COMP 551 Applied Machine Learning Project Report: Text Classifier TheThreeRedditteers

Shuonan Dong
shuonan.dong@mail.mcgill.ca

Owen Li
owen.li@mail.mcgill.ca

Yuanhang Yang
yuanhang.yang@mail.mcgill.ca

I. INTRODUCTION

This paper demonstrates our work in classifying conversations from Reddit websites with eight different topics (*hockey, movies, nba, news, nfl, politics, soccer* and *worldnews*). The training data set contains 165,000 samples with around 20,000 conversations in each category. Naive Bayes and decision tree algorithms are implemented by our team and the performances are compared with that of the support vector machine (SVM) and Naive Bayes methods from scikit-learn library. According to Kaggle, our highest accuracies for these three algorithms are 76.49%, 70.11%, and 96.70%, respectively. Discussion of these three approaches are provided in Section VI.

II. RELATED WORK

A. Naive Bayes in text classification

Naive Bayes initially caught our interest for its relative simplicity in implementation, and further so after Prof. Cheung mentioned it commonly being used for text classification in real-world products. In reviewing the literature on Naive Bayes for text classification, McCallum and Nigam's paper [1] pointed out some differences between two Naive Bayes probabilistic models: multi-variate Bernoulli model, and the multinomial model. This paper summarized that, among other aspects, the multinomial model generally outperforms multi-variate Bernoulli models given large vocabulary sizes. Our implementation using multinomial Naive Bayes functionality from scikit-learn indeed outperformed our implementation of the multi-variate Bernoulli model.

B. SVM in text classification

We cast a good deal of interest in using SVM in this challenge, initially because of the idea that word frequencies in texts can represent high-dimensional vectors, and the fact that SVMs learn regardless of the dimensionality of feature spaces. In addition, one particular paper by Joachims [2] pointed out the (theoretical) advantages of using SVM for text classification: *high dimensional input space, few irrelevant features, sparse document vectors, and linear-separability of most text categorization problems.* Since SVM implementation can be tedious, we decided to apply SVM functionalities provided in the scikit-learn library.

III. PROBLEM REPRESENTATION

A. Data pre-processing methods

Our data pre-processing work relied on leveraging the capabilities of machine learning libraries: scikit-learn, NLTK, BeautifulSoup, etc. The pipeline of data pre-processing converts the raw data into arrays of integer-valued vectors, in the following steps:

1) Read the CSV file into data frames.
2) Parse each conversation from an HTML-formatted document into plain text.
3) Filter out any text element that isn't in the English alphabet: numbers, punctuations, accented letters, etc.
4) Get rid of stop words. The list of stop words are determined through a visual inspection of the frequency of words across the total data, specifically targettng words with a nonsensical meaning, as well as common words found in all categories of the data. This is done with the aim of filtering the raw data into distinct, proper words that will serve to discriminate topics of conversation from one another.
5) For the Naive Bayes implementation, an extra lemmatization step is performed here: verbs and nouns are brought down to their original form, such that e.g. "American" and "America" will be treated as the same word.
6) Convert the array of text into feature vectors. Since features used, and the form they take are different in each method, this step is further discussed in subsection III-B.

B. Feature design/selection methods

For Naive Bayes and SVM, the scikit-learn function CountVectorizer was used to convert the array of texts into a matrix of token counts, resulting in a much larger matrix: about 17,000 features for Naive Bayes and 93,059 for SVM. Note that Naive Bayes had fewer features because only words that appeared in more than 15 conversations were counted as a feature. Entries of this matrix are then further processed from simple occurrence counts into TF-IDF (Term Frequency - Inverse Document Frequency) counts. At this point, the feature matrix is ready to be used by the classifiers. TF-IDF metrics are used to represent the significance of a word in a collection of documents.

