Determining Community Interest in Text-Only Posts on Reddit

Project Proposal

Charles Lewis
University of Michigan
Ann Arbor, MI
noodle@umich.edu

Nathan Price
University of Michigan
Ann Arbor, MI
nrprice@umich.edu

David Purser
University of Michigan
Ann Arbor, MI
dpurser@umich.edu

ABSTRACT

This paper proposes a term project for the EECS 498 Information Retrieval course at the University of Michigan which seeks to analyze content on Reddit[1] to determine community interest in certain topics, keywords, and questions.

The project involves collecting a set of data from a number of subreddits (individual forums on Reddit) including textual content of each post, number of votes that the post received, number of comments on the post, and other information. This information will be used to estimate the community's interest in each post. These estimates will be compiled and indexed by keyword, allowing a user to query the system to determine the expected interest in a new post and whether a new post is very similar to previous posts.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

Keywords

Data analysis, Reddit, community interest

1. INTRODUCTION

Expanding on a slightly modified version of one of the suggested term projects, this project aims to be able to accurately predict or measure community response and interest in text-only posts (sometimes known as "self-posts") on the popular internet forum Reddit. These text-only posts are often found in dedicated "subreddits" such as `/r/AskReddit` or `/r/ELI5`.

The popularity of current and previous posts on Reddit can be estimated through the use of comment counts, upvotes/downvotes[2] given to the post by users, number of distinct users who commented, average length of user responses, etc.

This popularity measure can then be used to predict the expected popularity of a new question or topic by matching keywords or topics of the new post with those of past posts to determine whether the new question is similar to any previously posted content, and, if so, how popular the new post is expected to be.

2. PREVIOUS WORK

There are a number of websites dedicated to detecting "reposts" on Reddit, which are exact duplicates of previously posted content. One such website is KarmaDecay[3]. However, these websites only detect exact duplicates of hyperlinks and images, which make up the majority of posts in many subreddits. The textual content of posts in text-only subreddits such as `/r/AskReddit` or `/r/ELI5` is not analyzed. This gives these websites limited usefulness in text-only subreddits, where no images or external links are generally allowed.

Furthermore, existing systems are designed mostly to help users avoid reposting exact duplicates of existing material, and do not match based on a similarity measure. This means that related or very similar posts will not be matched, only exactly identical ones. By using a text-matching system based on similarity scores, our project will allow text-only topics to be analyzed not to detect exact duplicates (as these are rare in text posts due to the diversity inherently present in language), but to find similar posts from the past based on matching keywords, topics, phrases, and other content. These similar posts can then be used to judge or estimate potential interest in the new post and to predict useful information such as whether a post will be popular or successful or which subreddits it may be most successful in by analyzing the interest in the previous posts. In addition, the uniqness of the post will also contribute to determing how popular a post might be.

[1] `http://www.reddit.com/`

[2] Users on Reddit can give either positive or negative votes to posts and comments, known respectively as 'upvotes' and 'downvotes'. Upvotes generally increase the visibility of a post or comment by causing it to be located near the top of the page, whereas downvotes have the opposite effect.

[3] `http://karmadecay.com/`

Therre has been work done analyzing news content, not based on the article but upon the responses of the general public [2]. This paper describes a way of what they describe as "comment centric tagging" as a means to identify which articles were highly interesting and important. They use these comments within a post to alter and adapt the interest weighting on an article.

While this work is not the main focus of the paper, it does highlight how important the comments of a post can be. The study they performed showed that the comments of a post greatly impacted the community interest within that specific topic. This will help us direct or findings towards the comments found on Reddit, weighing them higher, than the actual post itself.

Some work has also been done by researchers to determine characteristics of articles that have been ranked by crowdsourced feedback from users [1]. The results of this study point out that while crowdsourced feedback mechanisms "can be relatively effective for ... promoting content of high quality, they do not perform well for ordinal objectives such as finding the best articles." In other words, while interesting topics and general themes are often highlighted as outstanding by user feedback, the best specific articles are not always at the very top of the list. Rankings which are based on user feedback will likely tend to positively identify good topics, but perhaps not the absolute best topics. This means that we should not use a post's score as an absolute ranking; that is, we should not conclude that one post is definitively better than another simply because it has a higher score. For our purposes, this means that all posts with a reasonably high score could be determined to be interesting to the community, but that comparing levels of interest (or ranking the level of interest) between two interesting topics will likely be very difficult.

The article also has some interesting discussion about crowdsourced scores over time, indicating that posts tend to take a while to gain momentum, but then explode rapidly if they are determined to be interesting. This means that some posts which are not "discovered" by users (they may be posted at the wrong time of day or when few users are online, so they are not identified by interested users) may fail to gain traction and perform as they should. This may mean that the same post, created at different times, may be extremely popular one time and not at all another time. In fact, the post may go unnoticed several times and only be discovered once. Our scoring algorithms will have to take this into account, likely by giving the positive feedback of posts which are determined to be interesting much more weight than the negative feedback of posts which are determined to be uninteresting (because a post that scores highly is almost certainly an indicator of community interest, while a post that scores poorly may have many different reasons for receiving that score).

