Sentiment Classification with Statistical Machine Learning
Yifan Bai (260562421)

Problem Setup

This experiment compares three statistical machine learning (stat ML) methods, namely logistic regression, SVM with linear kernel (linear SVM) and Naïve Bayes', with a fourth one trying to solve a binary classification problem. Two corpus each containing 5331 sentences, and all of those are assigned either 'negative' or 'positive' labels. The problem is then a supervised learning classification task. Packages used are ***NLTK, Numpy, scikit-learn.*** To standardize, we used only unigram counts and *CountVectorizer* in ***scikit-learn'***s feature extraction module. For each method, we standardized the pipeline as such: with only stemming or lemmatization, compare the performance of whether to include stop words over a range of less frequent words with frequency thresholds of *[2, 3, 4, 5, 8, 10],* with which those lower than these will be removed. This will test the algorithm's ability to handle various quality of data, as well as the effect of such feature engineering on overall result. This pipeline makes full use of ***NLTK*** package.

Algorithms and Experiment Procedures

With recent booms in deep learning (DL), the performance of NLP models have drastically improved. The three base algorithms are all linear classifiers within stat ML family, which also incorporates ensemble methods that leverages weak learners with decision trees and bagging/boosting. Apart from classics such as Random Forest, Gradient Boosting, a newly emerged Kaggle winner, called Extreme Gradient Boosting (XGB) [1], is gaining popularity and used for this experiment.

Section I described the pipeline for experiment. On top of aforementioned, K-fold cross validation is used to compare the accuracy of the algorithm coming test time, as well as testing the ability of an algorithm's ability to generalize and prevent overfitting. For this problem, we use 5-fold, therefore the data will be divided into 80/20-train/test split. To implement this, we make use of *StratifiedKFold* of ***scikit-learn***, which preserves order dependencies in the dataset ordering without shuffling [2]. We also perform a simple hyperparameter grid search for each algorithm, which is entailed in the following section.

Hyperparameter Choices

For this problem, we conduct experiments on a range of minimum word frequency listed in *Section I*. For each specific model, while maximizing the available computation resources of a personal laptop, we conducted a simple grid search of selected hyperparameters. The choices, exact ranges and best performing are entailed in Table 2 in the Appendix, and following paragraph explains the reasons.

Logistic regression and Linear SVM are both linear classifiers and as such, regularization of objective function is key in performance. The built-in package allows us to explore the inverse strength of regularization, coded as 'C', which is a float value. For both, we investigated whether strong regularization would help. Additionally, Lasso regression, or L1, is compared with Ridge regression (L2) for logistic regression. Naïve Bayes' needs smoothing to help improve the performance, and the built in package allows us to set a smoothing constant of float value, with 1.0 being Laplacian smoothing. Finally, as a tree-based boosting ensemble method, we focused on investigating the tree's structures (ratio of subsamples, maximum depth) and gradient boosting schemes (learning rate, number of estimators), which is partially inspired by an experienced Kaggler's blog [3] and general knowledge of ensemble methods.

Results and Discussions

Referring to Table 1 in the Appendix is a summary of the best results obtained for each model after hyperparameter grid search. Logistic regression and Naïve Bayes' are the best performing with over 76.7% accuracy while linear SVM and XGB trail at 74% to 75%. A bare minimum baseline, with equal chance of labelling either outcomes, would be 50% given there are equal amount of training samples for both. From this regard, we can conclude that the 4 algorithms have worked. However, XGB theoretically 20 tunable hyperparameters [4]. On the one hand there could be more improvements made, on the other hand it implies high computational efforts. Referring to Table 3 Note that values in 'Best Results with Grid Search' column are the absolute best of each algorithm. For fair comparison, the values in 'Result with Default Settings' are those corresponding to the same settings. This table aims to show 'potentials' of each algorithms. XGB showed the most significant improvements, and provided more time and computational resources, better results could be achieved.

