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 [8] metric. We conducted additional experiments like adding features for sentiment analysis and readability level of a post's text [10] 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 [4] with state-of-the-art techniques like LSTM. However, those techniques do not always work as well in scenarios with many classes [6]. Another issue with training on many-class datasets is that they have a very large number of labels (such as WikiLSHTC-325K dataset [9], which has 325K labels) and are sparse - most labels in the above dataset have less than 100 examples [7].

We aimed 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

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 [3]. 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 [13]) 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

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

words like "extremely" and "extreme" to be replaced with "extrem".

4. EXPLORATORY DATA ANALYSIS

We first explore the dataset by looking at the histogram of the post titles (Figure 1). We find that most titles are shorter than 100 characters in length, with a median of 35 characters and 7 words.

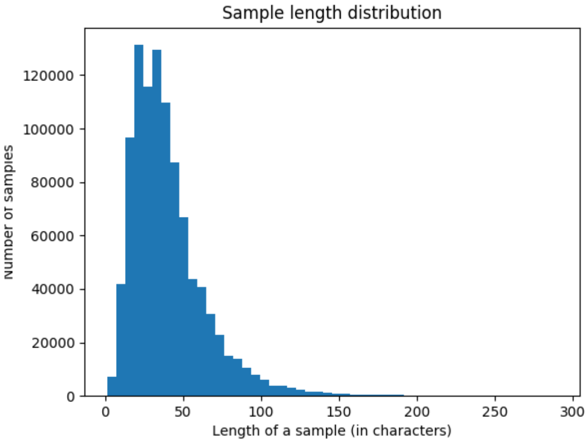

Figure 1: Sample Distribution for Length of Post's Title (in characters)

Similarly, the histogram for the post bodies (Figure 2) shows that most posts are shorter than 1000 characters in length, with a median of 500 characters and 105 words.

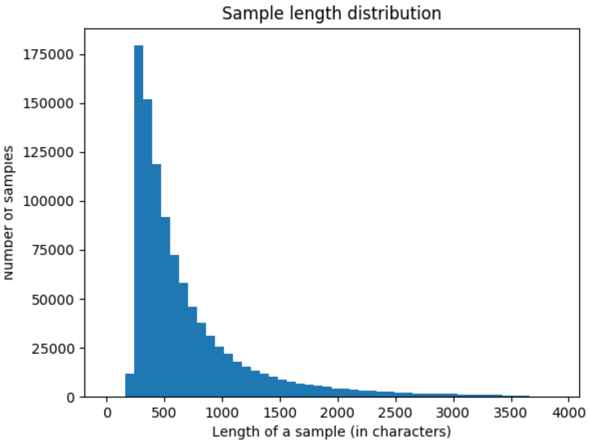

Figure 2: Sample Distribution for Length of Post's Body (in characters)

The t-SNE [14] plot for the subreddits (Figure 3) shows that the subreddits seem to be well-separated by topic. For example, we can see clusters of subreddits for gaming, cryptocurrency, and diseases.

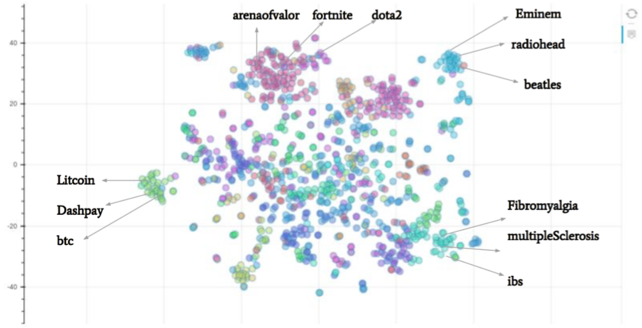

Figure 3: TSNE Plot of All the Self-Posts

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 [12]. 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 4. We tried the following values of `alpha`: 0.1, 0.5, 0.9, 1.0, and found the best validation accuracy at 0.1.

