Classification of Textual Data

Nikita Letov
McGill ID 260876580
nikita.letov@mail.mcgill.ca

Negin Moslemi
McGill ID 260939867
negin.moslemi@mail.mcgill.ca

Younggue Kim
McGill ID 260531176
younggue.kim@mail.mcgill.ca

McGill University
845 Sherbrooke Street West, Montreal
QC H3A 0G4, Canada

Abstract

The goal of this work is to analyze different machine learning methods, implement and test them on two popular textual datasets. Moreover, hyperparameters for these machine learning models were tuned to provide a higher accuracy. This paper serves as a final report for COMP 551 (Applied Machine Learning), 2020 taught by Professor Reihaneh Rabbany and Professor Mohsen Ravanbakhsh at McGill University [6].

1. Introduction

In general, text classification is the task of assigning an input text one label from a fixed set of categories. The goal of textual data recognition is recognizing different categories of topics in passages of text. Humans perform well in such task as we learn object classes since our birth by analyzing the world around us and learn our native language for describing the world and objects in it. Yet, categorizing a significantly large set of textual data can be exhaustive and time-consuming for a human being. This immediately suggests investigating the possibility of automation of the text categorization process. However, computers lack the ability to learn by default and thus have to be trained to do so.

The problem of textual data classification that is proposed to be solved in this work can be defined as developing a natural language processing (NLP) algorithm for estimating whether each analyzed text belongs to some particular category. There are two main challenges in answering this question. The first one embeds enabling the algorithm to ignore irrelevant factors of a text (*e.g.* not relevant words, grammatical mistakes, etc.). The second challenge is making the algorithm ignore insufficient differences between objects (*e.g.* it must be able to tell that two similar texts actually belong to some particular class, ignoring the fact that there is no so many completely identical texts in the world).

The work is organized as follows. Section 2 covers the theoretical background of the models used in this work. Section 3 describes the methods used for data pre-processing and the process of learning for each model. Section 4 concludes this work with comparison of the models used in this work. Tables A.4-A.13 show hyperparameters tested with cross-validation for the 20 newsgroups dataset.

2. Models structure

The models proposed for this work significantly differ from each other and of different nature: some of them are probabilistic while others are frequentist. The models are logistic regression, decision trees, support vector machines (SVM), Ada boost, Random forest. It was also decided to independently analyze the multi-

nominal naive Bayes classifier as it is easy to implement and fast to train. All these models are adapted from the Scikit-learn library for Python [5].

Most of the model descriptions are adapted from the lecture notes of the Applied Machine Learning course at McGill University and are described in the following subsections [6].

2.1. Logistic regression

Logistic regression is a linear classifier which is a modification of linear regression obtained by squashing the decision boundary $w^\top x$ with the logistic function, i.e.

$$f_w(x) = \sigma(w^\top x) = \frac{1}{1 + e^{-w^\top x}}, \tag{1}$$

where f_w is the function that defines the model, σ is the logistic function, x is the vector of input features, w is the vector of model parameters. The decision boundary remains linear. Logistic regression utilizes the cross-entropy loss function defined as follows

$$L_{CE}(y, w^\top x) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y}), \tag{2}$$

where y is the vector of target values and $\hat{y} = \sigma(w^\top x)$.

2.2. Decision tree classifier

Decision tree classifier divides the input space into regions and learn one function per region. It splits regions successively based on the value of a single variable called test. The model is defined as follows

$$f_w(x) = \sum_k w_k \mathbb{I}(x \in \mathbb{R}). \tag{3}$$

The regions are learned adaptively but the difficulty is $O(NP)$. Decision trees are not very stable and easily overfit.

2.3. Support vector machine

Support vector machine (SVM) is a form of binary classification, classifying a set of data into two classes. SVM is a frontier which best segregates the two classes by finding the decision boundary with maximum margin M as follows

$$\begin{aligned} &\max_{w, w_0} M \\ &\text{such that } M \leq \frac{1}{||w||_2} y^{(n)} w^\top x^{(n)} + w_0 \ \ \forall n. \end{aligned} \tag{4}$$

2.4. AdaBoost

AdaBoost is a classification algorithm that adaptively assigns new weights to the train features such that the outliers have a higher weight. Thus, each next decision boundary is affected by the outliers and being attracted to them. This can be described as follows

$$f_t(x) = \text{sign}\left(\sum_t w^{\{t\}} \phi(x; v^{\{t\}})\right). \tag{5}$$

2.5. Random forest classifier

The random forest classifier is an expansion to the decision tree classifier covered in Subsection 2.2 that attempts to solve the overfitting issue of the latest. The idea is to further reduce the correlation between decision trees such that only a random subset of features are available for split at each step.