For logistic regression, the default L2 loss and regularization strength gave the best result. Time and resources permitted we could try other solver + loss (elastic net, linear combination of L1/L2); for linear SVM, however, having stronger regularization provided better results. It is worth noting that the improvements of these combinations could go up to 3% in accuracy. For Naïve Bayes', having smoothing does improve performance, but the largest value tried provided the best results. However, the improvements are less significant. For XGB, with the limited combinations tried, number of estimators has the biggest influence on performance, as the accuracy generally improved from around 65% to 75%. Learning rate also has its influence as too low of that increases computation time but not necessarily provide better results. Tree depth also has influence, as the best results were given by that equals to 5, closest to the default value of 6. With deeper and broader trees, there is the risk of overfitting, so one has to investigate the tradeoff.

In conclusion, this experiment directly compares algorithms. We learned that simple stats ML method would provide decent results, which is valuable for low computational resource settings. For real life scenarios, one would try different feature engineering methods, such as using different embeddings, e.g. word2vec, adding manual smoothing features for Naïve Bayes'. Pretrained language models are also commonly used, and some pretrained embeddings could significantly improve results, such as BERT. Lastly, DL methods could provide much better results provided appropriate architecture and sufficient data.

References

[1] V. Morde, "XGBoost Algorithm: Long May She Reign!," https://towardsdatascience.com/, 07-Apr-2019. [Online]. Available: https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d. [Accessed: Sep-2020].

[2] "sklearn.model_selection.StratifiedKFold," scikit-learn.org. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html. [Accessed: Sep-2020].

[3] P. Serguro, "Hyperparameter Grid Search with XGBoost," *Kaggle.com*, 2017. [Online]. Available: https://www.kaggle.com/tilii7/hyperparameter-grid-search-with-xgboost. [Accessed: Sep-2020].

[4] "Scikit-Learn API," XGBoost. [Online]. Available: https://xgboost.readthedocs.io/en/latest/python/python_api.html#module-xgboost.sklearn. [Accessed: 2020].

Appendix

Stemming	Lemmatization	Stop Words	Min Word Frequency	Accuracy
Logistic Regression				
Yes	No	No	2	76.77%
Yes	No	Yes	2	75.28%
No	Yes	No	2	75.23%
No	Yes	Yes	2	75.09%
Naive Bayes				
Yes	No	No	2	76.72%
Yes	No	Yes	2	76.72%
No	Yes	No	2	76.72%
No	Yes	Yes	2	76.72%
SVM				
Yes	No	No	2	74.67%
Yes	No	Yes	2	74.67%
No	Yes	No	2	74.67%
No	Yes	Yes	2	74.67%
XGB				
Yes	No	No	2, 3, 4, 5, 8	74.26%
Yes	No	Yes	2, 3, 4, 5, 8	74.26%
No	Yes	No	2, 3, 4, 5	74.26%
No	Yes	Yes	2, 3, 4, 5	74.26%

Table 1 – Best Results

Smoothing	Regularization Penalty	Inverse Regularization Strength	Subsample Ratios	Learning Rates	Max Depths	Number of Estimators
Logistic Regression						
	[L1, L2]	[0.2, 0.5, 1.0, 2.0]				
	L2 - Default	*1.0 - Default*				
SVM						
		[0.2, 0.5, 1.0, 2.0]				
		0.2				
Naive Bayes						
[0.0, 0.5, 1.0, 2.0]						
2.0						
XGB						
			[0.6, 0.8]	[0.1, 0.2]	[3, 4, 5]	[200, 400, 600, 800, 1000]
			0.6	*0.2*	*5*	*1000*

Table 2 – Best Hyperparameters From Mini Grid Search

Algorithm	Result with Default Settings	Best Result with Grid Search	Improvement
Logistic Regression	76.77%	76.77%	-
Naive Bayes	76.59%	76.72%	0.13%
SVM	72.92%	74.67%	1.75%
XGB	70.89%	74.26%	3.37%

Table 2 – Effects of Hyperparameter Tuning