Predicting the Popularity of Reddit Posts

General Machine Learning

Ahmed Shuaibi {ashuaibi@stanford.edu}

***Abstract*—Accurately predicting the attention a Reddit submission receives elucidates the relative significance of inherent and engineered post features, thereby granting us a greater understanding of popularity on the site as a whole. While predicting the popularity of articles in news domains is well charted, the task of predicting the popularity of Reddit submissions is not as significantly explored. Furthermore, any attempts to do so do not utilize natural language processing techniques and features such as the use of existing, well-developed word embeddings. In this paper, the task of predicting Reddit submission popularity given a post is tackled. Sentiment classification of a post's text is obtained and utilized in addition to the GloVe embeddings of its component words. Linear, K-Nearest-Neighbors, and Random Forest Regression are utilized and assessed.**

Code: github.com/ashuaibi7/RedditPopularityPrediction

I. Introduction

Reddit is a popular discussion and content rating social website. Individuals can submit image or text posts that are constantly evaluated by the site's other users through a system of *upvoting* and *downvoting*. The amount of upvotes a post receives determines its popularity and thereby impacts its visibility on the site. A variety of factors play a great role in determining the popularity of a post.

Reddit is composed of a variety of subreddits, which are essentially themed forums with outlining rules that regulate the type of allowable posts among other things. While many subreddits allow for posts that contain images as part of the content, many only allow for posts entirely comprised of text. I will focus my popularity prediction on a popular text-post only subreddit: AskReddit. I utilize a variety of features derived from a post and its contents to predict its popularity determined by number of upvotes.

As input, the model begins with a Reddit post from AskReddit. I then use Linear, KNN, and Random Forest regression to ouput a predicted upvote *score*, the number of upvotes a Reddit submission recieves. This score indicates the popularity of a post.

II. Related Work

- Segal and Zamoschchin first tackle the prediction of Reddit post popularity as a classification problem. They define an upvote threshold that determines if a post is popular and consequently use the metadata of a post as features to classify it. Such metadata includes the subreddit a post belongs to, the author of the post, and the time of day the post is created. Among the post's text, they only use frequency of occurrences of words in the title as features [5] and do not explore more in depth NLP word embeddings.
- Terentiev and Tempest similarly tackle the problem as a binary classification task. Contrastingly, they focused on predicting popularity based off the initial 10 comments a post received. Working only with 2000 training examples, they engineered features such as sentiment based off the initial comments. They consequently utilize a variety of classification models such as SVMs, Logistic Regression, and KNN clustering to predict popularity [2].
- The work of Poon, Wu, and Zhang aims to recommend posts to users based off their preferences and other submissions they upvote. They do so using models such as KNN, K-means, and SGD. Their work elucidates several impactful features in a Reddit recommendation system, thereby urging their consideration in a popularity prediction system [3].
- Chandramouli, Moni, and Elango predict the popularity of posts given features such as subreddit, author engagement in comments, and sentiment classification. Instead of analyzing the post's title or text, they prioritize a natural language evaluation of a post's comments, weighting the frequency of positive/negative comments of a post in their prediction [8].
- White, Togneri, Liu, and Bennamoun evaluate how well different combinations of word embeddings capture meaning. They assess how much information is retained in a sum and mean of word embedings (SOWE/MOWE) of a piece of text. They assess this question with the task of semantic classification using SVMs. Their insight into the utility of MOWE will be used to capture a representation of a post's content in this project [6].

A. Dataset and Features

The raw data comes from Pushshift.io [4], which contains a collection of reddit posts from 2011 in which the data dumps are divided by month. The raw data contains one post per line formatted as a json entry with 96 fields to encompass the post's information. Among these fields are $created\_utc$, $author$, and $subreddit\_id$. I focused my analysis on the first 6 months of 2018, which comprised a total of 14 gigabytes of raw data. I proceeded to filter out all posts that did not originate from the subreddit $AskReddit$. Following this initial filter, I filtered out posts that had their content deleted or removed in addition to posts with empty content (blank post with no title) [1]. Finally, I retained post features that would best inform the popularity of a post. Particularly, these features are:

- $created\_utc$ - UNIX timestamp of the moment of the post's creation.
- $num\_comments$ - The number of comments that a post received.

