Classification of Reddit Text Posts

Pradeep Kumar Srinivasan
Purdue University
sriniv68@purdue.edu

Mohammad Haseeb
Purdue University
mhaseeb@purdue.edu

ABSTRACT

In this project, we worked on the problem of large text classification, 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 used 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 evaluated the performance of each model using the Precision-at-K metric. We conducted additional experiments like adding features for sentiment analysis and readability level of a post to see if that increases the performance of our models.

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 [6], which has 325K labels) and are sparse - most labels in the above dataset have less than 100 examples [4].

We aim 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 used 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

© 2019 ACM. ISBN 123-4567-24-567/08/06…$15.00
DOI: 10.475/123_4

Reddit is a popular social link aggregation, web content rating, and discussion website. Reddit's users submit posts, which are then voted up or down by other members. This allows the highest rated posts to gain the most attention. 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.

We obtained the Reddit Self-Post Classification Task (RSPCT) dataset from Kaggle [1]. It is a text corpus containing 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 [8]) with 1000 classes and around 1400 examples per class. This potential is what made us find this project interesting.

3. PREPROCESSING

As mentioned in the previous section, the RSPCT dataset contains real self-posts from Reddit. This means that the text in these posts may 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 &gt, &amp, &lt, <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***

4. EXPLORATORY DATA ANALYSIS

4.1 Distribution of Characters in Post and Title

TODO: Explain this

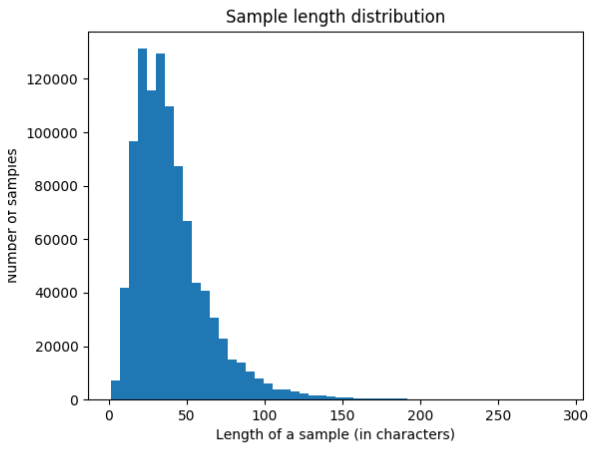

Figure 1: Sample Distribution for Length of Post's Title (in characters)

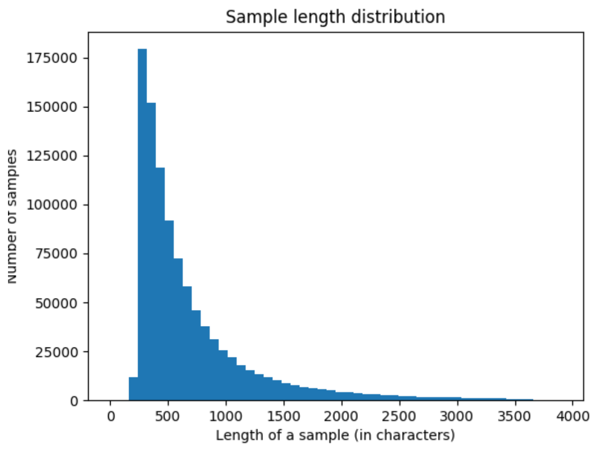

Figure 2: Sample Distribution for Length of Post's Body (in characters)

4.2 TSNE

TODO: What to include for this? To mention that we did TSNE or not? Or to just give a reference to this?

