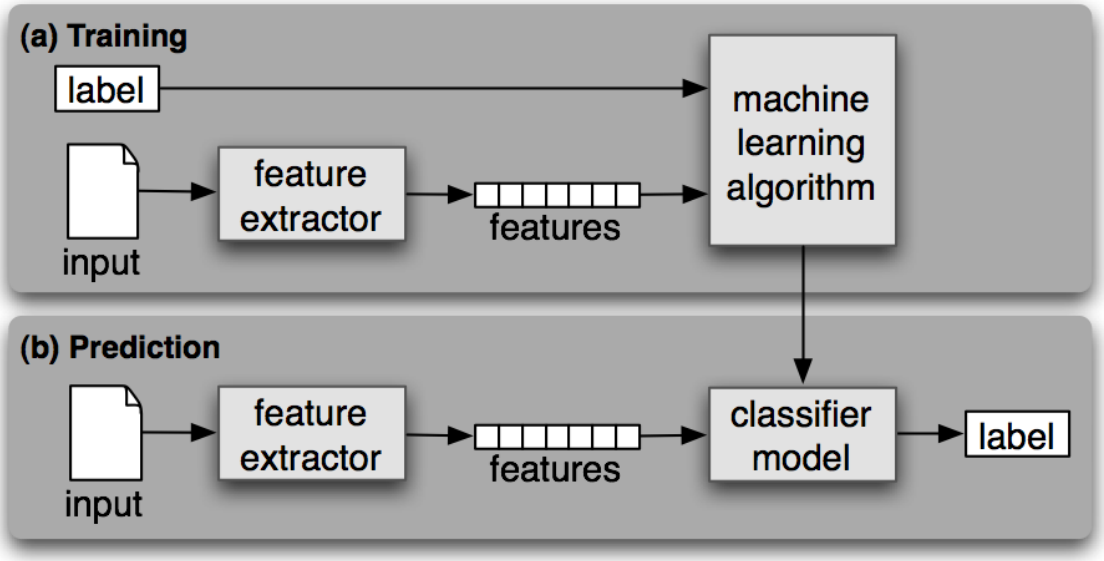

Figure 1: A general framework for the supervised classification.

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 first step is shared by both training and prediction, which tells us that data used in training and prediction should share the same feature space.

The focus of this data analysis challenge is to classify a large set of news articles, the details of which are discussed in Section 2. The challenge is a multi-class classification task, where the number of classes is greater than two, and each news article has only one class label.

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

```
ID tr_doc_1
TEXT Two German tourists have been found safe and well after
spending almost six hours lost in rugged rainforest at Finch Hatton
Gorge, west of Mackay, last night. It is the same area a young
Mackay man fell or jumped to his death last week. Sergeant Jon
Purcell says rescuers located the missing pair just before midnight
AEST.
EOD
```

(a) training data

```
ID te_doc_1
TEXT The Police Royal Commission in Western Australia is hearing
evidence from the first serving officer to testify about corrupt
activities in the service. The officer, code-named L-8, has been in
the service for more than 30 years. He reached the rank of
Inspector. He has testified that his improper behaviour began in
1979 when he and another detective took $200 from an armed robbery
suspect. He has told the commission that he subsequently assaulted
suspects, and made up a statement that was put before a court in an
armed robbery case. L-8 is the eighth officer to 'roll over' since
the Royal Commission started mid last year.
EOD
```

(b) testing data

Figure 2: Training and testing data format.

1 Aim of the challenging

The aim is to find a classifier that can classify news articles into a number of classes with best possible accuracy (see Section 6 for the accuracy measure). Please note that this is a multi-class classification task, where each article has only one label.

2 Dataset description

The dataset we consider here is a large set of news articles that are crawled from a news website. The corpus contains 133,055 new articles in total. 80% of news articles are randomly selected for training. The other 20% are use used for testing, the labels of which are withheld . The data folder provided contains the following files:

1. "training_docs.txt"
2. "testing_docs.txt"
3. "training_labels_final.txt"

Figure 2 shows the format of "training_docs.txt" and "testing_docs.txt". Each new article is stored as a chunk of text, the layout of which is as follows

- "ID": the line starting with "ID" contains an article ID. The ID has "tr" and "te" as the prefix for training and testing articles respectively. We will

```
tr_doc_1 C1
tr_doc_2 C1
tr_doc_3 C1
tr_doc_4 C1
tr_doc_5 C1
tr_doc_6 C1
tr_doc_7 C1
tr_doc_8 C1
tr_doc_9 C1
tr_doc_10 C1
tr_doc_11 C1
tr_doc_12 C1
tr_doc_13 C1
tr_doc_14 C1
tr_doc_15 C1
tr_doc_16 C1
tr_doc_17 C1
tr_doc_18 C1
tr_doc_19 C1
tr_doc_20 C1
```

Figure 3: Labels for the training articles

need the IDs to retrieve the class labels for the training articles from "training_labels_final.txt", and generate the label file for the testing articles.

- "TEXT ": the line beginning with "TEXT" contains the text content of an article.
- "EOD": it signals the end of an article.

All the chunks are separated by an empty line in the two txt files.

