SENTIMENT CLASSFICATION ON PRODUCT REVIEWS

JUNE 02, 2019

Table of Contents

1 Introduction

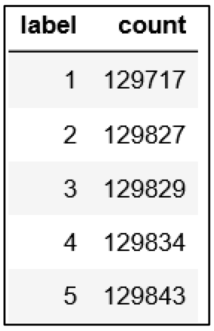
This report contains the discussions and results of developing a sentiment classifier performed on a large set of product reviews provided by Yelp. The aim of this challenge is to build a multi-class sentiment classifier to allow us to assign a large set of product reviews to five different levels of polarity of opinion, mainly *Strong Negative, Weak Negative, Neutral, Weak Positive & Strong Positive.*

We are given 3 different datasets, *train_data.csv* which contains 650000 product reviews which will be used as our training data; *train_label.csv* which contains the labels for the 650000 product reviews, in which the polarity was being labelled as *1* for *Strong Negative* and accordingly to *5* for *Strong Positive; test_data.csv* which contains 50000 unlabelled product reviews which will be solely for testing purpose.

The programming language used here is R and an open-sourced machine learning platform h2o was used for its ability to handle large-scaled data and build algorithms in parallel with multi-cores.

2. Exploratory Data Analysis

Based on the 650000 product reviews, none of the product reviews were empty, 827 contains duplicated reviews with similar label, and 950 contains duplicated reviews with different label. These records were removed from our training data. With the remaining 640950 reviews, balance of the classes was checked and as shown in Figure 1, the training data is balanced.



label	count
1	129717
2	129827
3	129829
4	129834
5	129843

(Figure 1 – R output of total samples across each sentiment class)

Due to the memory limitations of R Programming, we extracted only 200000 records as training data as our main intention was to extract as many features as possible. Steps were taken to ensure the 200000-training data were balanced across the five sentiment labels.

3. Pre-Processing & Feature Generation

As the product reviews are in free-text, text pre-processing is required before we can use them for classification purposes. The text pre-processing steps we have implemented were *tokenization, case normalization, punctuations removal, special characters removal, stopwords removal, removal of words of length one or two and filtering off the most/least frequent word.* After which, two feature selection steps were performed to extract more informative features and they were *bigrams generation* and *trigrams generation.*

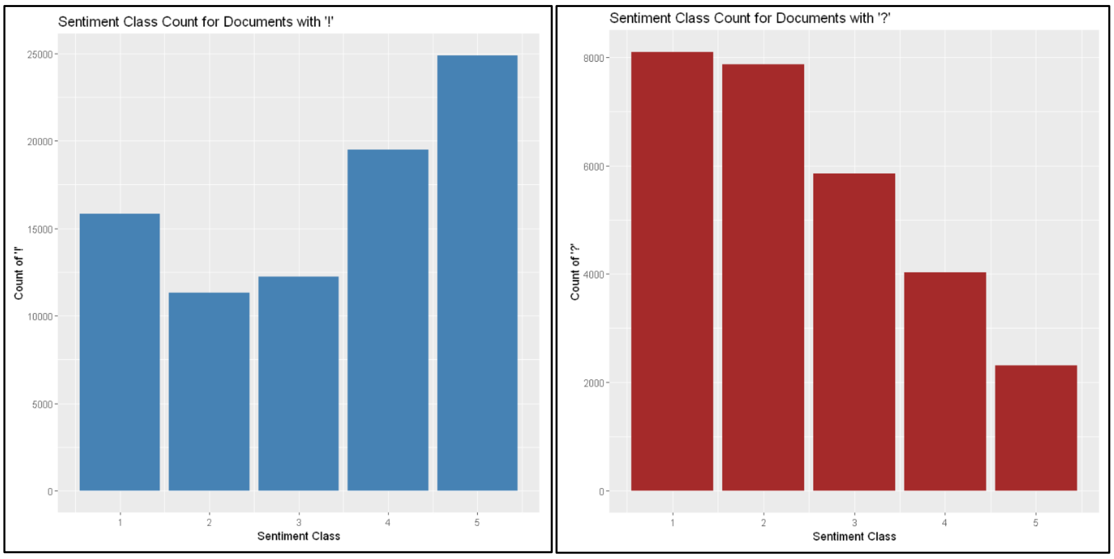
3.1 Pre-processing

3.1.1 Tokenization & Case Normalization

We first break the free texts into individual words which is a process termed tokenization. Each word/token is then considered as a single feature. For instance, "*Product is good"* will be stored as three individual tokens *"product", "is", "good"*. Furthermore, we converted all the tokens into lowercase for consistency across the whole vocabulary set/corpus.