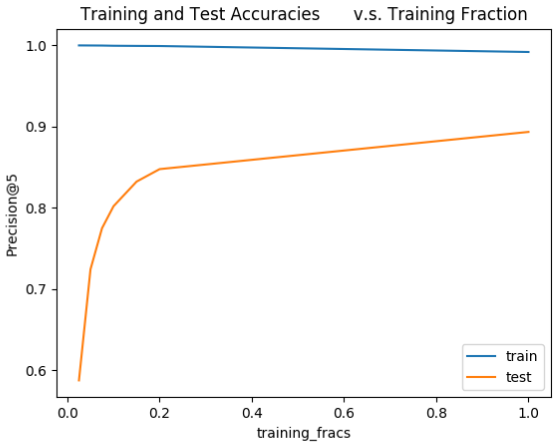

Figure 4: Learning Curve for Naive Bayes Classifier

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 [5] 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 5.

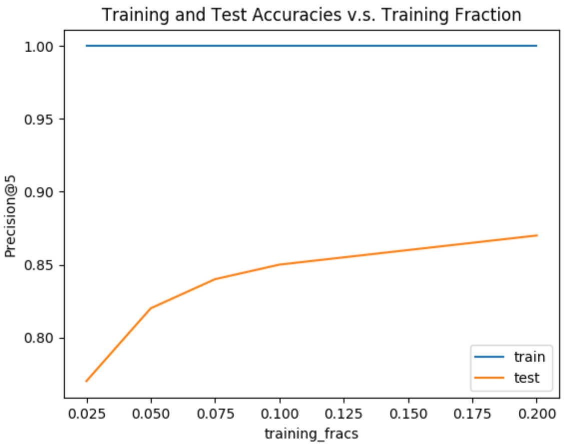

Figure 5: Learning Curve for Logistic Regression

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 [11]. 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. This architecure is shown in Figure 6. 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 7).

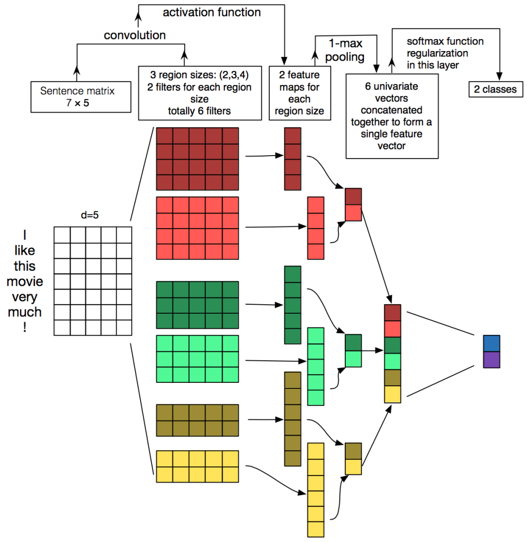
activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 x 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes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

Figure 6: CNN Architecture

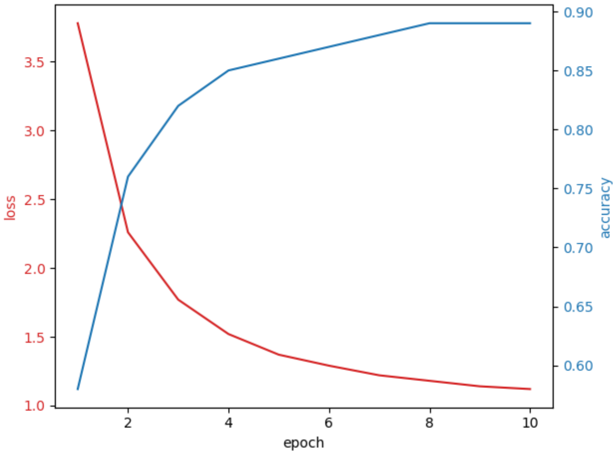

Figure 7: Learning Curve for CNN

5.5 Performance Comparison

We can see from the following Table 1 that Precision@1 for CNN is only 0.56. This is most likely because the model is not able to accurately predict the exact class label. On the other hand, CNN performs much better for Precision@5 with an accuracy of 0.89, where the true class label is in the top-5 predicted labels.

