

Classification of Reddit Text Posts

Pradeep Kumar Srinivasan
Purdue University
sriniv68@purdue.edu

Mohammad Haseeb
Purdue University
mhaseeb@purdue.edu

ABSTRACT

We work on a large text classification problem, specifically the Reddit Self-Post Classification Task (RSPCT). RSPCT is a dataset with around 1000 classes ("subreddits") with around 1000 examples per class, which is unique because most text classification datasets have sparse labels. We use two traditional machine learning algorithms and one deep learning algorithm to learn to predict the class (i.e., the subreddit) given the title and body of a post. We evaluate the performance of each model using the Precision-at-K metric and gain insight into the models and the dataset by using experiments on the title and body of the posts and by feature analysis.

Keywords

NLP; large text classification; multi-class classification

1. INTRODUCTION

Text classification with few classes, such as in sentiment analysis, has been well-studied [2] with state-of-the-art techniques like LSTM. However, those techniques do not always work as well in scenarios with many classes [3]. Another issue with training on many-class datasets is that they have a very large number of labels (such as WikiLSHTC-325K dataset [5], which has 325K labels) and are sparse - most labels in the above dataset have less than 100 examples [4].

We want to study this problem by using a dataset that has a large number of classes but also a large number of examples per class. To this end, we use the Reddit Self-Post Classification Task (RSPCT) dataset and trained three models and evaluated their performance.

We think this is a useful problem to solve because it has not been adequately studied, to the best of our knowledge, and because this could pave the way to future work on such many-class datasets.

2. DATASET

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

Reddit is a popular social link aggregation, web content rating, and discussion website. The content on Reddit is organized into "subreddits", which are sub-forums on Reddit dedicated to a specific topic, for example, r/geography, r/gameofthrones, r/MachineLearning.

There are two main types of posts a user can submit on Reddit: a simple URL link and a self-post. A self-post consists of a topic and a body of text and is generally much longer in length, thereby, providing more information to Redditors. Moreover, most of the self-posts talk closely about the subreddit that they were posted in, which makes for a good classification task.

The Reddit Self-Post Classification Task (RSPCT) dataset [1] is a text corpus containing such self-posts (i.e., text posts) from Reddit. RSPCT was collected to help spur research on models that could tackle a large number of classes by ensuring a large population of examples for each class. The data consists of 1.013M self-posts, posted from 1013 subreddits (1000 examples per class). For each post, its title, body and the subreddit that it belongs to are provided.

The aim is to allow for a situation comparable to that in the computer vision community, which was helped by the famous ILSRVC competition (ImageNet [7]) with 1000 classes and around 1400 examples per class. This potential is what made us find this project interesting.

3. PREPROCESSING

Since the RSPCT dataset contains real self-posts from Reddit, the text in these posts do not always conform to the grammar, spelling, vocabulary and other nuances of the English language. For this reason, we carried out a series of preprocessing steps.

Because this is a text classification problem and most of our models used the bag-of-words approach, we first lowercase all the text in the subreddit, title and body of each post. We noticed that many of the posts contained special HTML encoded text and tags like >, &, <, <lb>, <tab> etc. We used regular expression matching to replace these tags with a blank space. We also removed punctuation characters like !, ", #, \$, %, &, (, [, *, + etc. such that we were only left with words at the end of this step.

Finally, we used Python's NLTK's PorterScanner to replace all words with their stem words. This resulted in words like "extremely" and "extreme" to be replaced with "extrem". *Figure __ shows the whole process for pre-processing one self-post* (TODO)

4. EXPLORATORY DATA ANALYSIS

4.1 Distribution of Characters in Post and Title

We first explore the dataset by looking at the histogram of the post titles (figure TODO). We find that most titles are shorter than 100 characters in length, with a median of 35 characters and 7 words.