3.1.2 Punctuations Removal

Punctuations are often removed from the corpus as they serve little to no purpose [1]. However, we considered usage of `*!* ` and `*?* ` in product reviews of possible indicators towards the sentiment ratings (*"This is good!", "Why is it so bad?"),* and further investigations were carried out.

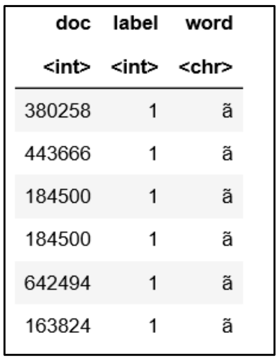


(Figure 2 – R plots of '!', '?' across each sentiment class)

From Figure 2, there's a clear relationship suggesting that having `!` in the product review indicated weak and strong positive (i.e. class 4 and 5), and`?` is associated with strong negative and weak negative (i.e. Class 1 and Class 2), Hence, we have decided to keep both `!` `?` and remove the rest of the punctuations from our corpus.

3.1.3 Removal of Special Characters

Checks were made to ensure the documents do not contain any non-English alphabets like Latin and as shown in Figure 3, there were some documents containing the Latin *"ã"* in our corpus and we should remove them as they might be possible noise to our classifier model.

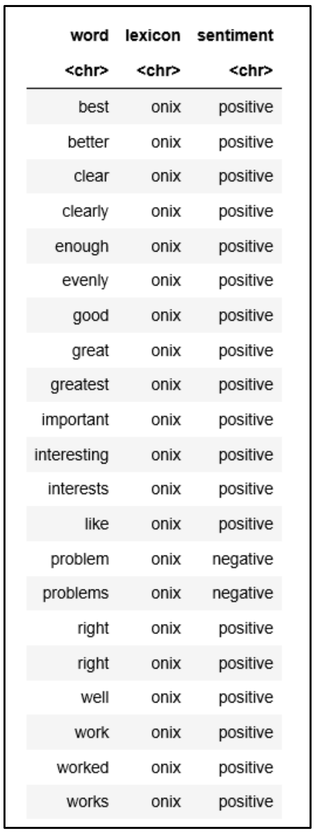


doc	label	word
<int>	<int>	<chr>
380258	1	ã
443666	1	ã
184500	1	ã
184500	1	ã
642494	1	ã
163824	1	ã

(Figure 3 – R output of samples containing "ã")

