Final Report - Reddit Comments Classification

Team Default: Yifan Bai (20153885), Maxime Daigle (20043325)

November 16, 2019

1 Introduction

Reddit is a forum with multiple subsections according to topics, namely subreddits. The task of this project is a to classify input short texts and output (labels) 20 pre-defined subreddits, such as "hocky", "nba", etc. Many supervised learning algorithms are available for this type of problem, generalized into two categories:

- **Classical algorithms**: using bag-of-words or **tf-idf** methods to get vector representations of raw input text, then use algorithms such as logistic regression on training data and get predictions on testing data
- **Deep Learning algorithms**: there are various deep learning algorithms used for this problem, including sequence models with LSTM/GRU, transformers, etc.

1.1 Naive Bayes

The first method used was Naive Bayes. With bag of words features, Laplace smoothing and the up-weighting of title words, it obtained an accuracy of 56.67% on the public leaderboard, meeting all baselines by Milestone I.

1.2 Logistic Regression with hand-crafted features

The second method was a classical method. It uses logistic regression with Naive Bayes' features. It was the the first attempt for second milestone, before deciding to focus on NN.

1.3 Single-layer neural network(NN)

The third method was a neural network. It had one hidden layer and used dropout.

1.4 RNN with LSTM-based sequence model

Another iteration on NN was a RNN, with LSTM based sequence model with pre-trained word embeddings. This method produced the best result, and its prediction was selected as the final submission on private leaderboard.

2 Feature Design

2.1 Pre-processing

Pre-processing is also known as text normalization in text classification, usually it includes:

- Removing stop words
- Handling special symbols and characters
- Case folding
- Spell correction
- Stemming and lemmatization
- Tokenization

The details of the pre-processing can be different depends on the algorithms which will be described more in details in the algorithms / methodology sections.

2.2 Feature Engineering

Feature engineering are vital to this type of problem. The performance of classical algorithms rely heavily on the quality of features. For text classification problem, there could be three different types of features:

- **Bag-of-words / Tfidf features**. Bag-of-words is simply the count of occurance of words in the text, and tfidf is simply apply the inverse-document-frequency on it to offset words with high occurance in corpus.
- **Word embedding features**. The disadvantage of it is sparsity. For instance, if the size of corpus is 100000 and the number of unique words is less than 100, the vector will have more than 99900 zeros, causing high spatial complexity on hardware. Word embeddings are also vector representations of text, but they are denser.
- **Text meta features**. This will required some experiences, such as length of the text, number of words in the text, number of capital letters in the text, number of punctutations (like "?", "!") in the text, etc.

For deep learning algorithms, hand craft features are less important, as the complicated model structure are suppose to be a "feature extractor" by itself. Instead, hyperparameters are the main differentiator in performance. Logistic regression and RNN are explained later in greater details. For this section, two case studies are shown:

- **Naive Bayes**

  The features are a bag of words. First, special characters and stop words are removed. We also tried to remove links and words made of one or two characters, but it slightly lowered the accuracy on the validation set. Next, we use lemmatization and then stemming on the words. The idea of using one after another is that, by using lemmatization first, we hope to obtain a better transformation than with just stemming and, because lemmatization doesn't find the root of every words, we still want to use stemming to process the words that weren't processed by lemmatization. The last pre-processing step is to up-weight title words [Ko et al., 2004]. The idea is that a text containing a subreddit name is more likely to come from that subreddit. So, a title word gets a bigger weight.

- **Single-layer NN**

  The feature used is the term-frequency times inverse document-frequency. Accents and stop words are removed. Words that occurred less than 2 times are ignored. We, also, used n-gram of size 1, 2 and 3. To reduce the number of features created by that pre-processing, the top 25,000 features with the best chi-squared statistic are selected.

3 Algorithms

3.1 Naive Bayes

Naive Bayes is a probabilistic classifier with the assumption that features are independents. The classifier is based on Bayes' theorem. By computing conditional probabilities, we select the most 'likely', i.e. the highest probability value.

3.2 Logistic Regression

Logistic regression is simply a linear classification algorithm / model. It generates linear decision boundary to classify different topics. The input feature used by logistic regression includes:

- **Tfidf** features on word level (without max features, i.e. use all features).
- **Tfidf** features on character level (with max features=500000, i.e. use top 500,000 frequent features).
- **Text meta features**, such as the length of the text, the number of words/punctutations in text, etc.

For hand-crafted feature, we drew inspiration from a Naive Bayes' approach which relies on log-count ratio, referencing *Baseline and Bigrams: Simple, Good Sentiment and Topic Classification* [Wang and Manning, 2012]. We created one-hot vector for the target, and train one model for each label. Each classification problem is defined as "whether this is topic X or not", then predicted topic will be the topic with highest probability out of 20.

