CS 4200 Final Report: Reddit Post Title Classification Using Huggingface's RoBERTa and Scikit Learn's MultinomialNB

Nicholas Calkins, Khoa Le, Matthew Levitt, Alex Liu
California State University Polytechnic, Pomona
NTCalkins@cpp.edu
khoale@cpp.edu
mzlevitt@cpp.edu
alexl@cpp.edu

Abstract

Classification of data based on clear data and algorithms has long been a staple of automation by computers. However, there are some categories of data that are not so cut and dry when it comes to classification. This is what our team experienced when attempting to classify titles of scientific posts made by users of Reddit.com. Using both the Multinomial Naïve Bayes algorithm for classification and Huggingface Transformer library, our team was able to achieve a reasonable level of success in determining the "category" of a post given the content of the title of the post.

Introduction

Reddit.com is a site that allows users to share multimedia across the world. The site itself subdivides into communities, wherein a user can curate a personal feed made up of the feeds from a subset of communities. One such community, dubbed a "subreddit", is called "science". In this science subreddit, users share scientific articles and tag the article with the relevant tag. For example, an article about squirrels recycling nutrients during hibernation might be "flaired" with the "animal science" or "environment" flair. An article talking about an AI model that is being developed to detect Alzheimers might be "flaired" with the "computer science" or "neuroscience" flair.

In the case of reddit.com, these tags must be determined by the user posting. Our team's goal was to create a model that could tokenize titles and train a model to draw relationships between titles and flairs such that it could determine the appropriate flair for a title if given just a title.

According to Amazon's website rankings, reddit.com is the 17th most popular site in the world, with 40% of the traffic coming in from the United States (Reddit Competitive Analysis, 2020). By developing a model that can more correctly recommend a flair to a posting user, users can be sure that their post is conforming to what other posts in that category are like.

This is an interesting problem primarily because our team had the opportunity to participate in the dataset creation as well as the model specification, training, and validation. We also got to compare two different approaches and draw our conclusions about what went wrong and right.

Related Work

Broadly, this problem is a classic sequence classification problem. There are titles each with a flair, and we want our model to be able to predict a flair given a title. Sequence classification is a broadly applied and researched field so our team will cover the most relevant.

For our project, we first made use of Huggingface's Transformer architecture. The Transformer is used in the field of Natural Language Processing and it was introduced only in 2017; therefore, it stands to reason that sequence classification was done using recurrent neural networks before 2017. As noted in the seminal paper *Attention is All You Need*, the Transformer allows for more parallelization on sequential data. This reduces training times greatly (Vasawani et. al, 2017). This has allowed for pre trained systems like BERT and RoBERTa, the latter of which our team used for this project.

Roberto Silviera was able to use this model from Huggingface to successfully predict the intent of a voice message given the English content of a query to a virtual assistant (Silveira, 2019). His work was a good reference for our team to figure out the specifics of using Huggingface's RoBERTa for sequence classification. This differed from our project in the sense that our sequences were much more variable and our dataset was not as curated as the one used by Silveira.

As for the Multinomial Naïve Bayes for classification, our process is not necessarily innovative. Our team was able to read through the documentation of Scikit Learn, Susan

Li's article on text classification models (2018), and other resources to determine what had to be done.

Method

Building the Dataset

Our project began with building the dataset. Using the Python Pushshift.io API Wrapper (PSAW), we were able to build general purpose tools for extracting data from reddit.com for use in building datasets. This allows for more future utility in that we could build larger and more diverse datasets when needed.

This modularity allowed us to repurpose these tools when our team discovered the difficulties and roadblocks with trying to determine personality from a user's posts. In our project, it allowed us to change our focus from detecting personalities to branches of science based on article names.

Once we collected our dataset, we began to tally the number of articles we had per science field. Delving further into testing, we realized that it was harder to distinguish certain fields from another as there was overlap between a number of different fields such as cancer, medicine, and health.

In addition, our team found that certain categories heavily resisted classification. Such a category was nanoscience, wherein the topics were so interdisciplinary that it was poisonous to the accuracy of our model.

	title	flair
0	Young Blood May Hold Key to Reversing Aging	Health
1	Mathematical model describes the physical prin...	Computer Science and Engineering
2	How one scientist is growing miniature brains ...	Medicine
3	A Second 'Big Bang' Could End Our Universe in ...	Astronomy
4	It's Possible To Cut Cropland Use in Half and ...	Environment

Figure 1: A small slice of a dataset from reddit. Note that the titles are truncated for display.

