Mathieu-Joseph Magri 260 928 498
COMP 550 September 27th, 2024

Report – Programming Assignment 1

In this first programming assignment, we needed to evaluate a variety of linear classifier models and preprocessing techniques on textual data. This was done in order to build models that were capable to distinguish real facts about cities from fake facts about cities. The environment/setup to accomplish this task was done in Python by leveraging the capabilities of Numpy, Pandas, Sklearn and NLTK. Briefly put, the goal of this assignment was to determine whether or not linear classifiers had the capabilities of distinguishing a fake fact from a real one, and they can.

To generate our data, ChatGPT 3.5 was used. The prompts used were the following: "Hello, give me 200 real facts about Montreal. Don't stop until you've given me all 200 facts." and "Hello, give me 200 fake facts about Montreal. Don't stop until you've given me all 200 fake facts." After entering these prompts, ChatGPT 3.5 generated all 200 true facts about Montréal as well as 200 fake facts. The 200 true facts were stored in "facts.txt" file and the fake facts were stored in "fakes.txt" with UTF-8 encoding as specified by the assignment. I did not share my data nor take more data from Ed. Furthermore, all true facts and fake facts were about Montréal and no other city. After preparing the data, mixing the datasets together in a balanced manner, and making a pandas dataframe to be used for testing, the total dataset was separated in a simple 80/20 split between the training set and the test set. The test and training sets were balanced with both facts and fake facts. No validation set was created for this first part of experimentation as no hyperparameter tuning was done here, this will come later under the form of 5-fold cross validation. For these first sets of experiments, we wanted to determine which vectorizer between countVectorizer() and TfidfVectorizer() would make the models perform better. The models used were BernoulliNB(), LinearSVC(), and LogisticRegression(). With countVectorizer(), the models had the following accuracy results respectively: 0.8750, 0.8500, 0.8325. With TfidfVectorizer(), they had the following accuracy results respectively: 0.9325, 0.9125, 0.9325. Following these experiments, it was deemed reasonable to continue with TfidfVectorizer() over countVectorizer() given the increase in accuracy for all models. This can be due to the fact that TfidfVectorizer() not only counts the amount of times a word is present in a corpus but also gives it an added importance if it seems more relevant than others.

Following that first experiment, we tested the effects of lemmatization and stemming over the dataset. In both cases, accuracy results dropped by 0.02 to 0.05 points, and therefore determined that it would not be wise to continue with either pre-processing technique. This can be due to the fact that we do not have much data, therefore trying to modify each word can actual represent a hinderance rather than a benefit given the lack of data and therefore diversity present in the dataset. We then decided to test removing stop words from the dataset. The BernoulliNB() drastically decreased in performance by going down a total of 0.10 points while the other models remained the same. It was therefore determined that we would not be removing stop words as this actually hurt our accuracies. The reasoning behind this may also be caused by the lack of data we have, therefore removing words, even if they are stop words, hurts us. We then tested using n-grams. At first, unigrams and bigrams were used and then we tested unigrams, bigrams and trigrams together. When considering unigrams and bigrams, accuracy increased by about

0.02 points for every single model, while when adding trigrams, the models actually did not increase in accuracy and the Bernoulli model actually decreased by a great amount (0.05 points).

The last set of tests done was hyperparameter tuning on the different models using GridSearchCV which also implements k-fold cross validation where, in our case, k was 5. The LinearSVM was trained with C values of 0.1, 1, 10, 100, the hinge and squared hinged loss functions, and the l1 and l2 classifier penalties. The best combination of parameters was C = 1, using hinge and using l2 which led us to an accuracy of 0.9375. For Logistic Regression, we tested the same C values as for LinearSVM but with penalties l1, l2 and elastinet. The best combinaison was found to be C = 100 and the penalty being l2 with an accuracy of 0.9375. For the BernoulliNB model, the alpha values we tested were 0.1, 1, 10, 100 and the fit_prior option was set to both true and false. The best combination was alpha being 0.1 and the fit_prior being set to True which gave us an accuracy of 0.96. All models increased in accuracy after fine-tuning their hyperparameters. The BernoulliNB model seems to perform the best after fine-tuning which seems to make sense given that predicting whether or not something is a fact or fake fact is a binary output, something with which Bernoulli models should be good at handling.

As a final experiment, we combined all best performing pre-processing techniques and hyperparameters together with the Bernoulli model, giving that is was the best performing one. With TfidfVectorizer(), an n-gram range of 1 to 2, alpha set to 0.1 and the fit prior set to true, we get an accuracy of 0.9625, a minimal increase.

As for the limitations of this study, its biggest limit is the fact that our dataset is too small. To be able to detect whether something is a fact or not in a more accurate fashion, we would need to have tons more data, and data about multiple cities as well which was not the case here. Furthermore, fine-tuning the hyperparameters would need to go hand in hand with preprocessing techniques to be able to get, not only the best possible parameters for a given model, but also the best combination of hyperparameters and preprocessing techniques used together which could have been done in much more depth here. As for the generalizability of this study, I do not believe it to be very generalizable given that our dataset is extremely small and facts about Montréal were the only facts used. Indeed, to create something that would be able to be generalized, we would need tons more data on many more cities. Additionally, our data comes from a generative AI model, and we've assumed that everything it has generated for us is correct, when in reality, the model may have supplied inaccurate facts, which can lead to training a model with bad data.

NOTE: When wanting to re-test models, it is important to reset the environment so as to not continuously train over the training data which would cause our model to overfit as well as minimizing data leakage from training sets to test sets.