Lost Redditors: A Machine Learning Story

EECS 349 Northwestern University
Michael Threatt, Jason Liu, Bennett Hensey

Abstract

We decided to create a content classifier for Reddit which is a popular online forum. Reddit uses subreddits to categorize content matter for its viewers. Unfortunately, there are people known as "trolls" who go about creating irrelevant content to harass to Reddit community. Moderators are the line of defense to prevent and control this but Reddit is a massive forum with hundreds of thousands of different subreddits.
Our task is to create a content classifier to aid Reddit moderators in their difficult and tedious task of scanning through titles and ensuring they remain on topic for that particular subreddit. This task would be daunting to people alone so we developed a model that would serve as a tool to help moderators scrub improper content from Reddit. We used Bayes net, J48 decision tree, and neural network models and compared the results for each scheme to see which would be most effective as a content classifier. We are able to acquire data using the Reddit API along with NLTK in python. We then used Weka and Keras for the machine learning portion of our design. This design takes the word frequency of the titles to judge the relevance of a post to a particular subreddit. After training a machine learner on the top posts of all time from the 6 most subscribed subreddits, we tested the model on datasets derived from different time periods to assess the generalizability of our classifier.

Dataset

Data was acquired by scraping the top 6 subscribed subreddits from Reddit using the official Reddit API. We gathered the titles of the top 1000 posts (a string between 1 and 300 characters) for each subreddit, and created 4 different datasets for top posts from the time periods "all-time", "year", "month", and "week" — these are the only time periods allowed by the API and the maximum number of posts we can request, yielding a total of 5960 valid data points for each set. Using the NLTK (Natural Language Toolkit) python library, we extracted word frequency as a bag of words then tested a variety of different feature selection methods described in the "Bayes Net" section below. As stated in our task description, we set out to optimize a machine learner on the "all-time" top post dataset. This learner was later tested with the datasets from different time-periods to investigate the generalizability of our model.

Machine Learner Results

1. Bayes Net

While we initially began testing with a Naive Bayes learning model, our API-limited dataset performed and trained quickly enough on the Bayes net learner that we moved forward with the more robust learning model. The Bayes Net learner in Weka offered a good combination of high accuracy and quick learning for our bag-of-word approach.
Our first task was to find an optimum number of words to include in our bag of words (figure 1). By default, these words are simply the most commonly occurring tokens (words or punctuation) in the joint dataset. We moved forward with feature selection using 2000 words for quicker learning times, and since increase in accuracy with more words was marginal, as shown in figure 1.

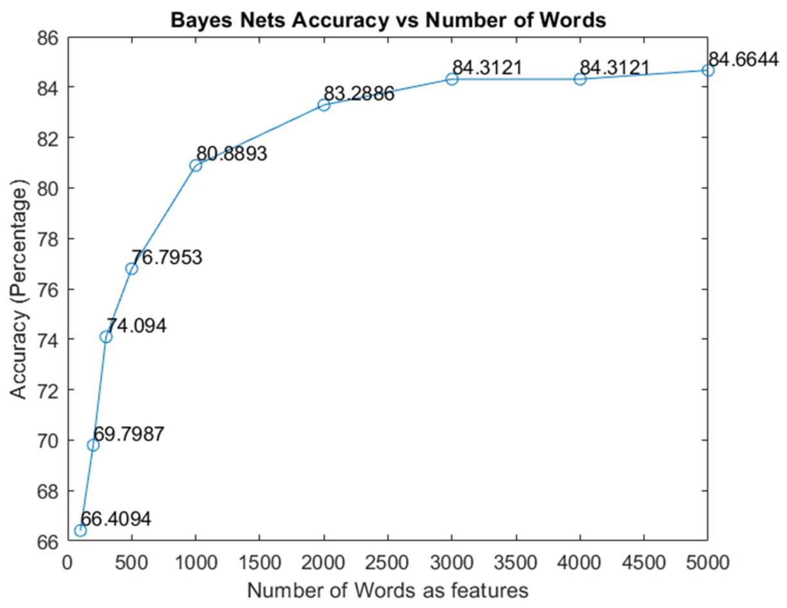


Figure1: Graph showing accuracy increasing in Bayes Net as the number of words increases, with diminishing returns.

