Classification of Textual Data using Naive Bayes and Softmax Regression

Nicholas Dahdah (260915130), Jonayed Islam (260930902), Jasper Yun (260651891)

Abstract

Machine learning models are increasingly being applied to text classification tasks in which they extract features from data. Compared to simpler models, more complex models can fit textual data better to achieve higher accuracy. We implemented Gaussian Naive Bayes (NB) and Softmax Regression (SR) models, trained on benchmark text datasets, 20 newsgroups and sentiment140. In the training process, we cleaned the data to remove features that would have negligible impacts on prediction accuracy, then we tokenized and vectorized it. We used 5-fold cross-validation to compare the training accuracy against the validation accuracy and choose optimal hyperparameters for each model. On the 20 newsgroup dataset, we achieved testing accuracies of 49.4% and 60.8%, for NB and SR, respectively. On the sentiment140 dataset, we achieved testing accuracies of 66.9% and 79.9%, for NB and SR, respectively. To improve performance, we augmented our vectorization using term frequency-inverse document frequency (TF-IDF). While this performance does not surpass human expert performance, our classifiers can be used to roughly and quickly sort text data into different categories.

Introduction

Text classification is a popular application of machine learning (ML). For humans, parsing and analyzing text data is time-consuming, but ML models can quickly and cost-effectively operate on text data. With accurate ML models, tasks such as text organization, topic labeling, and sentiment extraction can be performed on a large scale. We implemented Gaussian Naive Bayes (NB) and Softmax Regression (SR) models for topic labeling and sentiment extraction using the 20 newsgroups [1] and sentiment140 [2] datasets. Various previous papers have used the 20 newsgroup dataset to compare the accuracy of their ML models. For instance, [3] developed a Bayesian subspace multinomial model using a logistic regression classifier which achieved 84.65% test accuracy on the 20 newsgroup dataset. The group [2] that gathered the sentiment140 dataset used a set of classifiers to determine the sentiment of Twitter tweets. The maximum test accuracy they achieved was 83.0% using a maximum entropy classifier.

Datasets

The two datasets used are the 20 newsgroups and sentiment140. The 20 newsgroups dataset contains 18,846 messages separated into 20 subjects. The training set contains 11,314 messages with each category having between 377 (3.3%) and 600 (5.3%) examples. The test set contains 7,532 messages with each category having between 251 (3.3%) and 398 (5.3%) examples. The sentiment140 dataset contains over 1.6M tweets that are annotated with either a positive or negative sentiment. Given limited computational resources, we sample a quarter of the tweets. Thus, the training set contains 400,000 messages with the happy and sad categories containing 199,891 and 200,109 examples, respectively. The test set contains 359 messages with the happy and sad categories containing 177 and 182 examples, respectively.

For both datasets, we only keep the data from the main text as well as the label and ignore other features such as the subject line. The main text was then processed in three steps: cleaning, tokenization and vectorization.

The cleaning phase involved removing all stopwords and punctuation from the original string of the main text. We then tokenized the string, before lemmatizing the words and setting the words to uppercase. Stopwords were removed as their frequency is too common across all texts. We believe that the cleaning phase, while not required, is beneficial in improving classification accuracy.

The text was then transformed with a `CountVectorizer` which calculates the frequency of each word as it occurs in the text. To avoid words that rarely appear and are likely to have little impact on our accuracy, we only kept words that appeared at least five times across all texts. We then used the result from the `CountVectorizer` to obtain the inverse document frequency vector using the `TfidfVectorizer`. The TF-IDF vectorizer helps evaluate how relevant a word is to a particular message compared to all the other documents. We hypothesize that this is very useful for classification as words which are more relevant to a particular document are given a higher score.

Dealing with these datasets has ethical implications that must be acknowledged. These datasets are filled with text pulled directly from public platforms. The fact that these platforms are public absolve the data-collectors from claims of violation of privacy. However, the authors of these messages/posts may not have consented to or expected for their information being used in the datasets, which can be an ethical concern.

Results

After tuning the hyperparameters of the NB and SR models to their optimal values via cross-validation, we evaluated our models using a test set of data unique from the training and validation data. These results are presented in Table 1. The corresponding confusion matrices for all eight trials can be found in figures 1-4 and in Tables 2 and 3.