Worldnews	NBA	Hockey	Politics	Soccer	NFL	Movies	News
4217 isis	5634 nba	7322 game	5210 obama	4220 world	6924 nfl	15939 movie	4910 police
4207 israel	5363 game	4600 nhl	3563 right	3861 goal	6848 twitter	6423 movies	2515 news
3952 russia	4035 twitter	4245 imgur	3354 republicans	3578 league	4655 game	6258 imgur	2409 right
3886 uk	3972 lebron	4223 hockey	3120 org	3433 players	3390 season	5144 film	2289 going
3732 world	2883 player	3836 twitter	2672 want	3414 player	2794 qb	4474 youtube	2181 make
2969 co	2498 youtube	2674 thread	2625 government	3365 cup	2536 play	3410 first	2109 want
2909 news	2420 season	2669 season	2617 vote	3321 gfycat	2498 going	3311 trailer	2019 new
2704 war	2263 best	2385 play	2599 money	3129 imgur	2490 best	2715 best	1990 shit
2426 right	2227 play	2283 last	2505 gop	3075 co	2396 teams	2646 great	1940 state
2331 country	2207 players	2199 fans	2454 state	3045 game	2358 last	2548 make	1937 someone
2223 going	2159 kobe	2171 youtube	2360 republican	2994 season	2328 player	2506 watch	1931 money
2082 want	2037 better	2130 first	2335 police	2858 football	2284 first	2458 new	1886 years
2011 ukraine	1973 back	2119 going	2324 president	2790 uk	2224 bowl	2446 star	1824 cops
1985 russian	1918 going	2110 back	2289 make	2784 match	2221 years	2318 made	1820 way
1941 way	1904 imgur	2094 games	2091 party	2717 best	2192 brady	2254 films	1751 something

TABLE I

15 MOST FREQUENT WORDS IN EACH CATEGORY

For the decision tree classifier, the first step is to select the highest frequency words for each topic. Each word represents a "feature" that will be evaluated when deciding on the optimal splits in the tree. The number of words chosen to represent each topic was chosen through a trial process, although it is worth noting that beyond 100 most common words, further inclusion of distinguishing words becomes insignificant to the results. These words are then merged into a comprehensive list, which represents the most commonly used words in the entire dataset. For each conversation, if the conversation does indeed contain the word, it will be marked with a 1, and a 0 otherwise. The goal of this boolean assignment is to attempt to identify patterns of similarity between these occurrence vectors. This list of occurrences will be used as input in the recursive construction of the tree.

IV. ALGORITHM SELECTION AND IMPLEMENTATION

A. *Linear classification: Naive Bayes*

Two kinds of Naive Bayes implementations are commonly used for text classification purposes, as discussed in section II-A. The method implemented from scratch was the multivariate Bernoulli model. The Naive Bayes algorithm gives a predicted class C_i by solving $argmin_{C_i} P(C_i|W)$ given the word vector (bag of words and their occurrences) W of an unseen data point x. The Naive Bayes classifier is a generative model, where we compute the likelihood $P(W|C_i)$ based on the training data and then use Bayes rule to estimate $P(C_j|W)$. In this context, C_i represents the ith category, and W_j the jth word in the bag of words. Note that for text classification purposes,

$$P(Wj|Ci) = \frac{N_j + 1}{N + |V|},$$

where V is the set of words, $|V|$ is the number of unique words in V, N_j is the frequency of word j in all conversations in C_i category, and N the sum of total occurrences of all words in C_i (that is, the "word count" of C_i). The size of the uni-gram used in the algorithm is around 17,000, which is summed up from the most frequent 2000 words of each category. To reduce the computation time and redundant work in the code, lemmatization used during the training process contributed to reducing the size of the token set.