Our next task was to determine which feature selection techniques work best for our data. We compared the change in accuracy of our default 2000-word dataset when we applied each feature and plotted it in table 1. While eliminating stop words, eliminating punctuation, and using bigrams is generally considered a good approach for traditional text classification, the short length of our titles meant that every punctuation mark and stop word plays a very large role in classification. Our features worked best when they had high occurrences and were therefore highly generalizable. Various combinations of features other than the stemming yielded even poorer results. Table 2 shows the most informative features, derived from a classifier subset evaluator while using the Bayes Net classifier. Attributes tended to be quite intuitive; punctuation like '?' tend to denote questions appropriate for a subreddit like "AskReddit", and 'trump' tends to denote politics. While the most useful features do not always have an obvious class, they occur in many examples and work with the rest of the features to boost accuracy.

Feature vs Default (2000 words)	**Change in Accuracy**
Stemmer	+1.1242%
Stopwords	-0.3691%
Punctuation	-1.6275%
Bigrams	-3.3557%

Table 1: Feature Adjustment Techniques and their effects on overall accuracy. We chose to stick with just stemming as it had positive impact.

Most Informative Features (ClassifierSubsetEval)

Rank	Word	Rank	Word	Rank	Word
1	?	6	.	11	]
2	what	7	,	12	your
3	trump	8	to	13	in

4	you	9	my	14	:
5	The	10	[	15	that

Table 2: Most informative features shows that some features were more powerful than others.

As a final test, we compared the use of stemming between the 2000 and 5000-word datasets. Surprisingly, the 2000-word dataset with stemming outperformed the 5000-word dataset with stemming with accuracies of 84.4128% and 79.9329%, respectively. We expect this difference to be attributed to overfitting, where less useful features in the 5000-word dataset are generalized with the stemmer — the 2000th word occurs 6 times in the dataset while the 5000th word occur only twice. The 2000-word dataset with stemming performs with an accuracy just under the 5000th unstemmed dataset (-1.2516%). Ultimately, we moved forward using the 2000-word dataset with the expectation that higher frequency words and stemming will make it more generalizable when we test it with our other datasets.

Confusion Matrix

Classified as ->	**worldnews**	**AskReddit**	**movies**	**politics**	**funny**	**nba**
worldnews	712	1	29	204	46	5
AskReddit	5	969	3	1	11	5
movies	18	5	836	21	71	35
politics	115	2	15	848	14	4
funny	8	11	35	15	887	32
nba	8	10	24	24	162	779

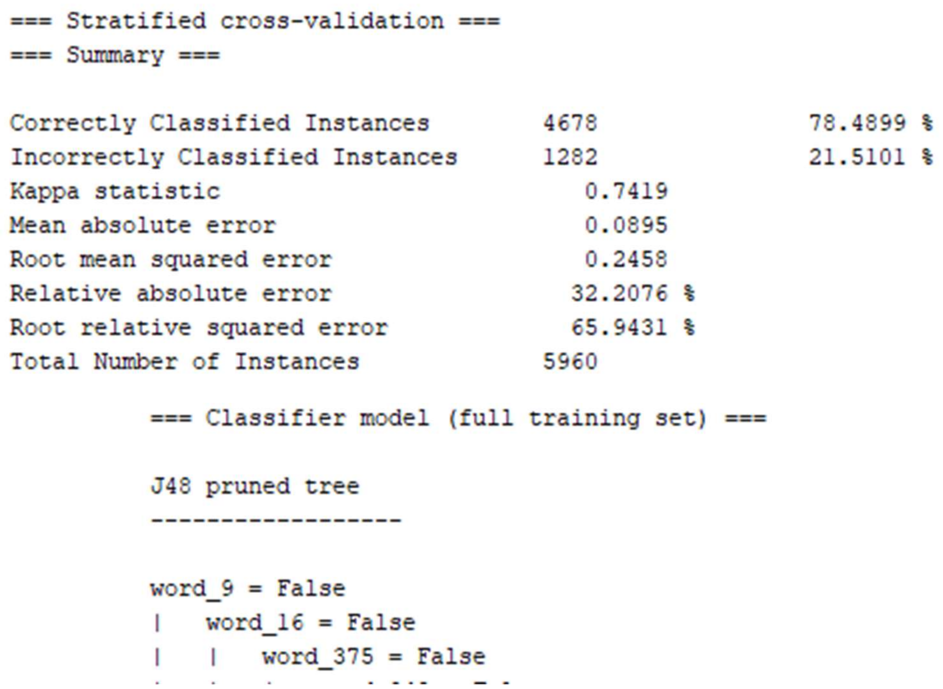
Table 3: Confusion Matrix shows that accuracy was very high for all classes.

