Classification of Horror Stories from Reddit

By: Dean Zhou, Chris Kim, Sohit Gatiganti

1. Introduction

Modern forms of interpersonal communication are highly reliant on the internet for its accessibility and ease of use in sharing information, ranging from social media, online forums, and visual entertainment. Among text-based communication methods, web content rating and discussion forums are most popular. A remarkable feature of this model is the inherent self-governing platform for evaluating the quality of posts, answers, and comments. Within each online community, the discussion can be moderated through a metric system that determines the popularity, relevance, and overall quality of shared content; metrics range from YouTube likes/dislikes, Instagram hearts, to Stack Exchange upvote/downvote systems.

While this metric system allows individual communities to showcase information, opinions, and ideas, the cumulative attention towards massively popular, and often higher quality, posts result in ostracization of numerous posts that are created in a similar timeframe but are lost in obscurity. As a result, sharing content online towards a larger audience is commonly a difficult task. Thus, there is a need for an assistance tool to evaluate posts and representation of ideas to promote higher quality content creation. While quality of posts are inherently subjective to each individual and community with their respective values and interests, it is feasible to determine patterns for successful text-based posts that will be well-received in that community. Pattern recognition of written speech is often a difficult metric to determine, necessitating a machine learning approach.

2. Related Work

Lexicon of LeBron - As part of our initial search for viable datasets, we came across this humorous representation of text of a specific subreddit. This proved to us initially that our idea was feasible and that the data was available. We found 1.7 billion reddits comments @ 250 GB compressed, available to be downloaded on torrent. In the end, we did decide to go with a different extraction method, but this was a backup that was feasible with fast internet speeds.

Detection of Depression-Related Posts in Reddit Social Media Forum - This is perhaps the most famous reddit related paper and the one that inspired our methods the most. Their paper focused mainly on feature extraction using LIWC (a dictionary), LDC (Linear Discriminant Classifier) and differently sized N-grams. Although they proposed a myriad of models (Logistic Regression, SVM, Random Forest, Adaptive Boosting, and MLP), in the end they went with MLP only. We decided to take a novel approach by attempting to try various different models since we had significantly less text to featurize and because we aimed at stories. In a horror story, every line and word is important and keeps you on the edge, while in depression-detection, some words (death, job, break-up) have much more weight while most other words have no weight at all.

Popularity Prediction of Reddit Texts - Although slightly outdated (2016), this paper investigates prediction models within medium-sized subreddits, it was a strong indicator of previous results. Using TF-IDF, Naive Bayes, and SVM, they were able to achieve results ranging from 60-75% accuracy.

3. Dataset

The primary dataset used in this investigation consisted of a collection of posts from the subreddit community r/nosleep from the online platform Reddit. With over 430 million monthly active users, Reddit ranks as the 7th most popular social media platform in the United States and 13th most popular in the world [1]. With over 100,000 active communities, we have decided to limit our analysis to r/nosleep, a community focused on posting fictional horror stories. This subreddit is particularly suitable for its large daily traffic/post activity and community focus on designing impressive stories, suggesting a greater

feasibility of finding a definitive relationship between high and low quality posts. Furthermore, community guidelines and rules indicate a lower likelihood of encountering online trolling, or posts that would otherwise disturb our models.
Data collection was conducted using the PushshiftAPI as part of the psaw Python package. Data was collected under three different groups ranging from 10,000 posts (small), 100,000 posts (medium), and 200,000 posts (large) since January 1, 2015. Due to filtering and pre-processing, a significant portion of our datasets were removed to further specify our target objective (See Section 5.1). During cross-validation, each dataset was split into an 80% training group and 20% testing group.

4. Feature Selection and Pre-Processing Methods

4.1. Initial Filtering 1- Upvote Posts

An unusual feature of Reddit is that after the author publishes a post, the upvote mechanic automatically sets their default score of 1 rather than 0. Since the majority of posts are of lower-quality and often ignored, our investigation only registered posts with a score greater than 1 and removed invalid posts from the dataset. As a result, the dataset only contains story posts that have at minimum one satisfied reader. Thus, we can ensure that the difference measured in the dataset is between comprehensible and truly extraordinary stories. This initial filtering removed a large portion of our datasets.

4.2. Quantile Ranges

In determining the criterion for positive and negative labels, our first approach was to label scores that were approximate to the median as positive and label all other scores to be negative. Upon further analysis, this method produces ambiguity relative to scores that are closer to the median. In response, our revised approach labeled scores above the 3rd quartile as positive and scores below the 1st quartile as negative. As a result, our models do not consider the medial 50% of scores. This separation is justified by the significantly large standard deviation from mean, indicating that most story scores are either extremely high (popular post) or extremely low (obscure post). Furthermore, the mean of scores are severely skewed to the left, indicating that most story scores are extremely low, indicating that an average story is more likely to be obscured.