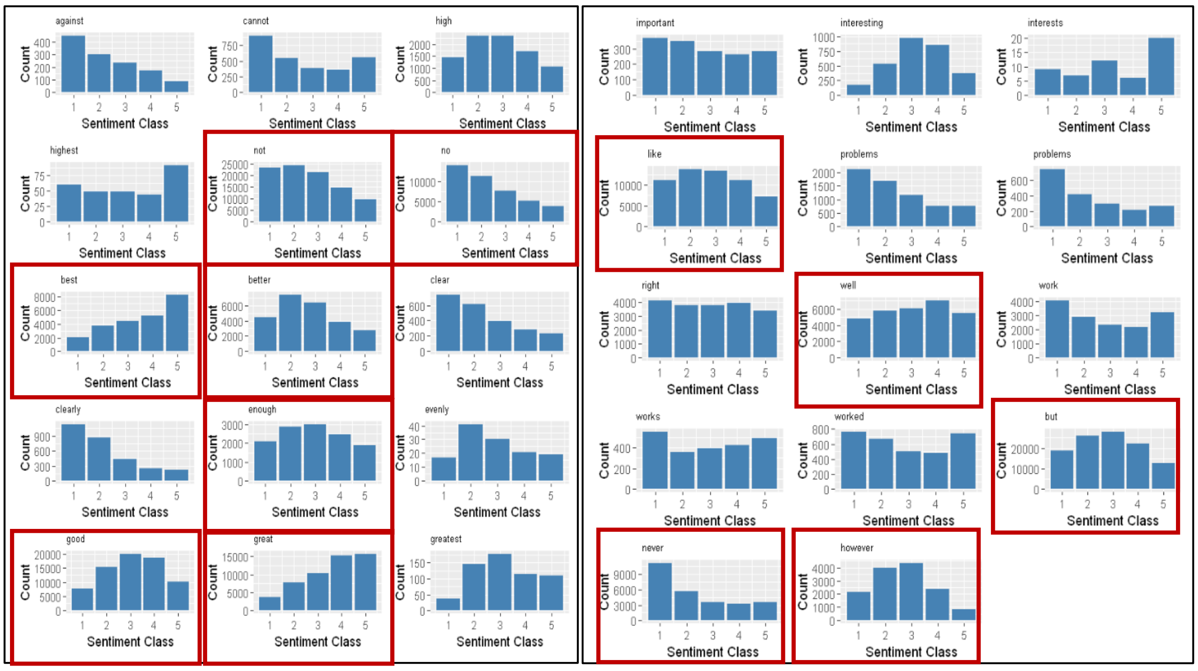
3.1.4 Removal of Stopwords

Stopwords are words that carry very little lexical content and are extremely common in corpus and removing them is often recommended [2]. Using onix stopwords list [3], we identified stopwords that might be influential in the sentiment for the review especially if they suggest negative or positive sentiments. We used `sentiment` function in the `tidytext` R package which return words that are positive or negative in sentiments [4]. There were 20 such stopwords being identifed as shown in Figure 4 below and a further 9 stopwords were considered for exploration.



word	lexicon	sentiment
<chr>	<chr>	<chr>
best	onix	positive
better	onix	positive
clear	onix	positive
clearly	onix	positive
enough	onix	positive
evenly	onix	positive
good	onix	positive
great	onix	positive
greatest	onix	positive
important	onix	positive
interesting	onix	positive
interests	onix	positive
like	onix	positive
problem	onix	negative
problems	onix	negative
right	onix	positive
right	onix	positive
well	onix	positive
work	onix	positive
worked	onix	positive
works	onix	positive

(Figure 4 – R table of 20 stopwords identified by sentiment package)

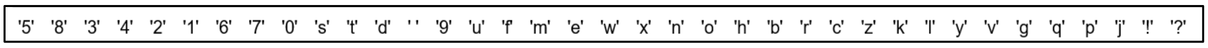


(Figure 5 – R plot of the 29 identified stopwords)

Looking at Figure 5, we have identified 12 words based on their frequencies towards a sentiment rating; *"not", "no", "best", "better", "enough", "good", "great", "like", "well", "but", "never", "however".*

3.1.5 Removal of Words Length 1 & 2

Next, tokens of length 1 are generally not useful in product reviews and investigations were done on such tokens.

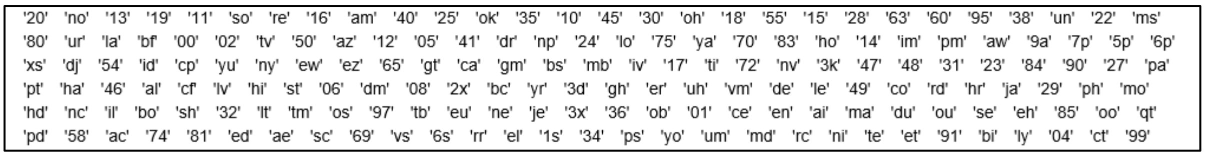


```
'5'  '8'  '3'  '4'  '2'  '1'  '6'  '7'  '0'  's'  't'  'd'  ' '  '9'  'u'  'f'  'm'  'e'  'w'  'x'  'n'  'o'  'h'  'b'  'r'  'c'  'z'  'k'  'l'  'y'  'v'  'g'  'q'  'p'  'j'  '!'  '?'
```

(Figure 6 – R output of length 1 tokens)

Based on Figure 6, single length tokens were mostly alphabeting and digits. We decided to keep the digits as they might be indicative towards the sentiment rating and remove the remaining single alphabets which do not mean much in product reviews context.