We tested the accuracy of our model using both CountVectorization and TF-IDF data. It is interesting to note that both models with both datasets have equal or better prediction accuracy when using their TF-IDF representation.

Table 1. Naive Bayes and Softmax Regression Test Accuracies (optimal hyperparameters)

	Naive Bayes		Softmax Regression	
	Sentiment140	**20 Newsgroups**	**Sentiment140**	**20 Newsgroups**
CountVectorization	66.9%	49.4%	79.9%	60.8%
TF-IDF	70.2%	49.4%	80.8%	65.4%

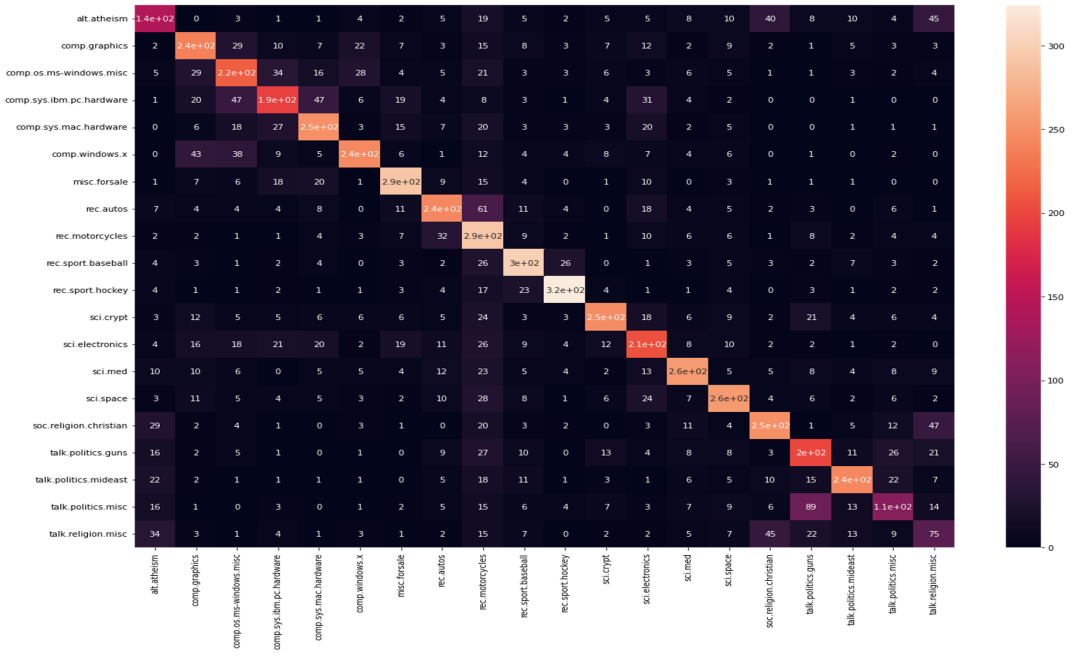

Fig. 1. Confusion Matrix of the 20 Newsgroups Prediction on Test Set with Softmax Regression Model (CountVectorization, C = 0.75, max-iters = 200)

