COMP 6721 Project: Natural Language Processing

Naitik Bhise Paras Kapoor

November 25, 2019

1 Introduction and Technical Details

Natural Language Processing forms an integral part of Artificial Intelligence, dealing with the analysis of words and their generation. Our project in Natural Language Processing was to analyze a Kaggle Dataset of Hacker News by modelling the dependency of the tweet posts on their types. It was completed as a part of project 2 of the course COMP6731 Introduction to Artificial Intelligence.

The Hacker news Data is a compilation of the news posts from the years 2018 and 2019 with each classified according to four types namely story, poll, show_hn and ask_hn. We implemented a Naive Bayes model of classification for understanding the structure of the relationship between the classes and their posts. The data is bifurcated into training dataset of 2018 and the testing dataset of 2019.

Data preprocessing involved reading dataframe from csv, training bigram and unigram part of speech tagger from various standard datasets in nltk library and then tokenization, lemmatization and part of speech tagging of words using the NLTK library. The libraries used frequently during the project were the NLTK library of python for Natural Language Processing and pandas for Data Operations.

The python libraries employed during the project are NLT-K, pandas, math, matplotlib and Scikit-learn. The report starts with the Methods section involving the application of the python libraries along with the methods used for the construction of the classifier. Once we construct the classifier, our classifier performs on the testing data. It is followed by the small result sections for tasks constructed in order to provide a better idea of the dataset and its functioning. The results sections are divided into the tasks of Stopword-filtering, Word-length filtering, Infrequent word filtering, frequency words and Delta smoothening. The conclusions section gives an excerpt of the analysis of the 5 experiments and is aimed to achieve a better insight into the dataset. Lastly, we conclude with supposed future work which could be inspired from this whole project along with the tough decisions faced during the project.

2 Baseline Experiment

The titles are divided based on their year. 2018 data is set as training set and 2019 data is set as testing set. Data preprocessing invloves converting Titles into word-arrays individually by tokenizing sentences, part of speech tagging and lemmatizing out by nltk functions. Lemmatize causes a word to retain its basic form according to its part of speech. We trained our own part of speech tagger instead of using the default one offered by the library. We trained a unigram and a bigram tagger with backoff tagger as Noun. Taggers were trained on a combination of Brown, Treebank, Colln2000 and Colln2002 tagged datasets. Adjacent words with noun as part of speech are combined to a single token.

Once we have a final set of tokenized and lemmatized word set, we construct the Naive Bayes classifier by calculating the frequencies of the words in the dataframes with their occurences in 4 classes. Then the conditional probabilities are constructed by using the frequencies of the words and the Smoothing parameter according to the equation.

$$P(word|PostType) = \frac{Word\_Count + \delta}{Total\_Words\_Per\_PostType + \delta(Vocabulary)}$$

In the expression above, the δ represents the smooothing parameter which controls the probability occurrence of the words absent in the class but could be introduced to it as the given data isn't exhaustive for the classes. The presence of the classes in the 2018 data were measured and it concluded the presence of four classes with story class getting the majority of attention. It is concluded that the data is an imbalanced dataset.

Identical preprocessing techniques are applied on Testing set (year 2019) to generate. We checked the probability of occurrence of the words in the set to calculate the sentence probability to occur in the 4 classes by using the generated test tokens. The class probability with the maximum value is the class assigned to the given sentence. If any word in the testing set is not covered in the training set of words, we ignore it.