We investigated tokens of length 2 as well.



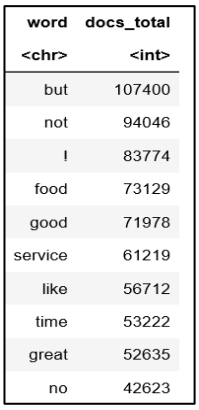
```
'20'  'no'  '13'  '19'  '11'  'so'  're'  '16'  'am'  '40'  '25'  'ok'  '35'  '10'  '45'  '30'  'oh'  '18'  '55'  '15'  '28'  '63'  '60'  '95'  '38'  'un'  '22'  'ms'
'80'  'ur'  'la'  'bf'  '00'  '02'  'tv'  '50'  'az'  '12'  '05'  '41'  'dr'  'np'  '24'  'lo'  '75'  'ya'  '70'  '83'  'ho'  '14'  'im'  'pm'  'aw'  '9a'  '7p'  '5p'  '6p'
'xs'  'dj'  '54'  'id'  'cp'  'yu'  'ny'  'ew'  'ez'  '65'  'gt'  'ca'  'gm'  'bs'  'mb'  'iv'  '17'  'ti'  '72'  'nv'  '3k'  '47'  '48'  '31'  '23'  '84'  '90'  '27'  'pa'
'pt'  'ha'  '46'  'al'  'cf'  'lv'  'hi'  'st'  '06'  'dm'  '08'  '2x'  'bc'  'yr'  '3d'  'gh'  'er'  'uh'  'vm'  'de'  'le'  '49'  'co'  'rd'  'hr'  'ja'  '29'  'ph'  'mo'
'hd'  'nc'  'il'  'bo'  'sh'  '32'  'lt'  'tm'  'os'  '97'  'tb'  'eu'  'ne'  'je'  '3x'  '36'  'ob'  '01'  'ce'  'en'  'ai'  'ma'  'du'  'ou'  'se'  'eh'  '85'  'oo'  'qt'
'pd'  '58'  'ac'  '74'  '81'  'ed'  'ae'  'sc'  '69'  'vs'  '6s'  'rr'  'el'  '1s'  '34'  'ps'  'yo'  'um'  'md'  'rc'  'ni'  'te'  'et'  '91'  'bi'  'ly'  '04'  'ct'  '99'
```

(Figure 7 – R output of length 2 tokens)

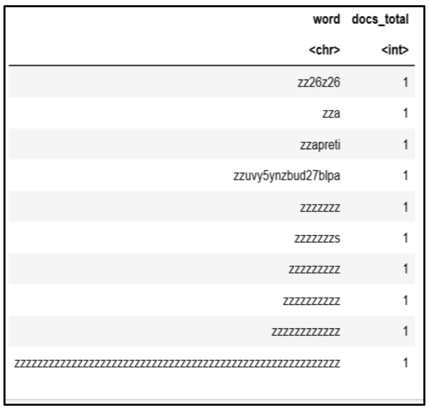
Based on Figure 7, most of the tokens carry little sense in the context of product reviews and most likely be acting as noise if chosen by our classifier model. However, three tokens were identified and kept: *"no"* which was identified in Section 3.1.4, *"10"* and *"ok",*

3.1.6 Remove Most/Least Frequent Words

The final step in our text pre-processing was removing the most/least frequent words based on their document frequency, in order to reduce the vocabulary size. Looking at the top 10 most/least frequent words, the top 10 seen in Figure 8 contains mainly the stopwords we have identified in Section 3.1.4 while the least 10 consist of words that are of little to no meaning in the context of product reviews.



word	docs_total
<chr>	<int>
but	107400
not	94046
!	83774
food	73129
good	71978
service	61219
like	56712
time	53222
great	52635
no	42623



word	docs_total
<chr>	<int>
zz26z26	1
zza	1
zzapreti	1
zzuvy5ynzbud27blpa	1
zzzzzzz	1
zzzzzzzs	1
zzzzzzzzz	1
zzzzzzzzzz	1
zzzzzzzzzzzz	1
zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz	1

(Figure 8 – R table of top/last 10 tokens)

As our objective was to generate as much features as possible with the 200000 training data, we decided to filter away the words that appeared in more than 99% of the documents (i.e. 198000) or less than 0.03% of the documents (i.e. 600), and keep the remaining as our features.