Improving Dataset Building

Although PSAW was proficient at gathering large amounts of data, our team found that the data generated often had a lot of garbage. For example, posts made by users would be low quality, unrelated to anything scientific, and possibly spam. Other times, the posts would be non-unique which made for highly duplicate data. Our team was able to mitigate these issues through several avenues.

First, PSAW was run with a directive to go through each month of every year since 2013. Within this month and year, PSAW would fetch every post made in score-descending order. Scores are given incrementally by users and serve as an imperfect but useful measurement of the general quality of a post.

In addition, the URL of the related source for every post would be saved into a set. If an upcoming post had a URL that had already been in the set, then this post would be eschewed.

Finally, a post would only be included if the user had given it a flair. We were relying on the good will of the user to properly flair their post, something that we'll discuss later on in the results.

Through this attention to detail in building our datasets, our team was able to curate and produce many quality, unrepeated, relevant datasets for testing. Our major dataset that was used had 12 categories with 259 quality, unrepeated posts in each category. On top of this, our team generated several other datasets, none with quite the quality and size.

Preprocessing the data for The RoBERTa Model

Once the data has been collected from reddit using the PSAW module, the data then can be preprocessed to be fit into the RoBERTa model. To first do the preprocessing, the data was read from a comma separated value spreadsheet using the Pandas Python module. Once the data has been read from the spreadsheet, a function is used to gather the labels (flairs) of each post in the dataset. This function is used to check if the dataset might have an abundance of a particular label.

Having a skewed dataset can train the model to only recognize flairs based on the skewed flairs, so this function allows to check for a skewed dataset and balance out the dataset. Once the flair amounts are calculated for every reddit post in the dataset, the dataset is then divided into a training dataset, a validation dataset, and a testing dataset. Finally each unique flair in the dataset is assigned a number, so that RoBERTa can use that number to predict a flair based off a reddit post.

Creating the RoBERTa Tokenizer

In terms of the libraries that were used to create the model, it was created using Facebook's PyTorch machine learning library and PyTorch's transformers architecture for the RoBERTa model. Our team used the standard RoBERTa configuration from 'roberta-base', only changing the number of hidden layers from 12 to 16.

The amount of labels for the RoBERTa model were also configured based on the amount of unique flairs identified in the entire dataset. Once the model was configured, a RoBERTa tokenizer is instantiated to encode words into numbers to be passed through the model. The tokenizer is derived from the GPT-2 tokenizer and uses byte-level Byte-Pair-Encoding (BPE) (Huggingface, "RoBERTa") .

Byte-Pair-Encoding works through compressing data by iteratively replacing the most frequent pair of bytes in a sequence (in this case a reddit post) with a single, unused byte (Senrich et. al, 2016). In the case for byte-level BPE, once a sequence of words have been fed through the tokenizer, the tokenizer will generate a set of

unique words and it will generate the frequency at which each word occurred in the training data.

BPE will then create a base vocabulary that consists of all symbols that occur in the set of unique words. It then learns merge rules to form a new symbol from two symbols of the base vocabulary. It continues to do this until the vocabulary has achieved the desired vocabulary size which is a hyperparameter that can be tuned before training the tokenizer.

For byte-level BPE, all the base vocabulary uses bytes instead of characters to ensure that every base character is included in the vocabulary (Huggingface, "Summary of the Tokenizers"). To get better accuracy on the dataset, undesirable characters such as punctuation marks like "?" or !" were removed as these marks don't give much context for classifying different reddit posts based on their flair titles.

	BPE based on bytes	BPE based on characters
1	'I like cats.'	'I like cats.'
2	'I', ' like', 'cats', '.'	'I', 'like', 'cats', '.'
3	['0x49'], ['0x20', '0x6c', '0x69', '0x6b', '0x65'], ['0x20', '0x63', '0x61', '0x74', '0x73'], ['0x2e']	–
4	'I', 'Ġlike', 'Ġcats', '.'	–
5	'I', 'Ġli', 'ke', 'Ġca', 'ts', '.'	'I', 'li@@', 'ke' 'ca@@', 'ts', '.'

Figure 2: Illustrates BPE encoding based on bytes and based on characters for the phrase 'I like cats.' (Github, 2020).