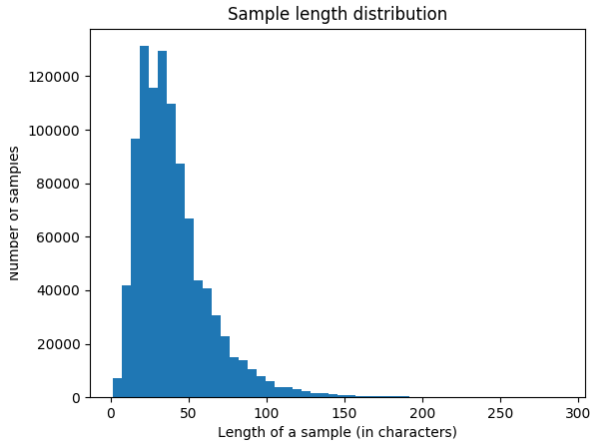


Figure 1: Sample Distribution for Length of Post's Title (in characters)

Similarly, the histogram for the post bodies (figure TODO) shows that most posts are shorter than 1000 characters in length, with a median of 500 characters and 105 words.

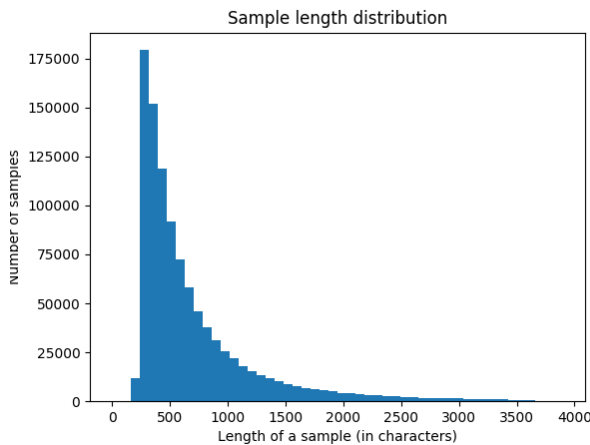


Figure 2: Sample Distribution for Length of Post's Body (in characters)

The t-SNE plot for the subreddits (figure TODO) shows that the subreddits seem to be well-separated by topic. For example, we can see clusters of subreddits for gaming, cryptocurrency, and diseases.

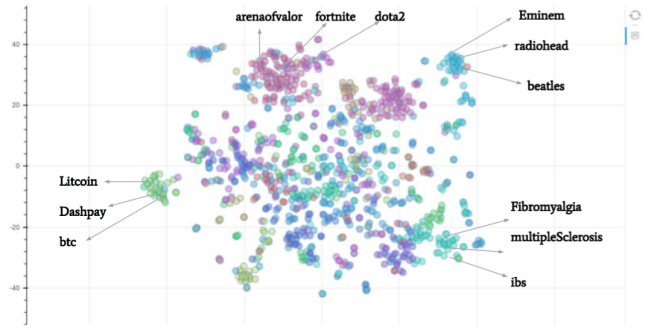


Figure 3: TSNE Plot of All the Self-Posts

5. ALGORITHMS IMPLEMENTED

5.1 Data Preparation

For the two traditional algorithms, we used a label encoder to transform each output label (subreddit name) into a number from 0 to 1012. Then, we used a TF-IDF vectorizer to get 30,000 top words, while excluding stop words and choosing only words that appeared in at least 5 documents and appeared in a maximum of 50% of the documents. For all three algorithms, we combined the title and body of the posts into one field.

5.2 Naive Bayes Classifier (NBC)

The first algorithm we looked at was the Naive Bayes Classifier because of its long use in text classification tasks (TODO). Specifically, we used `MultinomialNB` from `sklearn` with the hyperparameter `alpha`.

This model was the fastest to train among the three and gave us a quick baseline with which to compare the others. We also extended the maximum number of feature words to be 100,000 since it did not impact the training time too much. The training curve can be seen in figure TODO. We tried the following values of `alpha`: 0.1, 0.5, 0.9, 1.0, and found the best validation accuracy at 0.1.

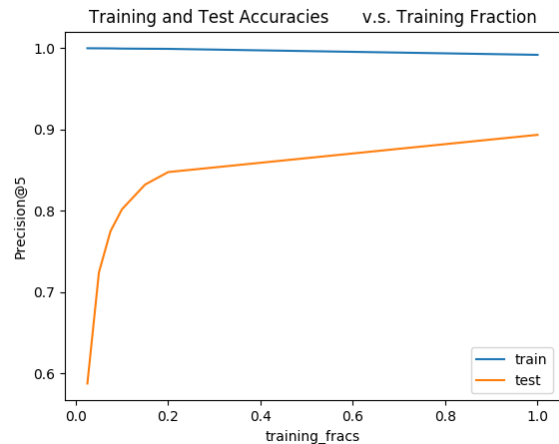


Figure 4: Learning Curve for Naive Bayes Classifier

5.3 Logistic Regression (LR)

The next traditional algorithm we wanted to consider was

Logistic Regression. We used `LogisticRegression` from `sklearn.linear_model` with the hyperparameter `C`.