Another interesting observation is that the Precision@5 accuracy for all the models caps out at 0.89. We think this is due to some underlying property or limitation of the dataset and requires further experimentation.

	Precision@1	Precision@3	Precision@5
NBC	0.74	0.86	0.89
LR	0.71	0.84	0.89
CNN	0.56	-	0.89

Table 1: Precison@1, 3 and 5 for each Model

6. FURTHER EXPERIMENTS

6.1 Toggle Title and Body

Table 2 shows the Precision@5 accuracy for both NBC and LR with only the posts' titles, only the posts' bodies and finally, with both.

The median length of the titles is 35 characters and 7 words. It is interesting to see that, even with such a small size, we can still distinguish between 1000 labels and get an accuracy of 0.59 and 0.64. This is pretty remarkable and can lead to further research on similar problems.

However, we also find that, given the body, the title alone does not add too much information and we do not see much increase in the accuracies of the models. This confirms our earlier intuition that the content of the title and the body are similar for a post and are very closely related to the subreddit that it was posted in.

	Naive Bayes Classifier	Logistic Regression
Title Only	0.64	0.59
Body Only	0.87	0.86
Both	0.89	0.89

Table 2: Precision@5 with Title and Body Toggled

6.2 Feature Importance Analysis

Figures 8- 10 show the feature importance for three subreddits: r/gameofthrones, r/rickandmorty, and r/cricket. Note that these were obtanied from stemmed data, so the graphs only show stemmed features.

We can see from these figures that our model seems to capture the key features of the respective subreddits. For example, "king", "dragon", "north" are of high importance in r/gameofthrones (Figure 8); "rick", "portal", "federation" are of high importance in r/rickandmorty (Figure 9); and "cricket", "match", "bat" are of high importance in r/cricket (Figure 10).

However, we can also see that some features were mistakenly given high importance. For example, "http" was given a high importance in r/cricket, which does not really make sense. This suggests scope for improvement in future work.

