Classification of Textual Data

Mini project 2 – COMP 551 (Winter 2022)

ABSTARCT

Here, we have a text classification problem to solve which is one of the most researched problems due to continuously increasing number of electronic documents and digital data *(Meena, M. Janaki, et al. 2009)*. We evaluated the performance of generative models (Multinomial Naïve Byes and Transformed Weight-Normalized Complement Naïve Bayes (TWCNB)) and discriminative models (Logistic regression (LR)) when doing document classification. We used two benchmark datasets in this assignment, the 20Newsgroups dataset, and the Sentiment140 dataset. When working with the 20Newsgroups dataset, the goal was to do the multiclass classification to 20 groups, and when working with the Sentiment140 dataset, the goal was to do the binary classification of the sentiment behind a text (if it is positive or negative). We investigated how to get to numerical feature spaces from the text environment using count vectorizer and TF-IDF vectorizer methods. We saw how using TF-IDF can help us with model generalization and accuracy improvement. We also evaluated the accuracy of models on multiple scenarios or instances of selected datasets, and we observed that when train data size goes toward infinity, Logistic Regression models perform better than generative models, while generative models can reach their maximum performance using fewer train instances. Comparison between ordinary naïve Bayes models and TWCNB also showed that TWCNB could perform better in multiclass classification. The best accuracies obtained in the case of the 20Newsgoups dataset was about 71% (TWCNB) and in the case of Sentiment140 was about 81% (LR). As the train data size is great in the case of Sentiment140, LR's better performance is acceptable. However, in the case of 20Newsgroups, generative models seem to show a better accuracy, which is acceptable since the train data size is not that big; we assume that if we had more train data in the case of the 20Newsgroups dataset, LR could perform better. We also investigated the effect of changing smoothing parameter in Naïve Bayes and TWCNB models and used a 5-fold cross-validation approach to get to the best hyperparameter. In the case of LR, we tried to find the best penalty and regularization factor using 5-fold cross-validation.

INTRODUCTION

Text categorization is the problem of assigning a label to data *(Ming-Wei Chang, et al. 2008)*. Instead of manually classifying documents or hand-making automatic classification rules, many machine learning algorithms are used to automatically classify unseen documents based on human-labeled training documents *(Xu, S. 2018)*. In this study, using the 20Newsgroups, we aim to classify texts into well-known categories. On the other hand, using the Sentiment140 dataset, we aim to learn how to determine the emotion behind a series of words, generally expressed over an online platform (in this case, Twitter). Category selection or sentiment analysis is widely used in areas such as social media and allows us to gain an overview of the more comprehensive public opinion *(Raghuwanshi AS, et al. 2017)*.

We used generative and discriminative learning to classify texts in this assignment. Generative classifiers learn a model of joint probability $p(x, y)$, of the inputs x and labels y and make their predictions by using Bayes rules to calculate $p(y|x)$ and then picking the most likely label y. Discriminative classifiers model the posterior $P(y|x)$, directly or learn a direct map from inputs x to the class labels *(Andrew Y.Ng et al. 2002)*. We considered naïve Bayes model (and also TWCNB) and its discriminative analog, logistic regression. We realized that the generative models have a higher asymptotic error than the discriminative model, but the generative model can approach its asymptotic error much faster than the discriminative model. As the number of training examples increases, there can be two distinct regimes of performance, the first in which the generative model has already approached its asymptotic error and is thus doing better, and the second in which the discriminative model approached its lower asymptotic error and does better *(Andrew Y.Ng, et al, 2002)*.

We also investigated different methods for feature extraction from text data, count vectorizer and TF-IDF, we evaluated models' performance while using these two methods, and realized than in the case of the first dataset, TI_IDF could improve model accuracy. Moreover, we tried to optimize the models by setting the hyperparameters within them using 5-fold cross validation. The hyperparameter in naïve Bayas and TWCNB was smoothing factor and in the case of logistic regression the hyperparameters are regularization factor and also the type of penalty used in model generation.

