COMP 6721 Introduction to AI: Project 2

Subhannita Sarcar[1]

[1] 40059367 subhannitasarcar1994@gmail.com

1 Introduction

The problem presented in this project has found applications in various fields, most of which concern classification of a colossal number of documents or even sentiment analysis. We were presented with a dataset containing information about posts on a social news website called Hacker News [1]. The objective was to train a machine learning algorithm on the vocabulary of the post titles and the given category of the post so that it will be able to classify further posts. The input data has been preprocessed to produce a vocabulary of lemmatized words while removing special. It has been noted to not remove apostrophe or underscore punctuation symbols as they convey meaning to the word. Initially, it was attempted to observe the results by removing the words containing numbers, but further investigation of data revealed some words such '3d' that contributed to the meaning of a sample towards classification. However, further effort could be directed towards better cleaning of the data so that words that convey meaning in phrases could be considered as a single token or meaningless words according to English grammar could be removed. This would have reduced redundancy in the training set. It has been observed that the removal of redundant cases has improved the classification performance as well as reduced the training time proportionally [2]. The data model consists of the smoothed conditional probability of each word in the processed vocabulary which is further used to deploy a multinomial Naive Bayes classifier [3] that fits the data. A study shows that among the naive Bayes assumption techniques, the multinomial model usually performs better than a multivariate Bernoulli model over all vocabulary sizes and results in a 27% reduction in error [4]. This is because the unigram language model takes into account the integral count of every word in the vocabulary to determine the probabilistic score of a document with respect to a class. The testing data is made to go through the same procedure of preprocessing, after which it is classified by the fitted classifier. The results show a high accuracy in estimation. This was followed by an investigation of the linguistic data properties such as

1. removing words that appear more or less frequently than set boundaries,
2. excluding words that exceed a limit in length,
3. filtration of words that must be excluded from impacting the classifier (stop-words)
4. variation in smoothing property of the naive Bayes classifier

The programming language used for this purpose is Python 3.7.3 and the environment is Jupiter notebook 6.0.1. The major libraries used are NLTK(Natural Language Toolkit) [5] and Scikit-learn [6].

2 Experimental Results and Comparisons

2.1 Baseline Experiment

Experimental Setup. The first experiment was conducted with the entirety of the processed vocabulary set consisting of n = 75751 words and a smoothing δ = 0.5. A total of 2 training samples were removed since they ended up being an empty string after preprocessing. The percentage of training samples and testing samples is 67% and 33% respectively. It was enforced that posts from the year of 2018 should be considered in the training set whereas those from the year of 2019 should be considered in the testing set. The training set is transformed such that the data is represented as word vector counts. Each vector $\theta_y = (\theta_{y1}, \ldots., \theta_{yn})$ for n features or words in the vocabulary and yth sample. Each element θ_{yi} is the conditional probability of the ith word when the class of the yth training sample is given, i.e., $P(x_i|y)$. Similarly, the testing dataset is preprocessed to produce the testing corpus. The predictions made by the fitted Multinomial Naive Bayes has an overall good performance in the metrics as shown in Table 1 and Fig 1.

Table 1. Performance of classifier in baseline experiment

Metrics	Class 'story'	Class 'ask_hn'	Class 'show_hn'	Class 'poll'
Accuracy		0.9786		
Precision	0.9988	0.8180	0.7472	0.0
Recall	0.9789	0.9925	0.9555	0.0
F1 score	0.9888	0.8969	0.8386	0.0
Cohen's Kappa score		0.8652		

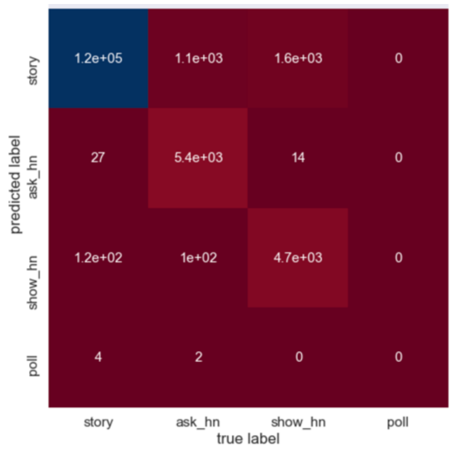


Fig. 1. Confusion matrix for baseline experiment.