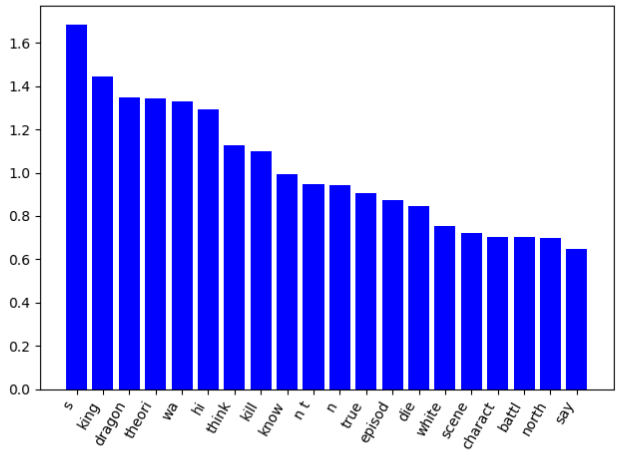

Figure 8: Important Features for subreddit r/gameofthrones

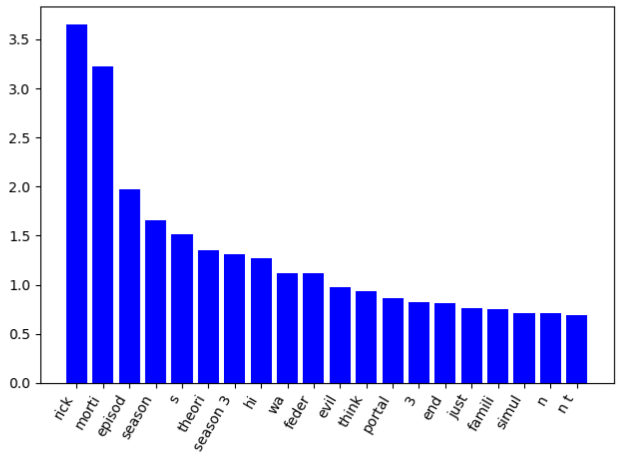

Figure 9: Important Features for subreddit r/rickandmorty

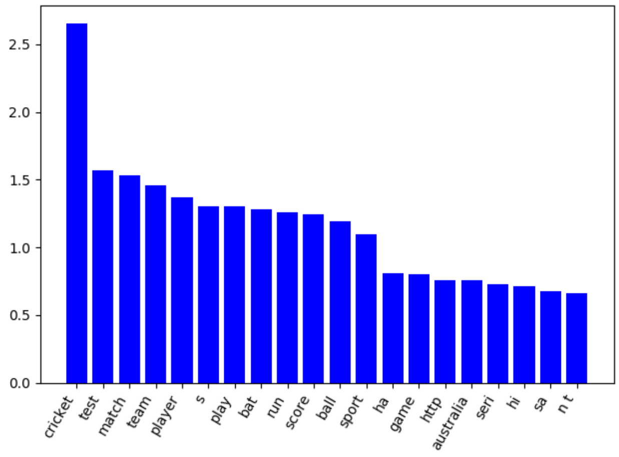

Figure 10: Important Features for subreddit r/cricket

6.3 Performance Using Features for Sentiment Analysis and Readability Level

We used Python's `TextBlob`[1] library to get the sentiment value of each post. Similarly, we used Python's `textstat`[2] library to compute the readability score of each post.

We compared the behaviour of Logistic Regression with and without those features on 100k rows of the dataset:

	Prec@1	Prec@3	Prec@5
With sentiment and readability	0.01	0.02	0.02
Without those features	0.58	0.75	0.80

Table 3: Sentiment and readability level: Precision

Unfortunately, adding those two features degrades performance completely. We believe it is because we add those two features to the sparse matrix that we get after vectorizing the bag-of-words, which leads the model to overweight those features even when they are not highly informative. In any case, we report our experimental results above.

6.4 Confusion Matrix

To analyze which subreddits are frequently mispredicted, we look at the confusion matrix. As the 1000x1000 matrix for all the labels was hard to read, we have zoomed in on the subreddits that were most frequently confused.

Figure 11 shows the top 40 most-confused subreddits, where we can see the characteristic diagonal of dark squares along with a few lighter squares denoting the mispredicted labels.

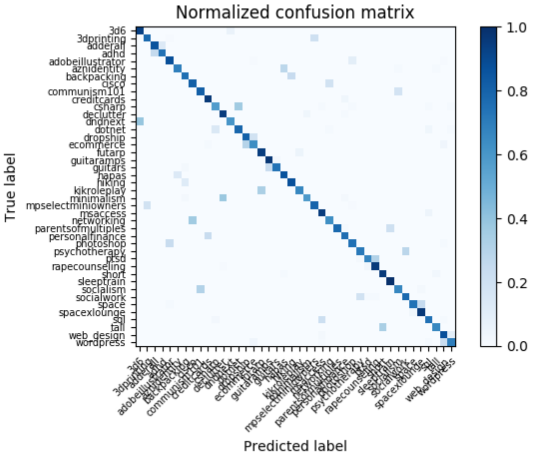

Figure 11: Confusion Matrix of the Top-40 Most-Confused Subreddits

Another version is in Figure 12 where we can see that r/Adderall and r/ADHD are often confused, as are r/dropship and r/ecommerce.