- *gilded* - The number of times a post was gilded. Gilding is a when a user grants another user's post Reddit gold. This gold can be purchased and utilized to unlock extra features on Reddit.
- $over\_18$ - A boolean tag indicating if the post is intended for mature audiences.
- *title* - Raw text post title. All AskReddit posts only contain a title and no body text.

This left a total of $1,427,490$ Reddit posts. I split the data into $70/15/15$ train/dev/test and left the test dataset untouched until the final evaluation. I ensured that the data was randomly split (using a seed for reproduciblity).

I also use the Natural Language Toolkit (NLTK) Twitter dataset. Due to the lack of manually labeled sentiment reddit post data, I decided that I would use Twitter data for the sentiment classification of the post's title. Although it would not be perfect since tweets will not perfectly generalize to Reddit posts, I found it sufficient to use for the purposes of engineering a title sentiment feature. This NLTK Twitter dataset contains a set positive tweets and negative tweets that are manaully classified. The tweets are preprocessed by first removing hashtags and hyperlinks. Stopwords, very common words such as *the*, are then filtered out. Finally, the tweets are tokenized using an NLTK tokenizer.

B. Feature Engineering

I will walk through the above features listed, their modifications, and their use in the final models. With *created_utc*, I felt that it would be most useful if the day of the week and the time of day was encoded. Thus, I created two variables: *weekday* and *hour* that represented the day of the week $(0-6)$ and the hour of the day $(0-23)$ respectively. With linear regression models, I need to convert these categorical variables to ones that I can utilize. I therefore utilized dummy variables to encode these categorical variables as dichotomous variables. I kept the numerical features of *num_comments* and *gilded* as is. Just as with *weekday* and *hour* I utilized a dummy variable to encode the boolean $over\_18$.

With regards to *title*, I wanted to highlight three important aspects. The first was the length of the *title* itself in which I created the feature *title_length* that represented the character length of the *title*. The second was the sentiment that the *title* expressed: positive or negative. I created a basic sentiment classification model using logistic regression based on NLTK Twitter data. As features, I used:

- *pos_words* - number of positive words in a tweet
- *neg_words* - number of negative words in a tweet

The output of this classification model will be the sentiment of a particular tweet. I used the weights learned in the logistic regression model for the classification of AskReddit posts after obtaining the respective pos_words and neg_words features for each post. This new feature *sentiment* will take on a value of $+1$ for positive sentiment and -1 for negative sentiment. Just as with $over\_18$, I created a dummy variable out of this boolean variable to use.

Lastly, I utilize GloVe to obtain embeddings of all the words in a specific title [7]. GloVe has pretrained embeddings that are effectively informative vector representations of individual words. These embeddings are trained by evaluating co-occurrence between words in a large corpus of text data. This similarity and co-occurrence allows us to encode particular words similarly. Consider the words and co-occurrence vector distance measurements between them as an example in Figure 1.