Creating the RoBERTa Model

Once a reddit post was passed through the RoBERTa tokenizer, the RoBERTa model was ready to be created. To have the tokenized data be in a format to be processed into the model, a python class called Science was created to get each title and flair from a post. The title would be tokenized and the flair label would be assigned a number based on its unique flair label.

The training, validation, and testing datasets would be passed through the Science class and turned into new sets to have each post be preprocessed and ready to be passed through the model. These new sets would then be turned into PyTorch DataLoader to be turned into a format that the model could recognize.

Finally, the model itself was created where each post and flair would be forward passed through the RoBERTa model. The model would then calculate the weights and those weights then be backpropagated through the network. The optimizer would then perform a parameter update based on the gradients that were backpropagated.

After every 100 iterations, the model will check for each post in the training data if it has accurately predicted a flair. The current iteration, loss calculated by the loss function at that iteration, and the accuracy of that iteration are then outputted to the screen. This training continues for the amount of epochs specified.

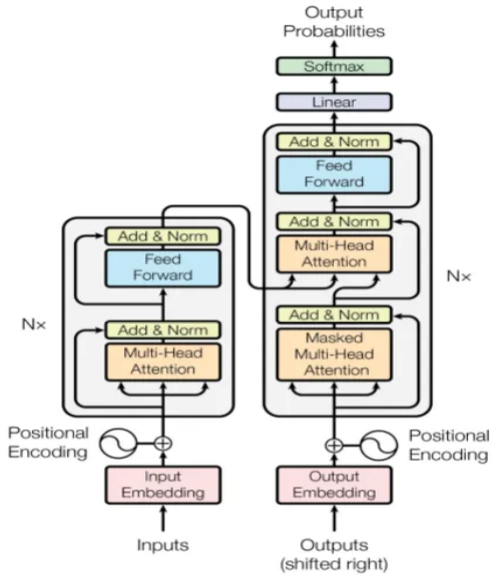

Figure 3: Transformer architecture like RoBERTa. Illustrates what is happening to each tokenizer post when fed through the model. (Vaswani, et. al.)

Training the RoBERTa Model

The model was trained using the cross entropy loss function as this loss function is good for classification problems. There are multiple categories to classify, so cross entropy is used over binary cross entropy. The optimizer used was the Adam optimizer at a low learning rate of 0.000001, so that the model wouldn't diverge. Finally, epoch size could be changed to let the model train on more instances of the training dataset. Increasing this hyperparameter generally gave us good results, but it resulted in the network taking a longer amount of time to train.

Multinomial Naïve Bayes Overview

Naïve Bayes methods are algorithms based on applying Bayes' theorem. It is naïve because it assumes that there is conditional independence between every pair of features. Given a flair of y and a dependent feature vector x_1 through x_n, courtesy of Scikit (Scikit, "Naive Bayes):

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

Because $P(x_1,\ldots,x_n)$ is constant, a classification rule is generated, in that:

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

We use the Multinomial Naïve Bayes algorithm as we assume that our textual data is distributed multinomially. This particular algorithm is also used commonly in text classification. Our team researched the implementation of the algorithm for pedagogical purposes but were shielded from implementation through Scikit Learn.

Preprocessing the data for the Multinomial Naïve Bayes Model

The text cleanup is much like RoBERTa but there is no tokenizing of the words. Instead, we remove "stopwords" that don't add any context and have no predictive value in this context. We also make everything lowercase so two words that are the same aren't confused due to capitalization. In the pipeline, a CountVectorizer will be used to convert the titles of posts into a matrix of token counts.

Training the Multinomial Naïve Bayes Model Using a Pipeline

We use Scikit's pipeline to first convert our titles into a matrix of token counts. We then transform that count matrix to a tf-idf representation. Tf-idf is a statistical measure for evaluating how relevant a word is to a class of titles, in our case (Scikit, "Feature Extraction"). Finally, we use Scikit's Multinomial Naïve Bayes for classification.

Results

RoBERTa for Sequence Classification Results

Our team trained our RoBERTa model for 30 epochs, with each epoch having 9 iterations. This was a long process but one that ultimately yielded a fruitful model. During training, the model went from ~8% accuracy to just under 30% accuracy.

Our team considered this a win primarily because with 12 categories, 8% is the average success of random guessing. Given the complexity of this problem, our team was happy with 30% primary accuracy.

