COMP 6721: INTRODUCTION TO AI, FALL 2019

Project 2: Hacker News Classification

Submitted by:

Shreya Biswas	40018190
Dinesh Kumar Sankuri	40041026

Introduction and Technical Details

In this project we are designing a Naive Bayes Classifier to classify the posts of Hacker News dataset. In this project we are given a dataset from Kaggle which contains posts from year 2018 and 2019. Our task is to classify the posts of 2019 into their likely class. Our training data consists of posts from 2018, and out test data consists of posts from 2019. We model our Naïve Bayes Classifier on the training data set and then classify the test data based on our model.

Baseline Model

The baseline model, is a probabilistic model of our training set with a smoothing value of 0.5.We first generate a vocabulary by tokenizing and lemmatizing the words in the "tTitle" column of our trating dataset. We then find the conditional probability of each word in out vocabulary given each "Post Type". Once we generate our baseline model, we build Naive Bayes Classifier and used log10 space to avoid arithmetic underflow. The results of the classification on testing data from 2019 are shown below:

	Precision	Recall	F1-Score	Support
story	0.93	1.00	0.96	126852
Ask-hn	0.99	0.12	0.22	5454
Show-hn	0.99	0.05	0.10	4903
poll	0.00	0.00	0.00	6
Accuracy			0.93	137215
Macro Avg	0.73	0.29	0.32	137215
Weighted Avg	0.94	0.93	0.90	137215

The overall accuracy of the Naïve Bayes Classifier is 0.93. In the classification report the metrics for poll is 0. We believe, this is due to the fact that there is no "poll" type post in the testing data set. For the classes ask_hn and show_hn we have high precision and low recall. This implies that we have missed a lot of positive posts. The precision and accuracy are good for the class story.

Results and Analysis of Stop Word Filtering

	Precision	Recall	F1-Score	Support
story	0.96	1.00	0.98	126852

Ask-hn	0.97	0.63	0.77	5454
Show-hn	0.98	0.19	0.32	4903
poll	0.00	0.00	0.00	6

Accuracy			0.96	137215
Macro Avg	0.73	0.46	0.52	137215
Weighted Avg	0.96	0.96	0.94	137215

The overall accuracy of the model increased from 0.93 to 0.96 when we removed the stop words. The precision of the story increased and the recall remained the same. This obviously implies that F1-Score increases too. The interesting thing about the experiment is for the class ask_hn the recall increased considerably from the baseline experiment but the precision dropped a little. The model performed better in terms of recall for class show_hn but still it is really low.

Results and analysis of the word-length filtering experiment

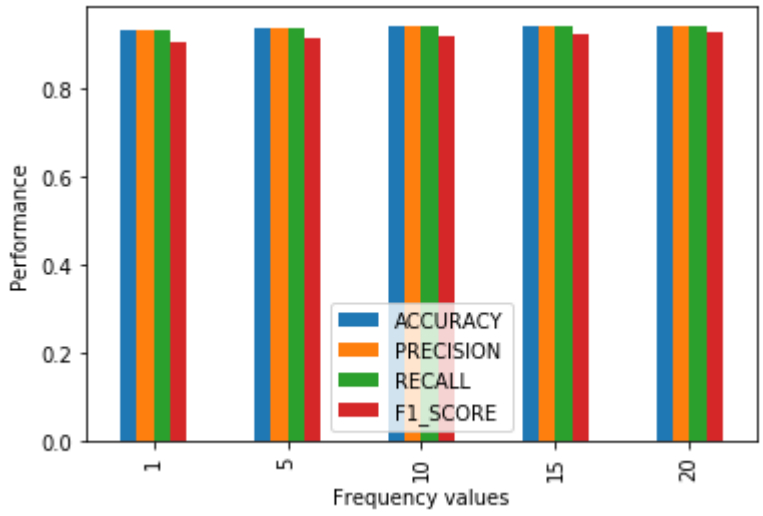
	Precision	Recall	F1-Score	Support
story	0.93	1.00	0.96	126852
Ask-hn	0.70	0.17	0.27	5454
Show-hn	0.90	0.09	0.16	4903
poll	0.00	0.00	0.00	6

Accuracy			0.93	137215
Macro Avg	0.63	0.31	0.35	137215
Weighted Avg	0.92	0.93	0.91	137215