After calculation of the word probabilities, we measure the performance of the classifier by calculating the accuracy of the given responses of the testing set. The accuracy for the baseline experiment is situated at 98.83%. We measure the Precision , Recall and the F1- score of individual class.

$$Precision = \frac{tp}{tp + fp}$$

$$Recall = \frac{tp}{tp + fn}$$

$$F1Score = \frac{2 * precision * recall}{precision + recall}$$

	pred_story	pred_ask_hn	pred_show_hn	pred_poll
true_story	125664	601	587	0
true_ask_hn	90	5358	6	0
true_show_hn	219	91	4593	0
true_poll	4	2	0	0

Table 1: Confusion matrix: Baseline Experiment

	precision	recall	F1score
story	0.997515	0.990635	0.994063
ask_hn	0.885327	0.982398	0.931340
show_hn	0.885654	0.936773	0.910497
poll	0.000000	0.000000	0.000000

Table 2: Precision-Recall : Baseline Experiment

2.1 Remove Words

During tokenization of words using nltk pos tagger we visualized some unwanted tokens. After, removed those tokens the accuracy on our test set boosted. Certain of those tokens are : ”, !, ”, etc. These tokens didn't add any meaning to the vocabulary.

3 Results

3.1 Stop-word Filtering

Stopwords are the most common words which generally appear in various sentences. The problem statement provided us with a list of stopwords to be filtered from the vocabulary. We remove the stopwords from both the training and the testing sets in the question and checked the accuracy of the model. The accuracy for the baseline experiment is situated at 99.26%. The gain in accuracy by removal of stopwords proved to be beneficial.

	pred_story	pred_ask_hn	pred_show_hn	pred_poll
true_story	126104	324	424	0
true_ask_hn	56	5396	2	0
true_show_hn	143	48	4712	0
true_poll	5	1	0	0

Table 3: Confusion Matrix : Stopwords Filtering

	precision	recall	F1score
story	0.998385	0.994103	0.996240
ask_hn	0.935344	0.989366	0.961597
show_hn	0.917088	0.961044	0.938552
poll	0.000000	0.000000	0.000000

Table 4: Precision-Recall : Stopwords Filtering Experiment

3.2 Word Length Filtering

In this task, we filtered tokens with (length¡= 2 and length¿=9) . The same procedure is followed in construction of the Naive Bayes Classifier and the accuracy is measured. We observe a decrease in accuracy of the classifier(97.55%) with respect to the baseline accuracy in this task. In the case of the precision and recall table, we observe a decrease in precision as well as recall leading to the solution that our hypothesis about word lengths between 2 and 9 might not be accurate. The poll class continues to be a class ridden with the misery of not accruing any precision or recall even by this method.

	pred_story	pred_ask_hn	pred_show_hn	pred_poll
true_story	124941	891	1020	0
true_ask_hn	474	4979	1	0
true_show_hn	947	11	3945	0
true_poll	6	0	0	0

Table 5: Confusion Matrix : Word-length Filtering

	precision	recall	F1score
story	0.988708	0.984935	0.986818
ask_hn	0.846625	0.912908	0.878518
show_hn	0.794402	0.804609	0.799473
poll	0.000000	0.000000	0.000000

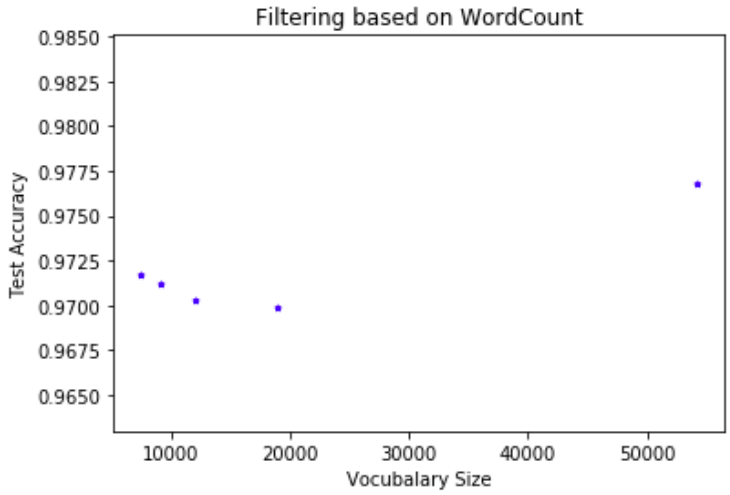
Table 6: Precision-Recall : Word-length Filtering