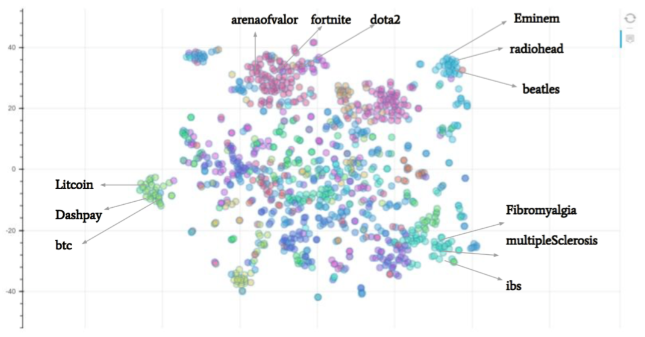

Figure 3: TSNE Plot of All the Self-Posts

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.

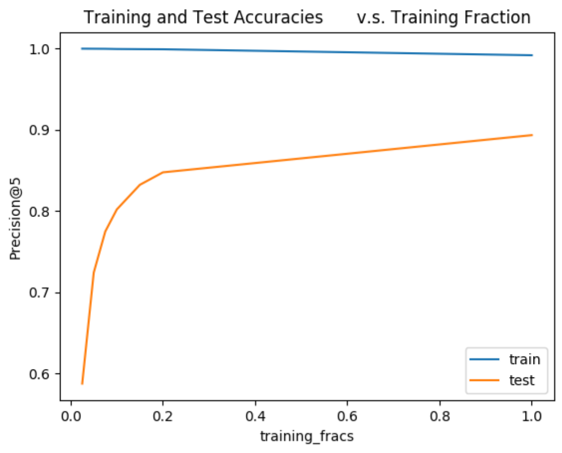

Figure 4: Learning Curve for Naive Bayes Classifier

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.

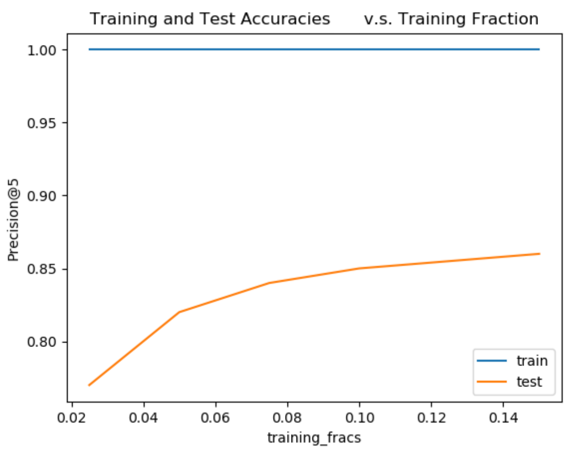

Figure 5: Learning Curve for Logistic Regression

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 then Recurrent Neural Networks.

To embed the words in the post, we used the GloVe (Global Vectors for Word Representation) embedding [7]. 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

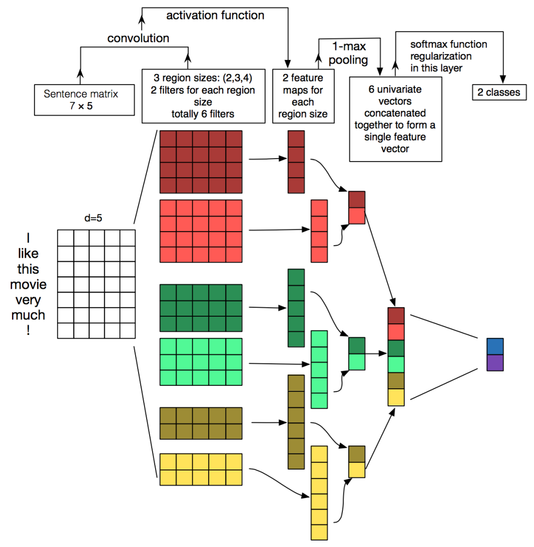
activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

