Reddit Post Classification by Flair

Harrison Kiang: hkiang2@illinois.edu
Hardik Naik: hardikn2@illinois.edu
Stan Lee: skl3@illinois.edu

Abstract

Reddit is a social news aggregation and discussion site that invites participants in various communities to post and reply to links relevant to said communities. By utilizing text and link extraction methods and text classification algorithms we are able to develop a process to assist moderators in classifying posts which allow for the organization of content and the growth of the community.

Introduction

Reddit is a social news aggregation and discussion site that invites participants in various communities to post and reply to links relevant to said communities[1]. Each community is organized into a subreddit or multireddit consisting of multiple subreddits. Each of these subreddits are moderated by moderators. These moderators are able to configure their subreddits as well as moderate them; they can set rules, set permissions, provide custom CSS, and even add other moderators[2].

One of the methods moderators use to organize and sort posts is to associate with them pieces of flair, which are custom CSS elements. Here's what a post with a flair looks like.

Figure 1 - A post in /r/machinelearning[3] with flair

The above post from /r/machinelearning has 284 upvotes, was posted 17 hours ago by a user called "gambs", has 52 comments, a link to a domain with the host "openai.com", and has a blue flair with "Research" as the flair text. Notice how the title of the post has an [R] at the beginning. This is how the /r/machinelearning subreddit, a community focused on topics surrounding machine learning, designates what flair is associated with each post[4].

A more notable example of the use of submission flairs comes from /r/science[5], a subreddit focused on all things science. Science itself is a pretty broad subject having many fields, e.g.,

---

[1] https://en.wikipedia.org/wiki/Reddit
[2] https://www.reddit.com/wiki/moderation#wiki_what_are_moderators_able_to_do_that_users_can.27t.3F
[3] https://www.reddit.com/r/machinelearning
[4] https://www.reddit.com/r/MachineLearning/about/rules/
[5] https://www.reddit.com/r/science

medicine, physics, health, astronomy, computer science, etc. This subreddit uses flairs to organize posts by field, to the extent where a user only interested in a particular subject matter can satisfy their curiosity without having to go through the entirety of /r/science and filtering out for themselves the posts they would initially be interested in.

Figure 2 - Some of the flairs supported by /r/science. Each flair corresponds to a field.

As a subreddit grows, it becomes more and more difficult for moderators to attach a flair to every post, as users who submit links and posts to these subreddits do not always post correct flair or post any flair at all. This limits the scalability of a subreddit.

Related Work

Some work is done on social media domain reddit for predicting upvotes, classifying post to subreddits, predicting best time to post etc. Most of the work we found (listed below not inclusive) were related experimenting different features selection, embedding and classification techniques on dataset from Reddit. Specific application to classify post into different flairs to reduce moderators overhead does not exist to the best of our knowledge.

- **r/Classifier**[6]
  This paper compares text classification of posts over 12 carefully selected different classes (subreddits) using techniques such as TF-IDF weighting, sentiment classification, parts-of-speech tagging and Latent Dirichlet Allocation along with feature selection technique like ‘’Title-Split’’ to take advantage of Reddit domain base knowledge.

  Though our work and the work discussed above has similar domain and classification problem, they differ in end goal and scope of the work. Our work includes playing with a live dataset by periodically scraping a given subreddit which includes text as well as links while above project limited to dataset which only include text based posts (ignores post with link, image etc). Our work goal is to minimize moderators’ input by classifying all posts in specific subreddit while their goal was to experiment with different classification methods to assign any post to any one of selected subreddits.

[6] http://cs229.stanford.edu/proj2014/Andrew%20Giel,Jon%20NeCamp,HussainKader,rClassifier.pdf

- **Text Classification of Reddit Posts**[7]
  This paper compares different feature extraction and classification methods on limited and diverse set of subreddits. Vector representations were used as input features for various supervised learning classification algorithms. In their feature extraction utilized bag-of-words and a single layer neural net.

  Our approach improves upon this by extracting live posts from any subreddit using the Reddit API instead of being limited to a dataset provided by Kaggle (see proposed method/algorithms). Subreddits that work best are those who have the vast majority of posts tagged, and each tag is used with enough posts for there to be able to build a large enough model.