From the corpus status in Figure 9 below, we have reduced the vocabulary size from 123770 to 2188 after five pre-processing steps. The lexical diversity [5] has also been reduced significantly from 141.314 to 42.12329.

	Before Text Pre-processing	Removal of punctuations	Removal of special characters	Removal of stopwords	Removal of words length 1 or 2	Removal of most/least frequent words
Total number of documents	200000	200000	200000	199999	199997	199978
Total number of tokens	28262800	25019385	25019385	10990246	10648051	8423732
Vocabulary Size	123770	123895	123721	123336	122420	2188
Average number of tokens (Lexical Diversity)	141.314	125.0969	125.0969	54.9515	53.24105	42.12329
Minimum number of tokens	1	1	1	1	1	1
Maximum number of tokens	10099	2787	2787	2787	2787	494

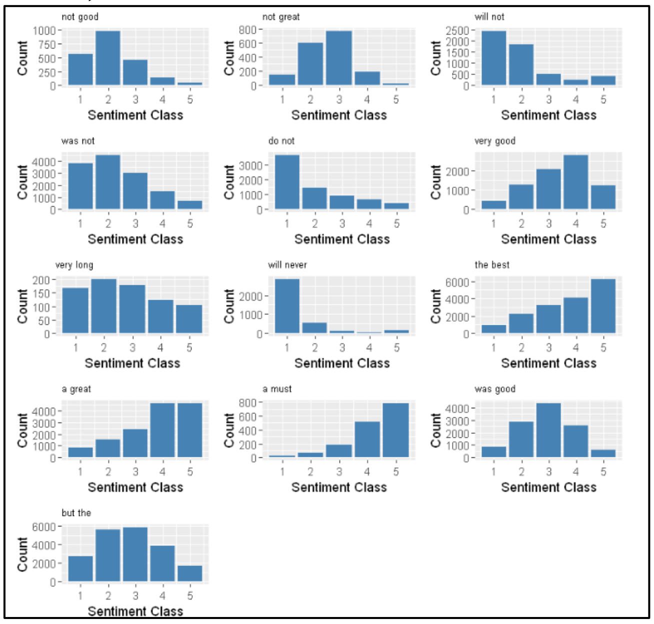
(Figure 9 - Status of Corpus after each pre-processing steps)

3.2 Feature generation

3.2.1 Feature Set 1 – Bigrams

First feature generation method considered was bigrams which are made up of two consecutive words in a given sentence. Generally, bigrams add more context into the words [6] and will prove to be more meaningful for sentiment analysis as a bigram of *"not good"* will be more accurate compared to two unigrams of *"not"* & *"good"* which might result in wrong classification.

Similar pre-processing steps in Section 3.1 were performed to bigrams, and we identified 13 stopwords bigrams that might be intuitive in representing the sentiment rating: *"not good", "not great", "will not", "was not", "do not", "very good", "very long", "will never", "the best", "a great", "was good", "a must", "but the"*.



(Figure 10 - – R plot of the 13 identified bigrams)

Based on the frequencies towards a sentiment class in Figure 10, all identified bigrams other than *"very long"* will be kept in our corpus. The top 100 bigrams based on the document frequency were then chosen for our feature set.

Therefore, the first feature set we considered was 2188 unigrams and 100 bigrams resulting in the vocabulary size to increase to 2288 and lexical diversity to 43.38678.

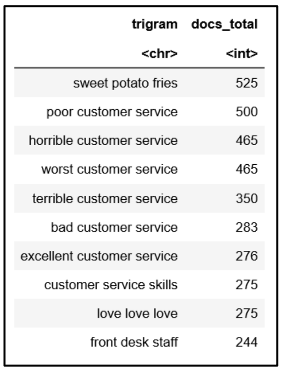
	Before Feature Generation	**Feature Set 1**
Total number of documents	199978	199978
Total number of tokens	8423732	8676402
Vocabulary Size	2188	2288
Average number of tokens (Lexical Diversity)	42.12329	43.38678

(Figure 11 - Status of Corpus of Feature Set 1)

3.2.2 Feature Set 2 – Trigrams

The second feature generation method considered was trigrams which are made up of three consecutive words in a given sentence that adds more context and information.