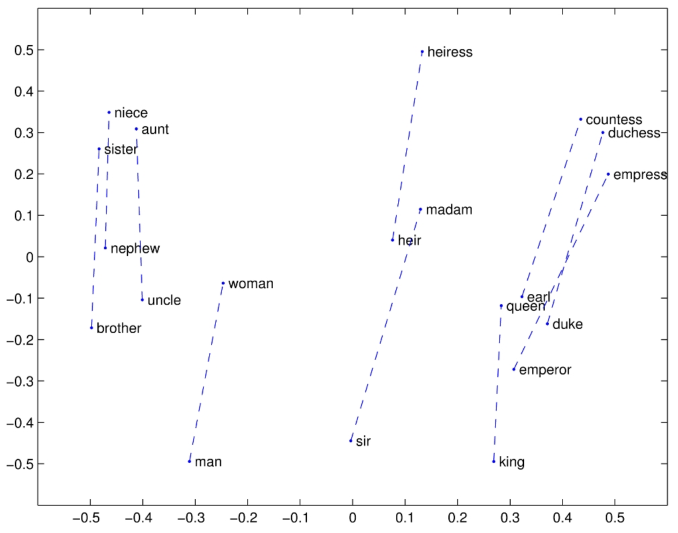

Fig. 1. visually is depiction of GloVe training with co-occurrence between words

I use the technique evaluated by White in *How Well Sentence Embeddings Capture Meaning* by obtaining the mean of all word embeddings (MOWE) for a particular Reddit post. This 300 dimensional vector sufficiently encodes a representation of the words in an AskReddit post's content. Note that standard tokenization and filtering applies before obtaining the word embeddings. Consider this simple example AskReddit post to see how this feature is generated:

"What are your new year resolutions?"

After tokenizing, the sentence is converted to lowercase and any required words are stemmed. Stemming is the reduction of related words to the root word they are derived from. It often includes cutting of affixes. For example, *"planning", "planned", and "plans"* would be stemmed to *"plan"*. In this example, *"resolutions"* is stemmed to get *"resolution"*. After this, we obtain the resulting list:

["what", "are", "your", "new", "year", "resolution", "?"]

The 300-dimensional representations of each of these tokens is obtained from the downloaded GloVe embeddings and averaged to get one final 300-dimensional representation for this AskReddit post.

In short, the final models utilize these existing and engineered input features:

- *hour* - Hour of day of post creation: 24-dimensional one-hot-encoded-vector.
- *day* - Day of week of post creation: 7-dimensional one-hot-encoded-vector.
- *num_comments* - Number of post comments: $\in \mathbb{N}$

- $gilded$ - Number of times gilded: $\in \mathbb{N}$
- $title\_length$ - Length of title: $\in \mathbb{N}$
- $over\_18$ - Is for mature audiences: 2-dimensional one hot encoded vector.
- $sentiment$ - Sentiment (positive/negative) of post content: 2-dimensional one-hot-encoded-vector.
- $embeddings$ - Mean of word embeddings of all words in post title: 300-dimensional vector.

III. Methods & Experiments

A. Metrics

I utilized two metrics in order to evaluate the models. The first was RMSE, defined as:

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\Big(\frac{d_i - f_i}{\sigma_i}\Big)^2}$$

Effectively, this a measure of the standard deviation of the residuals.

I also used $R - squared$ to measure how well a regression line fits the data:

$$R^2 = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - \bar{y})^2}$$

Effectively, this indicates how well the dependent variable of a Reddit post (the upvote score) is explained by the independent variables (highlighted above).

B. Models & Experiments

Prior to an in-depth ablative analysis, I tested all these models with and without the incorporation of the GloVe word embedding data. Therefore, I will highlight their performances with and without the incorporation of the 300 dimensional vector encoding of the title. As a simple baseline model, I predicted the score of a post to be the median of the scores of the post in the training dataset. This simple baseline gives an *RMSE* of 645.19 and an R^2 value of 0.001.

1) Linear Regression: Linear Regression with an ordinary least squared loss objective and default learning rate of 0.0001 was used:

$$Cost = \sum_{i}^{n} ||y_i - \hat{y}_i||_2^2$$

Evaluating the model gives an *RMSE* of 355.14 and an R^2 value of 0.7204. After incorporating the word embeddings as features, the model gives us an *RMSE* of 353.74 and an R^2 value of 0.7283, only marginally better performance.

It is important to note that a standard linear regression model is relatively inflexible and is subject to a high bias, and is therefore subject to underfitting. This prompts the consideration of models that have more flexibility and lower bias. One such model is K-Nearest-Neighbors Regression.

2) K-Nearest-Neighbors Regression: KNN Regression computes and considers examples that are closest together in the feature space. After finding the 5 nearest neighbors to a particular example, the average of these neighbors' scores is used as the predicted value. For this model, the standard Euclidian distance function is used. While other distance functions were tried, using Euclidian distance gave the best results:

$$distance = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Evaluating the model gives an *RMSE* of 353.22 and an R^2 value of 0.7289. After incorporating the word embeddings as features, the model gives us an *RMSE* of 344.32 and an R^2 value of 0.7408, slightly better performance.