2.6. Multinominal naive Bayes

Multinominal naive Bayes is an extension to naive Bayes that allows to use word frequencies by utilizing multinominal likelihood

$$p_w(x|c) = \frac{(\sum_d x_d)!}{\prod_{d=1}^D x_d!} \prod_{d=1}^D w_{d,c}^{x_d} \tag{6}$$

where $x_d^{(n)}$ is the number of times word d appears in document n and c is one of the classes to affiliation to which is to be predicted.

3. Implementation

This work is carried out using Google Colaboratory so that every team member could contribute to the work in a concurrent and agile way [2]. Also, some portion of the work was implemented in Python on a machine running Windows® 10 with an NVIDIA® Quadro P400 graphic card with GPU Memory of 2 GB GDDR5 and 12 Intel® Core™ i7-8700 processors (3.20 GHz each).

The Python implementation for the 20 newsgroups dataset and for the IMDB dataset come as an attachment to this work in the ZIP file `code.zip` and are named `COMP_551_Mini_Project_2_20_newsgroups.ipynb` and `P2-IMDB.py`, respectively.

Model	Accuracy on the train set, %	Accuracy on the test set, %
Logistic regression	74.52	68.02
Decision tree	44.52	41.20
SVM	75.55	68.93
AdaBoost	46.67	44.20
Random forest	61.42	58.92
Multinominal naive Bayes	75.31	69.41

Table 1: Accuracy of the tuned models on the 20 newsgroups dataset. The winner is highlighted in green.

Model	Accuracy on the train set	Accuracy on the test set
Logistic regression	0.87148	0.88528
Decision tree	0.735	0.74572
SVM	0.86844	0.88548
AdaBoost	0.84368	0.84592
Random forest	0.83208	0.83612

Table 2: Accuracy of the tuned models on the IMDB dataset. The winner is highlighted in green.

3.1. Data description

Two datasets were adapted for experiments:

1. The 20 newsgroups text dataset is a textual dataset of approximately 18,000 news posts on 20 different topics [7]. The dataset is provided within the Scikit-learn library. The headers, footers, and quotes are removed from the dataset.

2. The Internet Movie Database (IMDB) Reviews dataset is another textual dataset of 25,000 either positive or negative movie reviews, i.e. the dataset is binary [4]. Another 25,000 reviews are provided for testing.

For the purpose of learning and classification, each dataset consists of the train and the test datasets.

3.2. Data pre-processing

It was decided to remove stopwords and punctuation in the 20 newsgroups dataset to increase the relevance of the data. Stopwords are words that occur in most of sentences and which do not help to identify the class of the sentence. Such words include *a*, *an*, *and*, *the*, *have*, etc. Punctuation also appears in most of sentences and can be dropped. It was decided to use the Natural Language Toolkit (NLTK) library for Python as it has proven itself as a reliable tool for textual data pre-processing [1].

Moreover, for both dataset it was required to convert the initial textual data to feature vectors. Thus, the following steps were made. First, the textual data was tolkenized. Secondly the frequencies of words were obtained from the occurrences. Do do that, the term frequencies (TF) and then term frequencies times inverse document frequency (TF-IDF) were obtained.

For the second dataset, we did the same except the facts that after importing dataset, instead of using NLKT, we used a function to remove marks [3]. Also we didn't use regularization and didn't remove stopwords because we realized they were not helpful from first dataset.

3.3. Learning

As was mentioned in Section 2, each model tested in this work is significantly different from another. Thus, they all utilize completely different hyperparameters which have to be tuned. For every major parameter of every model in this work, sets of each of hyperparameters are chosen to find the most optimal set of hyperparameters for each model. It was decided to apply k-fold cross-validation for splitting the train dataset in several folds, one of which is used for vali-

Dataset	Accuracy on the test set, %
20 newsgroups	69.41
IMDB	88.55
Overall accuracy	84.12

Table 3: Best accuracy for the both datasets and the overall accuracy.

dation of each particular set of hyperparameters for every model. The number of folds is chosen to be $k = 5$.

