COMP 551: Mini Project 2 - Classification of Reddit Comments and Bernoulli Naive Bayes Implementation

GROUP-97 Nikhil Podila[1], Shantanil Bagchi[1], Manoosh Samiei[1]

Abstract— **In this project, we investigate the performance of text classification methods on reddit posts from over 20 subreddits. We preprocess the data using natural language processing techniques such as stopword removal, TF-IDF weighting, χ^2 test for feature selection. We used various classification algorithms and found that an Ensemble Classifier performed the best on this dataset. We also implemented Bernoulli Naive Bayes from scratch. Performances of all the classifiers and our implementation of Bernoulli NB are compared on the dataset. We achieved an accuracy of 61.02% on held-out validation set and 58.633% on Kaggle test set.**

I. INTRODUCTION

A. Task Description

Reddit is an American social news aggregation, web content rating, and discussion website. Registered members submit contents such as links, text posts, and images to it. Posts are organized by subject into user-created boards called "subreddits", with topics including news, science, movies, music [1].

Topic Labeling, which is our main task, is understanding what a given text is talking about. It's often used for structuring and organizing data such as organizing customer feedback by its topic or organizing news articles according to their subject [2].

The aim of our task is to analyze a test set of comments extracted from 20 subreddits, classify them to the appropriate subreddits, and compete in a Kaggle competition [3] with other groups to achieve the best accuracy. For this task, we are given a subreddit-labelled training set of comments.

B. Algorithm and approach

In this project, we first extract important features from the comments using Natural Language Processing (NLP) techniques detailed in Section III. Then, we use and compare the performance of various machine learning classification algorithms from the Scikit-learn library [5]. We also implement a Bernoulli Naive Bayes model from scratch to compare its performance with other classifiers.

[1]McGill University

C. Important findings

We found that preprocessing individual comment is essential but specifically for our dataset contraction removal, html removal, lemmatizing and stemming data actually reduced overall performance.

Next, we found that tuning TF-IDF vectorizer parameters and combining it χ^2-test helped us select the top 15000 features.

After extensive hyper-parameter tuning for our models, we found that an Ensemble Method with Ridge Classifier, Naive Bayes and OneVsRest classifier with Naive Bayes worked better than any other classifiers. We achieved 61.02% accuracy on validation set.

The rest of the paper is organized as follows. Section II covers works related to this task. Section III discusses the pre-processing methods. Section IV presents our proposed approaches and implemented models. Section V indicates our results and models' accuracy. Section VI contains discussions and conclusions made from the project, and finally Section VII highlights each members' contribution.

II. RELATED WORK

For the data preprocessing and feature engineering, Giel et. al. [6] have performed TF-IDF, Mutual information and PCA based feature selection, along with Latent Dirichlet Allocation (LDA). Another article [7] implements Count vectorizing and balancing the dataset. We explored further steps and alternatives to the procedures from these articles, including those from Scikit-learn [10] and NLTK [11] libraries.

For classification algorithms, Giel et. al. [6] implemented Multinomial Naive Bayes and Logistic Regression. The article by Martin [7] uses Support Vector Classifiers and Extreme Gradient Boosting to achieve high accuracy. However, since we have 20 completely distinct topics (subreddits) to classify, our approach could not be limited to output from a single classifier.

game
one
people
like
would
time
get
think
also
even
team
well
still
good
thing
make
know
right
http
really
much
actually
player
guy
see
point
play

Fig. 1. Word cloud of top words across all subreddits

We decided to explore ensembles of classifiers to achieve better accuracy.

We also referred to the implementations in [8], [9] and many other articles as part of our literature survey over the past weeks.

III. DATASET AND SETUP

A. *Dataset and features*

The reddit dataset contains 70000 comment samples across 20 subreddits with $subreddit$ as the target variable. The samples across all the subreddits are balanced.

We are using data from 20 subreddits :

AskReddit	*baseball*	*Global Offensive*
Overwatch	*canada*	*gameofthrones*
anime	*Music*	*league of legends*
europe	*funny*	*conspiracy*
hockey	*movies*	*worldnews*
nba	*soccer*	*nfl*
trees	*wow*	

B. *Data pre-processing methods*

We explored few individual comments across all the subreddits to understand the data. A large effort was dedicated to experimenting with different feature representations for comments. We processed the text to retain as much information as needed without involving unnecessary words. In this process, we removed 'Non-ASCII' characters, numbers, punctuation, spaces and single letters.