DATASETS

20NEWSGROUPS

The 20Newsgroups Dataset is a common benchmark used for testing classification algorithms. The dataset, introduced in *(Lang, 1995)*, contains 18,846 newsgroup posts, approximately evenly divided across 20 groups ranging in categories such as sports, religions, politics, etc. (*Ming-Wei Chang, et al. 2017*). The data is divided into train and test parts, the test set contains 7532 texts (approximately 40% of all data), and the train set contains 11314 texts (about 60% of all data). The following table illustrates the distribution of data across test/train sets and across different class labels.

Label	documents in test set	documents in train set	documents
alt.atheism	319	480	799
comp.graphics	389	584	973
comp.os.ms-windows.misc	394	591	985
comp.sys.ibm.pc.hardware	392	590	982
comp.sys.mac.hardware	385	578	963
comp.windows.x	395	593	988
misc.forsale	390	585	975
rec.autos	396	594	990
rec.motorcycle	398	598	996
rec.sport.baseball	397	597	994
rec.sport.hockey	399	600	999
sci.crypt	396	595	991
sci.electronics	393	591	984
sci.med	396	594	990
sci.space	394	593	987
soc.religion.christian	398	599	997
talk.politics.misc	310	465	775
talk.politics.guns	364	546	910
talk.politics.mideast	376	564	940
talk.religion.misc	251	377	628
Total	**7532**	**11314**	**18846**

SENTIMENT140

Sentiment140 is the dataset consisting of 1.6 million tweets (1- 140-character documents) which have been machine-annotated using an emoticon heuristic. It is a well-balanced dataset comprising 0.8 million positive and 0.8 million negative tweets, allowing us to construct datasets with a wide range of sizes. Here, the used test set contains 359 tweets where 182 are positive, and 177 are negative. The tweets in this dataset are labeled as 0 when it is a negative sentiment and 4 when it is a positive sentiment. The dataset consists of six features listed and described in Table 1. The only sections that will be important in this study would be the text and the sentiment sections, and the other parts are removed (*Gaye, B, et al. 2021; Prusa, J. et al, 2015*).

	sentiment	id	date	query_string	user	text
0	4	3	Mon May 11 03:17:40 UTC 2009	kindle2	tpryan	@stellargirl I loooooooovvvvveee my Kindle2. ...
1	4	4	Mon May 11 03:18:03 UTC 2009	kindle2	vcu451	Reading my kindle2... Love it... Lee childs i...
2	4	5	Mon May 11 03:18:54 UTC 2009	kindle2	chadfu	Ok, first assesment of the #kindle2 ...it fuck...
3	4	6	Mon May 11 03:19:04 UTC 2009	kindle2	SIX15	@kenburbary You'll love your Kindle2. I've had...
4	4	7	Mon May 11 03:21:41 UTC 2009	kindle2	yamarama	@mikefish Fair enough. But i have the Kindle2...

PREPROCESSING

Data extracted from online platforms contain unnecessary data which do not help in terms of classification. This makes preprocessing of the data a critical step prior to model development. This preprocessing step directly impacts the performance of the machine learning models.

PREPROCESSING STEPS

When preprocessing the 20Newsgroups dataset, the first step is to remove any subset of ('headers', 'footers', 'quotes'). Each of these is a kind of text that will be detected and removed from the newsgroup posts, preventing classifiers from overfitting on metadata and keeping the model's generalization intact. The other necessary step is removing stop-words (frequently used words in English like 'the' and 'a'), contractions, punctuations, and numeric values. For defining stop-word, we used a predefined list available in python. These removals are considered essential, as a large dataset necessitates more time for training, and data that are not correlated with the analysis reduce the accuracy of the prediction. Also, after cleaning the unnecessary data, the texts are converted to all lower-case, to be more generalized in terms of word list generation. The reason behind this step is that machine learning models are case-sensitive, which means that the same words with the upper or lower case will be considered as distinct words. Furthermore, in classification, one often wishes to group words with the same meaning to the same term; in this regard, we used 'nltk' library in python to do the lemmatization, which involves the conversion of words into their root forms by deleting affixes from the words (e.g. 'are' to 'be') (*Gaye, B, et al. 2021).* Sample texts from the dataset before and after preprocessing are illustrated below.

```
0  I was wondering if anyone out there could enli...
1  A fair number of brave souls who upgraded thei...
2  well folks, my mac plus finally gave up the gh...
3  \nDo you have Weitek's address/phone number?  ...

cleaned train data
0  i be wonder if anyone out there could enlighte...
1  a fair number of brave souls who upgrade their...
2  well folks  my mac plus finally give up the gh...
3  do you have weitek s address phone number  i w...
```

Figure 1 sampled texts before and after preprocessing - 20Newsgroups dataset

For the Sentiment140 dataset, the preprocessing step followed the same approach; however, some extra steps were added to it. For instance, removing URLs, Hashtags, Mentions, and HTML links. After all the processing, nan columns were omitted from the dataset.

