Reddit Post Classification using Machine Learning (RedMine)

Prepared By:

Harrison Walters (V00807994), Cameron Day (V00800896), Lance Chaplin (V00782427), Ryan Hutton (V00812291)

Department of Computer Science, University of Victoria

Abstract

This project examines posts submitted to the website Reddit, and determines whether they will become popular or not within the near future (several hours after posting). We used a prebuilt library for the extraction of data (PRAW) and a prebuilt library for the classification of data (Scikit-learn). After experimenting with various feature vectors and classifiers, it was found that a Random Forest classifier with a depth of 4 can classify posts using 20 minutes of data with a high degree of accuracy (87%). Our findings could be useful for frequent users of the reddit community, looking to make modules to increase the success of their submissions.

Acknowledgements

This Paper was completed as part of the final project aspect of the Spring 2017 Data Mining course. We would like to extend our thanks to Dr. Alex Thomo and Dr. Yudi Santoso for all their guidance during the course period. We would also like to thank the users of Reddit as—without them—this data would not be able to be collected.

Table of Contents

1 Introduction

Reddit [1] is a massive online user discussion website that sees over 80 billion page views and 70 million submissions each year [2]. Registered users of the site can submit content such as text posts, or direct links (which can then be gifs, webms, etc.) and then comment on posts. Users can also upvote or downvote on submissions which effectively scores the post and determines their position on the site's pages. Submissions with the most positive score appear on the front page or the top of a category otherwise known as subreddit. A subreddit is an area of interest that acts as a category for post types, examples include funny, world news, pictures, gaming, jokes, etc.

User accounts accumulate post-karma and comment-karma from submitting posts and comments and can be viewed on their profile. It is also possible for users to be gifted "Reddit gold" by other users if they believe a post or comment is of high-quality or funny. This process is known as "gilding" and requires users to purchase gold in order to give it, which financially supports the website. Although there is no reward for users accumulating lots of karma and gold, there is somewhat of an internal competition among many users who see themselves as superior to those with low karma and gold.

RedMine is Reddit Post Classifier that utilizes machine learning to analyze and predict which posts are most likely to be popular and hit the front page of Reddit. It scores posts based on the type of content (text, image, video, etc.), the amount of upvotes, downvotes, comments, word-count and time of post. RedMine monitors and adjusts its prediction based on real time intervals of 10 minutes, up to at most 60 minutes. With these attributes, it can then predict whether or not a post is likely to hit the front page, or in other words, be a popular post.

The objective of RedMine is to allow users to get to the popular posts early and comment on them.Comments on front page posts have shown to be more likely to be heavily upvoted and gilded than those not on the front page. Thus RedMine allows for users to increase their chances of gaining plenty of comment-karma and Reddit gold for their profiles to show off to others.

2 Related Work

As of late, the Internet has become integral in the world of data mining. Its increased presence has prompted much interest from professional and personal aspects alike. In this report, we present our own aspect towards this phenomenon: mining the metrics of Reddit posts in order to determine which type of posts will reach front page status as opposed to other types.

In the professional world, there are many approaches towards this topic. The most common of these approaches being for marketing and advertising. A prime example of this is Amazon's patent for anticipatory shipping [3] in which they ship an item "to the destination geographical area without completely specifying the delivery address at time of shipment," which is based on predicting the orders of various customers. These predictions are founded in mining the prior Amazon activity—"including time on site, duration of views, links clicked and hovered over, shopping cart activity and wish lists"— of users from given geographical locations and anticipating what those users will buy. This is a much grander scale of Amazon's suggested related items.

On the other hand, in the personal realm, people around the globe want to know more about the metrics produced on various social media platforms and what they correlate to. In the process of searching for projects that others have done, we came across a blog post describing the exact same process that we were working on, but on a larger scale [4].

This blog monitored the front page of Reddit for twenty-two days, scanning over four and a half million data points from five separate subreddits. The programmers working on this project soon came to the conclusion of ten findings ranging from "Starting at 9am PST is the fastest time for getting upvotes" to "The number of comments of posts on the front page are 5.5 times higher than of posts in the top 100 on average."

Along with this, the programmers were able to determine the most popular subreddits, complete a sentiment analysis of the various subreddits, and determine the average lifespan of a front page post.

All of this information is integral in a professional process called "Reddit scraping," which is used by news agencies in order to determine which of their stories will be viral—or popular. For example, a post that achieves a net score of two-thousand points on Reddit and makes the front page can be expected to make it to a news agency such as CNN or Buzzfeed within a few hours.