Results and discussion. It must be noted that the poor performance for class 'poll' could be the result of extremely low number of training samples for this class, thereby resulting in a classifier that has more tendency towards determining a sample as the other classes about which the model is more "learned" of. The dataset is clearly imbalanced with 91.78% of training samples bellowing to the class of 'story'. This imbalance in the dataset is causing problems since the smoothing parameter or Laplace correction, which accounts for irrelevant words, becomes responsible for the considerable increase in estimated probability of a test sample belonging to the over weighted class [7]. There are several alternatives suggested to tackle the problem of imbalanced datasets. One such option would be to select an optimal combination of positive and negative features, i.e., features that are most suggestive of membership and those that are devoid of the most characteristic features of a class [8]. A second alternative would be to change the performance metric that would reveal more about the results concerning imbalanced datasets. Cohen's Kappa [9] statistic has been found to work well with imbalanced classes since it computes the accuracy of classification by normalizing with the imbalance in classes in the training data. The Cohen's Kappa score for the baseline experiment is found to be 0.8651 which indicates that there is roughly 86% agreement between the predictions and actual labels. Another technique to handle imbalance in datasets is to generate synthetic samples using one of the popular algorithms – SMOTE (Synthetic Minority Oversampling Technique) [10]. This method creates synthetic samples by introducing attributes to existing samples of the minority class such that the distance measure of the synthetic sample from the existing sample does not cross a specified threshold.

2.2 Stop-word Filtering Experiment

The experiment has been repeated with a single modification – certain specified words are removed from the training corpus so that the classifier is trained to ignore the effect of these stop-words. The intention is to remove words that should be considered irrelevant to text classification. Consequently, it was made sure that the most characteristic words in each class was not included in the list of stop-words. The stop-words themselves are lemmatized to maintain consistency with the training corpus. The result is increased quality of the training samples which further increases the overall performance of the classifier. As shown in Fig 2, there is marked improvement for 'ask_hn' and 'show_hn' classes as there is an increase in the difference between correct and incorrect labels by 12.29% and 15.69% respectively. However, the imbalance in datasets continue to affect the performance of the classifier for 'poll' class.

Table 2. Performance of classifier in stop-word experiment

Metrics	Class 'story'	Class 'ask_hn'	Class 'show_hn'	Class 'poll'
Accuracy		0.9767		
Precision	0.9990	0.8960	0.8162	0.0
Recall	0.9871	0.9954	0.9680	0.0
F1 score	0.9930	0.9431	0.8856	0.0
Cohen's Kappa score		0.9128		

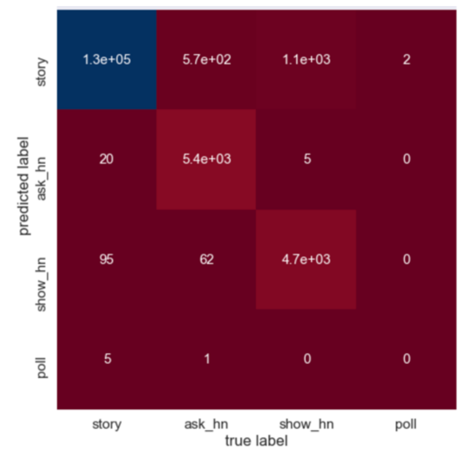


Fig. 2. Confusion matrix for stop-word experiment.

2.3 Word-length Filtering Experiment

In this case, the stop-words is a new set of words that exceed the length boundary of (2,9). The length boundary is imposed on unlemmatized words and both the training and testing corpus are made to go through the same restriction. In consequence, a total of 1632 and 837 samples are dropped from the training and testing dataset. The result overall accuracy drops than the baseline experiment. A comparison of Fig 1 and Fig 3 shows that the number of incorrect labels for 'story', 'ask_hn' and 'show_hn' classes have increased whereas the number of correct labels for the same classes have either remained same or decreased in this experiment. Clearly, filtration of the corpus according to the number of characters in the word is a poor feature to be considered for selection. This could either be due to the reduction in training sample size or removal of words that were integral to classification.