FEATURE EXTRACTION

Before running any machine learning algorithms, we need to convert texts into numerical features and form a feature space that is understandable to the machine learning classifier. To do feature extraction, two distinct approaches can be used. The first one uses the simple counting approach, and the other uses the Term Frequency — Inverse Document Frequency (TF-IDF) method.

1- The simplest way to obtain features is to use the Bag of Words (BoW) method. In this approach, each document is represented as a set of words and the number of times each word occurs in the document. So, each unique word will have a unique value in the dictionary. In this approach, the position of the words is ignored, and we just make use of the frequency of each word *(Ming-Wei Chang, et al. 2017; Lai CM, et al. 2022).*
There is a built-in class for implementing this feature extraction technique found in the Scikit-learn library of python called 'CountVectorizer'. It converts the text document collection into a matrix of integers (Garreta and Moncecchi). It is noteworthy that we first use 'fit_transform()' on the train data and then use 'transform()' on the test data. When using 'fit_transform()' on the train set, it means that the machine learns from the parameters in the feature space and also scales the training data, on the other hand, using 'transform()' on test data scale it according to the parameters learned from the training data. We did not use 'fit_transform()' on the test set since the goal of a test set is to test how well the model performs on the new data, which, in this case, means the unseen words in test data should not be considered as an essential factor.

2- To avoid larger files getting more weights after BoW the other used method is the TF-IDF (*Chun-Ming Lai*). In TF-IDF each term in the document is given a weight based on its term frequency (TF) and inverse document frequency (IDF). The FT-IDF transformation takes the original word frequency f and transforms it. Assuming that df is the number of documents containing the word under consideration and D the total number of documents, then the transformed attribute value becomes (*Kibriya, ashraf m. et al. 2004*):

$$TFIDF(word) = \log(f + 1) * \log\left(\frac{D}{df}\right)$$

This approach can be used in python using 'TfidfVectorizer' class in Scikit-learn library. Then again, the class is fitted on train data, not the test data (just because of the same reason explained in the previous section.)

In this assignment, both methods, CountVectorizer and TF-IDF, have been applied to transform the text document into a numerical vector that is then considered input to machine learning algorithms. Another point to mention is that when trying to build the feature space, in the case of the 20Newsgroups dataset, we ignored the words that were used in less than 2 documents, and we also ignored the words that we present in more than 40% of the documents. In the case of the Sentiment140 dataset, we just ignored the words that were used in less than 2 documents.

Note that we extracted unigrams (individual words within the text of the tweet) features not bi-grams, tri-grams; the reason behind this choice was that more complex features, such as bi-grams, tri-grams, and part-of-speech features had been shown to provide little additional value when conducting tweet sentiment classification, (*Prusa, J. et al., 2015*). Other research also states that in the case of the 20Newsgroups dataset, using unigrams can lead to higher F-score, precision, and recall *(Saleem, at el. 2021)*. However, one main drawback with unigram attributes is that they are in huge numbers and can lead to enormous vocabulary size.

TRAIN SIZE REDUCTION IN SENTIMENT140 DATASET

In the end, it is noteworthy to mention that we did not use all the 1.6 million tweets as a training set to avoid a high computation load when analyzing the second dataset. We shuffled the data and then chose 50,000 tweets of positive sentiment and 50,000 tweets of negative sentiment for further model development (the new training size would be 100,000).

