WESTERN SYDNEY
UNIVERSITY

Assignment

Applications of Big Data (COMP3002)

Due: 11 June 2023 AEDT 23:59

Centre for Research in Mathematics And Data Science
School of Computer, Data and Mathematical Sciences

Description

For this assignment, you will need to create a complete program to perform sentiment classification for movie reviews from feature extraction to classification. For a given review, your program should be able to predict whether it is positive i.e. like the movie, or negative, i.e. dislike the movie.

About the data

You will use a large movie review dataset containing a set of 25,000 movie reviews for training, and 25,000 for testing. You can download the data from the vUWS under the assignments folder named aclImdb.zip. You can also visit the following website for more information about the dataset at http://ai.stanford.edu/~amaas/data/sentiment/ or download data directly from there.

Unzip the data to your local directory. Enter the aclImdb/ directory created by the zip file (~500MB), you will find the following three items among others

- train/: feature files and raw text files for the training set
- test/: feature files and raw text files for the testing set
- README: the readme file for more information on the dataset

Read the README file carefully about the descriptions on the text files that contain the reviews and their naming convention. The directories we concern here are

- ./aclImdb/train/pos: raw text files of positive reviews in the training set
- ./aclImdb/train/neg: raw text files of negative reviews in the training set
- ./aclImdb/test/pos: raw text files of positive reviews in the test set
- ./aclImdb/test/neg: raw text files of negative reviews in the test set

A full version of this data set is available at hdfs://hadoop.cdms.westernsydney.edu.au:9000/users/bigdata/imdb/fullversion. A tiny cut down version with much less number of files (20 each for training and 10 each for test) is also available at hdfs://hadoop.cdms.westernsydney.edu.au:9000/users/bigdata/imdb/tinyversion. The tiny version is for experimenting purpose.

Task 1. Feature extraction (15 points)

Use the map reduce model to convert all text data into matrices. Convert *ratings* to vectors. These will be used for classification in Task 2. Use TF-IDF to vectorise the text files. See previous practical classes and lectures materials for TF-IDF. One step further though is to represent each text file (review) as a very long and sparse vector as the following. Assume `wordslist` is the final list of distinct words contained in all reviews and its length is D. Then each review will be a vector of length D, with each position associated with a word in `wordlist` and the value being either 0, if the corresponding word is absent in the review, or the word's TF-IDF. For example, if `wordlist = ['word1', 'word2', 'word3', 'word4']` and review 1 contains `word1` and `word4`, then the vector representation of review 1 is [0.1, 0, 0, 0.4] assuming TF-IDF of `word1` and `word4` in review 1 is 0.1 and 0.4 respectively. Note that TF is calculated from one single document while IDF is obtained from all documents in the collection.

Requirements:

1. Map reduce model is a must. Implement it using Hadoop streaming. All data are available on SCDMS HDFS. The recommendation is to work on the tiny version of the data to make the code work. You may try your code on the full version. However, the application to full version is not required.
2. Generate two matrices: `training_data`, `test_data`, and two vectors, `training_targets`, `test_targets`. `training_data` should have N rows and D columns with each row corresponding to each review in the training set, where N is the totally number of reviews in training set and D is the total number of words. N and D vary depending on which version of the data you use. `training_targets`

should have N elements each of which is the rating of the review is for. `test_data` and `test_targets` are similar defined.

Note:

- Ratings scores extraction can be purely python.
- Using map reduce model to extract TF-IDF is mandatory. If not used, a **50% penalty** for this task will incur. There is no constraint on how to form the training and test matrices and vectors. There are many versions of TF-IDF. There is no preference for which version to use.
- You can use data frame (using `pandas` package) instead of matrices and vectors to store training and test data and targets.

Marking scheme for task 1:

- Rating scores extraction (3pts): parse the name of text files to extract ratings.
- TF-IDF extraction (10pts): use map reduce model to extract TF-IDF for each text file.
- Forming matrices and target vectors (or data frames) (2pts): collect TF-IDFs to form training and test data for task 2.

Task 2. Classification (15 points)

Construct a classification model for review sentiment prediction, meaning that given a customer review (taken from test set) about a movie, your program should be able to predict whether it is positive or negative. There is no limitation on how many classifiers and what specific model you should use. You can simply pick one that works for you for this task, either from those covered in lectures and practical classs or any other classifiers from any python packages. A good starting point is the `scikit-learn` (i.e. `sklearn`) package. A few things you need to address in your python program are listed as requirements below.