- **Classification of posts on Reddit**[8]
  They used Reddit dataset from the Stanford Network Analysis Project and worked on predicting number of upvotes and further extended to classify post in different subreddits. Researchers have success with creating model to predict upvote however their model didn't perform well to classify posts into subreddits. The reason for this was attributed to large number of classes for classification and skewed dataset.

  Their restrictions are removed in our project by building our own live dataset via batch ingestion and classifying into limited number of classes (e.g., Research, Project, Discussion and News).

Novelty
Our approach improves upon these by scraping live posts from any subreddit using the Reddit API vai JRAW instead of being limited to a dataset provided by Kaggle or Stanford Network Analysis Project. Text is extracted using extractors like juicer and Apache Tika. This would hypothetically be an ideal setup to make a Reddit bot that would be able to train in real time and suggest post flairs based on the model constructed from the ingested batches of posts. (See "Extraction Phase" for more detail).

Problem Definition

As a subreddit grows, it becomes more and more difficult for moderators to attach a flair to every post, as users who submit links and posts to these subreddits do not always post correct flair or post any flair at all. This limits the scalability of a subreddit.

Our solution aims to automate the process of appending flairs to posts. We scrape a given subreddit or multireddit for a fixed batch of posts, extracts the relevant text and flair, and builds a model to classify posts by flair. This model will eventually be used to automatically classify

[7] https://jgutman.github.io/assets/SNLP_writeup_gutman_nam.pdf
[8] http://cseweb.ucsd.edu/~jmcauley/cse255/reports/fa15/021.pdf

new posts as they come in. The inputs are the scraped posts, and the outputs are the predicted flair per post.

We first discuss the existing and related work surrounding this task. We then cover our proposed method and algorithms used, first by describing how we obtain and construct our dataset, then by putting said dataset through our application. Finally, we analyze the results.

Proposed Method/Algorithms

There are two stages to our process: text extraction and classification. The extraction phase is where we extract the relevant text per post, excluding comments, for text analysis. The classification phase is where we use the text per post to train a model used to classify new posts based on flair.

Note that subreddits that work best are those who have the vast majority of posts tagged, and each tag is used with enough posts for there to be able to build a large enough model.

Extraction Phase

Reddit offers an API that allows exposure to its open source platform[9]. There is also a list of API wrappers that make working with the Reddit API in a development project using languages easier[10]. We chose to use The Java Reddit API Wrapper, or JRAW[11], due to our familiarity with the Java programming language, in addition to the ability to use Weka[12], an open source library for machine learning that is readily applicable to text mining applications, with our project.

Posts in Reddit come primarily in 2 forms: text posts and links. The host for a given text post corresponds to a host self.[subreddit], whereas a link corresponds to the its respective host. Figure 3 shows two post submissions from /r/machinelearning: a text post followed by a link. The hosts are colored in gray and are in parentheses adjacent to their corresponding titles.

Figure 3 - A text post posted 2 hours before the time of writing with 1 upvote, and a link posted 1 day before time of writing with 17 upvotes.

[9] https://github.com/reddit/reddit/wiki/API
[10] https://github.com/reddit/reddit/wiki/API-Wrappers
[11] https://github.com/thatJavaNerd/JRAW
[12] http://www.cs.waikato.ac.nz/ml/weka/

If a user were to click on a text post, they would be directed to a page that still resides on the reddit.com domain, whereas if a user were to click on a link, the would be redirected someplace else. This makes it difficult to reliably extract text from the use of the Reddit API, and by extension, JRAW alone. We primarily use two extractors: juicer and Apache Tika.

Juicer is “a web API for extracting text, metadata and named entities from HTML “article” type pages”[13]. The query with the key “url” in the api call corresponds to a link for which to extract article text from. A resultant json response is generated, from which text can be extracted via the “body” field (click the above link in Figure 4 to see the resultant json response). A sample query is shown below.

```
https://juicer.herokuapp.com/api/article?url=http://www.bbc.co.uk/news/world-africa-16377824
```

Figure 4 - A sample juicer API call for http://www.bbc.co.uk/news/world-africa-16377824

Apache Tika is a content analysis toolkit that detects and extracts metadata and text from a wide array of file types, which can all be parsed through a single interface[14]. This single interface, along with the fact that its Parser API[15] is written in Java, makes it an ideal tool to use in our project. There are two primary parsers used: AutoDetectParser[16] and PDFParser[17]. Juicer, Apache Tika’s AutoDetectParser, and Apache Tika’s PDFParser are used in a chain. If one fails, then the next is called. For example, AutoDetectParser is called when juicer returns a less than satisfactory result. A result is satisfactory only if the result contains a body of text larger than a given threshold (denoted as `TEXT_THRESHOLD` in our source code).

