Assignment 3: Text Classification

Niranjan Balsubramanian, Tianyi Zhao, and Tao Sun

CSE 352 Spring 2021

1 Due Date and Collaboration

- The assignment is due on **April 9 at 11:59 pm**. You have a total of four extra days for late submission across all the assignments. You can use all four days for a single assignment, one day for each assignment – whatever works best for you. Submissions between fourth and fifth day will be docked 20%. Submissions between fifth and sixth day will be docked 40%. Submissions after sixth day won't be evaluated.
- You can collaborate to discuss ideas to understand the concepts and math.
- You should NOT collaborate at the code or pseudo-code level. This includes all implementation activities: design, coding, and debugging.
- You should NOT not use any code that you did not write to complete the assignment.
- The homework will be cross-checked. **Do not cheat at all! It's worth doing the homework partially instead of cheating and copying your code and get 0 for the whole homework.**

2 Goals

Naive Bayes, Logistic Regression, Support Vector Machines, and Random Forests

This is an analysis assignment, where you will investigate the utility of different learning algorithms for a text classification task. You will be using the implementations available in the scikit python library here:
`http://scikit-learn.org/stable/supervised_learning.html#supervised-learning`

The task is to learn classifiers that can classify input texts into one of the K-classes. You will create classifiers with three configurations:

1. Unigram Baseline (UB) – Basic sentence segmentation and tokenization. Use all words.

2. Bigram Baseline (BB) – Use all bigrams. (e.g. I ran a race => I ran, ran a, a race.)

3. My best configuration (MBC) – You can create your own configuration based on the design choices below.

3 Dataset

For this assignment we're using the "20 newsgroup dataset" which contains 20,000 newsgroup documents partitioned into 20 news categories. We treat each of the categories as classes. We will use only 4 classes for this assignment.

1. rec.sport.hockey

2. sci.med

3. soc.religion.christian

4. talk.religion.misc

In each class, there two sets of documents, one for training and one for test. The format of each document is as follows:

Header	Consists of fields such as <From>, <Subject>, <Organization>and <Lines> fields. The <lines>field includes the number of lines in the document body.
Body	The main body of the document. This is where you should extract features from.

Note that you can ignore the header when you're extracting features.

4 Design Choices for your best configuration

4.1 Feature representations

Scikit provides few implementations of feature extractors. The extractors first segment the text into sentences, and tokenize them into words. Each document is then represented as a vector based on the words that occur in it.

1. CountVectorizer – Uses the number of times each word was observed.

2. TFIDFVectorizer – Uses relative frequencies normalized by the inverse of the number of documents in which the word was observed.

See Section 4.2.3 in `http://scikit-learn.org/stable/modules/feature_extraction.html` for details. You can also explore many different choices for getting a good representation. Preprocessing often has a big impact in text tasks. Some choices include:

a. Lower case and filter out stopwords (e.g., I ran a race => I, ran, race).

b. Apply stemming (e.g, running, runs, ran => run). You can find a stemmer in nltk. Use porter stemmer shown here: `http://www.nltk.org/howto/stem.html`

4.2 Feature selection

This is often key to the performance of any ML based application. Depending on the type of algorithm used, you will have different choices for doing feature selection. For instance, you can do an external feature selection where you find the most informative features or you can try L1, L2 or Lasso regularizers. You should read this blog and pick any two regularizations. `http://scikit-learn.org/stable/modules/feature_selection.html`

4.3 Hyperparameters

For most learning algorithms there are many different options that can be tuned. Finding the best set of options can be tricky and require many rounds of exploration. Try the following:

1. Naive Bayes – No hyperparameters.
2. Logistic Regression – Regularization constant, num iterations
3. SVM – Regularization constant, Linear, polynomial or RBF kernels.
4. RandomForest – Number of trees and number of features to consider.

5 Implementation

5.1 Requirements

1. You must implement in **Python 3.7+**. You can use scikit library for the algorithms.
2. You CAN discuss with your friends about which algorithms to use and to understand the algorithms but not share or discuss code.
3. All code will be tested by running through plagiarism detection software.
4. You should include cite any web resources or friends you used/discussed with for pseudo-code or the algorithm itself. Failure to do so will result in an F.
5. Your algorithm should be documented at a method level briefly so they can be read and understood by the TAs.

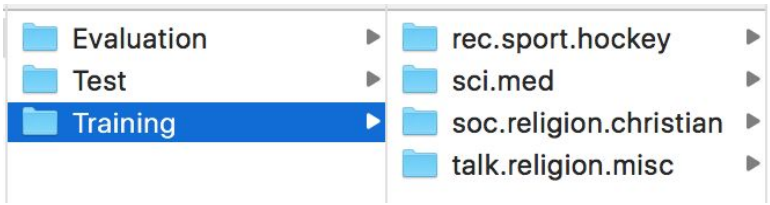
6 What should you turn in?

