COMP 6721
Introduction to Artificial Intelligence

Hacker News Data Analysis

Project-2 Report
Submitted on 25th November, 2019

Submitted by:

Student Id	Name	Email
40082906	Harsh Deep Kour	harshdeep.kour93@yahoo.com
40082312	Iknoor Singh Arora	iknoor438@gmail.com

Table of Contents

1. Introduction 3

1.1 Technical Details 3

2. Task 1:Details and Vocabulary Analysis 3

2.1 Task 2: Testing using Naïve Bayes Classifier 5

2.3 Task 3: Experimenting with the Classifier 5

3. Comparison of the results from all experiments 9

4. Future Work 10

5. RESPONSIBILITIES & CONTRIBUTIONS 10

6. REFERENCES 11

1. Introduction

The Project focuses on analysis of dataset of Hacker News fetched form Kaggle. Hacker News is a popular technology site, where user submits stories (known as "posts") which are voted and commented upon.

The project is divided into 3 tasks:-

- Task 1: Extract the data and build the model
- Task 2: Use ML Classifier to test dataset
- Task 3: Experiments with the classifier

1.1 Technical Details

IDE :- Jupyter notebooks

Language :-Python(version 3.7)

Libraries used:

- **Nltk**:- It is a suite of libraries and programs for symbolic and statistical natural language processing, we have used Nltk for tokenization, lemmatization as well as finding the POS tags for words.[2]
- **Sklearn**:- It is used for classification metric calculation as well as confusion matrix evaluation.
- **pandas**:- It is a software library for data manipulation and analysis , we have used pandas for reading hacker news data from the csv file and applying filter on column 'created_by' for separating training data(2018) and testing data(2019.
- **Numpy:-** It is a fundamental package for scientific computations, we have used numpy to calculate unique vocabulary list.
- **Matplotlib**:- It is a plotting library in python , we have used it for plotting the accuracy results against the number of words remaining in vocabulary in experiment 4 and against the smoothing factor in experiment 5.

2. Task 1:Details and Vocabulary Analysis

We have used the entire dataset provided which has 414196 records and has following columns :-

Object ID | Title | Post Type | Author | Created At | URL | Points | Number of Comments

The Training is done on **276981 records** filtered from the entire dataset using **column 'Created At'** , where value is **greater than '2018-01-01' & less than '2019-01-01'**.

Testing is done on **137215 records** filtered from the entire dataset using **column 'Created At'** , where value is **greater than '2019-01-01'**.

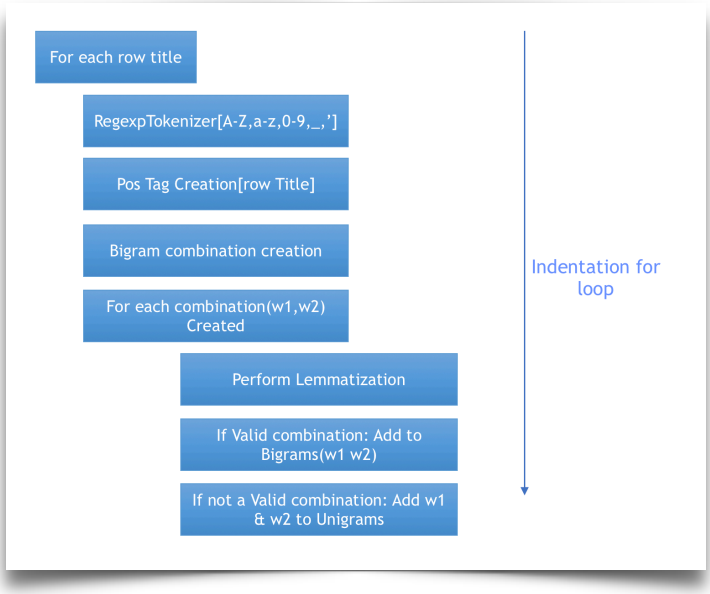
The training and testing data follows the following transformations and optimisations to make it useful for model creation. Here we have used same optimisations for training and testing.

RegexpTokenizer: RegexpTokenizer splits a string into substrings using a regular expression, which makes it flexible over other tokenisers. In order to avoid punctuations such as '- ', ':' etc. to be automatically removed and not part of the resulted tokens, we have used this tokenizer. We also tried performing the same action via word_tokenize but this tokeniser is incapable of removing the above mentioned punctuations. T**he punctuations above were resulting in less accurate vocabulary and was reducing the accuracy by 6 - 7 %.**

Pos Tag Creation: The entire title for each instance is passed directly to pos tag in order to create the correct pos tags as per the title structure. Here we have avoided to pass the tokenised title to pos tag creation as the individual tokens lost their meaning and thus pos tags results were not accurate.