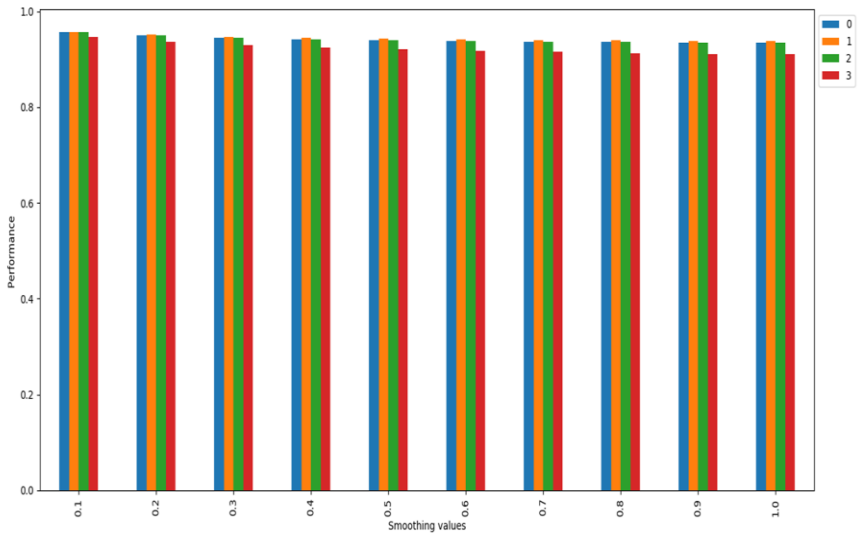
The overall accuracy of the model from baseline experiment remained same at 0.93 when we removed the words which are less than 2 and more than 9. In this model the recall is less and precision is high for class show_hn this implies that we missed a lot of posts of this class but what we predicted as show_hn class are indeed show_hn. The precision is not high for class ask_hn and recall is also low.

Results and analysis of Infrequent word filtering experiment:

	Precision	Recall	F1-Score	Accuracy
Frequency<=1	0.935	0.931	0.904	0.931
Frequency<=5	0.939	0.936	0. 914	0.936
Frequency<=10	0.942	0.94	0.92	0.94
Frequency<=15	0.942	0.941	0.924	0.941
Frequency<=20	0.941	0.942	0.926	0.942



Smoothing experiment



When we set the smoothing value to 0 the probability of the word which is not in the vocabulary will be log(0) which is not defined. So, when we are calculating the probability of sentence this will be not defined. So that the reason we started from smoothing value 0.1. The blue plot represents the overall accuracy. The accuracy varies a little when we change the smoothing value but the overall accuracy is almost the same. Precision(yellow), Recall(green) and F1-Score varies a little when we change the smoothing value

{'ACCURACY': [0.956, 0.95, 0.945, 0.942, 0.94, 0.938, 0.937, 0.936, 0.935, 0.934], 'PRECISION': [0.957, 0.951, 0.947, 0.945, 0.943, 0.941, 0.94, 0.939, 0.938, 0.938], 'RECALL': [0.956, 0.95, 0.945, 0.942, 0.94, 0.938, 0.937, 0.936, 0.935, 0.934], 'F1_SCORE': [0.946, 0.937, 0.93, 0.925, 0.921, 0.918, 0.915, 0.913, 0.911, 0.91]}

Comparison of four experiments

	Accuracy	F1_Score(weighted)
Baseline Experiment	0.93	0.90
Stop Word Filtering	0.96	0.94
Removing Infrequent Words	0.93	0.91
Frequency	0.93	0.92
Smoothing (Best at 0.1)	0.96	0.96

The table above represents the overall accuracy and F1_Score of all the experiments. It looks like best model is when we set the smoothing value to be 0.1.
From the baseline experiment if we remove the stop words the accuracy and the F1_Score increased a little. When we removed the infrequent words from the baseline experiment the overall accuracy remained same but the F1_Score increased.

Future Work

- If we were to continue working on the project, we would spend time to optimize the vocabulary so that we get better results for our classifier.
- We would run the data set against different models like decision trees, LSTM, SVM and see the performance.

References

- https://github.com/myh1234567/comp-6721-project2
- https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
- https://stackoverflow.com/questions/11348183/pandas-bar-plot-with-specific-colors-and-legend-location
- http://stackoverflow.com
- https://www.geeksforgeeks.org/confusion-matrix-machine-learning/