Project Description (COMP 7745/8745)

Due Date:April 22, 2022
Points = 100 (extra credit 10)

1 Summary

In this project, you will apply Machine Learning methods to classify fake reviews from real reviews. The dataset is a real-world dataset taken from [2]. The authors used real reviews from tripadvisor and fake reviews written by paid Amazon Mechanical Turk Workers. It is a standard binary classification problem. The training, validation and test datasets of reviews are provided. Each review is in a single line and the corresponding label is provided in a different file. There are 6 files, train.txt, test.txt and validation.txt that contains the reviews and trainlabels.txt, testlabels.txt and validationlabels.txt that contain the labels (1 denotes a fake review and 0 a real review). You will apply the full ML pipeline, i.e., feature extraction, fine-tuning ML algorithms and evaluation for this dataset.

- Feature Extraction: To perform feature extraction, you will use a state-of-the-art language representation model called Word2Vec [1] that is based on neural networks. Gensim is an open-source software package for this model that allows us to easily convert any document to real-valued vectors. Internally it uses a neural network architecture to do this. Note: In real-life you will use *pre-trained* models that have been trained on millions of examples to generate the vectors. But since this is computationally expensive you may not be able to load such large models on basic laptops and therefore you don't need to do this. But try to explore these for your own interest as shown in `https://radimrehurek.com/gensim/auto_examples/howtos/run_downloader_api.html`.

- ML Algorithms: Use 3 algorithms in your implementation, Neural Networks, Logistic regression and Random Forests. You can use the standard sklearn libraries for this. Use the validation dataset to tune the parameters to obtain as best performance as possible. That is, train on the training dataset and maximize the F1-score on the validation dataset by experimenting with different settings in the algorithms. E.g. for Neural-Nets vary the hidden-layer units, for Logistic Regression change the regularization and for Random Forests change the number of decision trees used and the depth of these trees.

- Evaluation: Evaluate performance on the test dataset for the optimal model parameters. Compute the precision, recall and F1-score.
- Analysis: Briefly describe your results. Can you think of any ideas to improve performance. You don't need to implement them but if you do implement at least one idea, you will get extra credit.

2 Implementation Steps

- Install Gensim from `https://radimrehurek.com/gensim/index.html`.
- Convert the training data to vectors using Gensim. Each line in the training data corresponds to a single review and the corresponding labels for the reviews (1 for fake and 0 for real) is in a separate file. To convert to vectors, first read the training data as tagged documents (code snippet provided in notebook). Then, use *gensim.models.doc2vec.Doc2Vec* to create vectors from the texts. This is a 2-step process. First you need to create a vocabulary, then you need to train the model on the reviews. An example is shown in `https://radimrehurek.com/gensim/auto_examples/tutorials/run_doc2vec_lee.html` under the heading "Training the Model".
- We will now train classifiers based on the vectors obtained in the previous step. To obtain the vector, use model.infer_vector(train_corpus[id].words) where id ranges from 0 to the number of reviews in the training data. An example is in `https://radimrehurek.com/gensim/auto_examples/tutorials/run_doc2vec_lee.html`. Stack the vectors together using numpy.vstack. Thus, we have created a matrix X that holds our feature vectors for the full training data. Create an array Y to hold the labels in "trainlabels.txt". We are now ready to start training the models.
- Train Random Forest, Neural Net and Logistic Regression Classifiers. Tune the parameters of the model using the validation data (try to experiment with different settings to maximize the F1-score). To do this, create the feature vectors using the previous 2 steps for the "validation.txt" file. Important Note: When you are creating the vectors for validation/test data do not re-train the Gensim model. Simply use model.infer_vector() to infer the vectors from the trained model.
- Finally, freeze the classifier parameters and evaluate the performance on "test.txt". Use precision, recall and f1-score to evaluate the performance (sklearn.metrics). Which classifier gave you the best performance? Did any classifier overfit the data (i.e., performance on training data was excellent but poor on test data).
- Can you think of any way to improve the performance of the classifiers for this task (no need to implement it)? If you implement any idea, mention this for extra credit.

3 Submission

Submit the code (Jupyter Notebook or other files). You can perform all your analysis within the notebook. If not, you can submit a separate text file/word document/pdf with a brief analysis of your experiment results.

4 Misc

You can do this project in groups of two. Just mention both your names in a single submission. Please do not plagiarize code from others and always give credit to other sources if you used them.

References

[1] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[2] Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T. Hancock. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 309–319, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.