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

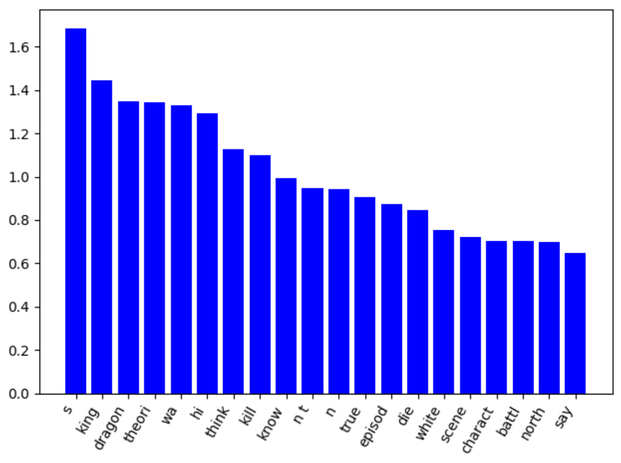

Figure 7: Important Features for subreddit r/gameofthrones

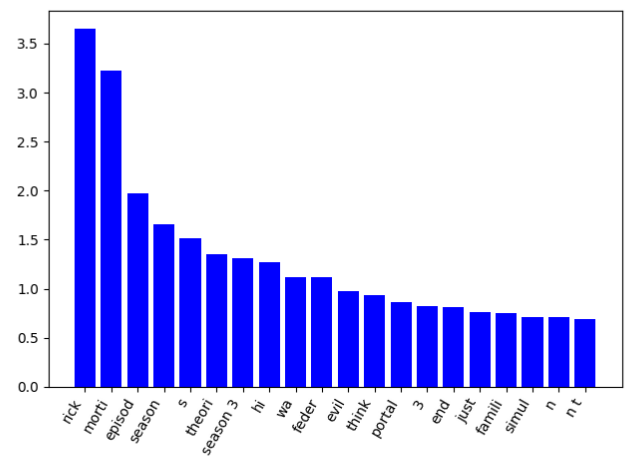

Figure 8: Important Features for subreddit r/rickandmorty

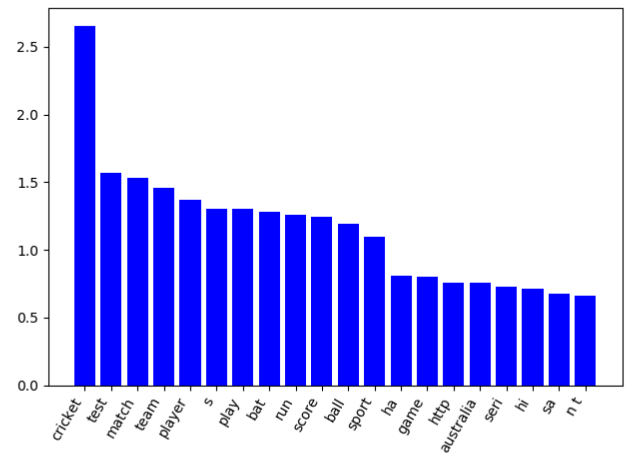

Figure 9: Important Features for subreddit r/cricket

7. EVALUATION OF PROJECT GOALS

TODO: take part from proposal

The key metric for the models will be the Precision-at-K metric [5], since this is a problem with a very large number of classes and we don't expect the correct label to be predicted as the top option. We will measure the precision on the top-1, top-3, and top-5 labels. As mentioned before, we will also experiment with adding a new feature for sentiment rating of the text to see if that improves performance. To understand the impact of the title on classification, we will measure performance by training and testing on just the titles of the posts, just the bodies of the posts, and then with both title and body.

In addition, for the deep learning model, we will plot the loss and accuracy curves vs number of epochs. For the traditional models, we will plot the learning curve vs training set size.

What we promised: 1. Preprocessing through TSNE and sample What we did not promise but delivered:

8. CONCLUSIONS

TODO

9. TEAM MEMBER CONTRIBUTIONS

10. REFERENCES

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

[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] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[6] I. Partalas, A. Kosmopoulos, N. Baskiotis, T. Artieres, G. Paliouras, E. Gaussier, I. Androutsopoulos, M.-R. Amini, and P. Galinari. Lshtc: A benchmark for large-scale text classification. *arXiv preprint arXiv:1503.08581*, 2015.

[7] 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.

[8] 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.