Table 3. Performance of classifier in word-length experiment

Metrics	Class 'story'	Class 'ask_hn'	Class 'show_hn'	Class 'poll'
Accuracy		0.9693		
Precision	0.9893	0.8072	0.7084	0.0
Recall	0.9776	0.9171	0.8156	0.0
F1 score	0.9834	0.8586	0.7582	0.0
Cohen's Kappa score		0.7996		

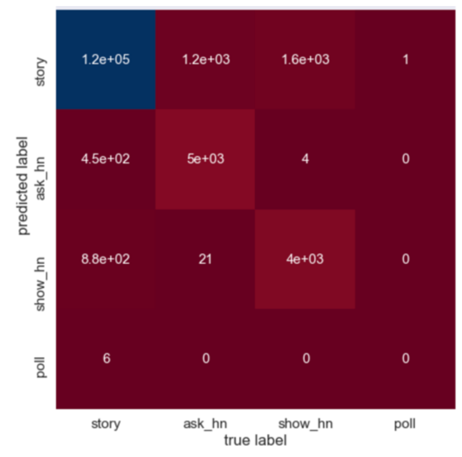


Fig. 3. Confusion matrix for word-length experiment.

2.4 Word-frequency Filtering Experiment

The effect of training corpus size on classification performance has been explored by filtering words based on their frequency in the dataset. The dataset has been filtered in steps of frequency = 1, ≤ 5, ≤ 10, ≤ 15, ≤ 20, top 5%, top 10%, top 15%, top 20% and top 25%. There was considerable debate on how to calculate the average metrics since this is multi class classification and not binary. A macro average of the metrics score was taken, i.e., the metrics for each label is calculated and their average was displayed. The results as shown in Fig 4a does not show any marked difference in the performance with varying training size. This is due to the continued poor performance of class 'poll' that has a consisted contribution towards lowering the average score. Intuitively, a better alternative would be to take the weighted average as it has been shown previously that the experiments are conducted on an imbalanced dataset. However, taking the weighted mean results in equal accuracy and recall metrics as shown in Fig 4b. This means that the proportion of correct predictions out of the actual labels of a class multiplied by its weight is equal to the overall accuracy. The best approach would be to observe the metrics for each class as shown in Figures 5a, 5b, 5c and 5d. It can be seen from these figures that for class 'story' precision is higher than recall whereas it is the reverse case for classes 'show_hn' and 'ask_hn'. This can be attributed to the fact that the number of correct labels as compared to the total number of samples labelled as 'story' is highest. For classes 'show_hn' and 'ask_hn', the number of true positives as compared to the actual labels is higher than when compared to the total number of predictions of that class. In general, there is an increase in performance with the increase in training size, except for the 'poll' class where dataset imbalance shows no difference in metrics with varying training set size.

It must be noted that the initial decrease in performance with increase in training set may be due to the addition of relatively irrelevant words to the vocabulary that increase incorrect labeling as these words may occur with equal probability in all classes.