While Linear regression was subject to high bias, KNN Regression is subject to having high variance. The flexibility of the model and computation of predictions based off only 5 neighbors causes this issue. KNN inherently depends on finding neighbors for a particular example in the training data. While having a very large training dataset reduces variance to an extent, KNN Regression may still not generalize very well. This prompts us to consider a model that is not subject to very high variance, Random Forest Regression.

3) Random Forest Regression: This form of regression utilizes a combination of multiple decision trees and the average score of these trees to arrive at a Reddit post score prediction. Random Forest Regression utilizes bootstrap aggregation to continuously randomly sample examples from the training data to fit decision trees. After doing this random sample with replacement and thereby obtaining B decision trees, the prediction for a test example is given by:

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x')$$

Evaluating the model gives an *RMSE* of 304.92 and an R^2 value of 0.7764. After incorporating the word embeddings as features, the model gives us an *RMSE* of 301.32 and an R^2 value of 0.7855.

Random Forest Regression is less likely to overfit the training data and is therefore more generalizable. The technique of bootstrap aggregating effectively reduces variance when compared to KNN Regression. Therefore, unlike Linear Regression that has a high bias or KNN Regression that has a high variance, Random Forest Regression presents a better bias-variance tradeoff balance.

IV. Results & Error Analysis

Relative to the baseline, we observe significantly decreased RMSE among Linear, KNN, and Random Forest Regression. Specifically, we note 45.17%, 46.63%, and 53.49% decreases in RMSE respectively among the models that account for the mean of the title's word embeddings as features. Random Forest Regression using word embeddings as features had the smallest RMSE at 301.32. Relative to one another, we can visualize the model performances using RMSE in Figure 2.

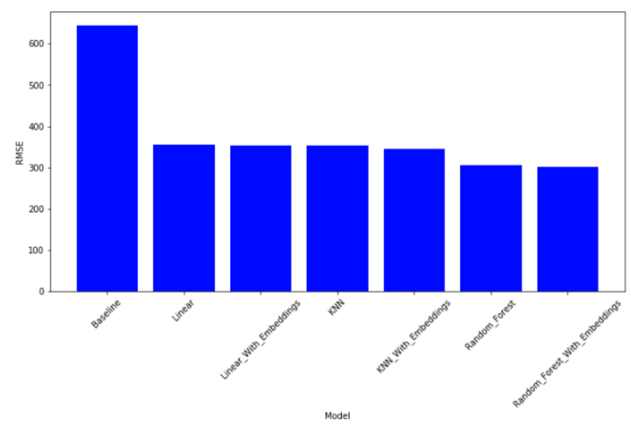

Fig. 2. Plot of RMSE vs. Model used, with and without accounting for embeddings feature

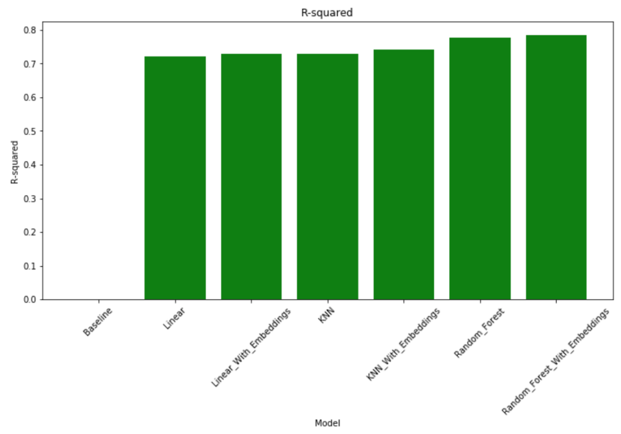

Fig. 3. Plot of R-squared vs. Model used, with and without accounting for embeddings feature

Word embeddings proved to be slightly impactful as features. Relative to the counterparts of the models that did not use them, there was only a 0.39%, 2.5%, and 1.1% decrease in RMSE for Linear, KNN, and Random Forest Regression respectively. Although they were not the most impactful, as can be seen in the ablative analysis section, they held some significance in indicating the popularity of a Reddit post.