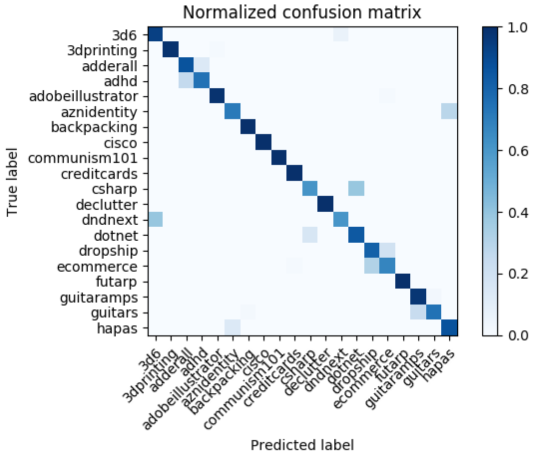

Figure 12: Another Version of the Confusion Matrix

Again, since the matrix is quite sparse it is hard to show them on the same plot, so we decided to visualize the true label along with its most-confused label in the same plot (Figure 13). We see that posts from r/hiking are attributed to r/backpacking, posts from r/cisco are attributed to r/networking, and, for some reason, posts from r/short are attributed to r/tall (probably because both talk about height).

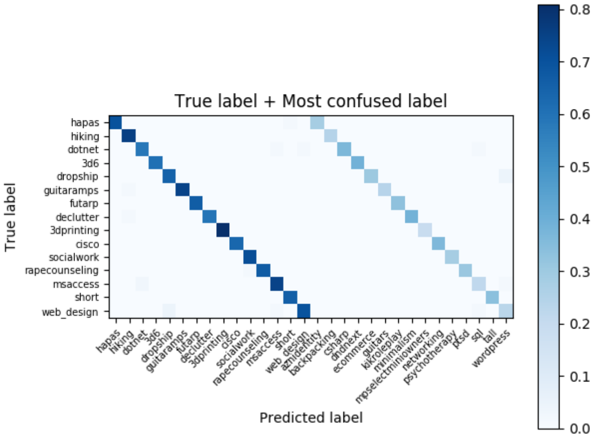

Figure 13: True Label and Most-Confused Label on Same Graph

To get a better idea of what confuses the model, we look at two concrete subreddits: r/hiking and r/backpacking. If we compare their most important features (Figure 14 and Figure 15), we find that many of them are the same, such as “trip”, “day”, “plan”, “trail”, and “camp”. This might be an inherent limitation of the dataset, because these are not well-separated subreddits and we may not be able to distinguish them given the posts alone.

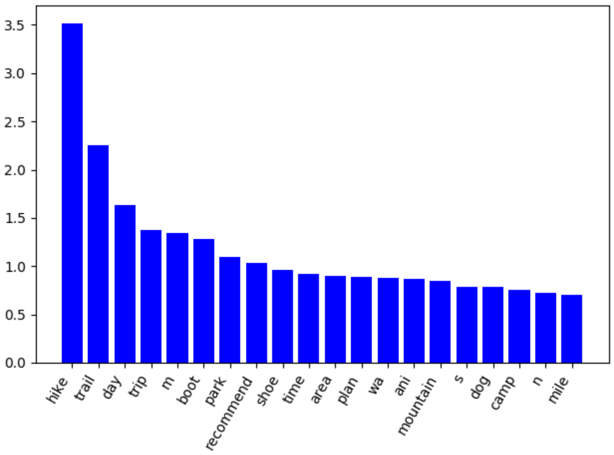

Figure 14: Important Features for subreddit r/hiking

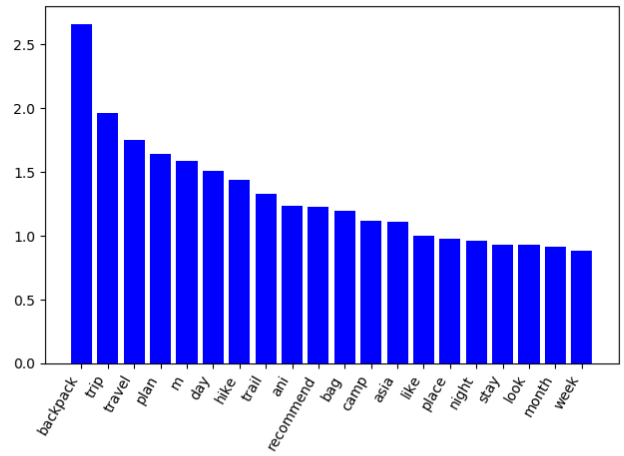

Figure 15: Important Features for subreddit r/backpacking

7. EVALUATION OF PROJECT GOALS

Our project goals consisted of:

1. Preprocessing

   We preprocessed the data by converting it to lowercase, removing encoded HTML and other tags, removing punctuations and stemming the words.