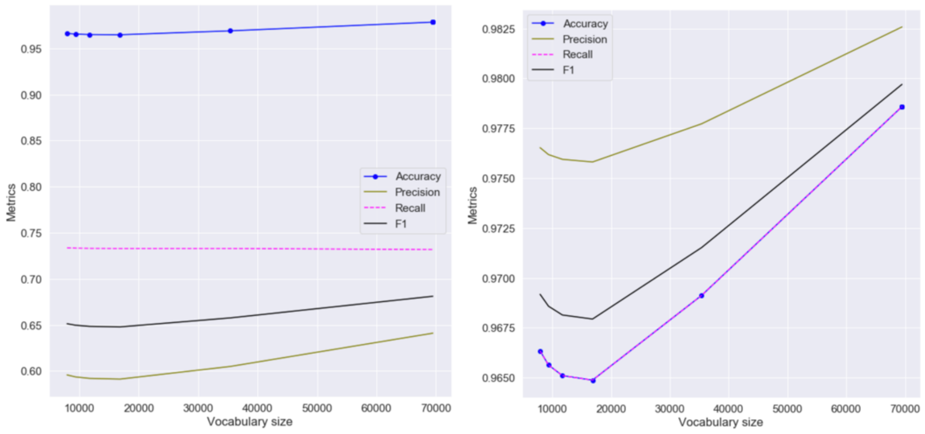
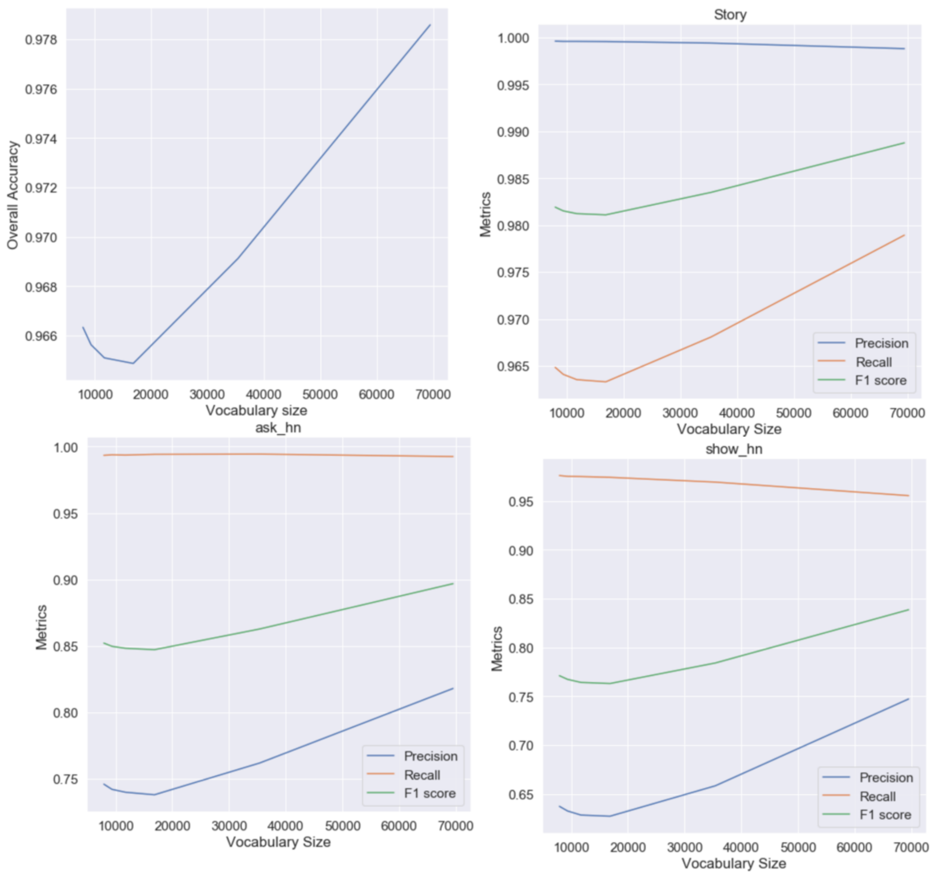


Fig. 4. left-to-right (a)Performance of classifier by "macro" averaging metrics scores across classes (b)Performance of classifier by "weighted" averaging metrics scores across classes.