Similar pre-processing steps in Section 3.1 were performed to trigrams. Based on the top 10 most frequent trigrams in Figure 12, mostly were related to *"customer service"* and intuitively, they are useful in determining the sentiment labels. The top 50 trigrams were chosen as studies have shown inclusion of large number of trigrams might result in overfitting [7].



trigram <chr>	docs_total <int>
sweet potato fries	525
poor customer service	500
horrible customer service	465
worst customer service	465
terrible customer service	350
bad customer service	283
excellent customer service	276
customer service skills	275
love love love	275
front desk staff	244

(Figure 12 – R table of top 10 trigrams)

Therefore, the second feature set we considered was 2188 unigrams, 100 bigrams and 50 trigrams resulting in the vocabulary size to increase to 2338 and lexical diversity to 43.43867.

	Before Feature Generation	**Feature Set 1**	**Feature Set 2**
Total number of documents	199978	199978	199978
Total number of tokens	8423732	8676402	8686779
Vocabulary Size	2188	2288	2338
Average number of tokens (Lexical Diversity)	42.12329	43.38678	43.43867

(Figure 13 - Status of Corpus of Feature Set 2)

4. Models

Two models were implemented, multinomial logistic regression and deep neural networks.

4.1 Model 1 – Multinominal Logistic Regression

We considered multinomial logistic regression as the first model as firstly, it assumes all predictor variables to be independent which is our assumption that all words are independent of each other in a product review. Furthermore, this model also assumes non-separable boundary in which there will not be a single boundary that separate the classes within the target variable clearly, which coincides with product reviews as product reviews are highly subjective [8].

The multinomial logistic regression was carried out in h2o using it's generalized linear model (GLM) function with family defined as 'multinomial'. 'L_BFGS' was defined as the solver method as it solves better for wider and dense datasets (thousands of predictors) [9].

GLM function has an in-built regularisation function and users can define alpha which controls the distribution between lasso and ridge penalties, as well as lambda, the shrinkage parameter. We have chosen not to implement regularisation in this model as overfitting did not occur when we cross-validated the training and validation error.

4.2 Model 2 – Deep Neural Networks

The next model is Deep Neural Network, which works by representing each word as a unit or neuron, these units are then connected to each unit in the hidden layers and finally to the output layer. This type of network is said to be fully connected. Purpose of hidden layer is to map the input into a feature space with a dimension corresponding to the hidden layer [10].

Another main reason why we have decided to implement neural networks were due to the reason that misclassification errors were especially high for class 2, 3, 4 with multinomial linear regression, and we would want an algorithm that can discover hidden features/information in these classes, and hopefully improve the accuracy.

The Neural Network was implemented in h2o. It consists of 5 layers, the input feature set, 3 hidden layers, each with 2048 hidden units, and the output layer of size 5.

We chose 10,000 for the `epoch_size` and left the parameter `overwrite_with_best_model` as the default, which returns the model with the lowest classification error independent of epoch number.

The regularisation term was set to L1, with a lambda value of 0.0001. The rationale was to eliminate features which had little important and were contributing noise to the model.

We also implemented an early stopping procedure which was based on the logloss criterion. The stopping rounds was set to 2 and the stopping tolerance was set to .01. In plain english, this means that if the logloss metric did not improve by more than 1% over 2 iterations, the training process would cease and return the best model.

We used the default activation function of tanh.

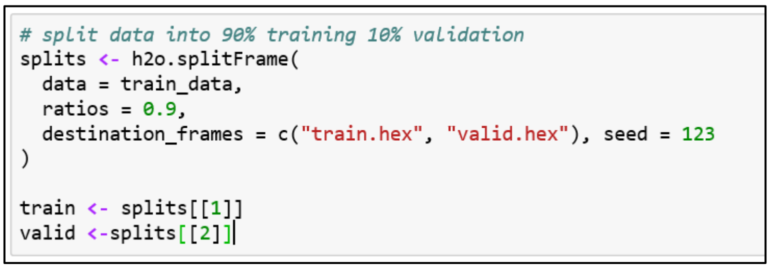
An assumption that the decision boundary for the classes was not linear, which is why we opted for a deep neural network with a non-linear activation function.