The file "training labels final.txt" contains the labels for all the news articles used in training. There are 23 classes in total, encoded as "Cn" where $n \in [1, 23]$. Those labels are also used for classifying the testing articles. Figure 3 shows the format used to store the labels for all the training articles. The training article ID and the corresponding class label are separated by a white space.

3 Data preparation

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.

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.

Besides those common steps listed above, **there is no limitation on the preprocessing 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".
- 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 the basic features 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.**

4 Classifier

Now, you should develop a classifier that can give you the most accurate prediction. 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.

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

	GoldLabel_A	GoldLabel_B	GoldLabel_C	
Predicted_A	**30**	20	10	TotalPredicted_A=60
Predicted_B	50	**60**	10	TotalPredicted_B=120
Predicted_C	20	20	**80**	TotalPredicted_C=120
	TotalGoldLabel_A=100	TotalGoldLabel_B=100	TotalGoldLabel_C=100	

Figure 4: An example confusion matrix for 3 labels: A, B and C. The numbers in the diagonal are the number of documents labeled with the right classses.

5 Prediction

The class labels for the testing articles are not provided. After you have trained your classifier, you should apply the classifier to the testing articles in "testing docs.txt". Your task is to predict the class label for each testing article, and save your prediction in a txt file (named as **"testing_labels_pred.txt"**). It should be in the same format as "training_labels_final.txt", where each row contains the test article ID and its class label, separated by a white space.

The class labels are exactly the same as those used in training, which means there are 23 class labels in testing as well, each of which is encoded as Cn for $n \in [1, 23]$.

6 Evaluation

The evaluation method used in testing is the macro-F1 score, which is defined as the mean of F1 scores for all the class labels.

Firstly, we should compute the per-class precision, recall and F1 scores. Figure 4 shows an example confusion matrix for 3 class labels.[6] Let's take Label A as an example.

- Recall of Label A

$$\begin{aligned} R_A &= \frac{TP_A}{TP_A + FN_A} = \frac{TP_A}{\text{Total Gold for A}} \\ &= \frac{TP_A}{\text{TotalGoldLabel\_A}} \\ &= \frac{30}{100} = 0.3 \end{aligned}$$

- Precision for Label A

$$\begin{aligned} P_A &= \frac{TP_A}{TP_A + FP_A} = \frac{TP_A}{\text{Total predicted as A}} \\ &= \frac{TP_A}{\text{TotalPredicted\_A}} \\ &= \frac{30}{60} = 0.5 \end{aligned}$$

[6] http://text-analytics101.rxnlp.com/2014/10/computing-precision-and-recall-for.html

- F1 for Label A

$$\begin{aligned} F1_A &= 2 * \frac{R_A * P_A}{R_A + P_A} = 2 * \frac{0.3 \times 0.5}{0.3 + 0.5} \\ &= 0.375 \end{aligned}$$

The macro F1 score for the three class labels is defined as

$$\begin{aligned} \text{marco-F1} &= \text{mean}(F1) \\ &= \frac{1}{3} \times (F1_A + F1_B + F1_C) \end{aligned}$$

The macro-F1 score will be computed based on the prediction on the testing articles given by the submitted classifier.

7 Files to be submitted

To finish the challenging, all the groups are required to submit the following files

1. **"testing_labels_pred.txt"**, where the label prediction on the testing articles is stored.
   - The format of "testing_labels_pred.txt" must be the same as "training_labels_final.txt".
   - The "testing_labels_pred.txt" must be reproducible by the assessor with the submitted R code.
2. **A Turnitin report**, which documents the development of the features and the submitted model. **The maximum number of words allowed is 1500**. The report must be in the PDF format, named as **"groupdName_ass2group.pdf"**. The report includes (but not limited to)
   - List of R libraries used
   - The preprocessing steps
   - The selected features and how they generated
   - The methodology used to develop the model/algorithm
   - A description of the model/algorithm used for learning
   - The signed cover sheet (A sample provided)
3. **The R code** for your classifier
   - Its input must be feature vectors generated for the training and testing articles.
   - The R code must run without any error.
   - The output of the R code must include the prediction file, i.e., "testing_labels_pred.txt".

4. **Feature file(s)** storing the feature vectors used by the R code.

    - There is no need to submit the code for preprocessing the text and generating features, which means you can use whichever programming language you prefer to preprocess the articles and generate features.
    - The generated feature vectors must be included in the submission in order to make sure the R code can run.

How to submit the files? The Moodle setup allows you to upload two files

- "groupdName_ass2group.pdf": A pdf file that is generated for the report. Please exclude the cover sheet and all the bullet points given as examples above.
- "groupName_ass2group.zip": All the the others will be submitted as one ZIP file.
- **Only one group member need to upload the two files.**
- 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!

More information about how to submit your assessment can be found in Moodle.

8 Useful resources

There are a lot of useful open-resources online, that can help you finish the assessment. For example,

- tm (R library)
- ngram (R library)
- text mining and n-grams [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
- "Compute classification evaluation metrics in R"[10]

[7] https://rpubs.com/brianzive/textmining

[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] http://blog.revolutionanalytics.com/2016/03/com_class_eval_metrics_r.html