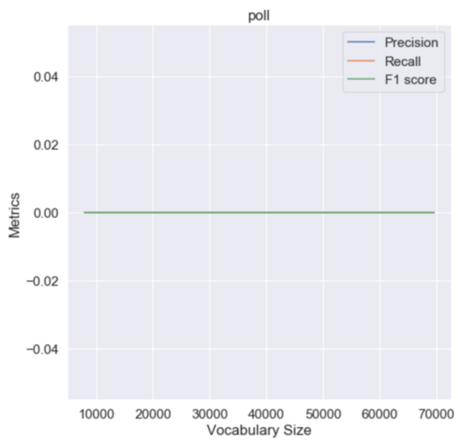
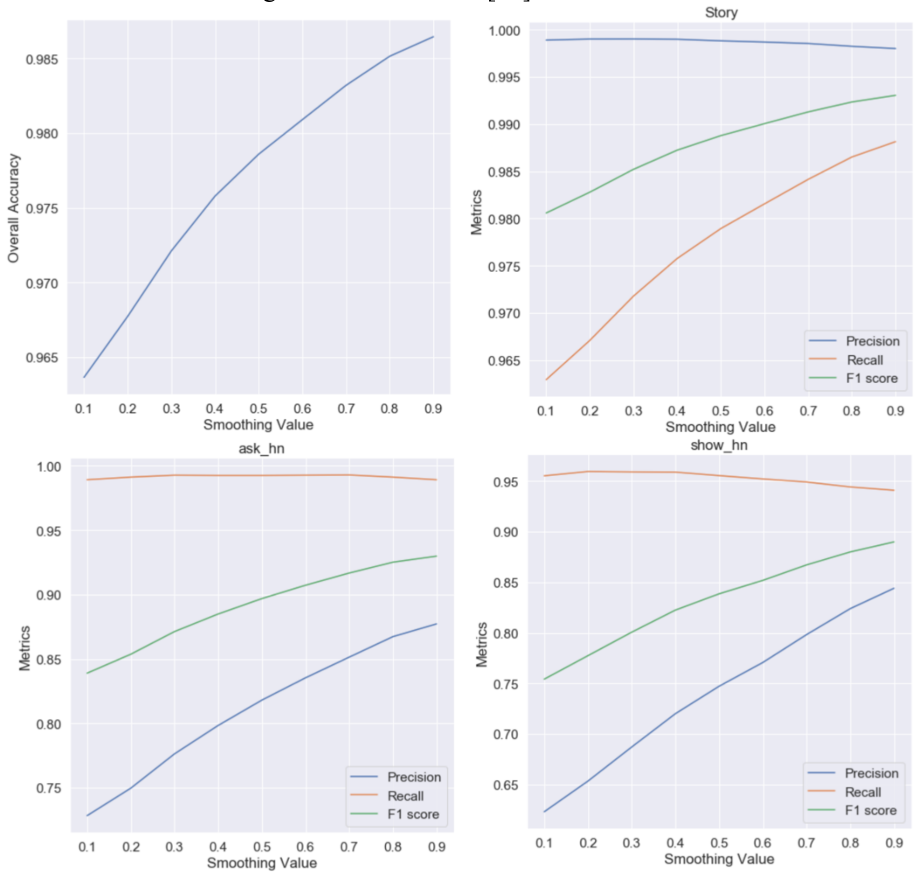


Fig 5. clockwise from top-left (a) Overall accuracy for all classes in word length experiment (b) Metrics score for class 'story' (c) Metrics score for class 'ask_hn' (d) Metrics score for class 'show_hn' (e) Metrics score for class 'poll'

2.4 Smoothing Experiment

The purpose of introducing a Laplace smoothing parameter in the multinomial model is to counter the dominant effect of a rare word occurrence in the testing sample. This experiment is conducted to observe the performance of classifier with variation in δ from 0.1 to 10 in steps of 0.1. Keeping δ = 0 results in an error as the logarithmic conditional probability of nonexistent words tends to -∝ and is therefore avoided. As shown in Figures 6a, 6b, 6c and 6d, the performance increases with the increase in smoothing with the exception of class 'poll'. This could be attributed to the fact that adding a prior pseudo-count makes the decision more conservative in its labeling of positive classes. However, Laplace smoothing could decrease classifier performance if the feature set or training size was increased [11].



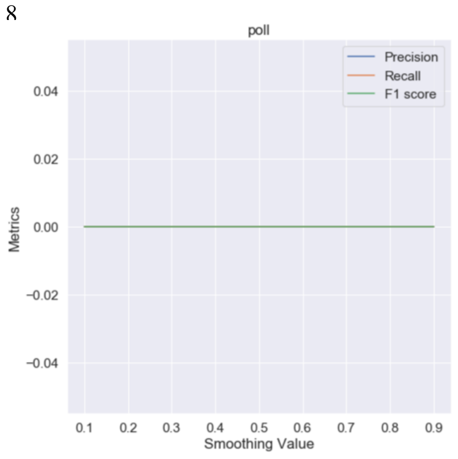


Fig 6. clockwise from top-left (a) Overall accuracy for all classes in smoothing experiment (b) Metrics score for class 'story' (c) Metrics score for class 'ask_hn' (d) Metrics score for class 'show_hn' (e) Metrics score for class 'poll'

3 Conclusion and Future scope

It must be noted that the results are concluded from a single run of the tests as opposed to the conventional method of repeating the same test 10 to 20 times to arrive at an average conclusion. However, averaging technique would have more significance if there was more variation in the results and a considerable deviation from the expected pattern was observed, which is not the case here.