4.3 Discussion of model difference(s)

Model	Advantages	Disadvantages
Multinomial Logistic Regression	Performs well with large sample size as it uses maximum likelihood estimation	Possibility of overfitting
	Simple and fast to implement	Each datapoint must be independent of each other
	Data need not be separable	
Deep Neural Network	Can approximate any function to an arbitrary precision given enough hidden units and layers	Difficult to interpret
		Long training time making cross-validation techniques impractical
		Large parameter sets for tuning

5. Experiment setups

A training-validation approach was taken to better measure the accuracy of our models in both Multinomial Logistic Regression and Deep Neural Networks. h2o has an inbuilt function `splitFrame` as shown in Figure 14, which will split the training data into training and validation sets based on a given ratio. We decided to split our data into 90% training and 10% as the validation set.

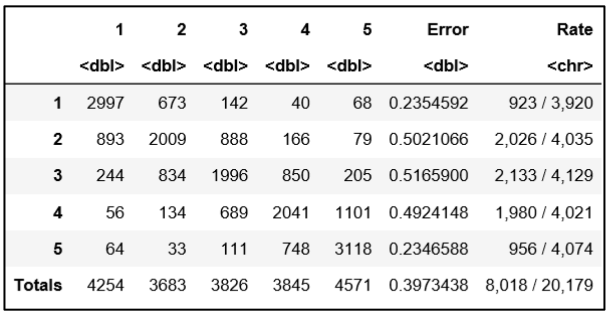


```r
# split data into 90% training 10% validation
splits <- h2o.splitFrame(
  data = train_data,
  ratios = 0.9,
  destination_frames = c("train.hex", "valid.hex"), seed = 123
)

train <- splits[[1]]
valid <-splits[[2]]
```

(Figure 14 – h2o splitframe function)

Inbuilt h2o function `confusionMatrix` was then used to compare the predicted labels and true labels of the validation set as seen in Figure 15. What will be ideal in the confusion matrix is to have a higher number on the diagonal of the matrix which means the predictions were accurate. The mean of the misclassification error across the 5 classes were then obtained which will be used as the criteria for our model comparison.



	1	2	3	4	5	Error	Rate
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
1	2997	673	142	40	68	0.2354592	923 / 3,920
2	893	2009	888	166	79	0.5021066	2,026 / 4,035
3	244	834	1996	850	205	0.5165900	2,133 / 4,129
4	56	134	689	2041	1101	0.4924148	1,980 / 4,021
5	64	33	111	748	3118	0.2346588	956 / 4,074
Totals	4254	3683	3826	3845	4571	0.3973438	8,018 / 20,179

(Figure 15 – h2o confusionMatrix output)

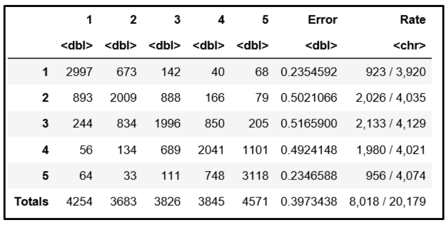
6. Experimental results

A total of 4 models were trained using the 2 feature sets in Section 3.2 on both models and the confusion matrix was obtained in each model to get the misclassification error as shown in Figure 16.

Feature Set	**Model**	**Validation Misclassification Error**
Feature Set 1 (Unigrams + Bigrams)	Multinomial Log Reg 1	0.3973
	Multinomial Log Reg 2	0.3979
Feature Set 2 (Unigrams + Bigrams + Trigrams)	Deep Neural Network 1	0.4008
	Deep Neural Network 2	0.4003

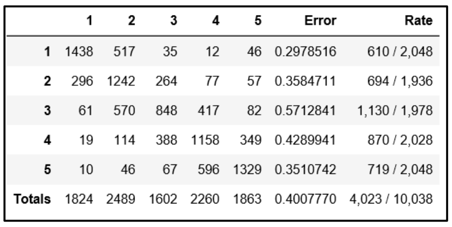
(Figure 16 – Validation Misclassification Error)

Based on the validation misclassification error, the best model was Multinominal Logistic Regression using feature set 1. Surprisingly, even though Deep Neural Network were able to classify class 2 and class 4 better than Multinomial Logistic Regression, the misclassification in class 1, 3, and 5 were worse and led to the total mean misclassification error to be higher.