Requirements:

1. Data pre-processing. In task 1, you have extracted the ratings vectors for training and test. They are raw ratings. As we are interested in sentiment prediction, i.e. to predict either the review is positive or negative. You need to convert all ratings>5 as positive class and ratings<=5 as negative class. Choose a coding scheme, e.g. 1 for positive, 0 for negative.
2. Normalisation. Apply at least one normalisation scheme and compare the performance of the classifier(s) with and without normalisation.
3. Training and model selection. Use cross validation to select the best parameters for your classifier. There may be many parameters to tune in some classifiers such as random forest classifier (RFC). You can focus on the most important one(s) such as `max_depth` and `n_estimators` in RFC. Refer to `scikit-learn` package documentation for details. Hint: you can start with a small subset of training set to test a few parameters to get a feel of what range the parameters should be that make the model perform well in terms of prediction accuracy. Then turn on large scale cross validation on the whole training set.
4. Test on test data. After model selection, apply the best model, i.e. model with the parameters that produces the best cross validation scores, to test data and make prediction for each review and record prediction accuracy (ACC).

Note:

1. Always train your classifier(s) ONLY on training data including cross validation. After model selection, apply the best model on test data to evaluate the performance.
2. Good performance, i.e. higher ACC on test data, is not essential for this task. However, if your classifier has ACC low than 60%, it usually means that there are some mistakes somewhere in your code. So try to score as high ACC as possible.
3. You are encouraged to try many classifiers. If the coding is right, this should not be too difficult. Remember model selection when you try different classifiers!

Marking scheme for task 2:

- Data pre-processing (1pts): convert ratings to positive and negative coding scheme.
- Normalisation and comparison (3pts): apply normalisation and compare performance difference with and without it.
- Training on training data (3pts): training performed on training data.
- Cross validation (6pts): apply cross validation on training data.
- Testing on test data (2pts): best model applied to test data and ACC produced.

Bonus Task (10 points)

This is a bonus task. It is not essential but if you could complete it as required you will receive 10 extra points towards your final results of this unit. The task is similar to the *Task 5* in prac 8.

Compute the correlation between features and response. Use the TF-IDF as features and review scores as response. Consider only training set, i.e. on `training_data`. Here is the details. Let $\mathbf{x}_i$ be the i-th TF-IDF *column* vector you extracted for the i-th review, and y_i its corresponding review score. To compute the coorelation, we need $\tilde{\mathbf{x}}_i$ and $\tilde{y}_i$, normalised version of $\mathbf{x}_i$ and y_i as the following.

$$\tilde{\mathbf{x}}_i = \frac{\hat{\mathbf{x}}_i}{\|\hat{\mathbf{x}}_i\|}$$

where $\hat{\mathbf{x}}_i = \mathbf{x}_i - \mathbf{m}$, $\mathbf{m}$ is the mean of all features, i.e. $\mathbf{m} = \frac{\sum_{i=1}^{N} \mathbf{x}_i}{N}$, and $\|\hat{\mathbf{x}}_i\|$ is the so-called ℓ_2 norm of $\|\hat{\mathbf{x}}_i\|$ which is defined as

$$\|\hat{\mathbf{x}}_i\| = \sqrt{\sum_{j=1}^{D} x_{i_j}^2}$$

i.e. the square root of the sum of squares of all the elements in vector $\hat{\mathbf{x}}_i$. $\tilde{y}_i$ is similar

$$\tilde{y}_i = \frac{y_i}{\|\mathbf{y}\|}$$

where $\mathbf{y} = [y_1, \dots, y_N]$ is the vector of all review scores. Then the correlation $\mathbf{r}$ is

$$\mathbf{r} = \sum_{i=1}^{N} \tilde{y}_i \tilde{\mathbf{x}}_i.$$

$\mathbf{r}$ will be a vector of length D.

Requirements:

1. Use map reduce computing model for this task is mandatory. Direct computing the correlation from the matrices obtained from Task 1, i.e. `training_data` is *not* acceptable.
2. Python code and Hadoop streaming commands must be supplied for this taks. If multiple map reduce steps are used, a step-by-step guidance must be provided as well.

Hint: you may consider several map reduce to compute mean, ℓ_2 norm, multiplication and etc.

Overall Marking Criteria for All Tasks

Your program will be marked against both functional and operational requirements. Functional requirements accounts for 80% of the mark, which measure how well your program achieves the expected functionalities and are further broken down into the items listed in marking schemes in those tasks.

In addition to function requirements, your program should also meet the operational and style requirements, which can be broken down into the following.

- Readability (5%): Comments should be included in your program to explain the main idea of your design; use meaningful variable and function names; do not declare variables that are not used in the program
- Modularity (10%): Your program should make use of functions or classes wherever possible to achieve modular design and maximise reusability.
- Useability (5%): Your program should be easy to use by the user. These include displaying messages for user interaction, performing adequate input validation, and allowing the users to choose the locations of the data file for model training.

Submission

Your python code solution including Hadoop streaming commands if any, documentation such as how to use the functions must be written in **jupyter notebook** with clear indication which is for which. A jupyter notebook version of this document called `comp3002_assignment.ipynb` is provided. Rename it to `comp3002_assignment_yourstudentid.ipynb` and work on it. Add code and markdown blocks as you like. Submit your complete notebook through vUWS before deadline.