Christie Jeung, Jiham Lee, Larry Wang
EECS 349: Machine Learning
Northwestern University

Predicting what Makes a Popular Reddit Post through Machine Learning Algorithms

Our task is to predict the scores of Reddit posts (defined as upvotes less downvotes) based on post attributes including date and time of post, comments, and receipt of gold. The model consumes a Reddit post and predicts whether it will be high-performing (ultimately defined as surpassing a threshold score of one standard deviation above the mean of the training data, though we also experimented with setting it as the mean of the training data). This inquiry is valuable in that optimizing the engagement levels of content posted on Reddit or other popular social media platforms is highly beneficial to content creators. Stakeholders range from the advertising industry and corporate commercial interests to individuals interacting with communities defined by subreddits.

We gathered data from 6000 different posts from 6 different subreddits to for training and testing. Weka was used as our approach to model our data using the algorithms: ZeroR, J48, MLP, BayesNet, Nearest Neighbor, IBK, LMT, and RepTree. We held out 1000 posts to test on. The features we used are time of day posted, day of the week posted, number of comments, gilded or not, and post score. We found that our solutions did not perform as accurately as we would have hoped as shown in the figure. Most models did not improve significantly over ZeroR, if at all. This is because no combination of features effectively differentiated between high-scoring posts and the rest of the sample. The best and most consistent algorithm that was used to train the data sets was BayesNet, Weka's Bayesian networks algorithm.

Data

Our data collection was motivated by the need to assess the impact of different post characteristics on the success of a post. We visited five different subreddits (funny, worldnews, gaming, movies, sports) and pulled the top 1000 posts from the json file of the page. We then appended each feature gathered into a corresponding list, creating one column for each feature in a CSV output. Refer to Appendix I for the python script performing this process. The final features of our data are as follows. Each observation corresponds to one post:

- **Post submission time of day:** categorized into 8 buckets lasting three hours each.
- **Post submission day of week**: i.e. Monday, Tuesday, Wednesday, etc.
- **Comments**: number of comments received on post
- **Gilded**: quantity of gold received by post
- **Class**: classification of post as successful or unsuccessful based on whether post score surpasses the threshold, which is defined as one standard deviation above the mean of the training data. Coded as "Y" or "N".

There are a few significant limitations to our data collection that merit mentioning. The first is that a limit in the amount of data possible to extract through Reddit's API in one call restricts us to 1000 posts per request. Even more critically, the other is that technical challenges in mastering PRAW (Reddit's API wrapper for python), combined with time limitations, prevented us from accessing a broader range of attributes which would have expanded our modelling power (e.g. user data and flair tags). For similar reasons, we also did not succeed in implementing plans for natural language processing analysis on post title and description through Tensorflow or a python script enabling bag-of-words analysis.

Christie Jeung, Jiham Lee, Larry Wang
EECS 349: Machine Learning
Northwestern University

Methodology

The approach we used was to use different machine learning algorithms of classification through Weka over various combination of thresholds to classify our data. There were four different approaches we took to give classification to our data, most of which centered around defining the classification threshold as one standard deviation above the mean (see Appendix 2). We validated the trained model using 10-fold cross-validation. We reserved the top 1000 posts from the relationships subreddit for testing, with the classification threshold as one standard deviation above the mean. We used Weka to train and test our data using the algorithms shown in Figure 1, included below:

Technique	**(1)**	**(2)**	**(3)**	**(4)**
ZeroR	85.77**	87.66	79.29	64.08
J48	85.77**	87.92**	78.99	64.62*
MLP	83.47	87.54	78.95	64.52*
BayesNet	85.77**	87.84*	79.45*	65.36*
Naïve Bayes	82.77	85.71	78.67	65.52*
Logistic	85.67	87.76*	79.72**	66.02**
SimpleLogistic	85.27	87.84*	79.68*	66.02**
IBK	76.25	79.32	69.45	57.44
LMT	85.27	87.80*	79.68*	66.02**
REPtree	85.47	87.56	78.69	63.47

Figure 1. Accuracy (in percentages) of machine learning techniques across datasets. (*) indicates technique's accuracy is greater than or equal to ZeroR. (**) indicates technique yields maximum accuracy for this range of approaches.

Results and Conclusion

Our models produced used different algorithms through Weka to classify our test data, and did not have much improvement over ZeroR. See Figure 1 above for accuracies of models against training data with 10-fold cross-validation. The most consistent improvement in accuracy over ZeroR was through the use of Bayesian networks. Even the Bayesian networks model, when tested against the reserved testing data, performed at ZeroR, 65.46%. We can test for the statistical significance of these marginal improvements by computing Z-scores testing for the null hypothesis that ZeroR and BayesNet return the same accuracies, which can be expressed by the following formula (p1 and p2 are the accuracies as proportions of the sample for ZeroR and

BayesNet respectively, and n1 and n2 are the sample sizes; in this case n1 and n2 are equivalent):

$$\frac{(\bar{p}_1 - \bar{p}_2) - 0}{\sqrt{\bar{p}(1 - \bar{p})\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}}$$

Figure 2, shown below, shows that improvements in accuracy between BayesNet and ZeroR is statistically insignificant across datasets.