Bigrams: For our vocabulary we have considered the combinations of words **belong to category of pos tag NN noun, NNS noun plural NNP proper noun, NNPS proper noun** and occurring together in the title column of

every instance**.** Consideration of only noun category for bigrams reduced our vocabulary to **370417 unique words.** We have not considered adverbs, adjectives and verbs for bigrams as this had no effect on the accuracy of the final results, but avoiding these bigrams removed noise from the vocabulary and bigrams such as ‘from year , in a’ are removed. Our final vocabulary has **328174 unique bigrams**.

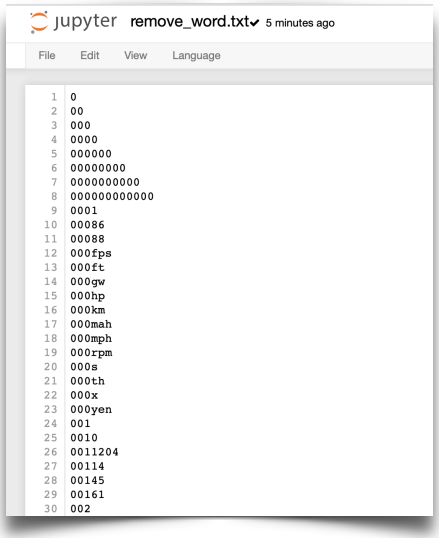


The Vocabulary created is written into a file called **vocabulary.txt** containing **370415 words** and the words not included in the vocabulary are written into a file called **remove_word.txt** containing **5044 words.** We have optimized the execution time of our training model and it took only **201 seconds** to run on the full training dataset and built the vocabulary.

We have removed words which do not aid our model creation and testing which are :-

```
Length of Training Vocabulary is :370417
Number of Bigrams in Vocabulary are: 328174
No. of words removed are :5044
Training time is : 199.79687595367432
```

- **alphanumeric words -** Alphanumeric words were removed from the vocabulary as they had no effect on the accuracy of the model and had tokens with repeated digits such as ‘0000000000’ , ‘000fps’, ‘000yen’ etc which were only creating noise in the vocabulary.



In order to create our model and compute the conditional probabilities for every class type we have followed the following algorithm**:**

Pseudo code :

1. for all classes c_i
2. for all words w_j in the vocabulary
3. Compute **$p(w_j | c_i) = (count(w_j, c_i) + smooth\_factor)/(total\ no.\ of\ words\ in\ c_j + size\ of\ the\ vocabulary)$**
4. The probabilities will be stored in **model-2018.txt** corresponding to each word.

2.1 Task 2: Testing using Naïve Bayes Classifier

The ML classifier that we have used here is Naïve Bayes. It is a classification technique based on Bayes' Theorem with an assumption of independence among features. Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.[1]

The log of Probability of every word in the Title for each instance belong to every class is calculated using the model-2018.txt file and class having the maximum probability is chosen as predicted.

The scores for each class as well as the class predicted is stored in the baseline-result.txt file corresponding each instance in the test dataset.

2.3 Task 3: Experimenting with the Classifier

Our result analysis is based on evaluation of the confusion matrix, classification report using the actual and predicted labels of the model obtained from Task 2 above.

A. Experiment 1(Baseline Experiment) :

Baseline experiment variation from above depends upon the vocabulary generated.

```
Enter the experiment no.(exp1/exp2/exp3/exp4/exp5)exp1
experiment:- exp1


----confusion matrix----
         ask_hn  poll  show_hn   story
ask_hn     4550     0        1     903
poll          1     0        0       5
show_hn      18     0     3617    1268
story        22     0       50  126780


----classification report----
/Users/harshkour/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.py:1437: UndefinedMetricWarning
: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
              precision    recall  f1-score   support

      ask_hn       0.99      0.83      0.91      5454
        poll       0.00      0.00      0.00         6
     show_hn       0.99      0.74      0.84      4903
       story       0.98      1.00      0.99    126852

    accuracy                           0.98    137215
   macro avg       0.74      0.64      0.69    137215
weighted avg       0.98      0.98      0.98    137215



----Accuracy report----
0.9834711948402143
Program finished
```

Experiment 1 result analysis:

- The accuracy of the classifier obtained is **0.9834**
- Most correctly classified class type is ‘story’ with 0.99 f1-score where as not even a single record for class type ‘poll’ is classified correctly.
- The f1-score for class type ‘poll’ is zero since there is not enough training data for this class and thus our model is not trained enough to classify this class type correctly.
- Class type ‘ask_hn’ is classified almost correctly with f1-score as 0.91
- For class type ‘show_hn’ the f1-score is 0.84 which is less than class ‘ask_hn’ and ‘story’ but a lot better than class ‘poll’.
- Overall, there is a major scope of improvement for the results for class type ‘show_hn’, which has been achieved via experiment-2. For class ‘ask_hn’ and ‘story’ the classification results are close to perfect and may improve with experiment-2.

B. Experiment 2 (Stop words Filtering) :

We have performed Tasks 1 and 2 again by removing the stop words from the vocabulary.

The stop words in natural language processing are useless words in training data. These are generally treated as noise in the data. Since these words did not aid in training so we remove these words from the vocabulary.

If the stop word is part of the bigram we are removing the bigram word from the vocabulary as well, for eg: for the removal of stop word ‘a’, if there is a unigram as ‘a’ in vocabulary, it is removed. But if there is a bigram such as ‘a year’ it is also removed as this bigram is formed of a stop word.

The list of stop words is obtained from the stopwords.txt file provided.

Experiment 2 result analysis :

```
Enter the experiment no.(exp1/exp2/exp3/exp4/exp5)exp2
experiment:- exp2


----confusion matrix----
         ask_hn  poll  show_hn   story
ask_hn     5148     0        0     306
poll          0     0        0       6
show_hn       2     0     4534     367
story        37     0       99  126716


----classification report----
              precision    recall  f1-score   support

      ask_hn       0.99      0.94      0.97      5454
        poll       0.00      0.00      0.00         6
     show_hn       0.98      0.92      0.95      4903
       story       0.99      1.00      1.00    126852

    accuracy                           0.99    137215
   macro avg       0.74      0.72      0.73    137215
weighted avg       0.99      0.99      0.99    137215



----Accuracy report----
0.9940458404693364
Program finished
```

- The accuracy of the classifier has been increased here from **0.9834** (baseline experiment with stop words in its vocabulary) to **0.9940** which indicates that stop words were noise in the training data.

- The classification results for class ‘ask_hn’ and ‘show_hn’ has been significantly increased which can be clearly also be seen from f1-score which has increased to 0.97 from 0.91 of the base experiment for class ‘ask_hn’ and 0.95 from 0.84 for class ‘show_hn’.
- The f1-score for class type ‘story’ has reached 1.00 from 0.99 of the base experiment.
- The f1-score for class type ‘poll’ is zero since there is not enough training data for this class and thus our model is not trained enough to classify this class type correctly and stop words had no effect on this result from experiment-1
- Overall, stop words had a brilliant effect on the classification for class ‘story’, ask_hn’ and ‘show_hn’ but no effect on the class ‘poll’.

C. Experiment 3 (Word Length Filtering):-

We have filtered words having length less than equal to 2 and all words with length greater than equal to 9. The new vocabulary size for experiment 3 is now **53999**.

The results are as follows:-

```
Enter the experiment no.(exp1/exp2/exp3/exp4/exp5)exp3
The length of vocabulary in experiment 3 is :53999
```

Experiment 3 result analysis :

```
Enter the experiment no.(exp1/exp2/exp3/exp4/exp5)exp3
experiment:- exp3


----confusion matrix----
         ask_hn  poll  show_hn   story
ask_hn     5448     0        1       5
poll          1     0        0       5
show_hn      36     0     4818      49
story       470     0      236  126146


----classification report----
/Users/harshkour/anaconda3/lib/python3.7/site-packages/sklearn/metrics/classification.py:1437: UndefinedMetricWarning
: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
              precision    recall  f1-score   support

      ask_hn       0.91      1.00      0.96      5454
        poll       0.00      0.00      0.00         6
     show_hn       0.95      0.98      0.97      4903
       story       1.00      0.99      1.00    126852

    accuracy                           0.99    137215
   macro avg       0.72      0.74      0.73    137215
weighted avg       0.99      0.99      0.99    137215



----Accuracy report----
0.9941478701308166
Program finished
```

- The accuracy of the classifier has been increased here from 0.983 (baseline experiment which also includes the words with length less than 2 and greater than 9) to 0.9941 which indicates that these words were nothing but noise in the training data.
- The classification results for class ‘ask_hn’ and ‘show_hn’ has been significantly increased which can be clearly also be seen from f1-score which has increased to 0.96 from 0.91 of the base experiment for class ‘ask_hn’ and 0.97 from 0.84 for class ‘show_hn’.
- The f1-score for class type ‘story’ has reached 1.00 from 0.99 of the base experiment, which is quite similar to what we have achieved in experiment-2. This shows that major stop words were removed as part of experiment-3 as well.