4.3. Text Pre-Processing

Natural Language Processing applications often require significant screening of input text before training. However, our dataset was unique in that the majority of story texts did not require significant editing, filtering, or parsing to be read in our model. Thus, the only modification made to our text data was to remove stop words.

4.4. Text Vectorization Methods

In order to provide numerical significance of our text data for our models, we implemented various vectorization techniques.

4.4.1. Bag of Words

The most common way of vectorising text would be through the Bag of Words model, which uses sparse vectors to easily represent the vocabulary and occurrence of specific words in a text. We can therefore use relative counts and distributions to make predictions.

4.4.2. TF-IDF

TF-IDF, or term frequency-inverse document frequency is a measure that is essentially a variation of a bag of words that takes into account the relative importance of a particular term in the corpus by using the frequency of word in the entire dataset, not just a particular example. It still generates sparse vectors.

4.4.3. Word Embeddings

Unlike the previously mentioned TF-IDF, and Bag of Words vectorisations, the Word Embedding method uses trained, dense word vectors to represent each example. This method essentially maps semantically close texts/words “close” to each other. We tried both training these word embeddings ourselves, and using pre-trained ones such as glov2vec.

4.4.4. Global Vectors

GloVe is an unsupervised learning algorithm developed by researchers at Stanford university that helps obtain vector repres
for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

We used the pretrained Wikipedia 2014 + Gigaword 5 pretrained vectors. We found overall that performance increased slightly with increased dimensions, at the cost of training time increasing linearly with the number of dimensions. We further hypothesize that performance would still increase a little if we used the larger pre-trained vectors such as Common Crawl or Twitter sets, but they come with increased download sizes (>1 GB) and increased training time.

4.5 Feature Selection

Apart from the text itself, we decided we would use other features in some of our more complex models. These were: text/length of title, time created, whether the post was for over 18s, and author information.

5. Supervised Learning Methods

5.1 Naive Bayes

Naive Bayes is a classifier that uses generated probabilities of words in text to come up with an estimate for an example belonging to a particular class. Since this was more of a baseline model, we filtered the data to only use the score and text of the story which was pre-processed as mentioned above. We used both the Bag of Words and TF-IDF vectorizer on numerous variations of NB (Gaussian, Multinomial and Complement NB). Complement NB was particularly promising as our data was imbalanced, and it often corrects for “misconceptions” that Gaussian/Multinomial have.

5.2 Random Forest

During our research, we found that Random Forest (RF) classifiers are promising for text classification. RF is essentially an algorithm that is robust to most types of data that generates a set of decision trees using random subsets of features. We used an RF classifier with the number of estimators set to 1000.

5.3 Logistic Regression & Stochastic Gradient Descent (SGD) Classifier

Logistic Regression is a MLE model that is often used to classify binary events. It learns the weights to use with the features, which are then input to the sigmoid function. The overall goal is to minimise the log-loss. Before we delved into the complexity of Neural Networks, we used a Logistic Regression/SGD classifier as a baseline for those models. We simply input either the sparse tf-idf vectors and labels without optimizing any of the pre-configured settings.

5.4 Sequential Neural Network

Neural Networks are a collection of interconnected nodes that can do an immense breadth of tasks. For this classification problem, we started with basic Sequential Neural Nets. After much experimentation, we settled on a Neural Net with two hidden layers, where the first hidden layer had 100 nodes and the second had 10. We used Binary Cross Entropy Loss with the Adam Optimised and ran for around 20 epochs with

a batch_size of 40. We ran the Neural Net on both the bag of words and word embeddings. We tried to train our own word embeddings and also use the pre-trained glov2vec vectors.

5.5 Multilayer Perceptron Neural Network

Multilayer Perceptron Neural Network (MLP-NN) is a classification model that is part of the feedforward artificial neural network (ANN) class. Likewise, a standard MLP consists of a minimum of three layers of nodes, including an input layer, hidden layer(s), and an output layer. Unlike a traditional linear perceptron, an MLP is able to distinguish data that is not linearly separable using its multiple layered framework and nonlinear activation functions. Aside from the input layer, every node uses a nonlinear activation function. In our investigation, we experimented with logistic, tan(h), and ReLU activator functions. The hidden layer is a hyperparameter of interest that can vary in both the number of internal layers and nodes per internal layer.

For this investigation, our MLP optimized for the following features: post text, post title text, length of post text, length of post title text, post flair (highlighted caption), length of post flair text, and time of day of creation in seconds. The post, title, and flair texts were converted into dense vectors using the word2vec package, while a logarithm was taken for each of their respective lengths. The input data for our perceptron was constructed using these text vectors and logarithmic lengths and was subsequently inserted into our 3-4 layered perceptron.

5.6. Recurrent Neural Network

Recurrent Neural Network (RNN) is a classification model that defines connections between nodes to create a directed graph in a temporal sequence, allowing time dependent behavior. This model is useful for writing and speech recognition applications. RNNs are based on feedforward neural networks and can utilize their internal node states to process different sequences and lengths of input variables.