For the most part, the classifier was quite accurate in classifying different posts, however there was some confusion in whether or not politics should be classified as world news and whether or not something mentioned in the NBA was funny. This can explain why our Bayes classifier doesn't achieve even higher accuracy, as there is always some grey area in where certain topics belong.

2. Decision Tree (J48)

Like the bayes net above we used the J48 decision tree with 10-fold cross validation from Weka on our pre-processed data. This method was relatively slow taking around 3 minutes to build the tree and 45 minutes to perform the cross-validation. The results shown in figure below show an accuracy of only 78.5% which was the worst of all three models. This combined with its long run time makes the model the least effective of all three models. The images below show the root node and overall accuracy of the tree. Word 9 in this case is a "?". J48 proved to be quite slow, so we decided to optimize other machine learners instead.

Figure 2: Accuracy and error measurements along with the root node for the J48 decision tree algorithm in Weka.



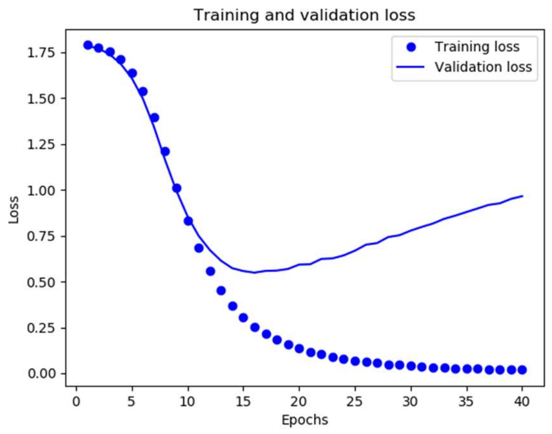
```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        4678               78.4899 %
Incorrectly Classified Instances      1282               21.5101 %
Kappa statistic                          0.7419
Mean absolute error                      0.0895
Root mean squared error                  0.2458
Relative absolute error                 32.2076 %
Root relative squared error             65.9431 %
Total Number of Instances             5960

          === Classifier model (full training set) ===

          J48 pruned tree
          ------------------

          word_9 = False
          |   word_16 = False
          |   |   word_375 = False
```

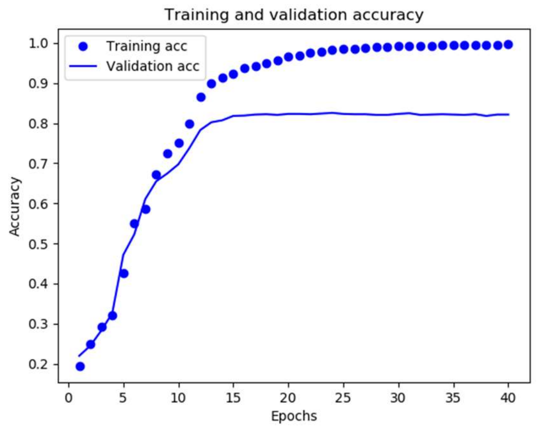
3. Neural Nets

In using neural nets to create a classifier, we used the tf.keras API to train different models. The inputs to the neural network were identical to those given to the other machine learners for fair comparison. Each node in the input layer received 2001 inputs, one for each of two thousand word counts and one for the bias. This input layer was then fed into the hidden layers, which both used relu activation functions. The output layer consisted of six nodes, one for each classification value and a softmax activation to denote which class the neural net believe to be the correct class.