Accuracy: ZeroR	85.77%	87.66%	79.29%	64.08%
Accuracy: BayesNet	85.77%	87.84%	79.45%	65.36%
Sample size	998	4991	4989	4991
Z-score	0	-0.2747	-0.198	-1.3405
Statistically significant (|Z|>1.96)	**No**	**No**	**No**	**No**

Examining the models generated through each machine learning technique and looking to identify the features most important to classification, we find that no features meaningfully aid in predicting score. As discussed in the data section, this is most likely attributable to our lack of success in collecting data outside of the limited variables available through .json retrieval in our python script. Therefore, our data is wanting in more attributes (e.g. flair tags, user likes, user comments, performance of other posts) as well as breadth of time scope (restrictions through the .json method limit data retrieval to relatively new data from the last 6 days, which is more recent than preferable for the purposes of training). These are data qualities which we would have acquired through PRAW had time permitted, and would likely have improved the predictive power of our model. Other interesting potential extensions of this project include natural language processing of the post title and description, image processing of post images, and constructing a model to consume a reddit post and output a predicted score number (a numeric value) instead of a binary classifier indicating high score performance.

Methodology creation: Christie Jeung, Larry Wang, and Jiham Lee
Data collection and preprocessing: Larry Wang
Methodology implementation: Christie Jeung
Website: Jiham Lee

Christie Jeung, Jiham Lee, Larry Wang
EECS 349: Machine Learning
Northwestern University

Appendix 1

```
import urllib2
import json
import time
import datetime
import calendar
from datetime import datetime
import numpy as np
#gaming, worldnews, sports, movies, funny

hdr = {'User-Agent': 'osx:r/gaming.multiple.results:v1.0 (by /u/<xerxen18>)'}
url = 'https://www.reddit.com/r/relationships/top/.json?sort=top&t=all&limit=100'
req = urllib2.Request(url, headers=hdr)
text_data = urllib2.urlopen(req).read()
data = json.loads(text_data)
data_all = data.values()[1]['children']

article_title = []
article_flairs = []
article_date = []
article_comments = []
article_score = []
article_gilded = []
article_id = []
scoreMean = 0
scoreThreshold = 0

counter=0
while (counter < 10):
        counter=counter+1
        time.sleep(2)
        last = data_all[-1]['data']['name']
        url =
'https://www.reddit.com/r/relationships/top/.json?sort=top&t=all&limit=100&after=%s' % last
        req = urllib2.Request(url, headers=hdr)
        text_data = urllib2.urlopen(req).read()
        data = json.loads(text_data)
        data_all += data.values()[1]['children']
for i in range(0, len(data_all)):
        article_title.append(data_all[i]['data']['title'])
        article_flairs.append(data_all[i]['data']['link_flair_text'])
        article_date.append(data_all[i]['data']['created_utc'])
        article_comments.append(data_all[i]['data']['num_comments'])
        article_score.append(data_all[i]['data']['score'])
        article_gilded.append(data_all[i]['data']['gilded'])
```

```
    article_id.append(data_all[i]['data']['id'])

scoreMean = np.mean(article_score)
scoreThreshold = np.std(article_score) + scoreMean
for i in range(0, len(data_all)):
    parsed_date = datetime.utcfromtimestamp(article_date[i])
    hour = parsed_date.hour
    weekday=calendar.day_name[parsed_date.weekday()]

    if hour>=12 and hour<15:
        timeofday="12pm-3pm"

    if hour>=15 and hour<18:
        timeofday="3pm-6pm"

    if hour>=18 and hour<21:
        timeofday="6pm-9pm"

    if hour>=21 and hour<24:
        timeofday="9pm-12am"

    if hour>=0 and hour<3:
        timeofday="12am-3am"

    if hour>=3 and hour<6:
        timeofday="3am-6am"

    if hour>=6 and hour<9:
        timeofday="6am-9am"

    if hour>=9 and hour<12:
        timeofday="9am-12pm"

    if article_score[i]>=scoreMean:
        good_bad="Y"

    if article_score[i]<scoreMean:
        good_bad="N"

    print str(timeofday)+","+str(weekday)+","+
str(article_comments[i])+","+str(article_gilded[i])+","+str(good_bad)
```

Appendix 2

1. Source: gaming subreddit. Classification threshold: one standard deviation above the mean.
2. Source: funny, news, gaming, worldnews, and sports subreddits. Classification threshold: the average of each subreddit's threshold as defined in (1).
3. Source: funny, news, gaming, worldnews, and sports subreddits. Classification threshold: one standard deviation above the mean of the five subreddits as one combined sample.
4. Source: funny, news, gaming, worldnews, and sports subreddits. Classification threshold: sample mean.