The number of feature words used in this bag-of-words approach affected the training time a lot. Similarly, when hyperparameter tuning, we found that while values of `C` like 0.0001 and 0.01 led to very short training time, they also led to poor validation accuracy. On the other hand, `C` = 100 (i.e., not penalizing large weight values) led to better validation accuracy but at the cost of orders of magnitude longer training time. The training curve can be seen in figure TODO (TODO: need to add training fraction 0.20).

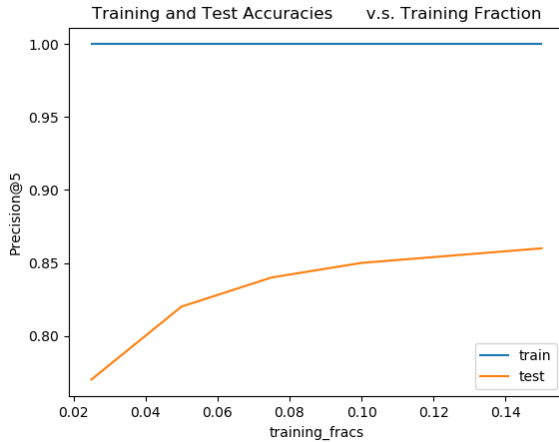


Figure 5: Learning Curve for Logistic Regression

5.4 Convolutional Neural Network (CNN)

Finally, we wanted to test a deep learning model on this dataset. We chose to use a Convolutional Neural Network because of its better average accuracy across different tasks and shorter training time than Recurrent Neural Networks.

To embed the words in the post, we used the GloVe (Global Vectors for Word Representation) embedding [6]. We tried a version with 100 embedding dimensions for each word and another version with 300 embedding dimensions and found the later to give quite a boost in overall accuracy.

We considered two architectures for the CNN, one with three successive layers of filters of size 5 and one with a multi-channel model of filters of sizes 2, 3, and 5. Upon testing, we found that the latter gave much higher accuracy even after a few epochs whereas the former seemed to overfit after a few epochs (TODO: maybe show the training curve for the old architecture). So, our multi-channel approach would consider two words at a time, three words at a time, and five words at a time. The later layers use max-pooling and softmax to predict one class label out of 1013 labels. Finally, we use dropout with probability 0.3. The optimizer we use is Adam because of its adaptive learning rate and the loss is categorical cross-entropy.

We trained the CNN for a maximum of 10 epochs over several days and obtained the following learning curve (figure TODO).