B. *Non-linear classification: Decision Trees*

The idea of decision trees in text classification is to construct a tree from distinguishing "words".In this implementation, the tree is constructed with a greedy top down recursive approach. The methodology follows a very intuitive line of thought. A root node is chosen on the first instance of recursion. Semantically, this node represents the most distinguishing word in the dataset, such that all words that will be partitioned to the left of it will fall into some subset of topics, and those partitioned to the right will fall into another subset of topics. These two partitions are disjoint. The process of evaluating which node produces an optimal binary split of the data is done through calculating entropy values of the dataset before and after the split. This calculation is repeated for each "feature" in the occurrence vector, such that the data will be split on each word. This provides a measure of information gain for each split, the highest of which will be chosen as the optimal split. The tree is built recursively in this manner.

This top down greedy approach, while intuitive and simple to implement, has drawbacks. The recursive splitting of the data has no means of termination prior to reaching an empty set. This condition leads to possible overfitting of the tree, as the branches will become overly complex and specific to the training data. The solution proposed in this implementation is to limit the depth of the tree, such that upon reaching a certain depth on any branch, the next node beyond a preconfigured threshold will be made a leaf node. The content of this leaf node is the most common topic of the remaining conversations in that branch.

Since the accuracy of the test set predictions are verified on Kaggle, the implementation concerned itself with only training and validation sets. K-Fold validation was used, with K=5.

C. *Libraries*

To leverage the capabilities of SVM, the built-in SVM feature in the sci-kit learn library was used. Given the pre-processed data, as described in section III-A, it is very straightforward to use the SVM functionality in a library: a class SGD Classifier is created and trained directly against the pre-processed data, with the following parameters:

1) Loss mode is "hinge". This is the default value and gives a linear SVM.
2) Penalty (regularization term) is "l2", the standard regularizer for linear SVM models.
3) Alpha (Constant multiplier in the regularization term) is set to 0.001.
4) Number of iteration is set to the default value of 5.
5) Random State is set to 42. This number is the seed value used for the random number generator. The seed must be explicitly defined to achieve repeatable results.

In addition to SVM, multinomial Naive Bayes and multi-variate Bernoulli Naive Bayes libraries were leveraged. The report will focus on the results of the multinomial Naive Bayes method, as it achieved the highest performance.

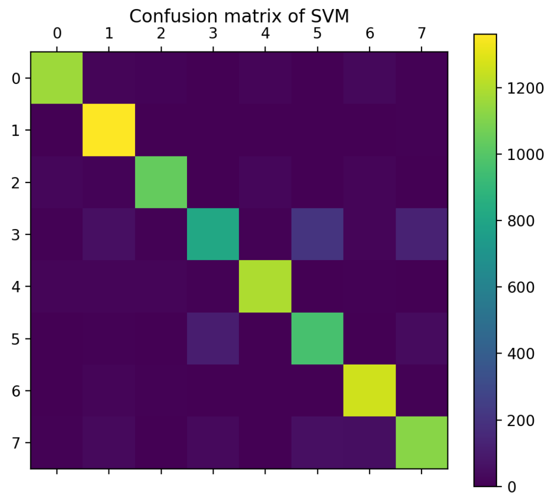

Fig. 1. Confusion Matrix of SVM algorithm.

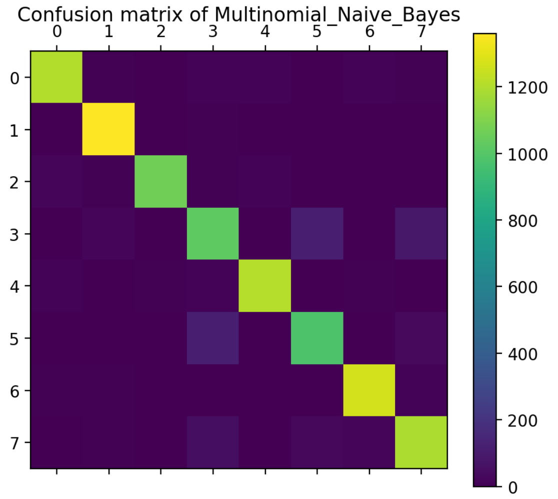

Fig. 2. Confusion Matrix of Multi-nomial Naive Bayes algorithm.