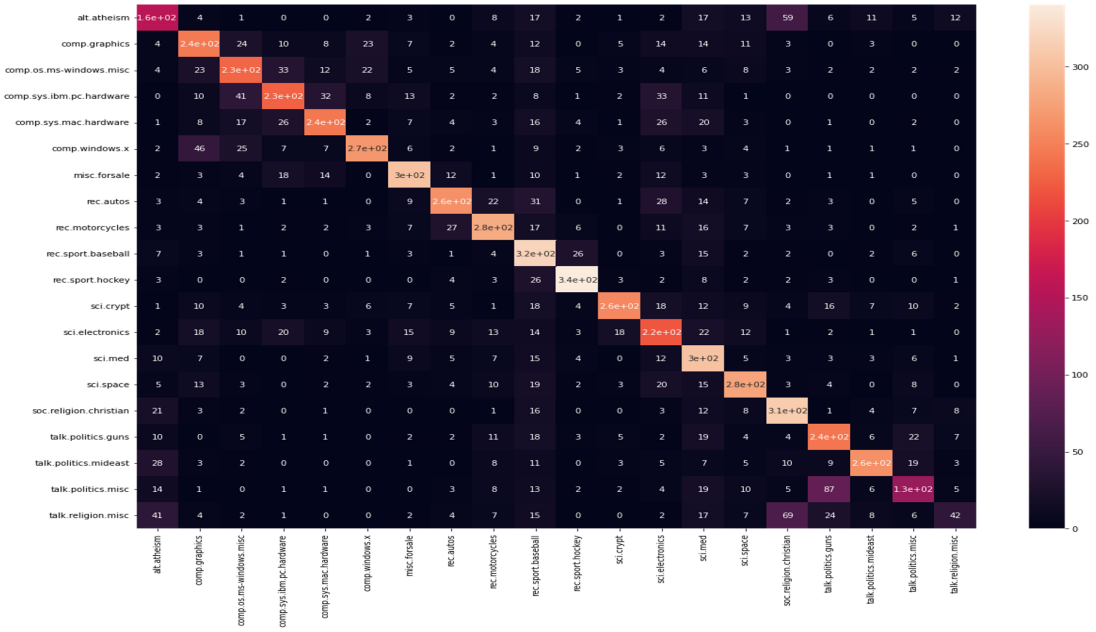

Fig. 2. Confusion Matrix of the 20 Newsgroups Prediction on Test Set with Softmax Regression Model (TF-IDF, C = 0.75, max-iters = 200)

Table 2. Confusion Matrices of Sentiment140 Prediction on Test Set with Softmax Model (C = 0.75, max-iters = 50)

	Softmax Regression (CountVectorization)		
ACTUAL	Total: 359	Happy	Sad
	Happy	135	42
	Sad	30	152

	Softmax Regression (TF-IDF)		
ACTUAL	Total: 359	Happy	Sad
	Happy	139	38
	Sad	31	151

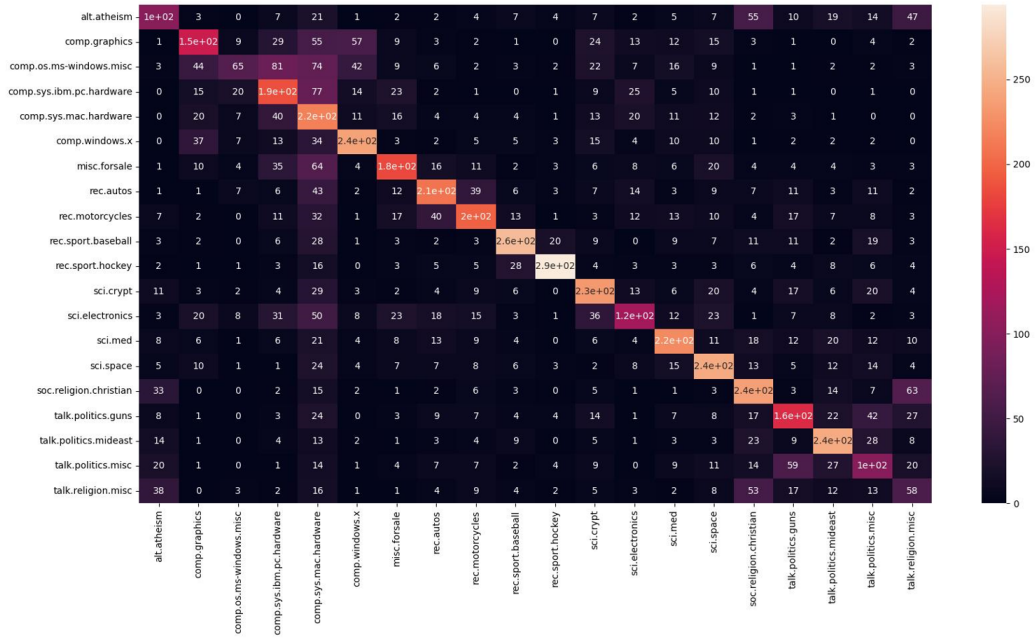

Fig. 3. Confusion Matrix of the 20 Newsgroups Prediction on Test Set with Naive Bayes Model (CountVectorization, smooth_val = 1E-6)