After extraction, the author, created date and time, text, and associated flair are processed and used to train a classification model designed to classify posts based on flair.

Classification Phase

We used two machine learning libraries to classify posts according to flair: Apache Spark and Weka. Spark provides a machine learning library called MLlib that “makes practical machine learning scalable and easy”[18]. Weka provides a collection of machine learning algorithms for data mining tasks that can easily be applied to text mining applications[19].

Two methods were chosen: Spark and Weka. Spark is specifically engineered to work with big data, or in this case many posts. This is useful for active subreddits with many posts or many fields that necessitate a large number of posts for a reliable model. Unfortunately, it’s also

[13] https://github.com/matth/juicer
[14] https://tika.apache.org/
[15] https://tika.apache.org/1.14/api/
[16] https://tika.apache.org/1.14/api/org/apache/tika/parser/AutoDetectParser.html
[17] https://tika.apache.org/1.14/api/org/apache/tika/parser/pdf/PDFParser.html
[18] https://spark.apache.org/docs/latest/ml-guide.html
[19] https://spark.apache.org/docs/latest/ml-guide.html

designed to run on data centers, often requiring a powerful machine with lots of memory (at least 16GB) to run well or run at all. Weka was chosen as it offers out of the box tools and machine learning models useful for text classification.

Spark Application

In Spark, we created a statistical language model for each flair, and a background statistical language model. For a given flair, we collected all the relevant posts from a given subreddit and their associated text to create the corresponding statistical language model. The background statistical language model was compiled from text from all posts, tagged and untagged, from the same subreddit. For example, in /r/machinelearning, we created a statistical language model for Research, Discussion, Project, and News, and created a background statistical language model consisting of posts from all the aforementioned flairs in addition to posts that were untagged (`submission.getSubmissionFlair()` returns `null`)[20].

We make a slight modification of the "Jelinek-Mercer Smoothing" scoring function by excluding the term frequency $c(w,q)$ and factoring into this scoring function a prior $p(d)$, changing the scoring function in Figure 5 to the following:

$$f(q,d) = \sum log\,[\,p(d) * p(w \mid d) / (\alpha * p(w \mid C)\,]$$

For all words $w \in d$ and all words $w \in q$

Figure 6 - The JM smoothing ranking function, with prior; as implemented in our project[21]

This function was used to calculate a score for each post for all flairs in a given subreddit, and a prediction was made according to the flair with the maximum score. Sample results are shown below

confusion matrix

		Predicted				
		Class 0	Class 1	Class 2	Class 3	Recall
Label	Class 0	138	22	0	0	0.8625
	Class 1	33	120	0	0	0.7843
	Class 2	64	47	0	0	0.0000
	Class 3	27	10	0	0	0.0000
	Precision	0.5267	0.6030	0.0000	0.0000	

Accuracy: 0.559652928416486

Figure 7 - Performance data from a sample run of the Spark implementation using the algorithm described above

[20] https://thatjavanerd.github.io/JRAW/docs/latest/net/dean/jraw/models/Submission.html#getSubmissionFlair()

[21] https://github.com/hkiang01/reddit-post-classifier/blob/bd7a63dce811a5ada810babc613ecf424b5fd4c0/src/main/java/com/cs410dso/postclassifier/App.java#L680

For more information, see source code:
https://github.com/hkiang01/reddit-post-classifier/blob/weka/src/main/java/com/cs410dso/postclassifier/AppSpark.java

Weka Application

Professor ChengXiang Zhai[22] suggested to experiment with Weka, an open-sourced machine learning library implemented in Java that can be readily applied to text mining applications. Weka was very simple to use out of the box, and actually ended up performing better than the above Spark application. We tried different classification methods and confusion matrix for different algorithms are listed below.