	1	**2**	**3**	**4**	**5**	**Error**	**Rate**
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>
1	2997	673	142	40	68	0.2354592	923 / 3,920
2	893	2009	888	166	79	0.5021066	2,026 / 4,035
3	244	834	1996	850	205	0.5165900	2,133 / 4,129
4	56	134	689	2041	1101	0.4924148	1,980 / 4,021
5	64	33	111	748	3118	0.2346588	956 / 4,074
Totals	4254	3683	3826	3845	4571	0.3973438	8,018 / 20,179

(Figure 17 – h2o confusionMatrix output for Multi Log Reg 1)



	1	**2**	**3**	**4**	**5**	**Error**	**Rate**
1	1438	517	35	12	46	0.2978516	610 / 2,048
2	296	1242	264	77	57	0.3584711	694 / 1,936
3	61	570	848	417	82	0.5712841	1,130 / 1,978
4	19	114	388	1158	349	0.4289941	870 / 2,028
5	10	46	67	596	1329	0.3510742	719 / 2,048
Totals	1824	2489	1602	2260	1863	0.4007770	4,023 / 10,038

(Figure 18 – h2o confusionMatrix output for DNN 1)

Furthermore, with the addition of trigrams, the performance of both models was not better as we have expected with more features yielding more information. Deep Neural Network performed even worse with trigrams compared to feature set 1.

7. Conclusion

The optimal classifier as discussed in Section 6 was Multinomial Logistic Regression with feature set 1 that consists of unigrams and bigrams. Our final predictions for the testing data was based on this model.

Lessons learned:

- Text Pre-processing appears to be by far the most important factor as the feature sets depend on it, it’s simply not enough to remove punctuation, stopwords and create n-grams.
- Multinomial Logistic Regression might not always work well with non-separable datapoints
- Neural networks are complicated and take a very long time to train. In terms of practicality, applying cross validation can take a very long time and requires powerful or distributed hardware. Neural networks are also difficult to interpret, we had trouble explaining the variance in the predicted classes in the confusion matrix between the two feature sets. It seemed that a trade-off between misclassification of label 3 and label 2 and 4 existed, the model would also seemingly vary in its misclassification of class 5 for an unknown reason.

Suggestions for future work:

- Use all training data available as more data will give more accurate representations of the sentiment classes
- Explore further text pre-processing method like part-of-speech (POS)
- More extensive cross-validation approach to obtain better accuracy of the models for better comparisons
- Hyperparameters [11] tuning for both multinomial logistic regression and deep neural networks in h2o to obtain more accurate models
- Explore L1 lasso and L2 ridge regression, as well as elastic net for regularisation
- Explore alternative algorithms like Support Vector Machines, Gradient Boosting or even Ensemble Modelling as it seems to suggest that there will not be one machine learning algorithm that gives good accurate predictions

References

[1] https://www.kdnuggets.com/2017/12/general-approach-preprocessing-text-data.html
[2] https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
[3] http://www.lextek.com/manuals/onix/stopwords1.html
[4] https://www.r-bloggers.com/using-tidytext-to-make-sentiment-analysis-easy/
[5] https://en.wikipedia.org/wiki/Lexical_diversity
[6] http://www.phontron.com/slides/nlp-programming-en-02-bigramlm.pdf
[7] https://github.com/hina86/DM2017_UT/wiki/Iteration-2-:-Text-Mining-(bigram-and-Trigram)
[8] https://it.unt.edu/sites/default/files/mlr_jds_aug2011.pdf
[9] http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/glm.html
[10] https://stats.stackexchange.com/questions/300543/if-each-neuron-in-a-neural-network-is-basically-a-logistic-regression-function
[11] https://www.h2o.ai/blog/hyperparameter-optimization-in-h2o-grid-search-random-search-and-the-future/