In tuning the neural network, some hyperparameters had very large influence on validation accuracy. Hidden layers were a very good example of overfitting, as adding more hidden layers only decreased the validation accuracy. As can be seen in the below figure which represents accuracy over time particularly with 4 hidden layers and a layer size of 16 nodes, even while the training accuracy increased, the validation accuracy levelled off and even decreased slightly. This meant that over time, the model began to overfit data. This is something we want to avoid.

Figure 3: Training and Validation Loss for Neural Nets showing accuracy as learner with 4 hidden layers trains on dataset. Validation Accuracy increases but exhibits diminishing gain in validation accuracy over time. After around 20 epochs, the validation accuracy even decreases slightly, which shows that the data is being overfit.





Adding hidden units also presented similar results to adding more hidden layers, with a maximum accuracy found with 16 hidden units. Varying dropout slightly harmed the accuracy of the neural network and caused it to learn more slowly than the other examples, so we opted to use an early stop at 20 epochs instead.

After comparing variations of different parameters, we decided to use a hidden layer size of 16, an early stop at 20 epochs, a batch size of 512, and 1 hidden layer to attempt to maximize validation accuracy. This resulted in a validation accuracy of 85.525%. While this is slightly better than our Bayes classifier, if we compare the accuracies with a chi-square test, we obtain a P value of 0.08, which is not statistically significant. We also preferred to use Bayes for its better interpretability.

Results on Differing Sets

We tested the Bayes Net classifier trained on 2000 features described above using the datasets of top posts from the past week, month, and year. Overlapping posts were removed from the dataset, which reduced the size of the datasets by 2%, 25%, and 59%, respectively. We report the accuracies of model testing in table 3. As expected, top posts from larger time periods are classified with higher accuracy by our “all time” top posts classifier; however, even the top posts from a recent week was classified with a respectable accuracy. An important observation is that the “year” dataset tested better than the “all time” dataset on its own classifier. The two time periods are highly similar, and the “year” dataset is likely more cohesive than the “all time” dataset. Our 1000 post API limit might be too small to accurately define the distribution of subtopics in every subreddit from all of reddit history, and these results indicate that a classifier trained on yearly data could provide an even higher cross-validation accuracy.

Dataset (top post time period)	**Accuracy**
week	78.4596%
month	82.9826%
year	85.1665%
All time (cross-validation)	84.4128%

Conclusion

We trained machine learning models that classify subreddits based on title strings using a Bayes net, J48 decision tree, and Keras neural net with accuracies of 84.4%, 78.5%, and 85.5% respectively. While neural nets presented higher accuracy, we believe Bayes classifiers are better for their interpretability. We tested our Bayes net model using posts taken from different time periods to determine the generalizability of our model. Our all-time top posts model achieved accuracies of 78.5%, 83.0%, and 85.1% with datasets of top posts from the previous week, month, and year. Performing stemming with our bag-of-words approach improved accuracy, but we expect that we higher accuracies will come from more complex feature selection (sentence structure, grammar, valence).   In the future, we could try increase our accuracy by using word embeddings in our classifiers as well as other preprocessing methods such as TF-IDF. Perhaps it might also be useful to look into clustering, as subreddit topics can often overlap and clustering can help to find other ways to group data together.

General Work Distribution

While everyone in the group took part in almost every aspect of the project, a rough breakdown of responsibilities could be shown as follows:

- Bennett worked on training and optimizing Naive Bayes.
- Jason worked on preprocessing and optimizing neural nets.
- Michael worked on the website and optimizing decision trees.