Whether it is used for professional or personal purposes, data which has been mined from user activity on consumer-driven sites and social media has proven to be invaluable for the world.

3 Data Preprocessing

We used two libraries to implement our project : scikit-learn for classifier methods, and the PRAW API to obtain data. The classifiers provided in scikit-learn work exclusively with numerical attributes, however a significant portion of the data we collected was categorical (for example, the "post-type" attribute comes as "image", "text", "url", or "video"). We used an implementation of a **label-encoder** to

transform each K-categorical attribute into a numerical range [0 to K-1] so that it was compatible with scikit-learn.

After running the miner for 2 days, we collected a dataset of approximately 20,000 points, of which 750 were “soon-to-be-popular” and the remaining 19550 were unpopular. This unmodified dataset would have been unideal because the number of ‘popular’ posts made up approximately 4% of the data and our classifiers would have run into **class distribution** issues with a ratio of 1:50 (would have resulted in skewed accuracy). To alleviate this issue, we truncated a significant portion of the “unpopular” noise, and reduced the set from 20,000 points down to 1,000 points so that the ratio of classes (popular vs unpopular) approached 1:1.

Normalization of the attributes was essential, since the relative scales of the attributes varied wildly. The attribute “comment karma” could have easily approach values of 1,000,000 whereas the “time-posted” attribute did not exceed 24. This would have caused “comment karma” to be the dominating attribute for classifiers such K-Nearest-Neighbours, which use Euclidean distance to measure similarity. Attributes were simply scaled to a range of [0 to 1].

Attributes used in each data point, for classifying data

Attribute	**Description**	**Why this might be useful?**	**Possible Values**
type	The type of the content of the post.	Are images more popular than text-posts? etc	**‘Image’, ‘video’, or ‘text’**
comment-karma	Measure of a user’s overall contribution to ‘discussions’ on reddit	Users with a history of popular comments might be likely to produce a popular post	**Numerical**
link-karma	Measure of how popular a user’s previous posts are	Users with a history of popular posts might be likely to produce another popular post	**Numerical**
subreddit	Category of post	Likely useful in conjunction with other attributes	**String**
time-posted	What time during the day the post was submitted	Posts early in the morning / late at night might be ignored	**Numerical**
num-words	Length of the body	Long posts might be	**Numerical**

	of the submission	ignored	
(every 10 minute interval) t0-comments	The number of comments in reply to the thread	Posts with more comments or fast growing might be more likely to be front page material	**Numerical**
(every 10 minute interval) t0-upvoteratio	Ratio of upvotes to downvotes	Measure of how positive the post is	**Percentage**
(every 10 minute interval) t0-upvotes	Total number of upvotes the post has	Measurement of how many users like the post	**Numerical**
(every 10 minute interval) t0-downvotes	Total number of downvotes the post has	Measurement of how many users dislike the post	**Numerical**
(every 10 minute interval) t0-num_gold	Total number of gold given to the post	Posts with more gold might be more likely to appear on the front page	**Numerical**
(every 10 minute interval) t0-score	Difference between upvotes and downvotes	Posts with a more positive difference might be more likely to appear on the front page	**Numerical**

4 Data Mining

The library we used to collect data was the PRAW (Python Reddit API Wrapper) API, which is a python API provided by the developers of Reddit itself. One of the significant challenges to mining the data for this project was the **temporal element** of the data (how new posts were “changing over time”). Since the data points were not instantly available to us, we had to constantly re-evaluate web posts in intervals of 10 minutes. We collected data such as “number of comments after 10 minutes”, “number of likes after 20 minutes”, etc. Data points weren’t considered “complete” until they had been evaluated 6 times, over a span of 1 hour, and then labeled 24 hours later (“popular” or “unpopular”). To implement this, we used a series of threads (one for each 10 minute interval), each of which had a queue of “data points to re-evaluate”. The threads were constantly re-awakened and evaluated every post on its queue, and then passed those evaluated posts to the next thread’s queue. Once posts reached the final queue, they were saved to disk in

bunches of 500 (this was done to mitigate **crashes** of the data miner, which were happening often due to random network connectivity loss). **Threads were required** because the PRAW implementation of the "collect new posts" method was asynchronous, and would cause the program to sleep until a new post arrived. This was problematic since the posts needed to be re-evaluated after *exactly 10 minutes,* and if the program was in a sleeping state for a long period of time, the temporal data would be inconsistent (might not get a chance to reevaluate posts until *12 minutes* instead of the desired 10 minutes). In addition, we also collected a handful of "general attributes" relating to the post - such as the type of post, renown of the poster, hour of the day it was posted, etc.