METHODS

IMPLEMENTED MODELS

MULTINOMIAL NAÏVE BAYES - BUILT FROM SCRATCH

Let us start with a brief description of multinomial naïve Bayes. Let N be the size of vocabulary and C be the number of classes, then MNB assigns a test document t_i to the class that has the highest probability $\mathrm{P}(c|t_i) = \frac{P(c)P(t_i|c)}{P(t_i)}$. Prior distribution can be estimated by dividing the number of documents belonging to class c by the total number of documents, and $P(t_i|c)$ is the probability of obtaining a document like t_i in class c and is calculated as: $\mathrm{P}(t_i|c) = (\sum_n f_{ni})! \prod_n \frac{P(w_n|c)^{f_{ni}}}{f_{ni}!}$ Wheref_{ni} is the count of words n in our test document t_i and $P(w_n|c)$ the probability of word n given class c, which is estimated from training document as: $P(w_n|c) = \frac{1+F_{nc}}{N+\sum_{x=1}^{N} F_{xc}}$ where F_{xc} is the count of word x in all the training documents belonging to class c. Hence, if we look at the classification rule for MNB given all the mentioned formulas, we get:

$$class(t_i) = argmax[\log(\mathrm{P}(c)) + \sum_n f_{ni} \log(\frac{\alpha + F_{nc}}{\alpha N + \sum_{x=1}^{N} F_{xc}})]$$

Where $\boldsymbol{\alpha}$ is the **smoothing factor**, which should be a positive integer greater than 0, if we do not have this smoothing factor, what happens is that division to 0 might occur in some cases, which makes the computations meaningless. This smoothing parameter has been chosen to be set as the **hyperparameter** of the model when performing k-fold cross-validation. In the end, it is important to mention that when using this algorithm, an assumption of conditional independence is considered, meaning that we assume all features are independent of each other, given the class variable.

TWCNB - BUILT FROM SCRATCH

Besides the multinomial naïve Bayes model, another approach is built from scratch: Transformed Weight-Normalized Complement naïve Bayes (TWCNB). TWCNB is a variant on the MNB classifier that fixes many of its problems without making it slower or difficult to implement. While being similar to MNB, one of the differences is that the TF_IDF normalization transformation is part of the definition of this algorithm. However, the main difference between the two is that TWCNB estimates the conditioned feature probabilities using data from all classes apart from c (*Kibriya, ashraf m. et al. 2004*). Hence, we have such formulation in this kind of classification:

$$w_{nc} = \log\left(\frac{\alpha + \sum_{k=1}^{C} F_{nk}}{\alpha N + \sum_{k=1}^{C}\sum_{x=1}^{N} F_{xk}}\right) where\ k \neq c$$

$$class(t_i) = argmax[\log(\mathrm{P}(c)) - \sum_n f_{ni} \log(w_{nc})]$$

As the value of $\log(\mathrm{P}(c))$ is usually negligible in the total, we can simplify the equations as:

$$class(t_i) = argmin\left[\sum_n f_{ni} \log(w_{nc})\right]$$

The literature shows that TWCNB performs about equally or better than MNB. Note that in this case, we again have a **smoothing factor** as a **hyperparameter** of the model that we need to set using cross-validation.

LOGISTIC REGRESSION

For Logistic regression, both L_1 and L_2 penalty and various values for the inverse of regularization strength, C, are compared. If we use L_1 regularization in logistic regression, all the less critical features will become zero, making it look like a feature reduction procedure. If we use L_2 regularization, then weight values will become small (but not necessarily equal to 0).

L_2 regularization:

$$\hat{\beta} = argmin\ \|X\beta - y\|_2^2 + \lambda \|\beta\|_2^2$$

L_1 regularization:

$$\hat{\beta} = argmin\ \|X\beta - y\|_2^2 + \lambda \|\beta\|_1$$

In terms of the regularization parameter, when we increase λ (decrease C), we prioritize wights not getting large, which can lead to underfitting. If we decrease λ (increase C), we are not paying enough attention to the importance of weights, which makes us prone to overfitting. This explanation illustrates the importance of setting a correct value for C. In this assignment, we use cross-validation to find the best penalty and the best regularization factor.