The only hyperparameter tuned for logistic regression is the inverse of regularization strength C. The maximum number of iterations is the higher is the better, so it was decided to set it to a sufficient value of 500. The dependency of the accuracy of logistic regression validation on the inverse of regularization strength C for the 20 newsgroups is illustrated in Table A.4. The most optimal parameter for the 20 newsgroups and IMDB dataset is respectively $C = 3.0, 2.0$.

For the decision tree classifier, the minimum number of samples required to split an internal node and the minimum number of samples required to be at a leaf node were tuned. The dependency of the accuracy of decision tree validation on the the minimum number of samples for the 20 newsgroups dataset is illustrated in Table A.5 and the dependency on the minimum number of samples is illustrated in Table A.6. The most optimal minimum number of samples splits for the 20 newsgroups and IMDB dataset is respectively 0.2 , 0.06 and minimum number of samples leaf for both dataset is 0.001.

The only hyperparameter tuned for SVM is the regularization parameter C_{SVM}. The dependency of the accuracy of SVM on the regularization parameter C_{SVM} for the 20 newsgroups dataset is illustrated in Table A.7. The most optimal parameter for the 20 newsgroups and IMDB dataset is respectively $C_{SVM} = 0.5, 0.25$.

For AdaBoost, the maximum number of estimators and the learning rate α were tuned. The dependency of the accuracy of AdaBoost validation on the the maximum number of estimators for the 20 newsgroups is illustrated in Table A.8 and the dependency on the learning rate α is illustrated in Table A.9. The most optimal maximum number of estimators for the 20 newsgroups and IMDB dataset is respectively 150, 200 and for both dataset, $\alpha = 0.5$.

For the random forest classifier, the maximum number of estimators, the minimum number of samples required to split an internal node and the minimum number of samples required to be at a leaf node were tuned. The dependency of the accuracy of random forest validation on the maximum number of estimators for the 20 newsgroups is illustrated in Table A.10, the dependency on the minimum number of samples required to split an internal node is illustrated in Table A.12, and the dependency on the minimum number of samples required to be at a leaf node is illustrated in Table A.11. The most optimal maximum number of estimators for both dataset is 200, minimum number of samples required to split an internal node for the 20 newsgroups and IMDB dataset is respectively 0.002, 0.003 and minimum number of samples required to be at a leaf node for both dataset is 0.001. Note that even though the maximum number of estimators is 200, it was decided to take 150 as the maximum number of estimators, for the newsgroups dataset, since the accuracy is not affected much by this decision but the performance is.

Not very encouraging cross-validation results of the previous models (except for SVM which almost approaches 70% accuracy) motivated to implement the multinominal naive Bayes model for the newsgroups dataset. It is not only simple to implement, but also fast to train. The only hyperparameter tuned for multinominal naive Bayes is the additive smoothing parameter α_{MNB}. The dependency of the accuracy of multinominal naive Bayes on the additive smoothing parameter α_{MNB} for the 20 newsgroups dataset is illustrated in Table A.13. The most optimal parameter is $\alpha_{MNB} = 0.05$.

4. Discussion and Results

The resulting accuracy of all implemented models on the test portion of the 20 newsgroups dataset is shown in Table 1. The results obtained from the implementation of SVM and multinominal naive Bayes have proven to be solid for the 20 newsgroups dataset. Note that in all cases the accuracy on the test set is slightly lower than on the train set, which is due to overfit of the models to the train data.

Summarizing, SVM and multinominal naive Bayes show the best results on the 20 newsgroups dataset among the others presented models with multinominal naive Bayes having the best accuracy on the 20 newsgroups dataset of 69.41%.

It was also decided to test these two models on the normalized dataset of the 20 newsgroups. For the the normalized data, the models achieve a slightly better accuracy: SVM achieves 69.81% of accuracy on the test data and while multinominal naive Bayes achieves 70.02%.

The resulting accuracy of all implemented models on the test portion of the IMDB dataset is shown in Table 2. We see that for all the models except Decision tree, the accuracy is around 85% .

According to the fact that all algorithms that we used are the same, we guess that this significant difference between two datasets might be because of difference between the classification type of each dataset (categorical and binary).