Classifier: NaiveBayes confusion matrix

Label \ Predicted	Class 0	Class 1	Class 2	Class 3	Recall
Class 0	22	3	2	3	0.7333
Class 1	14	21	1	1	0.5676
Class 2	9	1	40	4	0.7407
Class 3	5	1	1	7	0.5000
Precision	0.4400	0.8077	0.9091	0.4667	

Error rate: 0.3334
Mean absolute error: 0.1704
Correct: 66.67%

Classifier: NaiveBayes Multinomial confusion matrix

Label \ Predicted	Class 0	Class 1	Class 2	Class 3	Recall
Class 0	26	3	1	0	0.8667
Class 1	15	21	0	1	0.5676
Class 2	12	1	38	3	0.7037
Class 3	5	0	0	9	0.6429
Precision	0.4483	0.8400	0.9744	0.6923	

Error rate: 0.3037
Mean absolute error: 0.1585
Correct: 69.63%

Classifier: J48 confusion matrix

Label \ Predicted	Class 0	Class 1	Class 2	Class 3	Recall
Class 0	14	8	7	1	0.8087
Class 1	5	26	5	1	0.8855
Class 2	5	6	38	5	0.8712
Class 3	4	3	4	3	0.5897
Precision	0.7949	0.8596	0.8712	0.7188	

Error rate: 0.1615
Mean absolute error: 0.0922
Correct: 83.85 %

Figure 8 - Performance data from a sample run of the weka implementation using different classifiers. Note: class mapping does not necessarily correspond to that of the Spark implementation).

For more information, see source code:
https://github.com/hkiang01/reddit-post-classifier/blob/weka/src/main/java/com/cs410dso/postclassifier/AppWeka.java

[22] http://czhai.cs.illinois.edu/

Dataset

Our implementation uses the Reddit API via JRAW to ingest posts in a live batch. The extractors (see “Extraction Phase”) then extracts the text for each post. In the case of /r/machinelearning, Ingestions of about 1000 posts typically yield around 570 posts with sufficient text for classification. This number varies according to the posts themselves, the quality of extraction, etc. We then export the resultant information in JSON format. This would hypothetically be an ideal setup to make a Reddit bot that would be able to train in real time and suggest post flairs based on the model constructed from the ingested batches of posts. This is a sample JSON document.

Figure 5 - A part from our dataset viewed using jsonviewer[23]

We did not use every property of the JSON objects as shown above. The fields that were of interest included identifiable fields such as author and created, along with the content: flair and text. The text is processed into words and the flairs are indexed. Note that the index of each flair may vary from run to run.

The method of which Spark and Weka transfer words to text is similar: they both create inverted indices, although to accomplish this task, they use their own transformers and filters,

[23] http://jsonviewer.stack.hu/

respectively. For example, Spark uses a RegexTokenizer[24] and CountVectorizer[25], whereas Weka has available the StringToWordVector[26].

```
root
|-- author: string (nullable = true)
|-- created: string (nullable = true)
|-- flair: string (nullable = true)
|-- text: string (nullable = true)
|-- words: array (nullable = true)
|    |-- element: string (containsNull = true)
|-- indexed_flair: double (nullable = true)
```

Figure 9 - A Spark 2 Dataset schema of the dataset that is generated after posts are ingested via Reddit API through JRAW.

The same dataset was used for both the Spark and Weka implementations.

For the Spark implementation, all rows are used for both training and evaluation, amounting to around 570 posts with text in the case of /r/machinelearning.

For the Weka implementation, a percentage of posts are used for training, and the rest are used for evaluation. We used a hard coded value of 72 percent in our implementation (see https://github.com/hkiang01/reddit-post-classifier/blob/weka/src/main/java/com/cs410dso/postclassifier/AppWeka.java#L117).

More Sample Results

We ran the Weka implementation on /r/science. There are 36 categories corresponding to 36 indexed flairs. Results are shown below. Heatmaps generated using the d3heatmap R library[27].

Naive Bayes Multinomial

Error rate: 0.3725
Mean absolute error: 0.0212
Correct: 62.75%

[24] https://spark.apache.org/docs/2.0.2/api/java/org/apache/spark/ml/feature/RegexTokenizer.html
[25] https://spark.apache.org/docs/2.0.2/api/java/org/apache/spark/ml/feature/CountVectorizer.html
[26] http://weka.sourceforge.net/doc.dev/weka/filters/unsupervised/attribute/StringToWordVector.html
[27] https://cran.r-project.org/web/packages/d3heatmap/d3heatmap.pdf