The most significant parameter involved with optimizing this function was the smoothing coefficient. The importance of the inclusion of this parameter stems from needing to solve the problem of value zero class conditional probabilities. The smoothing parameter which, was chosen as alpha < 1.0 to implement lidstone smoothing, prevented the problem of encountering zero probabilities.

Additionally, the choice of considering n-grams as features was also significant in the results. This is justified in the feature design section. The degree to which we set n is a matter of computational power and practicality. A higher n can dramatically increase computation time, as there are many more permutations to consider. Through trial, it was decided to include n-grams with n ranging from 1 to 5. This selection kept the computational cost relatively low, while providing a higher precision.

Topic	Precision	Recall	F1-score
hockey	0.71	0.64	0.65
movies	0.84	0.85	0.84
nba	0.70	0.60	0.65
news	0.43	0.68	0.52
nfl	0.70	0.68	0.69
politics	0.70	0.59	0.64
soccer	0.76	0.67	0.71
worldnews	0.74	0.61	0.67

TABLE II
SUMMARY OF RESULTS FOR DECISION TREE

V. TESTING AND VALIDATION

A. Decision Trees

With K-Fold testing over the entire dataset, the following performance metrics were calculated. Each number represents an average performance over the 5 folds.

The breakdown of precision for each topic is very indicative of the nature of the data set. A prime observation to make is that 'movies' as a topic was very well distinguished from the other categories. This makes intuitive sense, as looking at the other topics, it can be seen that nfl, nba, hockey, and soccer all fall within the realm of sports, and therefore share a relatively homgenous vocabulary. The same logic applies to news, politics, and worldnews. The precision of "News" is particularly low. This result can also be justified by the observation that it is a superset of politics and worldnews, making it difficult to distinguish from the other more specific subsets. However, this could also be a symptom of overfitting, as the tree is struggling to predict more general topics.

The support value represents the frequency of classes in the dataset, and is indicative that the selection of the sets used in K-Fold testing was balanced.

The results clearly demonstrate that the model does indeed exceed the performance of baseline random guesses, however it does not offer optimal performance compared with the other methods of classification explored in this paper. Main areas of non-optimal performance can largely be attributed to the design of the implementation. For example, the approach used a binary classification of the data: "did the conversation contain the word or not". This is a simple, but largely flawed approach. The frequency of specific words in a conversation can be a significant indicator in determining the topic. In addition, groups of words that often appear together may also be a significant indicator of topic. These issues are unaccounted for in the implementation of the decision tree, and are certainly major causes for its poor performance in this particular data set.

Topic	Precision	Recall	F1-score
hockey	0.76	0.72	0.74
movies	0.88	0.88	0.88
nba	0.70	0.92	0.65
news	0.72	0.45	0.56
nfl	0.82	0.73	0.78
politics	0.65	0.80	0.72
soccer	0.83	0.74	0.78
worldnews	0.84	0.62	0.71
total	0.74	0.72	0.71

TABLE III
SUMMARY OF RESULTS FOR NAIVE BAYES

B. *Naive Bayes*

In the implementation of Naive Bayes, the size of the training set was fixed at 120,000 and the validation set at 10,000. In order to find the best subset for the classifier, a random shuffle was used to increase performance on the validation set. However, for the final test set prediction, the entire data set was taken as training data. The following table represents the data gathered over the training set of 120000, and validated over a validation set of 10'000. As previously seen in the discussion for the results of the decision tree, much of the same logic applies. However, it is worth noting that the Naive Bayes implementation on this data set has a higher precision on most topics of conversation, and is able to more accurately discriminate between topics with similar vocabularies.