An interesting approach to further explore the project would be to introduce techniques (discussed previously in section 2.1) to handle imbalance in dataset. Also, a more exhaustive and meaningful list of stop-words could be constructed through linguistic research on the dataset, such as words that have almost same probability for all classes. This probability should be weighted with the number of samples belonging to the class. A different classification could also be implemented to compare the results with the multinomial model used here. Such a classification technique could use word vectors to measure the similarity between words and whether that contributed to classification. The term frequency could also be incorporated in the word space to calculate the weighted contribution during classification of a string of words [12]. An alternative approach to Laplacian smoothing also offers increased performance in classification as adding a pseudo-count may account for probabilities for unseen words but does not redistribute the probability mass over the set of possible words [13]. Instead an absolute discounting method [14] would assign a probability mass to ever seen word which is further redistributed throughout the vocabulary using a uniform or unigram distribution.

References

[1] P. Grahan, Y Combinator, 19 02 2007. [Online]. Available: https://news.ycombinator.com/. [Accessed 24 11 2019].

[2] H. S. F. A. O. L. Ohno-Machado, "Improving machine learning performance by removing redundant cases in medical data sets.," *Proc AMIA Symp,* vol. PMC2232167, pp. 523-528, 1998.

[3] E. F. B. P. G. H. Ashraf M. Kibriya, "Multinomial Naive Bayes for Text Categorization Revisited," in *Australasian Joint Conference on Artificial Intelligence*, Cairns, 2004.

[4] A. &. N. K. Mccallum, "A Comparison of Event Models for Naive Bayes Text Classification," in *AAAI*, 1998.

[5] S. Bird, "NLTK: the natural language toolkit.," in *Proceedings of the COLING/ACL on Interactive presentation sessions, Association for Computational Linguistics*, 2006.

[6] G. V. A. G. V. M. B. T. O. G. M. B. A. M. J. N. G. L. P. P. R. W. V. D. J. V. A. P. D. Fabian Pedregosa, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* no. 12, p. 2825–2830, 2011.

[7] R. B. Eibe FrankRemco, "Naive Bayes for Text Classification with Unbalanced Classes," *Lecture Notes in Computer Science,* vol. 4213, 2006.

[8] X. W. R. K. S. Zhaohui Zheng, "Feature selection for text categorization on imbalanced data," *SIGKDD Explorations,* vol. 6, pp. 80-89, 2004.

[9] S. Kampakis, "Performance Measures: Cohen's Kappa statistic," 8 May 2016. [Online]. Available: https://thedatascientist.com/performance-measures-cohens-kappa-statistic/. [Accessed 17 11 2019].

[10] K. W. B. L. O. H. W. P. K. N. V. Chawla, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research,* vol. 16, pp. 321-357, 2002.

[11] F. H. Ding, "Improving Naive Bayes Text Classifier Using Smoothing Methods," in *ECIR 2007: Advances in Information Retrieval*, Rome, 2003.

[12] L. S. S. B. B. G. K. F. Erica K. Shimomoto, "Text Classification based on Word Subspace with Term-Frequency," in *IJCNN 2018 : International Joint Conference on Neural Networks*, Rio, 2018.

[13] M. W. H. NeyS, "Statistical language modeling using leaving-one-out.," *Corpus-Based Methods in Language and Speech Processing, Text, Speech and Language Technology,* vol. 2, pp. 174-207, 1997.

[14] H. N. Alfons Juan, "Reversing and Smoothing the Multinomial Naive Bayes Text Classifer," in *Proceedings of the 2nd Int. Workshop on Pattern Recognition in Information Systems*, Ciudad Real, 2002.

[15] G. V. A. G. V. M. B. T. O. G. M. B. A. M. J. N. G. L. P. P. R. W. V. D. J. V. A. P. D. Fabian Pedregosa, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* no. 12, p. 2825–2830, 2011.