Next we removed stop-words, a list of common words that are not context specific, provided by the NLTK package [11], along with a few additional words. Additionally, Word Net Lemmatizer with Part of Speech Tagging and Snowball Stemmer are used to lemmatize and stem the document but are found to reduce the performance for our dataset so we dropped them from our implementation. We also considered both unigrams and bigrams during preprocessing, as a pair of words occurring together often might be more specific to a particular context. Unigrams proved to be performing better so we only considered using unigrams in our implementations.

We used two different approaches to calculate the weights of a term:

- **Binary**: If a word shows up in a post, its weight is held 1 else it is 0.
- **TF-IDF**: This type of weighting represents the term-frequency and inverse document frequency of the reddit comment rather than just a binary value. [12].

We noticed that extracting large vocabulary from the preprocessing step results in a large feature vector dimension which is computationally costly and is prone to overfitting. Also, the Bag-of-words (BOW) approach is simplistic and unable to capture essential information from the data. We performed feature space dimension reduction to counter this problem.

C. *Feature Space Dimension Reduction*

After preprocessing, we obtain 68000 different words(max) from the training dataset, represented as features. To reduce the feature space, we tried two feature selection methods:

- **TF-IDF Vectorizer Parameter Tuning**: Using a pipeline to find the best parameter for our set, we found that using minimum document frequency as 2 and maximum document frequency as 70% gives the best result.
- $\boldsymbol{\chi^2}$ **test**: The Chi-square test, χ^2, measures how well the observed distribution of data fits with the distribution that is expected if the variables are independent [13]. We select only a set of 15000 features having the top score, using the $SelectKBest()$ implementation of the sklearn package [10].

D. *Latent Dirichlet Allocation*

Our most experimental feature vector representation was based on the topic distribution vectors inferred by Latent Dirichlet Allocation (LDA) which is a generative model. It assumes that a document is created as a mixture of a finite number of topics, and each topic has some distribution over words. Given a corpus of reddit comments and the number of topics k, LDA infers the topic mixture model and the topic-to-word distributions. Given a new reddit comment, LDA will

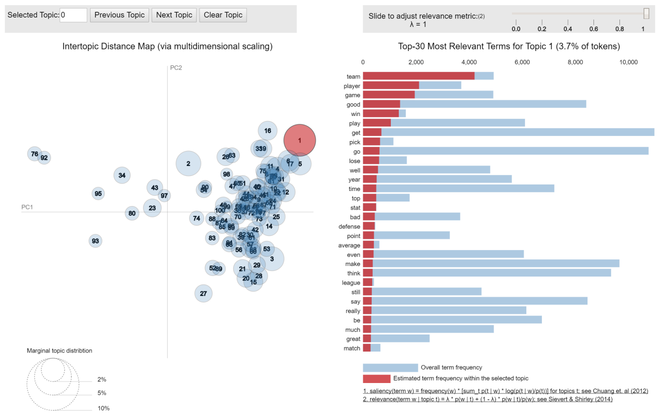
Selected Topic: 0
Previous Topic
Next Topic
Clear Topic
Slide to adjust relevance metric:(2)
λ = 1
0.0
0.2
0.4
0.6
0.8
1
Intertopic Distance Map (via multidimensional scaling)
PC1
PC2
Marginal topic distribtion
2%
5%
10%
Top-30 Most Relevant Terms for Topic 1 (3.7% of tokens)
0
2,000
4,000
6,000
8,000
10,000
team
player
game
good
win
play
get
pick
go
lose
well
year
time
top
stat
bad
defense
point
average
even
make
think
league
still
say
really
be
much
great
match
Overall term frequency
Estimated term frequency within the selected topic
1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

Fig. 2. Latent Dirichlet Allocation display

predict the topic distribution in the form of a vector whose elements sum to 1 [6].

Once these models were trained, we ran our dataset through the models again, giving us the predicted topic distribution vectors. The overall result including LDA output in our model implementation decreased our final performance. It may be because of improper tuning of our LDA training. So we decided not to use it in the final model implementation.

E. Doc2Vec Implementation

Doc2vec uses unsupervised learning approach to learn a document representation. The goal of doc2vec is to create a numeric representation of a document, regardless of its length. But unlike words, documents do not come in logical structures such as words. So, when training the word vectors W, the document vector D is trained as well, and in the end of training, it holds a numeric representation of the document.[14].

We executed the implementation of Doc2Vec with logistic regression for 300 feature space vectors and achieved 45.7% accuracy on the test set. We wanted to increase the feature space to 15000 to get better results but our systems were not able to handle such large computation.