2. Exploratory Data Analysis

   Exploratory Data Analysis was done by exploring the sample distribution length for the title and body of the posts and by plotting a TSNE plot to see if similar subreddits are clustered together.

3. Implementing algorithms

   We implemented two traditional learning algorithms (Naive Bayes Classifier and Logistic Regression) and a deep learning model (Convolutional Neural Network). We also tuned the hyper parameters of these algorithms to get the best possible performance on our dataset.

4. Toggling post's title and body

   To understand the impact of the title on classification, we measured the 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.

5. Experiment with adding new features

   In order to see if we can extract even more information from the data, we added features for the sentiment value and readability score of every post in the dataset.

In addition to the above, we also computed the confusion matrix and carried out qualitative analysis by computing the feature importance for our models.

We have, therefore, achieved our goals that we set in the project proposal.

8. CONCLUSIONS

Overall, we were able to get good results on the RSPCT dataset, which supports our hypothesis that one of the main bottlenecks in large text classification tasks is the sparsity of examples for labels. All three models were clearly able to learn over the dataset and curiously produced very similar results for the top-5 classification accuracy. Future work might focus on either increasing the accuracy on this dataset using better model and more tuning or aggregating such a large number of examples per label for other tasks (perhaps using Reddit itself). Another question to answer is whether adding more than 1000 examples per label might improve accuracy even further or whether the benefits would taper off.

Our experiments on the title and body showed us that as few as 7 words could allow a reasonably-high degree of accuracy even with 1000 labels. Our investigation of the feature importance shows us the strengths of the models and points to clear steps for future work. Finally, our analysis of the confusion matrix points to the most-confused subreddits, which could inform future work in this area.

9. ACKNOWLEDGEMENTS

We would like to thank Professor Ming Yin for her suggestion about using a confusion matrix for pinpointing the subreddits that frequently confused our model.

10. TEAM MEMBER CONTRIBUTIONS

Pradeep Kumar Srinivasan: Worked on model implementation and tuning, report, confusion matrix, presentation, slides, feature importance

Mohammad Haseeb: Worked on preprocessing, exploratory data analyis, sentiment value and readability score features, slides, presentation, report

11. REFERENCES

[1] TextBlob: Simplified Text Processing. https://textblob.readthedocs.io/en/dev/.

[2] TextStat: Calculate Statistics From Text. https://pypi.org/project/textstat/.

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

[4] A. Agarwa. Sentiment analysis of twitter data. *Association for Computational Linguistics*, 2011.

[5] A. Besbes. Overview and benchmark of traditional and deep learning models in text classification. https://www.kdnuggets.com/2018/07/overview-benchmark-deep-learning-models-text-classification.html, 2018.

[6] R. Ghani. Combining labeled and unlabeled data for multiclass text. 2001.

[7] M. S. Jones. The reddit self-post classification task (rspct): a highly multiclass dataset for text classification (preprint).

[8] 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.

[9] 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.

[10] G. C. Pastor. Translation universals: do they exist? A corpus-based NLP study of convergence and simplification. https://www.researchgate.net/profile/Viktor_Pekar/publication/254434062_Translation_universals_do_they_exist_A_corpus-based_NLP_study_of_convergence_and_simplification/links/0046353b6bfcaa134b000000/Translation-universals-do-they-exist-A-corpus-based-NLP-study-of-convergence-and-simplification.pdf, 2014.

[11] 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.

[12] J. D. Rennie. Improving multi-class text classification with naive bayes. 2001.

[13] 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.

[14] L. van der Maaten. Visualizing Data using t-SNE. https://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf, 2008.