3.3 Infrequent Word Filtering

3.3.1 Count Based

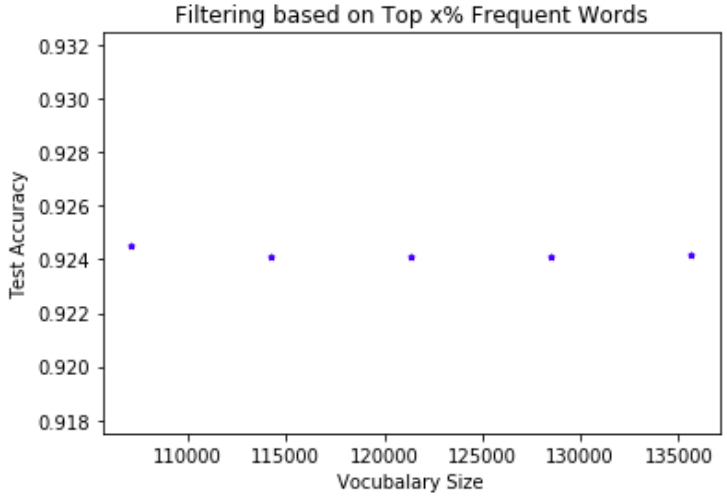
In this experiments tokens were filtered based on their count in the vocabulary. Specific count thresholds were chosen from as list [1,5,10,15,20]. As we see with the accuracy, we see that it changes with the reduction of words with frequency 1, we see the number of total words reduced to 54180 and accuracy reduced to 97.68%. The accuracy goes to a dip with lowest accuracy of 96.98% coming

with a word count of 18901 for the frequency of length 6 or more. The accuracy increases slightly to 97.16% for words with frequency of length 21 or more.



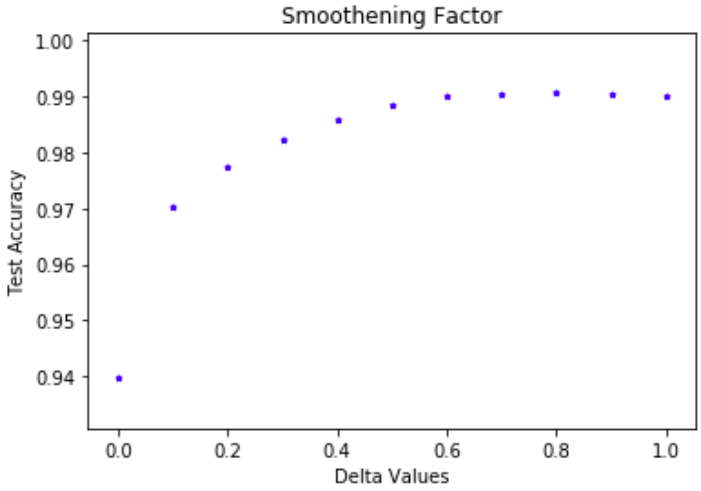
3.3.2 Frequency based

In this experiments tokens were filtered based on top percentile values. We removed the top 5%, 10%, 15%, 20%, 25% most frequent words from the vocabulary and measured the performance on test set. In the plot , we can observe a different pattern with the accuracy of the classifier. The classifier accuracy takes a big drop to the accuracy of 92.4% with the removal of top 5% most frequent words. The accuracy of the classifier plateaus thereafter.



3.4 Smoothing

We study the effects of the smoothing parameter on the classifier performance. We tweak the smoothing parameter δ from 0 to 1 with steps of 0.1 to observe the change in the accuracy. Smoothing parameter is the most important parameter in regards with handling of the new words introduced to the vocabulary by assigning it a small probability. Starting from δ at 0 to 0.8, the accuracy increases monotonically reaching its maximum values at approximately 99%. The accuracy of the classifier plateaus thereafter.