EFFECTIVENESS- BUILT FROM SCRATCH

The performance tests are based on the accuracy of the suggestions made by the classifier in assigning the right category to a particular document.

$$ACC = \frac{true\ positive + true\ negative}{true\ positive + true\ negavite + false\ positive + false\ negative}$$

RESULTS AND INTERPRETATION OF RESULTS

HYPERPARAMETER TUNING – USING 5-FOLD CROSS-VALIDATION- BUILT FROM SCRATCH

We employ five-fold cross-validation, which splits the data into five equal partitions. By the term ‘data’, we mean all the data except the test sets, which we call unseen data and put them aside first. In each iteration of CV, four folds are used as training data, while the remaining fold serves as a test dataset. This is repeated five times with a different partition used for validation in each iteration. After all, the

hyperparameter that has led to the least mean validation error will be chosen as the best hyperparameter. All data except the unseen part will be used to fit a model having that hyperparameter, and accuracy on the unseen data will be reported as the model performance.

MULTINOMIAL NAÏVE BAYES

In the case of the 20Newsgroups dataset, the best smoothing rate for MNB is equal to 0.05. This value has been obtained by using a 5-fold cross-validation approach. After finding the best hyperparameter, we build the model using this value for smoothing factor and by using the whole training set. The proposed model is evaluated using the test/unseen dataset. The obtained accuracy is 69.41% in this case.

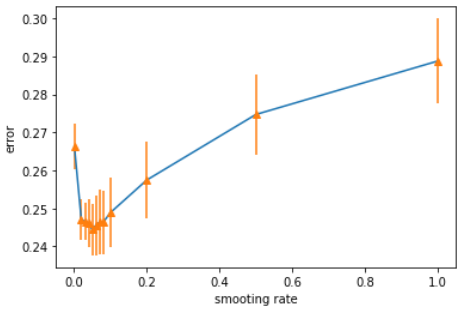


Figure 2 finding the best hyperparameter for the MNB classifier – 20Newsgroups dataset

In the case of the Sentiment140 dataset, the best smoothing factor would be 1, and the corresponding accuracy on the test/unseen data is 78.83%.

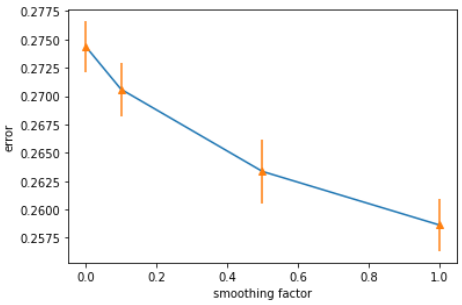


Figure 3 finding the best hyperparameter for the MNB classifier – truncated Sentiment140 dataset

TWCNB

The best smoothing rate for TWCNB is equal to 0.5, which is found using the 5-fold cross-validation. If we build a model using all instances in the training set and select 0.5 as alpha's value, we get test set accuracy equal to 70.56%.

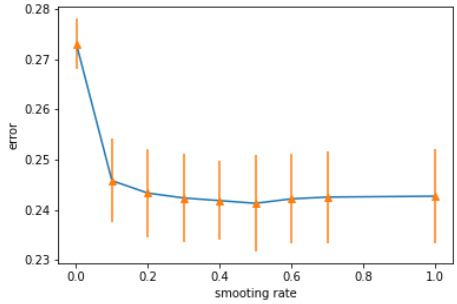


Figure 4 finding the best hyperparameter for the TWCNB classifier – 20Newsgroups dataset

We do not use TWNBC in the case of the Sentiment140 dataset since it is a simple binary classification problem which makes it unnecessary to implement TWCNB.

LOGISTIC REGRESSION

As illustrated in the figure below, it is evident that in the 20Newsgroups dataset, logistic regression with L_1 penalty has a higher validation error in compared with logistic regression model with L_2 penalty. Hence, the preferred penalty would be L_2. Furthermore, 5-fold cross-validation suggests C equal to 4 as the best hyperparameter. If we build an LR model using L_2 penalty and C equal to 4, we get an accuracy of 67.9% on the test/unseen set.