5.4.1 Performance

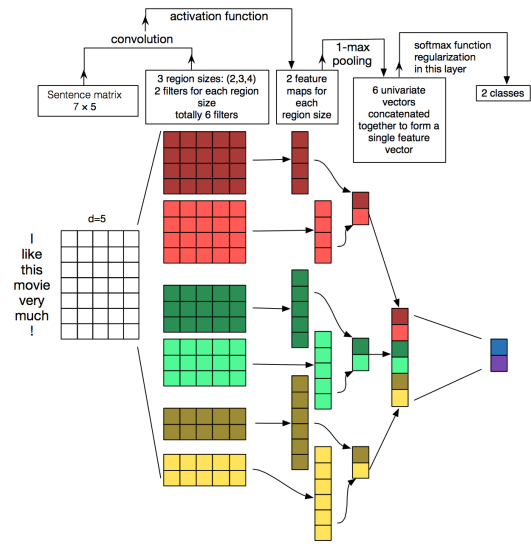


Figure 6: CNN Architecture

5.5 Performance Comparison

TODO

6. FURTHER EXPERIMENTS

6.1 Confusion Matrix

TODO: add matrix and explain it

6.2 Toggle Title and Body

TODO: add comparison table

6.3 Performance Using Features for Sentiment Analysis and Readability Level

TODO: add graphs

6.4 Qualitative Analysis

TODO: talk about this

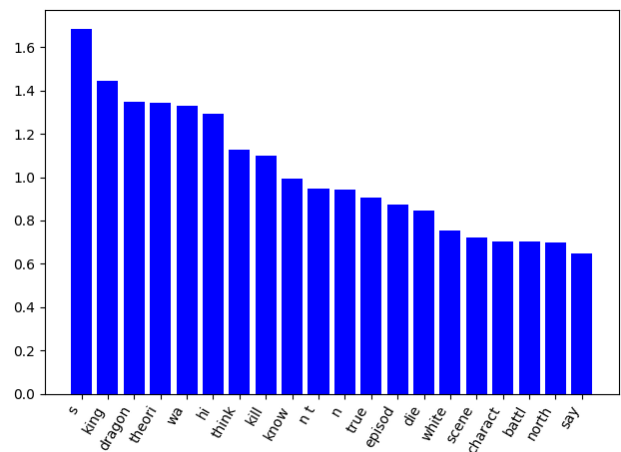


Figure 7: Important Features for subreddit r/gameofthrones

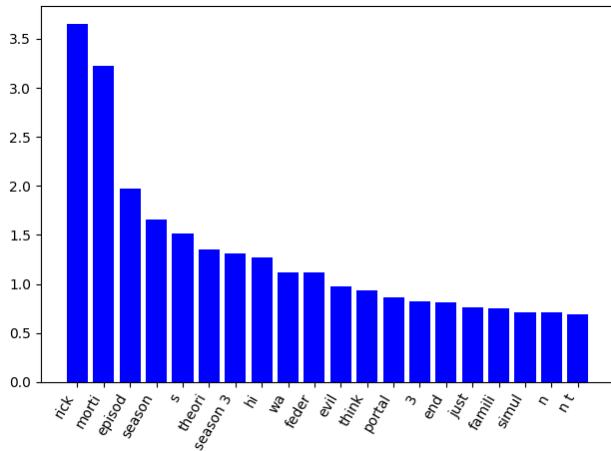


Figure 8: Important Features for subreddit r/rickandmorty

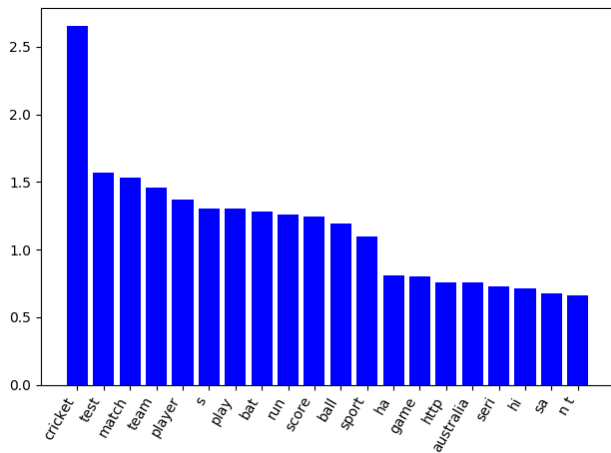


Figure 9: Important Features for subreddit r/cricket

- [2] A. Agarwa. Sentiment analysis of twitter data. *Association for Computational Linguistics*, 2011.
- [3] R. Ghani. Combining labeled and unlabeled data for multiclass text. 2001.
- [4] M. S. Jones. The reddit self-post classification task (rspct): a highly multiclass dataset for text classification (preprint).
- [5] I. Partalas, A. Kosmopoulos, N. Baskiotis, T. Artieres, G. Paliouras, E. Gaussier, I. Androutsopoulos, M.-R. Amini, and P. Galinari. Lsttc: A benchmark for large-scale text classification. *arXiv preprint arXiv:1503.08581*, 2015.
- [6] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.
- [7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F.-F. Li. Imagenet large scale visual recognition challenge. *CoRR abs/1409.0575*, 2014.

7. EVALUATION OF PROJECT GOALS

As promised in our initial proposal, we measured the top-1, top-3, and top-5 precision for Naive Bayes, Logistic Regression, and Convolutional Neural Networks. We also presented the learning curves for all three, including the loss vs number of epochs for CNN.

Further, we experimented with the title and body of the post and measured the precision for three different cases. Also, we added features for sentiment rating and readability level to measure the impact on the precision.

8. CONCLUSIONS

TODO

9. TEAM MEMBER CONTRIBUTIONS

10. REFERENCES

- [1] The reddit self-post classification task. <https://www.kaggle.com/mswarbrickjones/reddit-selfposts>.