3. APPROACH

Our approach for this project involves three major areas: data collection, processing, and retrieval.

First, data must be collected from Reddit. This will involve collecting and indexing a number of past posts in text-only subreddits. The posts and their comments will be analyzed, looking for common keywords, themes, or topics which seem relevant to the post in order to match the post with other submissions that have similar topics.

Next, the popularity or interest of the post will then be measured through various statistics including number of comments, number of upvotes/downvotes received (through Reddit's voting system), number of distinct users who commented on the post, average length of the comments, average depth of comment trees[4], number of upvotes/downvotes on comments and their distribution among the comments, time of day that the post was made, and how long the post has been on Reddit.[5]

It is currently unclear whether these metrics will be adequate or accurate measures of community interest in a post, but we assume that one or more of these measures in combination will be able to give insight into the popularity of posts and comments based on certain topics or keywords. For example, we assume that posts with greater numbers of comments must have been more interesting or popular among users. Likewise, we assume that posts with large numbers of upvotes were likely interesting. However, posts with many downvotes present a less clear picture: the post may just be very controversial, but still interesting—or the post may be badly written or uninteresting. We will have to analyze the metrics to determine which of these scenarios is more likely, given the other information we have collected about the post.

The various metrics may have different weighting as to analyzing the interest in the topic. As previously mentioned, the numerical "score" seen on Reddit decays over time. Also, the upvote/downvote system can be abused by bots and other humans attempting to abuse the system. Therefore, we believe that while the number of upvotes/downvotes will be useful, it will likely have a lower weight than the number of comments and the comparison to other, similar posts.

Finally, the keywords, and posts matching those keywords, will then be used to create a database which can be used to judge popularity of a post based on which keywords appear in the post title or text body. This database can then be queried to determine expected popularity of a new post given its text. This database will also be designed in such a way that new posts on Reddit will be automatically incoporated into the corpus. The query system may also be able to suggest similar keywords (words which are frequently used with the ones in the given text), topics, or information to include in the post which may increase the post's popularity score (and thus hopefully lead to a more interesting post on Reddit). [6]

4. TESTING

[4]On Reddit, users can comment on other comments, forming a tree-like structure of comments.

[5]Reddit's algorithm works in such a way that posts that have been on Reddit a long period of time have a lower "score". We may have to inflate the score given to older posts as their ranking has decayed over time.

[6]This is a possible extension of the project and will need to be explored further.

We will have to analyze how our system performs in comparison to actual level of interest. To accomplish this, we will provide the system with queries matching those of known range of documents. We will take a sample of topics found on Reddit and have our system generate a interest score[7] for them. We will then compare our generated interest score to the interest score of those in the sample.

We will make attempts to account for the time decay of upvotes/downvotes and the time of day the post was made. We will also make sure to take a wide sample of posts, ranging from the most popular to least popular, to make sure that our system is not biased to a specific subset of posts and manages to accurately predict an Interest Score which matches the measured reality of those posts.

5. IMPLEMENTATION

We will implement the project in Python, using Python libraries and tools to download, analyze, store, index, and query content from Reddit.

Reddit provides a web-based API[8] for accessing its content directly, eliminating the need for complicated scraping mechanisms and reducing server load introduced by repetitive page generation during crawling. This API appears to provide all of the necessary functions that we would require for this project, such as the ability to obtain posts, their upvote and downvote counts, comments, comment scores, information about users, and many other useful pieces of information.

Reddit has some guidelines on their API usage[9], and expects users to rate-limit their gathering of information to approximately one request every two seconds (technically, thirty requests every minute). Other restrictions on API usage, such as valid and correct User-Agent strings which include version numbers, must be obeyed as well. Care must be taken to properly use the API according to all of Reddit's rules.

There are pre-existing libraries for Python which can interface with the Reddit API, such as the *Python Reddit API Wrapper*[10]. The PRAW API is easy to use and is well-documented, and also follows all of Reddit's API rules mentioned previously, making it easy to obtain data from Reddit without worrying about the raw data processing. This will give us more time to spend analyzing and retrieving information from the databases we build.

Furthermore, Reddit's underlying server-side source code is also written in Python and is available on GitHub[11]. Analysis of this code may be useful for determining the best uses of the statistics provided by Reddit. For instance, Reddit's own ranking algorithm which determines which posts should appear near the top of a given subreddit uses the post's age and upvote/downvote counts in order to determine the post's position, and analyzing this code may give us some useful insight into measures which we may want to use to judge interest in posts. It also may give us a baseline comparison point for recent posts.

Additionally, Reddit "fuzzes" the values it reports for upvotes/downvotes on currently popular posts, in order to avoid vote manipulation by users. Analyzing the source code for Reddit should allow us to avoid retrieving incorrect data because of this issue.