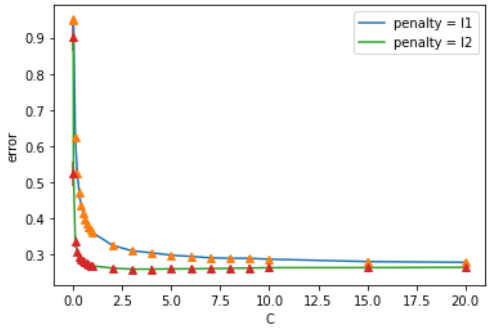


Figure 5 finding the best regularization factor and type of penalty in LR – 20Newsgroups dataset

In the case of the Sentiment140 dataset, 5-fold cross-validation suggests L_2 penalty and C equal to 0.8 as the best model parameters. This model leads to an accuracy of 80.77% on test data.

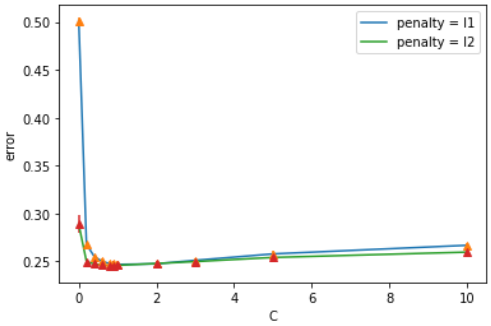


Figure 6 finding the best regularization factor and type of penalty in LR – truncated 140Sentimaent dataset

COMPARISON OF GENERATIVE AND DISCRIMINATIVE MODELS - BUILT FROM SCRATCH

In this section, we used different percentages of train data to build the model using the fixed test/unseen set to evaluate the model performance. The expectation is that discriminative logistic regression algorithm has a lower asymptotic error while the generative naïve Bayes classifier may converge more quickly to its higher asymptotic error, thus as the number of training examples m is increased, one would expect generative naïve Bayes to initially do better but for discriminative logistic regression to eventually catch up to, and quite likely overtake the performance of naïve Bayes. In cases that we find that the logistic regression's performance did not catch up to that of naïve Bayes, we assume that the dataset was small and m presumably have not grown large enough for us to observe the expected dominance of logistic regression in the large m limit *(Andrew Y.Ng et al. 2002).*

The following figure and table show different models' performance on various percentages of the test data (20Newsgorup dataset). You can see that the TWCNB model's performance is slightly better than the multinomial naïve Bayes classifier. Meanwhile, you can see that these generative models slightly perform better compared to logistic regression, which is a discriminative model. However, if we increase the train size, logistic regression is expected to perform better than generative models. The last thing to mention is that naïve Bayes models seem to need fewer train data to reach high accuracy and are trained with less effort, making them a potential candidate for online text classification.

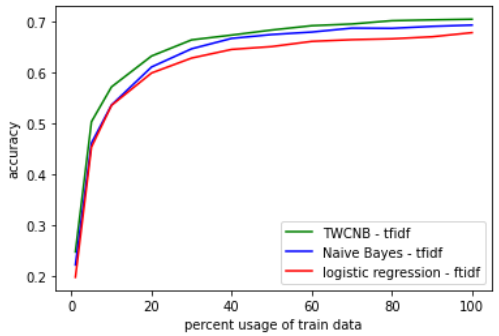


Figure 7 Accuracy of NMB. TWCNB and LR across different train set sizes – 20Newsgroups dataset

Training percent	*Accuracy MNB (0.05)*	*Accuracy LR Solver:'saga' max_iter:1000 penalty: L2 C: 4*	*Accuracy TWCNB (0.5)*
1%	0.2231	0.1988	0.2488
5%	0.4624	0.4544	0.5042
10%	0.5370	0.5363	0.5726
20%	0.6117	0.5999	0.6332
30%	0.6475	0.6293	0.6650
40%	0.6679	0.6463	0.6744
50%	0.6755	0.6518	0.6844
60%	0.6804	0.6621	0.6929
70%	0.6881	0.6654	0.6962
80%	0.6877	0.6671	0.7027
90%	0.6915	0.6711	0.7043
100%	**0.6941**	**0.6793**	**0.7056**

In the case of the second dataset, Sentiment140, what happens is that the logistic regression model outperforms other models when 100% of data is presented. While naïve Bayes works slightly better when introducing more minor train data. All in all, one can say that naive Bayes converges quicker but has a higher error than logistic regression when the train set is big enough.