- The f1-score for class type ‘poll’ is zero since there is not enough training data for this class and thus our model is not trained enough to classify this class type correctly and stop words had no effect on this result from experiment-1
- In comparison from experiment-2 and experiment-1, experiment-3 had better results for class type ‘show_hn’, since this class has shown the maximum f1-score during the experiment-3.

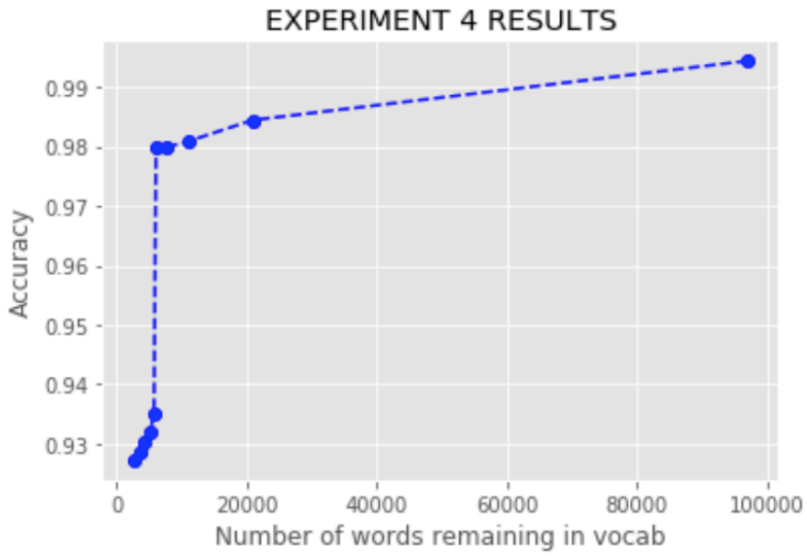
D. Experiment 4 (Infrequent Word Filtering):-

In this experiment we are gradually removing words with specific length and words with highest frequency in the Training Dataset.

The results are as follows:-

- The plot between the no. of words left in the vocabulary after removing frequent words against the accuracy after each subsequent removal has been shown below.
- The order followed is as follows:- [=1, <=5, <=10, <=15, <=20, top 5%, top 10%, top 15%, top 20%, top 25%]
- The Remaining words list(x-axis) and the Accuracy(y-axis) at each step is fed to the plot.

```
Remaining words in Vocab:
 [97047, 20994, 11205, 7873, 6149, 5842, 5258, 4470, 3576, 2682]
Accuracy:
 [0.994344641620814, 0.9843967496264986, 0.9808694384724702, 0.9798637175235944, 0.9798199905258171, 0.935130998797506
76, 0.9320263819553256, 0.9301825602157199, 0.9287541449549976, 0.9273111540283496]
```



Experiment 4 result analysis :

- There are large no. of words in the vocabulary with **frequency =1** since the vocabulary size reduces from **370415** to **97047 words** i.e **73.8%** of the vocabulary has words with frequency as only 1.
- Removing most(top) frequent words lowers the probability of classifying the title in the correct class and hence lowers the overall accuracy of the Naïve Bayes Classifier.
- We can see a huge dip in the curve as the length of the vocal decreases.
- Also since the most frequent words are gradually removed, the accuracy decreases since classification becomes difficult because of the major presence of the most frequent words in the testing dataset as well.

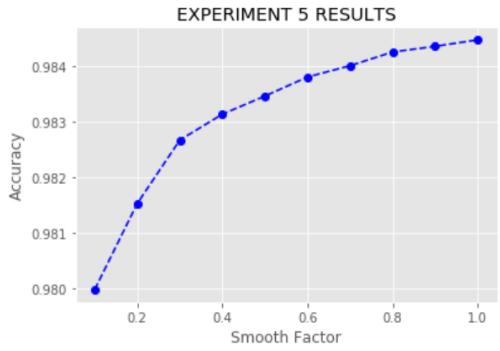
E. Experiment 5(Smoothing):-

In this experiment we have used different value for smoothing factor and analyzed the accuracy of the algorithm on each smooth value.