Unfortunately, we ran into a memory bottleneck with our RNN model. For our most basic RNN, we had to use the lower-dimensioned 50-d vectors. We only had enough memory to feed the first 200 vectors of every post into a very generic LSTM. Naturally, we implemented batch training and were able to increase dimensions up to 300-d and the first 1000 words. However, we found that training time increased linearly with both of these factors, meaning that our iteration was extremely slow. It was simply infeasible to tune hyperparameters and test theories with this approach given our limited resources.

6. Optimization Methods

6.1 Hyperparameter Optimization

After initial training, we noticed that the MLP model produced the best results. As such, we proceeded to run a hyperparameter optimization on the MLP model using the GridSearchCV optimizer class, where we optimized the hidden layer/node sizes, activator functions, solver algorithms, learning rate, and learning rate type (constant vs. adaptive). The results of our hyperparameter optimization was a decrementing 3 layered neural network (400, 200, 100), ReLU activation function, a stochastic gradient-based optimizer proposed by Kingma, Diederik, and Jimmy Ba, 0.0001 learning rate, and constant learning rate type. This hyperparameter optimization produced noticeable improvements in overall test accuracy.

6.2 Regularization

With our neural nets in particular, we often had the problem of overfitting where we ended up with a training accuracy of 90% and a test accuracy of 58%. To offset this problem, we used an L2 regulariser on the weights in each layer. Furthermore, we also tried adding a Dropout of 0.1 to make the model more robust. This led to noticeable improvements in test accuracy.

7. Supervised Learning Results

7.1. Error Matrix Table

Error Matrix Values				
Model	Precision	Recall	F1-Score	Accuracy
MLP-NN	0.737	0.739	0.738	0.75
NN (Bag of Words)	0.62	0.55	0.582	0.58
NN (Bag of Words, Regularized)	0.64	0.61	0.624	0.62
NN (Word Embeddings)	0.64	0.67	0.66	0.64
NN (Word Embeddings, Regularized)	0.66	0.66	0.67	0.65
Log Reg	0.6	0.59	0.59	0.59
SGD	0.58	0.88	0.7	0.61
Random Forest	0.6	0.69	0.64	0.61
Naive Bayes (Bag of Words, Gaussian)	0.63	0.44	0.64	0.58
Naive Bayes (TF-IDF, Complement)	0.64	0.67	0.66	0.64

8. Discussion

Our primary objective in training the supervised learning algorithms was to optimize for the F-Score (F1-Score) to maximize the harmonic mean of both testing precision and recall. This is because the focus of our investigation revolved around accurately identifying both high-quality and low-quality text-based posts, which necessitated both high precision and recall scores. However, a closer analysis of our datasets revealed that a significant portion of stories were very unpopular - average scores were closer to the minimum value - indicating that a large majority of stories were below the threshold for high-quality content and that only a select few were above the threshold. For this reason, accurately predicting low-quality posts would be more significant for the nature of our dataset. Furthermore, we were surprised that our maximum accuracy for the problem was only 75%. We hypothesise this is because there is likely an element of luck in determining whether a post takes off, and reading through some of our labeled “low quality” posts, it is easy to see that many great stories simply don’t get any upvotes. Therefore, our initial assumption of upvotes being directly correlated to post quality may not have been entirely correct.

At the end, we found that our Multilayer Perceptron Neural Network was the most accurate by a large margin. We believe this is due to MLP’s inherent flexibility, simplicity and suitableness for complex

classification problems. However, we are not yet sure why our sequential networks with custom and pre-trained word embeddings performed so poorly in comparison.

9. Next Steps

We strongly believe that there is more to be found in this problem. The eventual goal is to to be able to create a website where any Reddit user can input the parameters of their upcoming post and determine whether it has a chance of becoming popular. If we are able to get an accuracy of around 90% for our problem, this would be a feature that many people would use. In order to get better results, we plan on trying to use the BERT transformer which seems to give improved results for problems like ours.

Contributions

Each member made significant contributions to the final project.

References

[1] Dean, Brian. “Reddit Usage and Growth Statistics: How Many People Use Reddit in 2021?” Backlinko. Backlinko LLC, February 25, 2021. https://backlinko.com/reddit-users#reddit-statistics.
[2] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation
[3] *LeXicon of LeBron*, tompiona.github.io/lex_lebron/index.html.
[4] M. M. Tadesse, H. Lin, B. Xu and L. Yang, "Detection of Depression-Related Posts in Reddit Social Media Forum," in IEEE Access, vol. 7, pp. 44883-44893, 2019, doi: 10.1109/ACCESS.2019.2909180.
[5] Rohlin, Tracy, "Popularity Prediction of Reddit Texts" (2016). Master's Theses. 4704.
DOI: https://doi.org/10.31979/etd.d7nw-6gx7
https://scholarworks.sjsu.edu/etd_theses/4704