IV. PROPOSED APPROACH

A. Algorithm/Model selection and implementation

We experimented our dataset with several classifiers from sklearn.[10]

1) Naive Bayes classifiers: Naive Bayes is a family of statistical text classification algorithms. In Naive Bayes, we assume that each feature is independent of other features, which help us to greatly simplify calculating the probabilities (in this task the probability that each comment belongs to a sub-reddit). In this project, we utilize the Multinomial(**MNB**), Bernoulli(**BNB**) and Complement(**CNB**) versions of Naive Bayes.

2) Support Vector Machine (SVM): SVM draws a "line" or hyper-plane that divides a space into two subspaces: one subspace that contains vectors that belong to a group and another subspace that contains the rest of the vectors. Those vectors are representations of training comments and a group is a tag/subreddit we would like to tag our texts with. We used (**LinearSVC**) similar to SVC but it has more flexibility in the choice of penalties and loss functions and scales better to large numbers of samples.

3) Logistic Regression and Stochastic Gradient Descent: Logistic regression(**LR**) is a classification algorithm that uses a logistic function to model a binary dependent variable [15]. Stochastic Gradient Descent(**SGD**) is an iterative method for optimizing an objective function with suitable smoothness properties (e.g. differentiable). SGD can be regarded as a stochastic approximation of gradient descent optimization, since it replaces the actual gradient (calculated from the entire data set) by an estimate (calculated from a randomly selected subset of the data). With large datasets, this reduces the computational burden, achieving faster iterations in trade for a slightly lower convergence rate. [16]

4) Ridge Classifier: The Ridge Classifier(**RC**) is an implementation in Scikit Learn package that uses the Linear least squares loss function to perform linear regression. Unlike other classification methods, this predicts the probability of each class and then assign a class instead of directly predicting the class from the inputs [17]. This classifier works on a One vs. Rest manner in multi-class datasets.

5) One Vs Rest Classifier: One-vs.-rest strategy involves training a single classifier per class, with the samples of that class as positive samples and all other samples as negatives. This strategy requires the base classifiers to produce a real-valued confidence score for its decision, rather than just a class label[18]. We use this classifier along with various Naive Bayes model implementations. (**OVRMNB, OVRCNB, OVRBNB**).

6) Ensemble Methods: Ensemble methods are meta-algorithms that combine several machine learning techniques into one predictive model in order to decrease variance (bagging), bias (boosting), or improve predictions (stacking). We used the Voting Classifier (**EM**) for our modelling. The idea is to combine conceptually different machine learning classifiers and use a majority vote (hard voting) to predict the class labels. Such a classifier is useful for a set of equally well performing models in order to balance out their individual weaknesses. [19].

B. Hyperparameter Tuning

We implemented different strategies for tuning the hyperparameters. Initially, we created a pipeline $model\_eval$ to select best parameters for TFIDFVectorizer along with χ^2 selection using Multinomial NB.

In order to maximize our performance, we tuned our hyperparameters using 5-fold cross validation. We optimized our performance by evaluating on F1 (the harmonic mean between precision and recall). We used GridSearchCV to find the best parameters for a certain classifier. The results are showed in Table. I. By tuning the parameters, we were able to find large increases in the performance of our overall system.

C. Bernoulli Naive Bayes implementation from scratch

We implemented a multi-class Bernoulli Naive Bayes from scratch using the naive assumption that all features are independent from each other. The equation that determines the probability that a data point is from class k can be written based on the marginal probability of class k and the probability of the observed features given class k, as follows:
$P(y = k|x) \propto P(y = k)P(x|y)$
Assuming all features are independent:
$P(y = k|x) \propto P(y = k) \prod_{j=1}^{m} P(x_j|y)$
We will use the notation below (these are the parameters we need to learn):
$\theta_k = P(y = k)$, and $\theta_{j,k} = P(x_j = 1|y = k)$
We take logarithm from both side of the equation for numerical stability, and after replacement we will have:
$log(P(y = k|x)) = log(\theta_k) + \sum_{j=1}^{m} x_j log(\theta_{j,k}) + (1 - x_j)log(1 - \theta_{j,k})$ For an easier implementation, we combined all terms containing x_j as one term and converted the equation to a linear decision boundary model, as follows:
$log(P(y = k|x)) \propto log(\theta_k) + \sum_{j=1}^{m} log(1 - \theta_{j,k}) + \sum_{j=1}^{m} x_j(log(\theta_{j,k}) - log(1 - \theta_{j,k}))$ Here, the terms $log(\theta_k) + \sum_{j=1}^{m} log(1 - \theta_{j,k})$ act as W_0 and $log(\theta_{j,k}) - log(1 - \theta_{j,k})$ acts as W. Therefore after finding W and W_0 we will compute $W_0 + X^T W$, which gives us $log(P(y = k|x))$ for each class k. The advantage of this method is that we can benefit from matrix multiplication which is much faster than looping over all elements. We calculated the probability of being in each of 20 subreddits for each input sample(comment), and then we chose the class with the highest probability for that comment, as its subreddit category. We also used Laplace smoothing to deal with words which are not observed in the training set but exist in the test set. To do so, we added 1 to the number of instances with $x_j = 1$ (existence of a particular word in the comment) and $y = k$ (being of a particular sub-reddit), and added 2 to the number of examples with $y = k$, which are respectively the numerator and denominator of $\theta_{j,k}$ relation. We keep θ_k relation as the number of examples where y=k to the total number of examples.