3.3 Single-layer NN

The neural network used is a fully connected feedforward network trained by backpropagation. It contains one hidden layer. The activation functions are Relu and, for the last layer, Softmax. The loss function is cross-entropy. It uses dropout for regularization.

3.4 RNN with LSTM-based sequence model

LSTM based sequence model structure is popular in Kaggle text classifications.We drew inspirations from a Kaggle post about a previous competition [Burmistrov, 2017], which shares some similarities with this one. Since we switched late onto this method, the main foucus shifted to tuning hyperparameters. Furthermore, the following hyperparameters are identified to be tuned:

1. Number of layers and number of hidden units of those layers
2. Optimization algorithms
3. Training learning rate
4. Training batch size
5. Max features and sequence length (for text data)

4 Methodology

The training/validation split are same for both algorithms:5-fold cross validation. All training data are split into five equally sized sets, each time four folds sets (80% of data) are used to train the model, one fold (20%) for validation purpose. Each time training it, we also generate the predictions (probabilities for all 20 topics) for testing set, then average out the predictions across 5 folds as the final prediction, and the predicted topic is the topic with highest probability. The advantage of this approach is it has weak ensemble effect by providing five model predictions, which could prevent over-fitting and give more robust results. The disadvantage is it takes five times longer to train.

4.1 Naive bayes

The accuracy obtained by removing urls or by removing words with 2 character or less were tried, but both slightly decreased the accuracy. Also, the accuracy using stemming, lemmatization and both methods were compared. Both methods separately gave similar results, but using them together gave a slightly better accuracy. Laplace smoothing value and weight of title words were found with grid search.

4.2 Logistic Regression

The parameter used to tune is the penalty coefficient (regularization), experimenting in range from 0.5 to 100. However, it has little effect on the cross validation score. We imported *LogisticRegression* from sklearn.linear_models, and occasionally it gave warnings about *"non-convergence due to small max_iter"*. To resolve, it was increased to 10000.

4.3 Single-layer NN

First, grid search was used to find a good structure for the neural network. The structure found was one hidden layer with 32 neurons. With early stopping, the neural network was significantly overfitting. So, L1, L2 and dropout were tried. L1 and L2 were not successful, but dropout was sastisfactory.

4.4 RNN with LSTM-based sequence model

Deep learning models have much more hyperparameters, including model structure, optimization algorithms and strategies, etc. We cannot experiment every combination of hyperparameters given we have limited time and resource, therefore we selected those we think are critical to model performance, they are listed in the Algorithm section.

- **Model structure**: More complex models tend to be better at extracting more features from data and achieve better accuracy, but require more training data and longer training time, and more likely over-fit. Our model is constructed as follows:
  1. Attach bidirectional LSTM layer after embedding layer with width set to 128
  2. Attach bidirectional GRU layer after bidirectional LSTM layer
  3. Apply max pooling and average pooling layer, concatenate together with output state from GRU layer
  4. Attach fully connected layer with width set to 192

5. Output layer with width set to 20, use softmax activation function

- **Optimization algorithm and training settings**: Stochastic optimization algorithms are used to cope with large hyperparameter set. The algorithm used is **Adam** with learning rate of 0.001. Another important hyperparameter is **batch size**, number of data used to calculate the gradient for loss function each time. During training, it calculates the gradient based on one batch and then update, then use next batch till all data are used (one epoch). The model needs to be trained for multiple epochs. Batch size can be different as epoch increases, and by experiments satisfactory results were achieved using 5 epochs with batch size being 64, 64, 128, 256, 512.

5 Results

5.1 Naive Bayes

The most important hyperparameters for Naive Bayes were the Laplace value and the weight given to title words. As shown in Figure 1, adding smoothing α greatly improves accuracy. However, as the value is increased to a certain point (around 0.013), the accuracy slowly decreases.

The x axis is the number of additional times a title words is counted when occurred. For example, upweighting with the value 2 means that when a title word occurs in a text, we count it as if the word occurred 2 additional times in the text. This strategy significantly improves accuracy. However, as the weight increase, the accuracy slowly starts to decrease, as shown in Figure 2. Naive Bayes with Laplace smoothing and upweighting title words obtained 56.67% on public leaderboard.