Bots and users on Reddit commonly downvote old posts for several different reasons, which can skew results. It is unclear whether these downvotes create a significant impact on the total score of posts in the long term, but care must be taken to watch for and account for these factors so that our data is correct and accurate.

As "community interest" is a relatively vague and subjective term, it is difficult to define a strong baseline comparison for measuring the accuracy of our algorithm. However, we can use a number of techniques to gauge relative accuracy. For instance, being able to accurately predict, based on text alone, whether a new post will reach the front page of a subreddit (top 25 of recent posts, according to Reddit's ranking algorithm) should be a good start. Similarly, the algorithm should assign high interest ratings to all posts which appear on the front page, when supplied only with the text of the post, indicating agreement with the users of Reddit and with Reddit's own algorithms. Therefore, we could create snapshots of the front page and of recently submitted posts at regular intervals, and use these to measure the effectiveness of our algorithms by routinely calculating interest for all posts on the front page and for all recently submitted posts. We expect the posts on the front page to have a higher interest rating than the random newly submitted posts. Furthermore, by running different versions of the algorithm on the same set of front page posts, we can update and tweak variables in the algorithm until it ranks these posts more highly, indicating agreement with Reddit's own ranking algorithms.

Our project implementation will likely use a database storage system to store gathered information permanently so that it can be re-analyzed, re-processed, and re-indexed as our algorithms are developed and refined, without having to download all of the information again. This will allow us to have a nearly continuously growing database from which to pull data from, so our algorithms should get more and more accurate over time as they gain access to more and more data. However, we have not decided on a particular storage mechanism.

Using a complex database storage mechanism as opposed to simple data structures will allow us to more easily do complex analysis of information. For instance, summing the upvote/downvote totals for all comments on a given post can be done in a single, simple SQL statement, whereas it would take several loops to do this from Python. On the other hand, a database storage engine adds complexity and

[7] We define our Interest Score as being an approximation of the number of upvotes/downvotes, and comments that a query could expect to attain, or another similar measure. We may also include an ideal time of day to post, in addition to a ranking on how it compares to all posts on a given subreddit (in the form of an Interest Percentage).

[8] `http://www.reddit.com/dev/api`

[9] `http://github.com/reddit/reddit/wiki/API`

[10] *PRAW*, `https://praw.readthedocs.org/en/latest/`

[11] `https://github.com/reddit/reddit`

can be more difficult to manage. An alternative would be to use Python's `pickle` library, which stores native Python data structures (lists, tuples, and dictionaries) directly in byte-form into a file, allowing them to be saved and loaded at any time. However, depending on the amount of data collected, it may be unreasonable to store all of the data in memory, or even in one file, and it may be necessary to use additional file storage as well.

Some additional work will have to be done to create an interface for querying the database. This interface could be implemented in a terminal-based system for basic querying, or in a browser-based setting (for example through Python's `SimpleHTTPServer` modules) to allow for better data visualization of the results. The relevant information to be displayed has not yet been determined, so the implementation details of the user interface will be determined later, but will likely include both the ability to estimate score of new posts and also to view scores and other information about posts that have been indexed previously.

6. GROUP TASKS AND RESPONSIBILITIES

This project has many pieces that need to fit together properly to produce a working product. As mentioned in prior sections, the main parts of the project are: gathering and storing data from Reddit; developing algorithms for determining the interest in posts; matching new posts with similar or related posts from the past; creating some sort of interface for displaying the results to the user; and possible further extensions on this project.

These parts are all heavily dependent on one another, so it would be difficult to completely split the work of the project among group members from the very beginning. Charlie and David will be heavily focused in gathering, storing, and handling the database, querying, and API usage. Charlie will also be focused on implementing our system on heroku and possibly building a bot to actively gather and assess new data. Nathan will primarily focused on the ranking and anlyzing posts based on various factors, and will also help with the similar post analysis. The similar post analysis will primarily driven by David. The display and output of our system will be a collective work of all of us, and will the final remaining part of our project, besides the extention features.

While each of these tasks have been split up for the sake of listing what each project member will be primarily focusing on, they are all heavily intergrated with one another and will require each of us to have an in-depth understanding of how each part functions. We will just specailize in one area more than others such that one person will have a greater knowledge of there specific part. This will allow us to provide a better final presentation and be able to contribute more to our collective team.

Once the main portion of the project is working, we can begin working on improving parts separately as necessary to improve the algorithms, collection, and display of information to the user.

All work besides the programming work will be a joint effort and worked on through various pair work resources (Google Docs, git for this proposal management, etc.).

7. REFERENCES

[1] G. Askalidis and G. Stoddard. A theoretical analysis of crowdsourced content curation. In *The 3rd Workshop on Social Computing and User Generated Content*, 2013.

[2] X. Liu and V. von Brzeski. Computational community interest and comments centric analysis ranking.