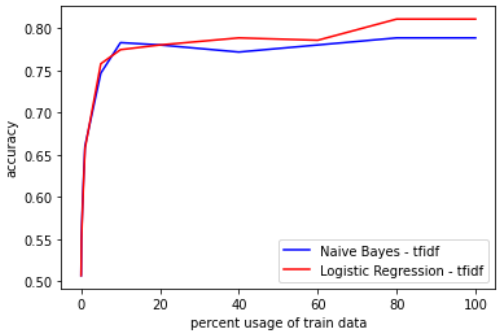


Figure 8 Accuracy of NMB. TWCNB and LR across different train set sizes – truncated 140Sentiment dataset

Training percent	*Accuracy MNB (1)*	*Accuracy LR Solver:'saga'max_iter:1000 penalty: L2 C: 0.6*
0.01%	0.5069	0.5069
0.5%	0.6072	0.6016
1%	0.6601	0.6573
5%	0.7465	0.7576
10%	0.7827	0.7743
20%	0.7799	0.7799
40%	0.7715	0.7883
60%	0.7799	0.7855
80%	0.7883	0.7883
100%	**0.7883**	**0.8105**

As mentioned before, it is expected to see that when the training size reaches infinity, the discriminative model: logistic regression performs better than the generative model Naive Bayes. Such a thing is observed in the second dataset, where we have 100,000 samples in the train set, while we cannot observe it yet for the first dataset.

TF-IDF OR COUNT VECTORIZER

As mentioned before, when working with text data, we have two approaches to extract features. One method is using count vectorizer, and the other is using TF-IDF. We tried to investigate how the TF-IDF approach can help with the model's generalization. In this regard, we did train set splitting. In each iteration, we once extracted features using TF-IDF vectorizer and once pulled them using count vectorizer, then fitted the test data on each of them and classified the test data using MNB and LR classifiers. Here, you can see the accuracy of the two models as a function of the size of the dataset, once the models were based on features obtained from the count vectorizer and once the models were based on features obtained from TF-IDF.

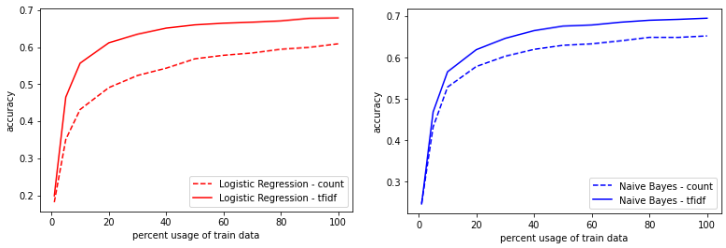


Figure 9 different feature extraction methods lead to different performances in ML classifiers – 20Newsgroups dataset

It is clear that in this case, TF-IDF results in better accuracy in models, presumably because it focuses on the frequency of words present in the corpus and provides the importance of the terms. However, in terms of the second dataset, things are a bit different.

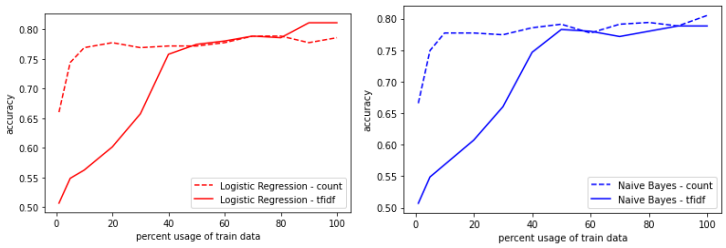


Figure 10 Figure 11 different feature extraction methods lead to different performances in ML classifiers – Sentiment140 dataset

SUMMERY OF THE RESULTS

Model / Dataset	MNB	TWCNB	LR
20Newsgroups	69.41%	**70.56%**	67.93%
Sentiment140	78.83%		**80.77%**

DISCUSSION

We have learned the classic problem in NLP, text classification. We learned about important concepts like count vectorizer, TF-IDF for extracting features from text data and implemented important algorithms NB, TWCNB, and LR. We investigated the effect of various feature vectors on model performance and realized that TF_IDF approach could cause a more generalized model. In this assignment, our truncated 140sentimente was classified with the accuracy of around 80,77% using the logistic regression model, and the 20newsgoups dataset was classified with the accuracy of 70.56% using the TWCNB approach. We also realized that when train set is big enough, logistic regression which is a discriminative approach works better than naïve Bays which is a generative model. Furthermore, naïve Bayes performs better when train size is not enough meaning that naïve Byes reaches its asymptotic error faster. The suggestion for further investigation is to try support vector machines and neural network classifiers to train your models and see how much accuracy one achieves.