Since the best model for the 20 newsgroups dataset is multinominal naive Bayes and for the IMDB dataset SVM is the best one, Table 3 includes results these results as well as the overall accuracy.

References

[1] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media Inc, 2009.

[2] Google. Colaboratory. `https://research.google.com/colaboratory/faq`. Accessed: 2020-03-10.

[3] Google. Sentiment-analysis. `https://towardsdatascience.com/sentiment-analysis-with-python-part-1-5ce197074184`. Accessed: 2020-03-10.

[4] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[6] Reihaneh Rabbany and Siamak Ravanbakhsh. McGill University COMP-551, Lecture notes: Applied machine learning, Winter 2020.

[7] Rennie, Jason. 20 newsgroups dataset. `http://people.csail.mit.edu/jrennie/20Newsgroups/`.

A. k-fold cross-validation results for models tested on the 20 newsgroups dataset

The following pages of this work include results of k-fold cross validation for all the models analyzed in this work with $k = 5$.

C	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	5.5	6.0
Average accuracy, %	71.99	73.25	73.86	74.24	74.50	74.52	74.47	74.49	74.47	74.42	74.41	74.39

Table A.4: Dependency of the average accuracy across $k = 5$ folds on the inverse of regularization strength C of logistic regression for the 20 newsgroups dataset.

Minimum number of samples	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Average accuracy, %	47.27	48.04	47.47	46.72	46.06	45.95	45.87	45.78	45.97	45.71

Table A.5: Dependency of the average accuracy across $k = 5$ folds on the minimum number of samples required to split an internal node of decision trees for the 20 newsgroups dataset.

Minimum number of samples	0.001	0.002	0.003	0.004	0.005	0.006	0.007	0.008	0.009	0.010
Average accuracy, %	44.52	42.84	39.72	38.26	36.93	36.37	34.78	33.27	32.19	31.98

Table A.6: Dependency of the average accuracy across $k = 5$ folds on the minimum number of samples required to be at a leaf node of decision trees for the 20 newsgroups dataset.

C_{SVM}	0.25	0.50	0.75	1.00	1.25	1.50
Average accuracy, %	75.38	75.55	75.23	75.13	75.03	74.84

Table A.7: Dependency of the average accuracy across $k = 5$ folds on the regularization parameter C_{SVM} of SVM for the 20 newsgroups dataset.

Max. number of estimators	50	100	150	200
Average accuracy, %	39.17	44.25	46.11	45.76

Table A.8: Dependency of the average accuracy across $k = 5$ folds on the maximum number of estimators of AdaBoost for the 20 newsgroups dataset.

α	0.01	0.1	0.5	1	1.5	10
Average accuracy, %	28.22	41.83	46.67	46.12	45.62	7.12

Table A.9: Dependency of the average accuracy across $k = 5$ folds on the learning rate α of AdaBoost for the 20 newsgroups dataset.

Max. number of estimators	50	100	150	200
Average accuracy, %	63.59	64.73	65.49	65.53

Table A.10: Dependency of the average accuracy across $k = 5$ folds on the maximum number of estimators of random forest for the 20 newsgroups dataset.

Min. number of samples	0.001	0.002	0.003	0.004	0.005	0.006	0.007	0.008	0.009	0.010
Average accuracy, %	65.94	66.04	65.79	65.76	65.55	65.40	65.62	65.92	65.64	65.70

Table A.11: Dependency of the average accuracy across $k = 5$ folds on the minimum number of samples required to be at a leaf node of random forest for the 20 newsgroups dataset.

Min. number of samples	0.001	0.002	0.003	0.004	0.005	0.006	0.007	0.008	0.009	0.010
Average accuracy, %	61.45	57.80	54.15	50.65	47.86	45.12	43.66	40.63	38.34	35.75

Table A.12: Dependency of the average accuracy across $k = 5$ folds on the minimum number of samples required to split an internal node of random forest for the 20 newsgroups dataset.

α_{MNB}	0.001	0.005	0.010	0.050	0.100
Average accuracy, %	73.61	74.72	75.305	75.314	74.56

Table A.13: Dependency of the average accuracy across $k = 5$ folds on the additive smoothing parameter α_{MNB} of multinominal naive Bayes for the 20 newsgroups dataset.