4 Comparison of all experiments

The Hacker Dataset was subjected to many experimented in order to better understand the classification of the tweets and their relation with their types with the help of Naive Bayes classification. We constructed the model by tokenizing the sentences followed by applying a trained position-tagger or a figure speech tagger. The tokens were then applied noun phrase grammar rules and were integrated into specialized tokens leading to final processed tokens after POS Lemmatization. We would like to explain the inferences about the experiments in an orderly fashion to analyze the dataset.

In the Baseline result, we observe that we have a good accuracy but mostly is owing to the fact that the dataset has a highly imbalanced classification with most titles of story class. We saw that the class poll, which was the least represented with merely 6 titles, had most of its titles distributed to ask_hn and story leaving it with absolutely no precision and recall. The class with the best F1 score was the story followed by ask_hn and show_hn with the conclusion of the score being proportional to the amount of vocabulary it brings along to the dataset.

The Stopwords[1] filtering causes a substantial rise in the accuracy of the

baseline accuracy of the model and it also has a magnifying effect on the precision and recall of all the classes. We can conclude that filtering very common words from the vocabulary removes bias from the classifier.

The word length filtering caused a slight decrease in the accuracy but the tremors were prominent in the precision-recall table which saw a slump in all the values. We can conclude that the word length of 2 to 9 maybe is not a significant filter to consider as there maybe other words of length 1 or more than 10 words which could be significant for classification.

In concern to the words happening just once, we conclude that they are a significant chunk of dataset as they are contributing to the maximum amount of words to the vocabulary(about 60%) and also elevate the accuracy in their presence. We see that 85% of the words in the vocabulary are concentrated in the low frequency zone of 0 to 5. The accuracy graph improves in the later part where we remove the words with frequency less than 20 which are the most words which are about 5.05% of the total vocabulary.

On the other hand , we remove the words in the higher frequency zone and it drops the accuracy to the ultimate minimum of 92.4% which is supposed to being closed to every sentence being classified as story.

The smoothing parameter[2] causes a direct effect on the accuracy with a direct proportionality of accuracy until we reach the maxima at $\delta = 0.8$ where we have the accuracy closer to 99% after which it plateaus. We can conclude that the smoothing value should vary between 0.1 to 1 depending on the dataset.

We can conclude that we could construct the best model with a certain modifications to the dataset by using stopwords which raises the accuracy along with the tuning of δ to higher values around 0.7 to 0.8 where it reached the peak value. We have observed that we need more balancing of data for better recall of over all the classes.

5 Future work

One new technique which could be applied is the application of word embeddings. Given the vocabulary is vast , we could utilize word relations to create word vectors. The words vectors could be fed directly to a Deep Learning network to produce the class labels.

The class imbalance problem can be tackled by capturing more titles of minority classes. We can also balance the classes by over-sampling. Also, various ensemble techniques can be applied to training which improves the recall of minority class.

The grammar exploration also rests as a possible path of approach to the problem as constructing the verb and noun phrases of any length which could be fed to the Naive Bayes classifier producing a trained model. The accuracy or the efficiency can't be speculated as combining words will lead to possibly more vocabulary but possibly finding similar phrases or some having the same sentiment might cast an effect of the performance.

Also, we can think about the dependency of the smoothing parameter and

checking other datasets to check whether change in smoothing parameters causes an impact on the precisions and recall of the classes. We can study the properties of the under-represented classes by isolating them and applying a proper filter on the types of words being considered in the vocabulary.

References

[1] A. Schofield, M. Magnusson, and D. Mimno, "Pulling out the stops: Rethinking stopword removal for topic models," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, (Valencia, Spain), pp. 432–436, Association for Computational Linguistics, Apr. 2017.

[2] Q. Yuan, G. Cong, and N. M. Thalmann, "Enhancing naive bayes with various smoothing methods for short text classification," in *Proceedings of the 21st International Conference on World Wide Web*, pp. 645–646, ACM, 2012.