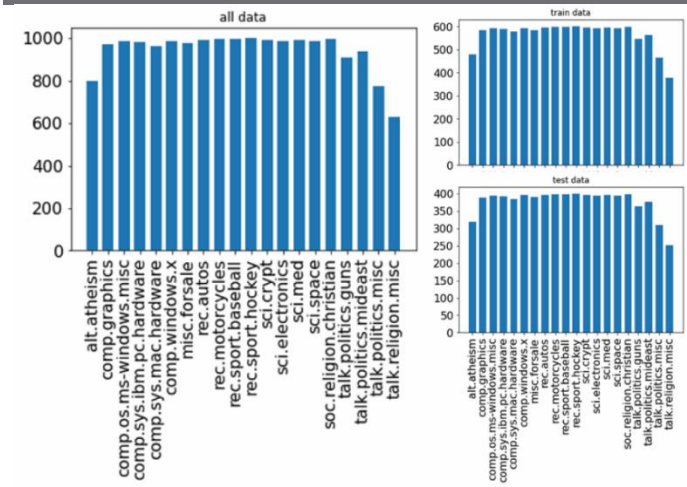
STATEMENT OF CONTRIBUTIONS

- Asa Borzabadi Farahani: report draft + preprocessing dataset 1 + code implementation
- Gurucharan Marthi Krishna Kumar: preprocessing dataset 2 + code review/debug + report review
- Yunhan Li: code review/debug + report review

REFRENCES

1. Andrew Y Ng and Michael I Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In Advances in neural information processing systems, pages 841–848, 2002.
2. Chang, Ming-Wei, Lev-Arie Ratinov, Dan Roth, and Vivek Srikumar. "Importance of Semantic Representation: Dataless Classification." In *Aaai*, vol. 2, pp. 830-835. 2008.
3. Raghuwanshi AS, Pawar SK. Polarity classification of Twitter data using sentiment analysis. International Journal on Recent and Innovation Trends in Computing and Communication. 2017;5(6):434-9.
4. Lang, K., 1995. 20 newsgroups.
5. Gaye B, Zhang D, Wulamu A. A Tweet Sentiment Classification Approach Using a Hybrid Stacked Ensemble Technique. Information. 2021 Sep;12(9):374.
6. Prusa, Joseph, Taghi M. Khoshgoftaar, and Naeem Seliya. "The effect of dataset size on training tweet sentiment classifiers." In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pp. 96-102. IEEE, 2015.
7. Lai CM, Chen MH, Kristiani E, Verma VK, Yang CT. Fake News Classification Based on Content Level Features. Applied Sciences. 2022 Jan;12(3):1116.
8. Kibriya, Ashraf M., Eibe Frank, Bernhard Pfahringer, and Geoffrey Holmes. "Multinomial naive bayes for text categorization revisited." In *Australasian Joint Conference on Artificial Intelligence*, pp. 488-499. Springer, Berlin, Heidelberg, 2004.
9. Meena, M. Janaki, and K. R. Chandran. "Naive Bayes text classification with positive features selected by statistical method." In *2009 First International Conference on Advanced Computing*, pp. 28-33. IEEE, 2009.
10. Xu S. Bayesian Naïve Bayes classifiers to text classification. Journal of Information Science. 2018 Feb;44(1):48-59.
11. Saleem Z, Alhudhaif A, Qureshi KN, Jeon G. Context-aware text classification system to improve the quality of text: A detailed investigation and techniques. Concurrency and Computation: Practice and Experience. 2021 Jul 21:e6489.

SUPPLEMENTARY INFORMATION



Supplementary figure 1 – Distribution of 20Newsgroups dataset across different class labels