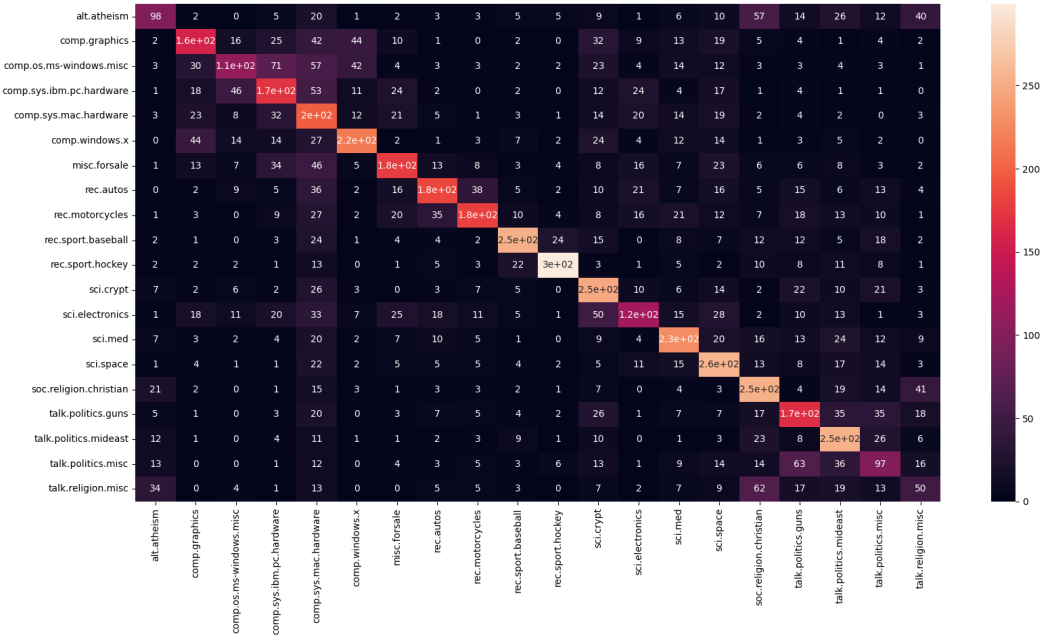

Fig. 4. Confusion Matrix of the 20 Newsgroups Prediction on Test Set with Naive Bayes Model (TF-IDF, smooth_val = 1E-6)

When observing the confusion matrix for the 20 newsgroup dataset, the majority of errors were caused by the algorithm choosing the wrong group that, however, still belongs to the same general category. For example, most of the misclassifications in the comp.graphics category were attributed to comp.os-ms-windows.msc or comp.windows.x.

Table 3. Confusion Matrices of Sentiment140 Prediction on Test Set with Naive Bayes Model (smooth_val = 1E-3)

	Naive Bayes (CountVectorization)		
ACTUAL	Total: 359	Happy	Sad
	Happy	142	35
	Sad	84	98

	Naive Bayes (TF-IDF)		
ACTUAL	Total: 359	Happy	Sad
	Happy	131	46
	Sad	61	121

It is critical to note that in order for the NB model to achieve a strong prediction accuracy, we introduced a smoothing parameter attached to the matrix of standard deviations, sigma. The smoothing parameter is added to each row of sigma and is proportional to the maximum variance in that current row of sigma. This enabled our model to avoid divide-by-zero errors by reducing the negative effect of zero-frequency elements.

Cross-Validation

As mentioned above, to obtain the optimal hyperparameters used when making predictions on the test set, we used 5-fold cross-validation. For the SR model, we varied the C value, which corresponds to the inverse of regularization strength, as well as the maximum number of iterations needed to converge. The optimal values for these parameters varied between datasets. For 20 newsgroups, these parameters were 0.75 and 200 for C and maximum iterations, respectively. For sentiment140, these parameters were 0.75 and 50 for C and maximum iterations, respectively. These results indicate that stronger regression can benefit prediction accuracy on both datasets. They also indicate that the optimal number of maximum iterations until convergence is dependent on the dataset being evaluated. For the NB model, we varied the smoothing parameter's prefactor. For 20 newsgroups, the optimal value of the smoothing prefactor was 1E-6. For sentiment140, the optimal value of the smoothing prefactor was 1E-3. These results indicate that stronger smoothing benefitted NB when evaluating sentiment140 compared to 20 newsgroups. Due to an exceedingly high computational workload, we were not able to perform cross-validation on more parameters, although we would have liked to. For SR, we would have liked to vary the solver and intercept fitting. For NB, we would have liked to try types other than gaussian, such as multinomial or Bernoulli. See Appendix A for detailed cross-validation results.