We also notice increasing R^2 values in using KNN Regression over Linear Regression and in using Random Forest Regression over KNN Regression. Random Forest Regression using word embeddings as features had the greatest R^2 at 0.7855. We compare the R^2 values of all the models in Figure 3.

V. Ablative Analysis

To evaluate the importance of the features used, I removed one at a time from the original model and observed the relative decrease in R^2 to see how well the new set of features explains the output score. We note that the most signficant features in prediction were the $num\_comments$ and $gilded$, which indicate the number of comments a post has and the number of times a post was gilded respectively. The third

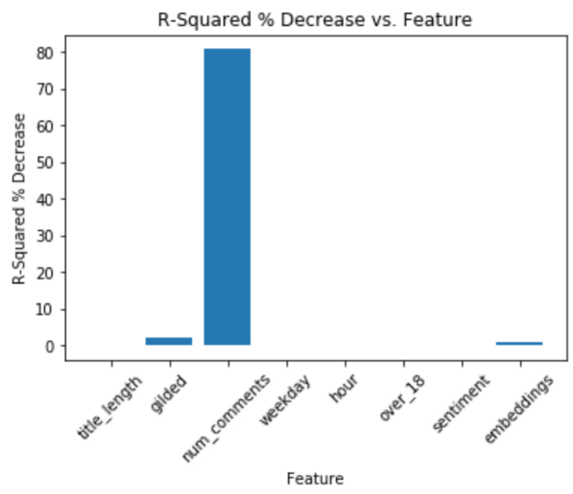

Fig. 4. Removing a feature and the relative decrease in R-squared when compared to the original R-squared accounting for all features using a Linear Regression Model

most important feature was the mean of word embeddings of the words in a post title. Although completely unique posts gain popularity, many recycled posts on AskReddit repeatedly gain popularity. The similarity between these posts and their repeated popularity in part intuitively explains the significance of embeddings in popularity prediction. Surprisingly, features such as sentiment were not significant at all in predicting the popularity of a post. In analyzing the data, I note that only 2.7% of AskReddit posts in the training data have a positive sentiment classification. Figure 4 fully outlines the relative decrease in R^2 that removing a feature brought about.

VI. Conclusion & Future Work

Using Reddit submission data, I was able to fashion features and evaluate models that predict the popularity of a Reddit post. Unlike existing approaches, I focused greatly on the text content of a Reddit post and used features such as sentiment classification and averaged GloVe word embeddings. Overall, the Random Forest Regression model had the greatest performance. Unlike the inherently high bias model of Linear Regression and the high variance model of KNN Regression, Random Forest Regression possess a better bias-variance tradeoff balance due to utlizing such techniques as bootstrap aggregation. Random Forest Regression achieved an RMSE of 301.32 and R^2 of 0.7855

While embedding representations for the title were generated, I believe embedding representations for a subset of the comments would also be a beneficial feature in popularity prediction. Furthermore, given time and computational resources, specific word embeddings can be generated with GloVe given a corpus of Reddit data that may be more suitable for this task. Finally, this popularity prediction can be applied to other subreddits and can utilize features that do not solely depend on text, such as attached images or links. Incorporation of such additional features in combination with the NLP techniques presented here can help create a better popularity prediction model for Reddit posts.

References

[1] Adam Reevesman. Reddit Comment Karma, Dec 2018.
[2] Andrei Terentiev and Alanna Tempest. Predicting Reddit Post Popularity Via Initial Commentary, 2014.
[3] Daniel Poon Yu Wu, and David Zhang. Reddit Recommendation System, 2011.
[4] Jason Baumgartner. PushShift.io: Directory Contents.
[5] Jordan Segall and Alex Zamoshchin. PREDICTING REDDIT POST POPULARITY, 2011.
[6] Lyndon White, Roberto Togneri, Wei Liu, Mohammed Bennamoun. How Well Sentence Embeddings Capture Meaning, 2015.
[7] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.
[8] Prasanna Chandramouli, Radhakrishnan Moni, and Varun Elango. Predicting reddit post popularity, 2017.