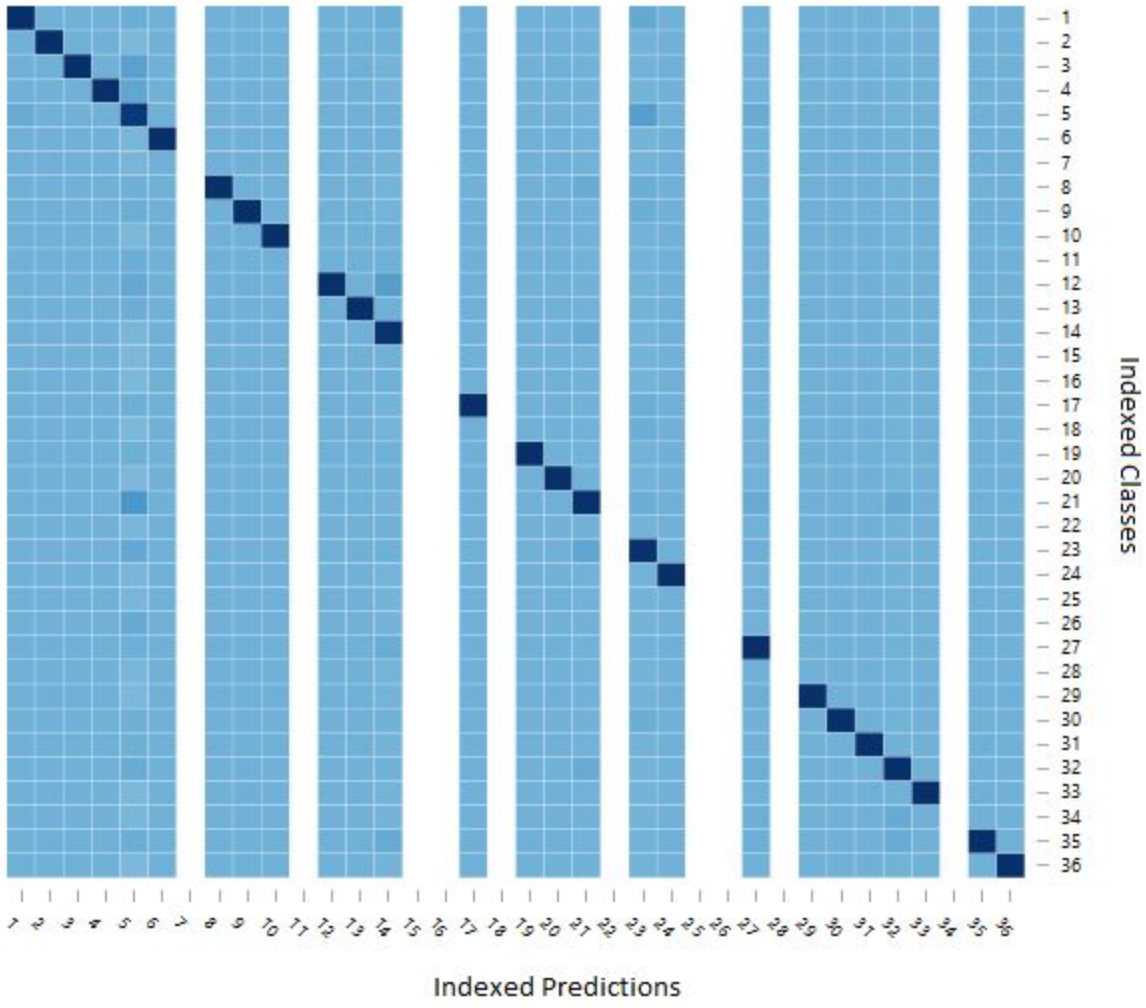

Figure 10 - Results for NaiveBayesMultinomial Weka Classifier for /r/science

Naive Bayes
Error rate: 0.2733
Mean absolute error: 0.0151
Correct: 72.67%

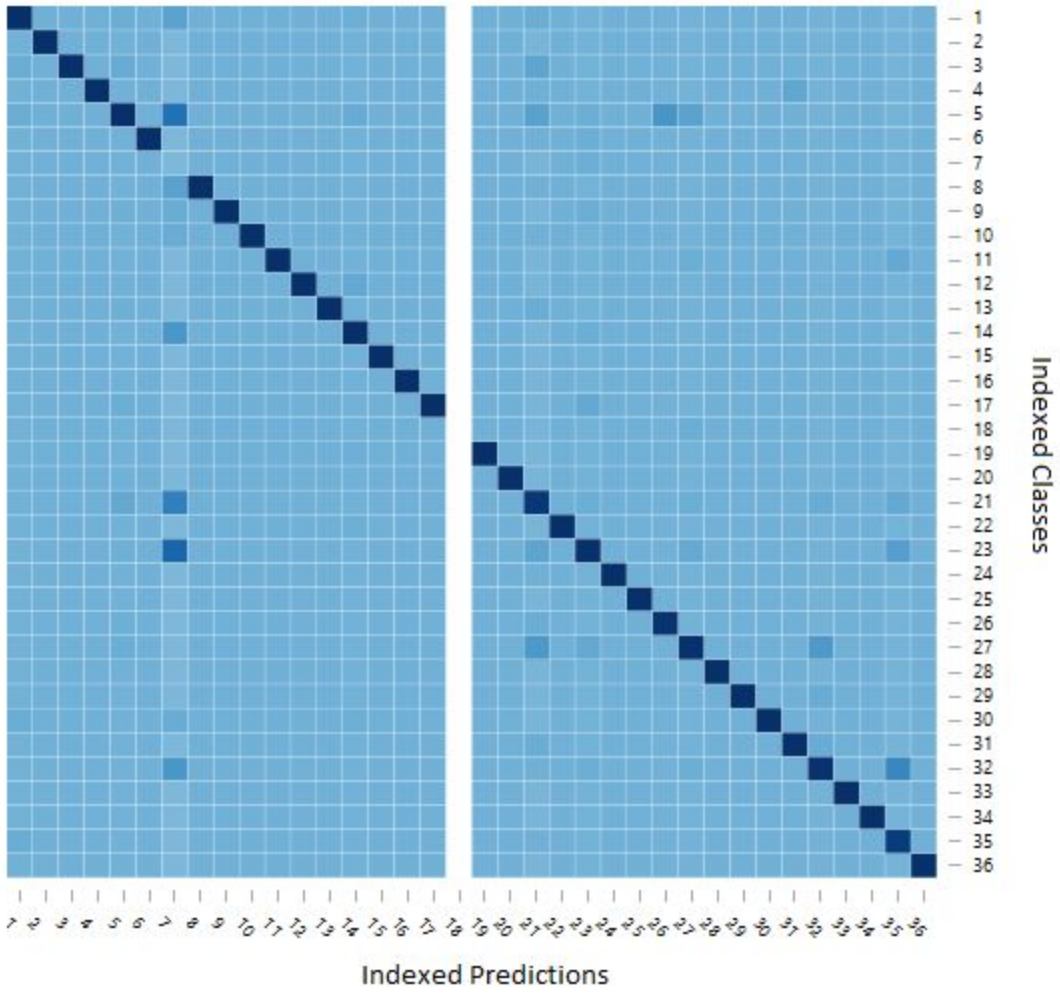

Figure 11 - Results for NaiveBayes Weka Classifier for /r/science

J48

Error rate: 0.3299

Mean absolute error: 0.0197

Correct: 67.00%

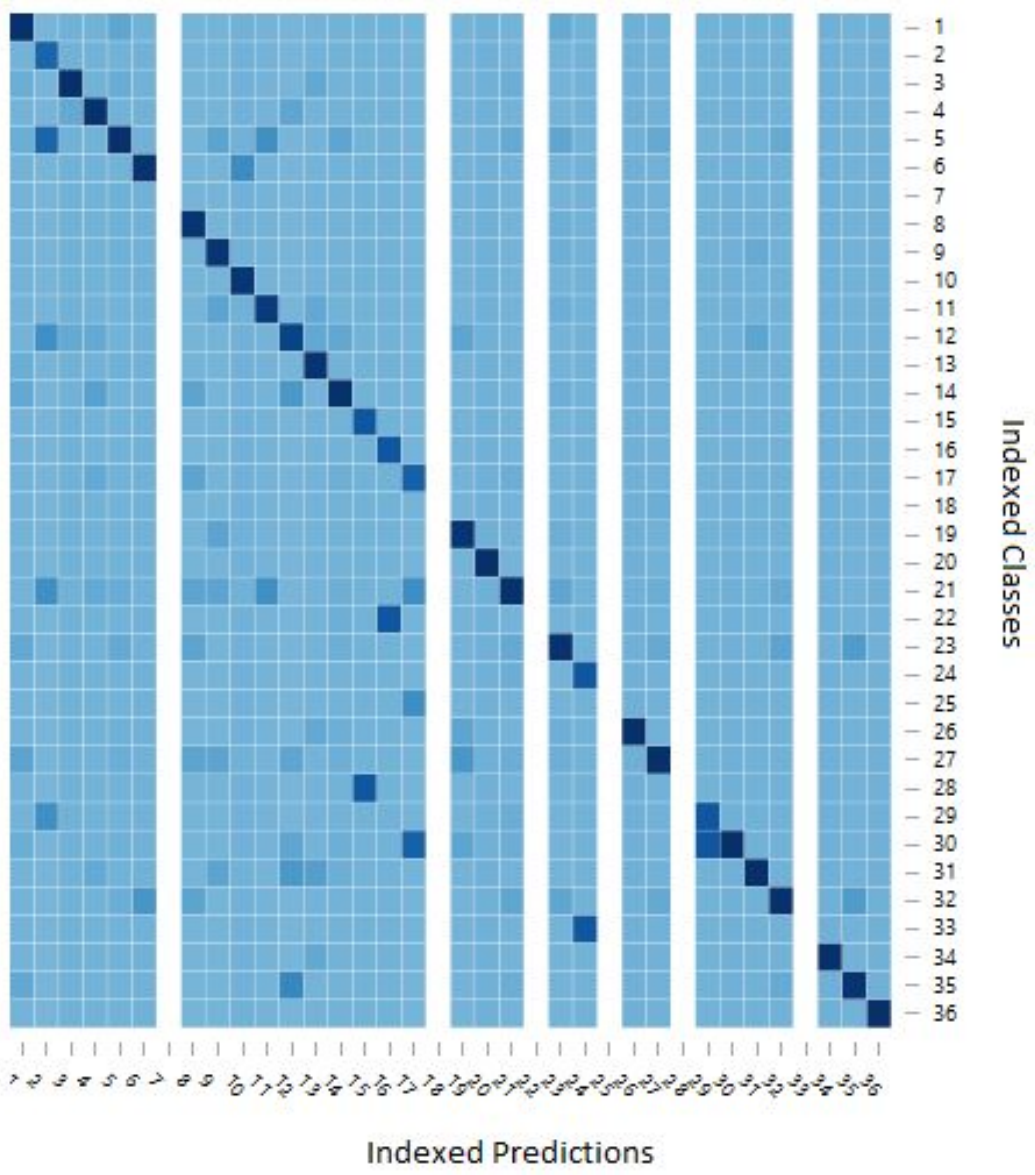

Figure 12 - Results for J48 Weka Classifier for /r/science

Future Work

While our work is far from complete, there are many extensions to the project that can be completed once the project is in a mature state. Examples of extension includes:

- Use of a feedback mechanism to continually train the model and including comment text as they come in to add it to the model
- Implement a Reddit bot to automatically classify posts on moderators' behalf and to create a browser plugin to suggest flairs as a user creates their post