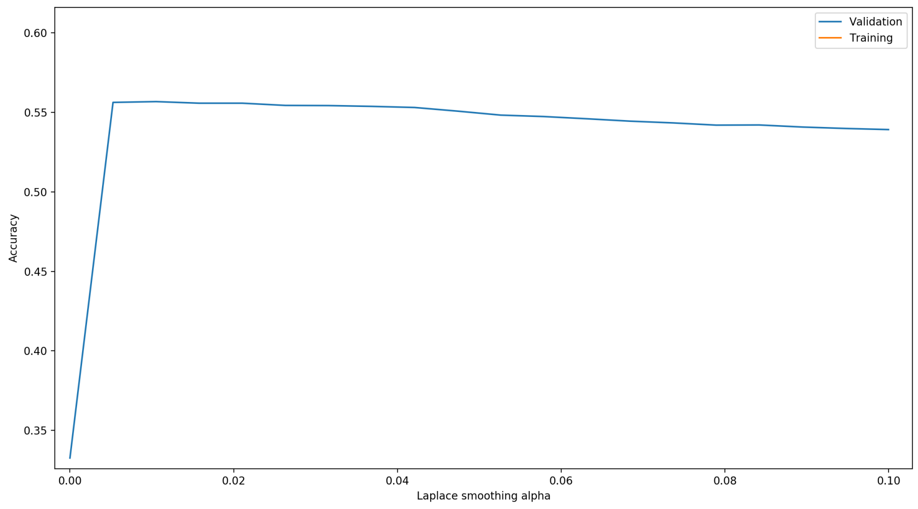

Figure 1: Accuracy in function of smoothing, Naive Bayes'

5.2 Logistic Regression

Initially we tried logistic regression only with **tfidf** and **text meta** features, the 5-folds cross validation accuracy is around 54%. This was a poor effort as it did not even beat Milestone I, forcing it to be abandoned. Next we tried logistic regression with Naive Bayes features. With the settings mentioned earlier, the 5-folds cross validation accuracy is around 56.2034%, and the public leaderboard score is 57.466%.

5.3 Single-layer NN

The most important hyperparameter for Neural network were the number of neurons in the hidden layer. Referring to Figure 3, if number of neuron is too low, the neural network is unable to learn as much as wanted. The accuracy grows until it reaches 32 neurons and plateaued thereafter. The neural network obtained 56.93% on public leaderboard.

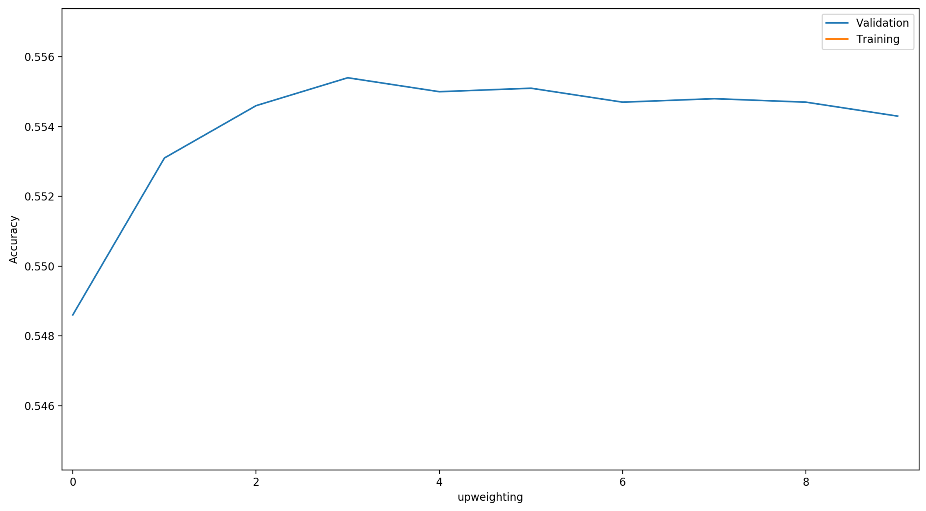

Figure 2: Accuracy in function of weight given to title words, Naive Bayes'

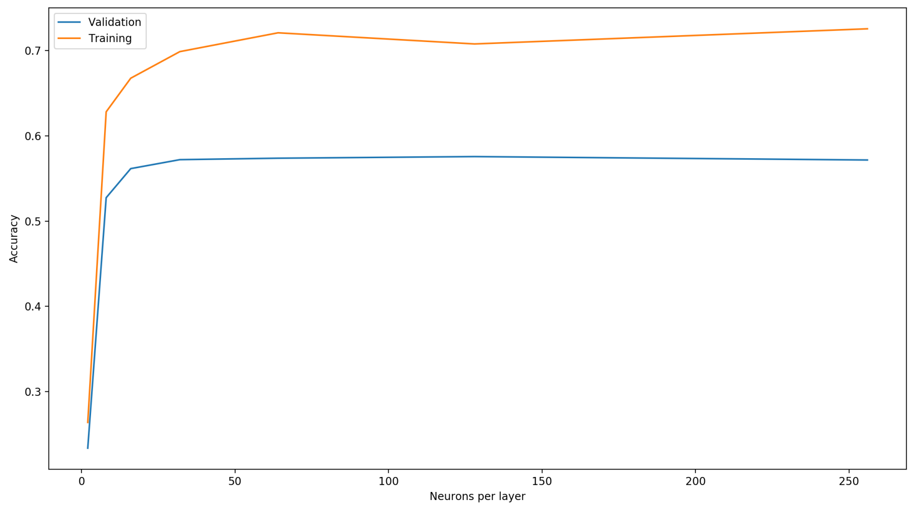

Figure 3: Accuracy in function of the number of neurons in the hidden layer, Single-layer NN