New posts were arriving to reddit at a rate of about 5 per second, which caused our "waiting queues" to become large, but did not cause any issues thanks to our rugged implementation.

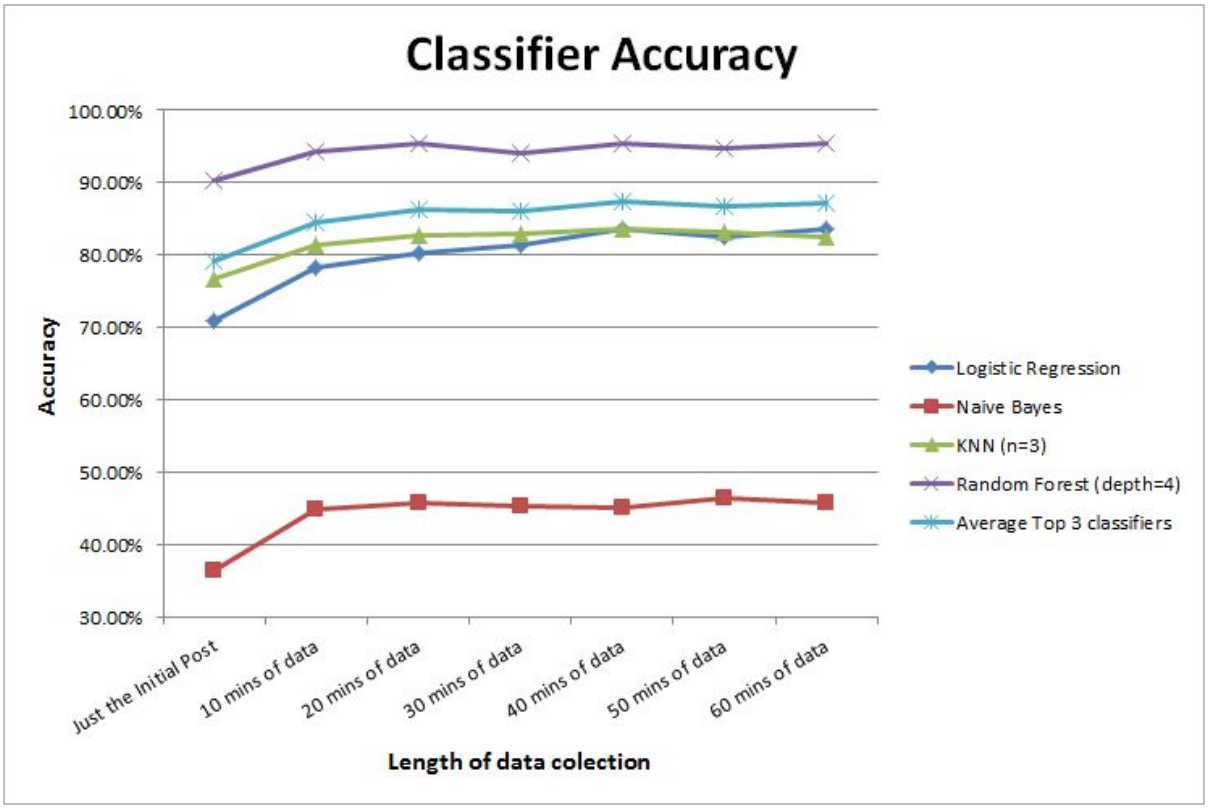
5 Evaluation

Ultimately, we were interested in whether the post would be labeled "popular" or "not popular", so we used four different binary classifiers from the different families of classifiers (probabilistic, decision-tree, clustering …) and recorded their respective accuracies. We were also interested in what the **minimal amount of temporal data necessary** was, to classify the post with a high degree of accuracy. We collected 60 minutes of data, but could it be done with 20 minutes of data? 10 minutes of data? We also computed the average between the top 3 classifiers, to note whether there was a general increase in accuracy with a certain amount of temporal data, irrespective of the classifier used. To compute the accuracy scores, we used the standard **k-fold cross-validation** with a fold size of 10.