All of these extensions will allow for greater accuracy and use of flairs by users as well as assisting moderators with organizing and growing their subreddit community.

Conclusion

Although system had correctness of around 84%, it still needs improvement before it can go through user study. We also validated that classification is empirically defined problem. We compared our models with two different subreddits /r/machinelearning and /r/science and we noticed that best performing model for both datasets were different.

This project also helped us strengthen our understanding of various topic covered in class like different classifier and feature selection techniques, creating datasets and data cleansing for specific need, text extraction techniques like tokenization, stemming, smoothing etc. We also realized difficulties in web scraping e.g. link scraping, pdf parsing etc which forced us to use multiple toolkits. Choice of toolset is very important when it comes to deliverables. We felt that due to choice of Java language and toolkit like spark we ended up putting more time in coding than we estimated. We could have avoided this by choosing rapid development language like python and its associated machine learning toolkit like scikit-learn[28] to use its out of box features.

Individual Contributions:

Harrison: JRAW integration, Spark and Weka setup, heatmaps for /r/science results
Hardik: Literature review, programing, testing and debugging, evaluations of different classifiers, creating /r/science dataset and comparing it with r/machinelearning
Stan: Testing and debugging, tool setup, programming, literature review

References:

See respective footnotes.

[28] http://scikit-learn.org/stable/documentation.html