- Basic Comparison with Baselines (50 points) – For **all** four methods (NB, LR, SVM, and RF), you should run **both** unigram and bigram baselines. You should turn in two results:

  a **UB_BB.py** (20 points) should run all 4 methods, each with 2 configurations.

  python UB_BB.py <trainset> <evalset> <output> <display_LC>

  where <trainset> is the parent folder to your training data and <evalset> is the parent folder to your evaluation data. For the example shown in the figure below, <trainset> should be 'Training', and <evalset> should be 'Evaluation'.



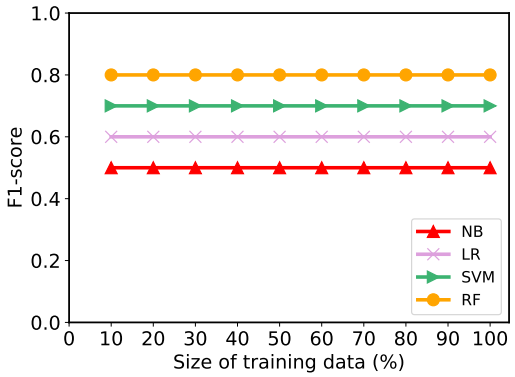
  <display_LC> is an option to display the learning curve (part b). '1' to show the plot, '0' to NOT show the plot. <output> is the path to a **comma-separated values file**, which contains 8 lines corresponding to 8 runs. Sample output file as below. Note that all elements are UPPER-CASE letters.

```
NB,UB,0.67,0.67,0.67
NB,BB,0.69,0.69,0.69
LR,UB,0.71,0.71,0.71
LR,BB,0.67,0.67,0.67
SVM,UB,0.69,0.69,0.69
SVM,BB,0.71,0.71,0.71
RF,UB,0.67,0.67,0.67
RF,BB,0.69,0.69,0.69
```

  Each line should be '<Classification Method>,<Configuration>,<Macro-Precision>,<Macro-Recall>,<F1-score>', evaluated on the **evaluation data**, not **training data**. **Include the content of your output file into your report.**

  Refer to `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html` for the evaluation metrics.

b (20 points) A learning curve (LC) result, where you show the performance of each classifier only with **the unigram representation**. The learning curve is a plot of the performance of the classifier (F1-score on the y-axis) on the **evaluation data**, when trained on different amounts of training data (size of training data on the x-axis). You can choose different training size for this figure. For example, you can randomly sample 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 100% training data. Note the output file only contains the results when using 100% training data. Plot all four learning curves in **a single figure**, and **include the figure into the report**. Use proper line styles and legends to clearly distinguish four methods. An example plot is shown below. Note the values are not real.



c (10 points) Describe your findings and make arguments to explain why this is the case. You can use any online source to understand the relative performance of algorithms for varying training data sizes. Make arguments in your own words. Provide a citation.

- My best configuration (50 points) – Pick the best performing classification method from the above experiments and then use the design choices to find the best possible result on the **evaluation data**. You will create a model that we can run on a hidden **test data**.

  a **MBC_exploration.py** (20 points) For **each** of four methods (NB, LR, SVM, and RF), pick **at least two** design choices and output evaluation results to a file.

  python MBC_exploration.py <trainset> <evalset> <output>

  The output file should contain 4 lines. Each line should be '<Classification Method>, <Configuration>, <Macro-Precision>, <Macro-Recall>, <F1-score>', evaluated on the **evaluation data**, not **training data**.

You can name <Configuration> to be anything you want, but <Classification Method> must be one of {NB, LR, SVM, RF}. **Include the content of your output file into your report.**

b **MBC_final.py** (20 points) – Export the best configuration training model (you can use any combinations) and give us code that we can run to get performance of your model on a hidden **test set**. You can call helper functions from MBC_exploration.py if you want. The organization of the test data we use is the same as the evaluation data you are given.

python MBC_final.py <trainset> <testset> <output>

The output file should contain **a single line** evaluated on the **test data** in the same format as previous ones. The grading for this part is decided by competing all results among students. It means that the threshold for grading this question is based on results of the whole class.

c Explanation (10 points) – Explain your result based on your best understanding of the configuration options. Should be no longer than 2 paragraphs. You can use any online source to understand the options and what they mean. Provide a citation of the sources.

- Report (PDF) – Include all results/observations above into your report, named SBUID-LastName-FirstName-A3.pdf.

Please put all 4 files (Report, UB_BB.py, MBC_exploration.py, and MBC_final.py) in the same folder. Your submission should be one zip file named SBUID-LastName-FirstName-A3.zip, containing all source code and report under a single folder named SBUID-LastName-FirstName-A3, with no other folders in the zip file. Please note that the file name for the source codes must be exactly the same as instructed.

7 Grading

See last section.