Learning to Rank Comments Within Subreddit Submissions

- Alex Jin, Don Mai, Jeff Setter

Problem Statement
Given a Reddit* submission, $s$, with $n$ comments, $c_1, c_2, \ldots, c_n$; Predict the ranking $R[i]$ in $\mathbb{R}^n$ where $R[i]$ is the ranking of $c[i]$ for all $i$ in $[1,2,\ldots,n]$. (i.e., predict the order of the top-level replies’ assigned scores in that submission). Ask Don for the ranking challenge!

Data Collection

ReddiT Server

Processing Unit

LevelDB (Key, Value)

60 Request per minute

Stemming

Regression

Linear Regression

Our regression model takes a post with varying number of comments, and outputs the predicted ranking of all of the comments. This prediction is made by extracting features (listed bottom right) from the comments, normalizing them with respect to all comments in a post, and then training a regression model of linear regression on an SVR (pictorially shown above). To calculate the effectiveness of our ranking, we used a normalized Spearman’s Footrule, which is the sum of the absolute difference of rankings for each post, divided by the maximum (Spearman’s) error (shown below right). We tested our model on different subreddits, but show only data from /r/murica. The learning curve of /r/murica shows that our model has high bias and underfits the data, as expected since we don’t have an enormous number of features (shown below). However, despite this lack of features, we still find that the average error decreases from 0.633 to 0.354 for /r/murica, as well as shift most errors below 0.5 (shown bottom left). The feature effectiveness varied across subreddits, including the most significant feature time, adding feature categories one at a time shows the effect of each for /r/murica (shown below right).

Clustering

The Meme Clustering algorithm is our attempt to use machine learning tools to provide intelligent features that are difficult to generate using a single formula. The algorithm can be described in the following four steps:
1. generate prior frequencies for Swords
2. identify (word, time) pairs of significant importance
3. run k-means on results from step 2
4. run a filter to remove undesired outputs
The output of this algorithm is a collection of centroids representing phrases that had a significant impact during a time period. Below are some human readable sample output:

Future Work

We plan on improving our k-means clustering algorithm or researching alternative means to more accurately group together words that convey a single idea, so that we can use features that check for the presence of their multi-word phrases. This would allow us to make better comment ranking predictions within subreddits where users up-vote comments simply because they included a particular grouping of keywords in their comments. For more complex subreddits, we would need to leverage the results of natural language understanding, as the semantics of the comment would play a much larger, possibly dominating role in the overall score of a comment.

References