Effects of Training Set Size

We also examined the effects of changing the size of the training set on testing accuracy. These results are presented in Fig. 5. Using a smaller portion of the training data to fit the models tended to yield a poorer testing accuracy. This is particularly evident when evaluating the 20 newsgroup dataset. We believe this is the case because the 20 newsgroups training set is already much smaller than the sentiment140 dataset trimmed to a quarter of its size. Occasionally, there is a downward trend while moving up in training set size. We believe this is coincidental and that the training data that was sampled happened to be better fit to the testing set that was predicted. In general, more data allows the models to learn better, yielding better training accuracies.

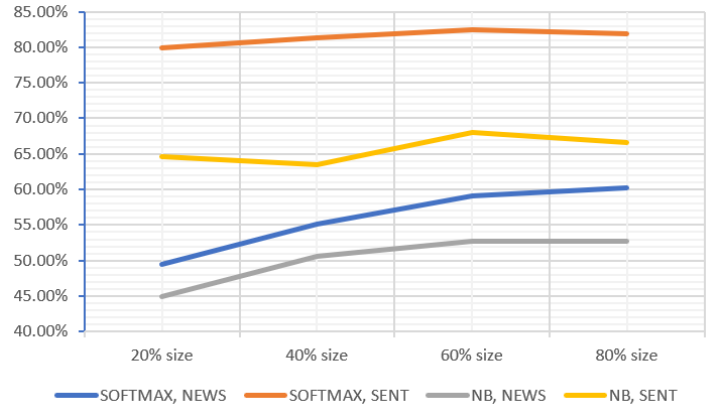

Fig. 5. Testing Accuracy vs. Percentage of Training Set Used (CountVectorization)

Discussion and Conclusion

Overall, the Softmax Regression model performed better than the Gaussian Naive Bayes Model on both the 20 newsgroups and sentiment140 datasets. We also note that using TF-IDF values had a considerable positive impact on our accuracy, due to the fact that it attributes more weight to relevant keywords. Each of our models performed well on the two datasets, but their test accuracies indicate that they are not more accurate than a human evaluator. Because of this, our models should only be used as a rough evaluative tool for classification.

In general, our models performed better on the sentiment140 dataset than on the 20 newsgroups dataset by a consistent margin of 15-20%. We believe this is due to the fact that the sentiment140 dataset, for our purposes, was a binary classification task. This made it much easier for our models to accurately predict the outcomes from the test data. The multinomial nature of the 20 newsgroups made it more challenging, as our models had to place text data in one of twenty different categories, rather than two. Even still, our models were capable of correctly classifying at least half of data.

It is also important to acknowledge the computational limits that we faced when completing this project. We were limited by both memory availability and runtime. When evaluating the sentiment140 dataset, we were not able to allocate enough memory to load intermediate matrices required for the prediction process. When performing cross-validation, we were forced to limit the parameters over which we performed grid search, since performing such an extensive task would take much too long. To work around these issues, we had to sample the sentiment140 data down to a quarter of its size and we had to limit the amount of parameters over which we grid searched during cross-validation.

To improve the performance of these models in future work, it would be interesting to explore different model variations. For example, evaluating the performance of a SR model with nonlinear data or a multinomial NB model rather than a gaussian one. We also believe that preprocessing the data more aggressively could yield strong results, as many words in text data are non-indicative of tone or subject, meaning they can be disregarded for potentially better prediction accuracy.

Statement of Contributions

Nicholas handled the implementation of cross-validation and hyperparameter optimization, as well as figure generation. Jonayed processed the data and handled the training and experimentation for Softmax Regression. Jasper handled the implementation and experimentation for Gaussian Naive Bayes.

Please note that a Google Colab .ipynb file is attached with this submission and contains all necessary code for the assignment, but not all of it will run due to issues with insufficient memory. As such, two extra .py files are attached, one which simply runs Naive Bayes, called “comp551_A2_runNB.py,” and one which runs cross-validation on both Naive Bayes and Softmax Regression, called “comp551_A2_runCV_NB_SR.py.”