V. RESULTS

We used two different methods in our Model validation pipeline to ensure that the data doesn't overfit to either Training set or the Validation set. First, we performed K-fold cross validation with Grid search for hyperparameter tuning. Second, we retained 20% of the data as "hold-out validation set".

We analyzed the performance of various classifiers and reported the time and their accuracies on both K-fold cross validation and held-out validation in Table I. Comparing the confusion matrices and performance of various classifiers so far, we observed that certain classifiers performed better on classifying few subreddits, while others performed better on some other subreddits. We used Voting classifier and obtained much better accuracies on the validation sets as shown in the Table I.

We also compared the Naive Bayes implementation results with the rest of the classifiers as shown in the table.

The best classifier as mentioned in Table I performed the best on the Kaggle competition test dataset, and obtained test accuracy of 58.633%.

VI. DISCUSSION AND CONCLUSIONS

Various preprocessing steps including the novel approach of using χ^2-test for feature selection yielded ideal features for training our classification models. Usage of ensemble methods and OneVsRest classifiers resulted in our best reported score on Kaggle competition.

Although our classifier performs decently, there are a few improvement ideas we never had time to pursue.

One improvement which was discussed was attempting to use Latent Dirichlet Allocation as more than a tool to give us feature vectors. More concretely, we would like to use LDA to be our classifier. Unfortunately, we never found a tractable way to approximate

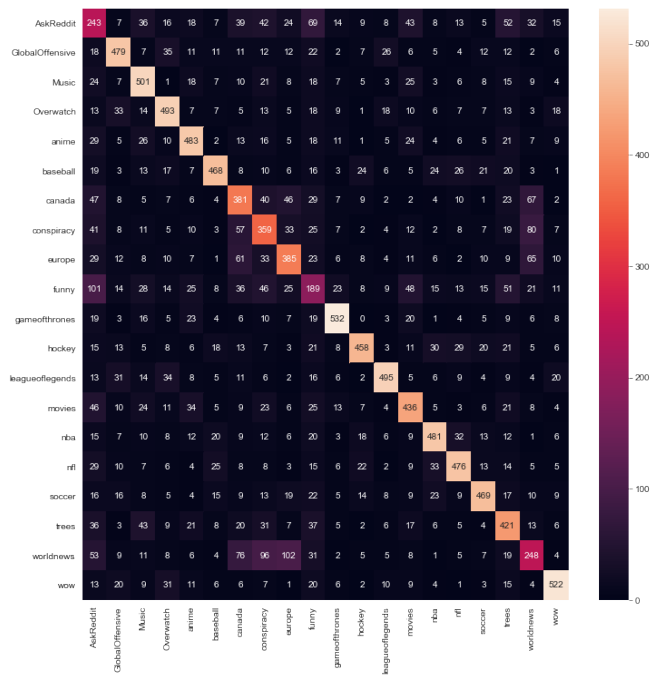

Fig. 3. Confusion Matrix for our best model

TABLE I
PERFORMANCE OF VARIOUS CLASSIFIERS

Classifier with GridSearchCV	**Run time (sec)**	**k-fold accuracy**	**Test set accuracy**
Bernoulli Naive Bayes from Scratch	4.09	55.9	56.8
MNB	3.76	59.1	59.3
CNB	2.56	59.4	60.02
BNB	2.81	57.6	57.82
OVR-MNB	1.27	60.7	59.45
OVR-CNB	4.3	60.2	60.52
OVR-BNB	2.19	59.6	59.8
LR	268	55.3	56.3
SGD	29.46	56.6	57.8
Linear SVC	86.39	57	58.4
RC	360	57.4	58.9
EM (MNB, CNB, RC)	25.23	60.3	60.94
EM (OVRMNB, OVR-CNB, OBVRBNB, RC) **Best classifier**	8.2	60.6	61.02
EM (OVRMNB, OVRCNB, OVRBNB, MNB, CNB, BNB, RC) **Proposed Best-couldn't submit**	10.45	60.8	61.25