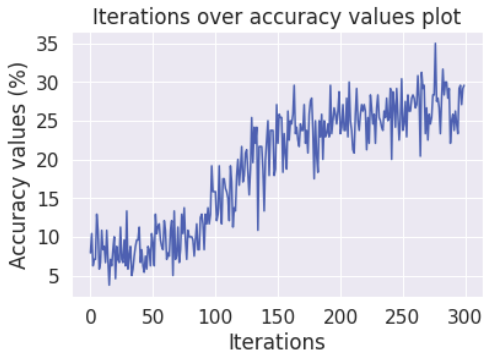

Figure 4: Iterations over accuracy values for RoBERTa training

Top 3 Accuracy for RoBERTa for Sequence Classification

While we were happy with 30% accuracy for the model, our team also evaluated what we called "top 3 accuracy." In validation, we prompted our model to not only check the true flair against the model's first prediction but also the second and third.

We found that 54% of the time, our model would predict the correct flair as either the first, second, or third prediction. This was a necessary check for us because several flair categories had much in common, like medicine, neuroscience, biology, and health. In this way, we could imagine this being deployed as a model that helps users determine an appropriate flair for their post, giving them 3 or more suggestions.

```
******FINAL RESULTS********
28.666666666666668% correct on first choice
13.666666666666666% correct on second choice
12.0% correct on third choice
54.333333333333336% correct across top 3 choices
```

Figure 5: Checking if the RoBERTa model could guess the correct flair in its first 3 choices

Confusion within the RoBERTa Model

Although our accuracy with RoBERTa is admittedly low, it's important to give context to this lowness. As alluded to earlier, several flairs had much in common. For example, health, disease, medicine, neuroscience, and biology all fall under a similar umbrella. Since our dataset relies on laypeople to accurately flair their posts, it's possible that many posts were given incorrect or misleading flairs by users.

In figure 5, our confusion matrix is shown. Astronomy was one of the more readily predicted categories, perhaps due to its abundance of unique words that are literally "out of this world".

The health category was predicted for many titles, notably in medicine, biology, environment, and

psychology (and also health). These types of relationships and problems were found throughout our testing and they ultimately improved as we continued to create a more quality dataset.

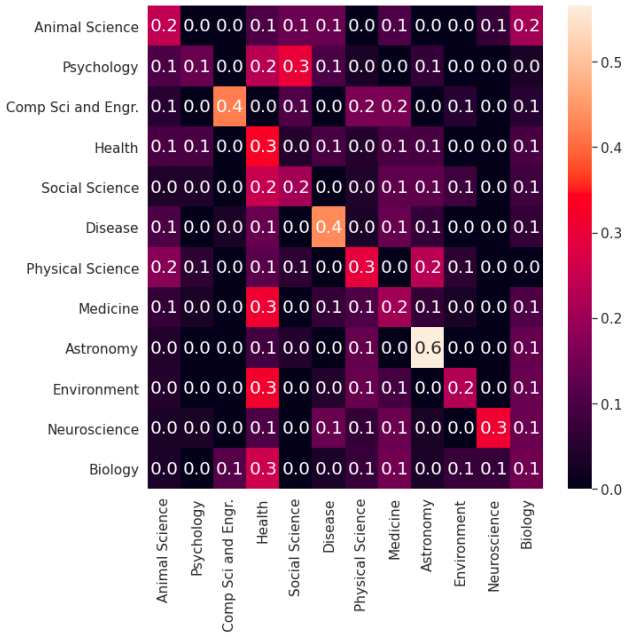

Figure 6: True flair of post title on vertical axis, predicted flair of post title on horizontal axis for RoBERTa model confusion

Multinomial Naïve Bayes Model Results

Our team found the Multinomial Naïve Bayes to be much more accurate in classifying posts. The first choice accuracy was around 56%. It was also much faster, training in a fraction of the time. We assumed this was the case due to simpler classification techniques as opposed to RoBERTa. The code was much simpler thanks to our use of Scikit Library for Python as well as Susan Li's implementation of the model.

It had traces of the same confusion as the RoBERTa model but ultimately was much more accurate at navigating the more intertwined flairs like biology, disease, medicine, and health. It seems that, in general, some categories are much more distinct from others. This can be seen between the confusion matrices of both classification approaches, wherein astronomy, computer science and engineering, and disease in particular do well.