References

[1] F. Pedregosa et al. “Scikit-learn: Machine learning in Python”, JMLR 12, pp. 2825-2830, 2011.

[2] A. Go, R. Bhayani and L. Huang. “Twitter sentiment classification using distant supervision,” [https://cs.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf], Jan. 2009.

[3] S. Kesiraju, O. Plchot, L. Burget and S. V. Gangashetty. “Learning document embeddings along with their uncertainties,” arXiv:1908.07599 [cs], Oct. 2019.

Appendix A

NEWS, LOGISTIC									
		cvals=0.75	cvals=0.75	cvals=0.75	cvals=1.00	cvals=1.00	cvals=1.00		
		maxiters = 50	maxiters = 100	maxiters = 200	maxiters = 50	maxiters = 100	maxiters = 200		
	L=1	0.579319487	0.665930181	0.664162616	0.595227574	0.670349094	0.670349094		
	L=2	0.627043747	0.673000442	0.669465312	0.624834291	0.672116659	0.669465312		
	L=3	0.612461335	0.677419355	0.672116659	0.615996465	0.659743703	0.667255855		
	L=4	0.591692444	0.668581529	0.673884225	0.59036677	0.682280159	0.676093681		
	L=5	0.590627763	0.66091954	0.67948718	0.587533157	0.669761273	0.67152962		
	avg(L)	0.600228955	0.669170209	0.671823198	0.602791651	0.670850178	0.670938712	0.671823	BEST: cvals=0.75, maxiters=200
		FALSE	FALSE	TRUE	FALSE	FALSE	FALSE		
SENT, LOGISTIC									
		cvals=0.75	cvals=0.75	cvals=0.75	cvals=1.00	cvals=1.00	cvals=1.00		
		maxiters = 50	maxiters = 100	maxiters = 200	maxiters = 50	maxiters = 100	maxiters = 200		
	L=1	0.7600625	0.7581875	0.758	0.7554375	0.755875	0.756		
	L=2	0.759625	0.759	0.758625	0.7586875	0.758125	0.7575		
	L=3	0.756625	0.7553125	0.755	0.755625	0.753375	0.753375		
	L=4	0.7575625	0.757	0.7565625	0.7563125	0.7555625	0.754125		
	L=5	0.7621875	0.7600625	0.759875	0.76175	0.758125	0.757125		
	avg(L)	0.7592125	0.7579125	0.7576125	0.7575625	0.7562125	0.755625	0.759213	BEST: cvals=0.75, maxiters=50
		TRUE	FALSE	FALSE	FALSE	FALSE	FALSE		
NEWS, GAUSSIAN NAÏVE BAYES									
		smoothing=1	smoothing=1E-3	smoothing=1E-6	smoothing=1E-9	smoothing=1E-12			
	L=1	0.057003977	0.393283252	0.571807335	0.570481662	0.566946531			
	L=2	0.053910738	0.379584622	0.569597879	0.56650464	0.572691118			
	L=3	0.052143173	0.498453381	0.594343791	0.592576226	0.581970835			
	L=4	0.05125939	0.332744145	0.57666814	0.572691118	0.564737075			
	L=5	0.054818745	0.383289125	0.589301503	0.585322723	0.577365164			
	avg(L)	0.053827205	0.397470905	0.58034373	0.577515274	0.572742145		0.580344	BEST: smoothing 1E-6
		FALSE	FALSE	TRUE	FALSE	FALSE			
SENT, GAUSSIAN NAÏVE BAYES									
		smoothing=1	smoothing=1E-3	smoothing=1E-6	smoothing=1E-9	smoothing=1E-12			
	L=1	0.5356875	0.6611875	0.65025	0.641125	0.633625			
	L=2	0.548375	0.648125	0.6244375	0.6113125	0.601375			
	L=3	0.555	0.6435625	0.613375	0.5976875	0.589125			
	L=4	0.589125	0.641875	0.6146875	0.5996875	0.5886875			
	L=5	0.617625	0.627875	0.6019375	0.5911875	0.5835			
	avg(L)	0.5691625	0.644525	0.6209375	0.6082	0.5992625		0.644525	BEST: smoothing 1E-3
		FALSE	TRUE	FALSE	FALSE	FALSE			