The smoothing values to be analyzed lies between 0 to 1 with a step of 0.1

The results are as follows :-

```
Enter the experiment no.(exp1/exp2/exp3/exp4/exp5)exp5
Smooth Factor:
 [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]
Accuracy:
 [0.9799803228510002, 0.9815326312720912, 0.9826768210472616, 0.9831432423568852, 0.9834711948402143, 0.9838137229894691, 0.984
0177823124294, 0.9842655686331669, 0.984367598294647, 0.984484203622053]
```




```
Program finished
```

Experiment 5 result analysis :

- The plot clearly shows the increase in the accuracy for every 0.1 increase in the value for the smoothing factor.
- Since there is a gradual increase in the smoothing factor, this has resulted in a very slight change in the accuracy.
- From the confusion matrix and the F1-score obtained from the individual variation of the smoothing factor, we observed that the class type ‘show_hn’ has responded to the variation of smoothing factor from 0.9 to 1.
- For smoothing factor 0.5, 0.6,0.7,0.8, 0.9 the f1-score for class ‘show_hn’ is 0.85 but at smoothing factor 1, show_hn has a f-1 score of 0.86 which has lead to a slight increase in the accuracy from 0.98436 to 0.98448, no the class type has responded to this variation of smoothing factor.
- For smoothing factor 0.1 to 0.2, class ‘ask_hn’ has a magnificent improvement of f1-score from 0.88 to 0.90, class ‘show_hn’ has shown a slight increase in f1-score i.e. from 0.81 to 0.82. No other class type has responded to this variation.
- Class type ‘poll’ has not responded to any variation of smoothing factor and the f1-score for class ‘poll’ is zero for all the variations.
- For variation in smoothing factor from 0.2 to 0.3, only lass ‘show_hn’ has responded with variation in f1-score from 0.82 to 0.84, no other class has responded for this variation.
- For variation in smoothing factor from 0.3 to 0.4, no class has responded and variation in f1-scores’s is not seen where as there is a minor change in the accuracy of the classifier.

3. Comparison of the results from all experiments

Following are the major observations from all the experiments

- The size of a dataset is often responsible for poor performances in ML projects. Since the size of the training as well as the testing data is too big our results have faired well.

- The size of the different post_types in the training data is story which has 254222 instances,ask-hn which has 12509 instances, show-hn which has 10225 instances and poll which has only 25 instances. As we see we have 68.6% instances of story which corresponds to its high accuracy in classification and only 0.00006% records for poll in the training dataset corresponds to its poor classification accuracy.
- Using bigrams in the model increases the semantic relationship between the words hence increasing the accuracy of the classifier.[5]
- The f1-score for class type 'poll' is zero for all the experiments since there is not enough training data for this class and thus our model is not trained enough to classify this class type correctly .
- For experiment-5, change in the smoothing factor has shown an increase in the f1-score of majority class 'show_hn' and very slight change for class 'ask_hn'. There is no change in f1-scores for class type 'story' or class type 'poll' which leads to an observation that smoothing factor in our case has not brought major effects on the class type with major and minor records. Since in our dataset, the major instances are for class type 'story' and minor for class type 'poll'.

4. Future Work

- We would have tried to reduce the size of the vocabulary(**370415 words**) to extract more meaningful words which can be used for training the model.
- It would have been interesting to investigate if any specific digit or numerical(**eg year**) inclusion in the vocabulary has an effect on the algorithm accuracy.
- According to the above analysis maximum correctly classified instances belong to class story since the training data has large number of instances which belong to story, we think that the training dataset should be balanced and should not have such baised towards class story.The future work involves balancing the dataset to have stronger analysis on class story.
- We are curious as to how other classification algorithms fair on this Hacker News dataset. If time would have permitted we would have tried more algorithms except naïve bayes on this dataset.
- If time would have permitted we would have used more feature selectors for the text classification.

5. RESPONSIBILITIES & CONTRIBUTIONS

Harsh Deep Kour	Iknoor Singh Arora
• Hacker News Dataset Analysis. • Understanding Libraries. • Vocabulary creation • Implementing experiments • Result Analysis • Optimizing the testing time. • Report Writing	• Setting up development environment. • Hacker News Data Analysis. • Understanding Libraries. • Training the model. • Result Analysis • Optimizing the training time • Report Writing

Table 1 : Contributions Table

6. REFERENCES

- https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/
- https://www.nltk.org/
- https://nlp.stanford.edu/pubs/sidaw12_simple_sentiment.pdf