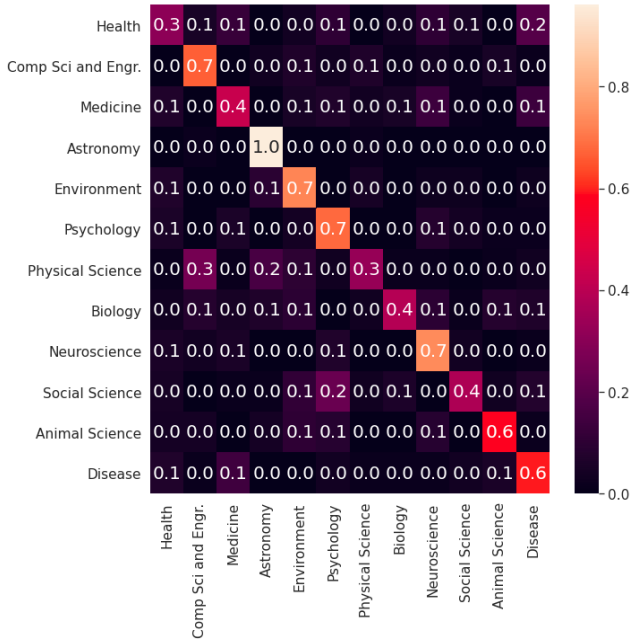

Figure 7: True flair of post title on vertical axis, predicted flair of post title on horizontal axis for classification using Multinomial Naïve Bayes

Conclusion

Our team managed to learn a lot over the course of this project. From curating datasets to running training epochs on a model, there was always something more to know and a resource or documentation to consult. For the two major components of our project, we have different conclusions that we came to.

Lessons About Data Curation

For datasets, our team learned the importance of curating our datasets. Many of our early efforts were thwarted by unknown accuracy issues, sometimes being just around the likelihood of a "shot-in-the-dark" guess.

We realized later than we would have liked that our data was rife with unmoderated posts that didn't meet the standard of Reddit's science community, much less our own dataset. In addition, repeats within the dataset poisoned our accuracy, particularly when the same posts had very similar titles, the same source, and very different flairs.

Lessons About Data Classification

Our team was incredibly optimistic about RoBERTa but it fell short of not only our expectations but classification using a much more naïve methodology. That being said, our major takeaway is that we ought to ask more questions and seek out more knowledge. We are confident that, with some adjusted parameters and better

data, RoBERTa could perform much better than we were able to get it.

Areas of improvement

If given the opportunity to improve upon what we learned, they are some areas of improvement we could investigate to get more accurate results. One way is to train on bigger datasets of reddit posts, so the model has more data to train to understand how to differentiate between different reddit posts. The next way would be to use many different text classification models like other BERT models or SVMs (support vector machines) and compare the accuracies between the models to find the model that is best for reddit post classification. The final way would be to experiment with different loss functions and optimizers to find the best ones to use for reddit post classification.

References

Github. *In the vocab of bart.bpe.bpe.decoder, what does* Ġ mean for those words prefixed with 'Ġ'?. 17 Feb. 2020, https://github.com/pytorch/fairseq/issues/1716

Huggingface. "RoBERTa." *RoBERTa - Transformers 4.0.0 Documentation*, huggingface.co/transformers/model_doc/roberta.html.

Huggingface. "Summary of the Tokenizers." *Summary of the Tokenizers - Transformers 4.0.0 Documentation*, Huggingface, huggingface.co/transformers/tokenizer_summary.html.

Li, Susan. "Multi-Class Text Classification Model Comparison and Selection." *Medium*, Towards Data Science, 6 Dec. 2018, towardsdatascience.com/multi-class-text-classification-model-comparison-and-selection-5eb066197568.

Reddit Competitive Analysis, Marketing Mix and Traffic. (n.d.). Retrieved September 18, 2020, from https://www.alexa.com/siteinfo/reddit.com

Scikit Learn. "Feature Extraction" *Scikit*, Scikit, scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html.

Scikit Learn. "Naive Bayes." *Scikit*, Scikit, scikit-learn.org/stable/modules/naive_bayes.html.

Senrich, R.; Haddow, B.; Birch, A.; 2016. Neural Machine Translation of Rare Words with Subword Units. arXiv preprint. arXiv:1508.07909v5 [cs.CL] 10 Jun 2016. School of Informatics, University of Edinburgh.

Silveira, Roberto. *Text Classification with RoBERTa*. 19 Aug. 2019, rsilveira79.github.io/fermenting_gradients/machine_learning/nlp/pytorch/text_classification_roberta/.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I.; 2017. Attention Is All You Need. arXiv preprint. arXiv:1706.03762 [cs.CL]. Cornell University.