Accuracy of the Classifiers

Amount of data used	**Logistic Regression**	**Naive Bayes**	**KNN** (n=3)	**Random Forest** (depth=4)	***Average*** *Top 3 classifiers*
Just the initial post	71.0%	36.5%	76.6%	90.2%	**79.2%**
10 mins of data	78.3%	44.8%	81.3%	94.3%	**84.6%**
20 mins of data	80.3%	45.8%	82.8%	95.5%	**86.2%**
30 mins	81.3%	45.3%	82.9%	94.0%	**86.1%**

of data					
40 mins of data	83.6%	45.16%	83.5%	95.3%	**87.5%**
50 mins of data	82.5%	46.5%	83.16%	94.83%	**86.8%**
60 mins of data	83.5%	45.8%	82.5%	95.5%	**87.2%**



In the above table and graph they show the accuracy of all the classifiers that were applied to the data over different collection period times, as well as the averages for the three best classifiers to show the ideal data collection period to obtain the highest accuracy. The graph shows that the random forest classifier with max depth equaling 4 gives the best results, with the lowest accuracy of 90% during the initial post with an increasing accuracy to 95.5% after 20 minutes. In general after a 20 minute period of data collection the accuracy of all four classifiers do not improve any great deal, but instead alternate within a range of 2% of their best calculated accuracies; however at around 40 minutes of data collection gave the best results overall for classification accuracy, this could mainly be do to more similar posts being made during that time which get upvoted quicker.

So why are some classifiers better at processing the type of data collected from Reddit, while others like naive bayes suffer with the loss of accuracy. Well, with naive bayes they are quite good at small data sets with their decision tree based approach, however as the data set increases in size and complexity the decision tree matrix also increases there is a loss of accuracy [10]. This is the reason why we see such a loss in classification accuracy when compared to the other three methods, for the data set that was collect from Reddit was quite large.

The other three classifiers all did roughly the same job based on their accuracy, when compared with how the naive bayes performed. Even though the remaining three classifiers all did a good job, it was random forests that did the best job. The potential reasons to why random forests performed the best in our case is because it works quite well on larger datasets, and since random forest are based on the group of items viewed as a whole rather than individually of unpruned classification or regression trees [9]. These ensemble of unpruned classification are created by using bootstrap samples of the training data as well as random element selection in the tree [9]. Finally the predictions for the classification are made by aggregating the estimated predictions of the ensemble, and because of this random forests are a very powerful tool and is one of the most accurate methods for classifications so far [9].

6 Conclusion

From this data mining experiment, we have shown that a Random Forest with a maximum depth of 4 was ideal for the task, using the handful of features that we chose. Ultimately, it was **not the actual content of the post** that was important in determining its future popularity, but rather how the “buzz” surrounding the post changes over time (comments, etc) and a few parameters about the structure of the post. A post can be classified (“soon-to-be-popular” or “will-die-out”) with a high degree of accuracy using **the initial post itself**, or, for an even more accurate result, a mere **20 minutes of data**.

Reddit users who frequently submit articles and images could make a **module** using our findings, that uses a RF classifier and scans their post for a few features to tell whether it is worth posting or not. Additionally, frequent commenters who are looking to “ride” popular posts could have the module scan incoming posts to tell whether commenting is worthwhile or not.

7 References

[1] “Reddit.” [Online]. Available: https://www.reddit.com/

[2] "Reddit in 2015." *Upvoted*. Reddit, 31 Dec. 2015. [Online]. Available: https://redditblog.com/2015/12/31/reddit-in-2015/

[3] : Ulanoff, Lance. "Amazon Knows What You Want Before You Buy It." *Predictive Analytics Times*. Mashable, 27 Jan. 2014. [Online]. Available: http://www.predictiveanalyticsworld.com/patimes/amazon-knows-what-you-want-before-you-buy-it/3185/

[4] : "We analyzed 4 million data points to see what makes it to the front page of Reddit. Here’s what we learned.4." *DataStories.*, 4 Feb. 2016. [Online]. Available: https://blog.datastories.com/blog/reddit-front-page

[5] Pedregosa, Fabian, Gael Varoquaux, Alexandre Gramfort, and Vincent Michel. *Scikit-learn*. Computer software. *Scikit-learn*. Vers. 0.18. [Online]. Available: http://scikit-learn.org/stable/

[6] Boe, Bryce. *PRAW.* Computer software. Vers. 4.0.0+. [Online]. Available: https://praw.readthedocs.io/en/latest/

[7] *RedditAPI.* Computer Software. *Reddit*. [Online]. Available: https://www.reddit.com/dev/api/

[9] Kohavi, Ron. *Scaling Up the Accuracy of Naive-Bayes Classifiers: a Decision-Tree Hybrid*. Tech. N.p.: n.p., n.d. [Print]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.462.9093&rep=rep1&type=pdf

[10] Svetnik, Vladimir, Andy Liaw, Christopher Tong, J. Christopher Culberson, Robert P. Sheridan, and Bradley P. Feuston. "Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling." *Chemical Information and Modeling*. 43 (6), 2003, pp. 1947-958. [Online]. Available: http://pubs.acs.org/doi/abs/10.1021/ci034160g