5.4 RNN with LSTM-based sequence model

This is the model used for final submission. We started with embedding size of 300 (averaging two pretrained word embeddings), input sequence length of 90, drop out rate of 0.2 (between embedding layer and LSTM layer), LSTM and GRU width of 40, training for 4 epochs with batch size 128, 256, 512, 512. The 5-folds cross validation accuracy is 56.9142%, public leaderboard score is 59.366%.

The next iteraion features an additional dense layer with width of 96 before output layer, increase LSTM and GRU width to 128, increase sequence length to 120 (more than 90% inputs are less than it) and training one more epoch with batch size 512. 5-folds cross validation accuracy is 57.3%, and public leaderboard score is 59.533%.

After more experiements, we finalized the model by increasing the dense layer width to 192, drop out rate to 0.5, embedding size to 600, sequence length to 150 and concatenating two pretrained word embeddings instead of averaging them. 5-folds cross validation accuracy is 59.43%, and public leaderboard score is 60.555%. This gave the confidence of getting 59% on the private leaderboard, required for achieving full performance marks.

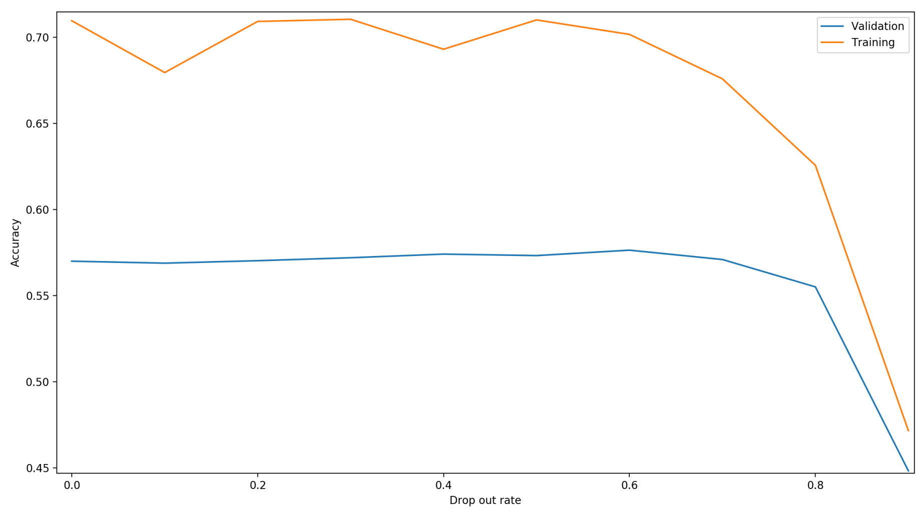

Figure 4: Accuracy Dropout, Single-layer NN

6 Discussion

Since Milestone I requires using the same algorithm, the discussion focuses on the methods for Milestone II.

6.1 Pros and Cons

For logistic regression, linear model can be trained in shorter time and is easier to interpret. However, it relies heavily on feature engineering, and **tfidf** could take very long time to run. In addition, it does not perform as good as ensemble model or deep learning model. RNN could achieve higher accuracy and requires less feature engineering. However, it has many hyperparameters which increases tuning effort. It could also easily over-fit, which makes interpreting the model more difficult.

6.2 Suggestions for Improvement

On the one hand, the pre-processing used, mainly text normalization, are relatively simple, ideally we can conduct more the feature analysis to improve pre-processing and feature engineering. In addition, handling out-of-vocabulary words are also one direction to improve. On the other hand, if time had permitted, more hyperparameter tuning could be done, as well as tweaking hyperparameter set. For the same reason, we had to adopt a simple-to-build model and gave more time for hyperparameter tuning, which leaves room for improvement.

7 Statement of Contributions

Both team members contributed equally for each component of the competition (e.g. defining the problem, developing the methodology, coding the solution, performing the data analysis, writing the report, etc.). We hereby state that all the work presented in this report is that of the authors.

References

Alexander Burmistrov. About my 0.9872 single model, 2017. URL `https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/discussion/52644#latest-662601`.

Youngjoong Ko, Jinwoo Park, and Jungyun Seo. Improving text categorization using the importance of sentences. *Inf. Process. Manage.*, 40(1):65–79, January 2004. ISSN 0306-4573. doi: 10.1016/S0306-4573(02)00056-0. URL `https://doi.org/10.1016/S0306-4573(02)00056-0`.

Sida Wang and Christopher Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 90–94, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/P12-2018`.