TABLE II
PRECISON, RECALL AND F1 SCORES FOR OUR BEST CLASSIFIER (ENSEMBLE)

Subreddit	**Precision**	**Recall**	**F1**
AskReddit	0.30	0.34	0.32
baseball	0.78	0.68	0.73
Global Offensive	0.68	0.69	0.69
Overwatch	0.67	0.70	0.69
canada	0.49	0.54	0.51
gameofthrones	0.78	0.76	0.77
anime	0.66	0.69	0.67
Music	0.62	0.72	0.67
league of legends	0.78	0.71	0.74
europe	0.58	0.59	0.59
funny	0.29	0.26	0.27
conspiracy	0.43	0.51	0.47
hockey	0.75	0.64	0.69
movies	0.60	0.65	0.62
worldnews	0.41	0.36	0.38
nba	0.73	0.67	0.70
soccer	0.74	0.64	0.69
nfl	0.72	0.69	0.71
trees	0.52	0.58	0.55
wow	0.77	0.77	0.77

this, and thus never was able to use LDA in more than trying it out in a feature-space sense.

We wanted to improve the Doc2Vec implementation. Our feature space only consisted of 300 vectors giving an accuracy of 45% with logistic regression. The number of vectors could be increased for a better performance but due to system capacity and time limitation, we couldn't implement it.

For future investigations, we believe RNN (Recurrent Neural Networks), LSTM (Long Short Term Memory) networks can further improve the accuracy of our predictions, although their implementation will require significant computing power.

This project provided hands-on experience of trying out the various sklearn classifiers and improving it until it can perform well on prediction. This work has helped us understand the algorithms and their tuning more deeply. Using ensemble methods help us reduce our variance and make sure that our best model doesn't over-fit.

VII. STATEMENT OF CONTRIBUTIONS

The collaborative work of this team involved everyone contributing to the report, Shantanil and Nikhil worked on data preprocessing and analysis, various feature extraction and classifier implementation. Manoosh worked on implementing multi-class Bernoulli Naive Bayes from scratch.

References

[1] https://en.wikipedia.org/wiki/Reddit

[2] https://monkeylearn.com/text-classification/

[3] https://www.kaggle.com/c/reddit-comment-classification-comp-551/overview

[4] Hastie, Trevor, Robert Tibshirani, and J. H. Friedman. The elements of statistical learning : data mining, inference, and prediction. New York: Springer, 2001. Print.

[5] https://scikit-learn.org/stable/

[6] http://cs229.stanford.edu/proj2014/Andrew%20Giel,

[7] https://www.learndatasci.com/tutorials/predicting-reddit-news-sentiment-naive-bayes-text-classifiers/

[8] Li, Shoushan, et al. "A framework of feature selection methods for text categorization." Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2. Association for Computational Linguistics, 2009.

[9] McCallum, Andrew, and Kamal Nigam. "A comparison of event models for naive bayes text classification." AAAI-98 workshop on learning for text categorization. Vol. 752. No. 1. 1998.

[10] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

[11] Steven Bird, Ewan Klein, and Edward Loper (2009). Natural Language Processing with Python. O'Reilly Media Inc. http://nltk.org/book

[12] http://www.tfidf.com/

[13] https://www.ling.upenn.edu/ clight/chisquared.htm

[14] https://medium.com/scaleabout/a-gentle-introduction-to-doc2vec-db3e8c0cce5e Jon%20NeCamp,HussainKader,rClassifier.pdf

[15] https://en.wikipedia.org/wiki/Logistic_regression

[16] https://en.wikipedia.org/wiki/Stochastic_gradient_descent

[17] Avila, Julian, and Trent Hauck. Scikit-Learn Cookbook: over 80 Recipes for Machine Learning in Python with Scikit-Learn. Packt, 2017.

[18] Bishop, Christopher M. (2006). Pattern Recognition and Machine Learning. Springer.

[19] https://scikit-learn.org/stable/modules/ensemble.voting-classifier

[20] https://en.wikipedia.org/wiki/Mutual_information

[21] https://cs.mcgill.ca/wlh/comp551/schedule.html

[22] https://pythonmachinelearning.pro/supervised-learning-using-decision-trees-to-classify-data/