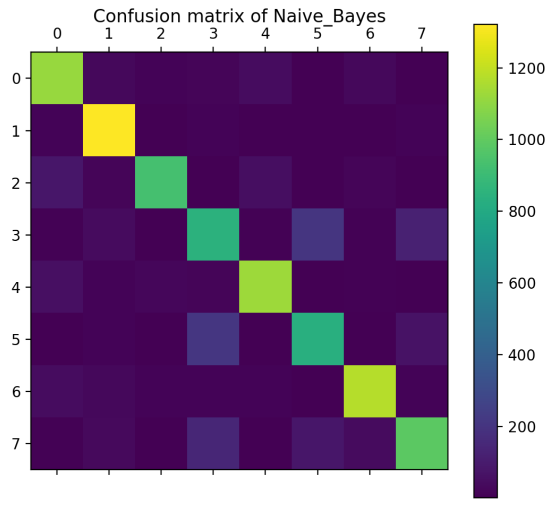

Fig. 3. Confusion Matrix of Bernoulli Naive Bayes algorithm.

The rows in the confusion matrix represent the actual categories the conversations fall in, and the columns are the predictions given by the classifier. Therefore, the brighter the diagonal is, the more accurate our predictions are. Bernoulli Naive Bayes' overall performance is similar to that of the package versions in sci-kit learn, as indicated by the remarkable similarities in the generated confusion matricies. It is important to note that some categories are more often confused than others. This can be explained similarly to the results of decision trees. The nature of the categories lends itself to a notable overlap of common ideas, and therefore, common vocabularies.

VI. DISCUSSION

A. *Decision Trees*

A major challenge in the implementation of decision trees was the time taken to run the model. The implementation discussed has a run-time of about 90 minutes to build the tree, and an additional 20 minutes to perform the data preprocessing. For practical reasons, the model could not be more complex, as it would increase the run-time too much. The appeal of this approach lies in its intuitiveness. It serves as a suitable stepping stone for developing ideas on improvements that may generalize to other approaches. The shortcomings of the decision tree implementation were addressed in the implementation of the other methods to much success, particularly in the Multinomial Naive Bayes approach.

B. *Naive Bayes*

Given the results, the Naive Bayes algorithm implemented from scratch did not perform as well as the implementation from the scikit-learn library. Possible justifications for this shortcoming may stem from the limited number of tokens used in the generated bag of words. The set included less than 8500 uni-grams. The initial intention behind using less words in the training set was to make the classifier more sensitive to the most frequent words, which turned out to be less in number than expected. After enlarging the number of uni-grams to approximately 17,000, as well as including bi-grams, the implementation achieved improved results. Due the limited budget of computation, we did not try out a larger scale of data. A noticeable detail while implementing Naive Bayes was the importance of eliminating the frequently used words shared by multiple topics. For example, 'love' seems to be a meaningful word in the classification problem, especially for the sentinel classification. However, for the classification problem in this paper,the goal is not to classify the conversations into a binary result, but rather categorical. Therefore, words that may be more discriminative between topics were chosen for the token set. This was done with the aim of increasing the sensitivity of the classifier towards distinct classes.

VII. STATEMENT OF CONTRIBUTIONS

This work was overall a team effort, but each member's contribution can be roughly accredited as following:

1) Dong: Implementing Naive Bayes classifier. Coding SVM and Naive Bayes methods with scikit-learn library. Optimizing the scikit-learn library SVM and Naive Bayes approaches.
2) Li: Implementing decision trees. Optimizing the scikit-learn library SVM and Naive Bayes approaches.
3) Yang: Coding SVM method with scikit-learn library. Report writing.

We hereby state that all the work presented in this report is that of the authors.

REFERENCES

[1] A. McCallum, K. Nigam *et al.*, "A comparison of event models for naive bayes text classification," in *AAAI-98 workshop on learning for text categorization*, vol. 752. Citeseer, 1998, pp. 41–48.

[2] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *European conference on machine learning*. Springer, 1998, pp. 137–142.