Insincere Question Classification

Srashti Agrawal, Billy Ermlick, Nick Newman

Abstract

This paper discusses and applies various machine learning approaches for detecting illegitimate questions in online question answering (QA) forums. A dataset obtained from *Kaggle* regarding questions posted from users on *Quora* is used for both testing and evaluation. The approach focuses on the implementation of combinations of neural networks built upon a model averaging framework to avoid overfitting. The best performing model is a statistical ensemble classifier combining statistiical features with a LSTM/CNN network. Results proved comparable to other top performing teams in the competition. Improvements are possible in refining the tuning parameters of current architecture as well as exploring cyclical learning rates, other statistical features, and combinations of statistical features with attention based networks.

keywords: text classification, statistical ensemble classifier, attentional neural network, model averaging.

1) Introduction

The issue of "*trolling"* is a hot topic on the modern internet scene. As perhaps best illustrated through select posts found on *Yahoo! Answers,* internet trolls act by posting divisive content on websites in order to provoke controversy and emotional reactions from readers[1][2]. While less scrupulous sites may not mind the added traffic trolling provides, question and answer (QA) forums that focus on providing credible information must take steps to moderate these types of posts[3].

This project will focus on detecting insincere questions posted by members on the *Quora* QA community forums. Reflecting on the present challenges in this area, *Quora* sponsored a *Kaggle* competition in 2017 in aim of improving their current machine learning systems[4].

The challenge in the task revolves around detecting questions which are intended to make a statement rather than looking for answers. Identifying if a question is legitimate is fairly intuitive by hand. Insincere questions typically have a clear agenda (usually political), have a non-neutral tone, and disparage or convey an extreme view of a group. More subtle questions also exist that are based on extreme or fake assumptions, such as "Did Julius Caesar bring a tyrannosaurus rex on his campaigns to frighten the Celts into submission?".

Training machine learning models to detect such features is not a straightforward endeavor. While topics of conversation may provide a clue for a question's intentions, features for clearly dividing "troll" type questions from those seeking real answers are not obvious and depend on the context of the sentence in its entirety.

2) Related Work

Since explicit features for dividing the classes are difficult to determine, the best performing teams on the task implement different types of deep learning models. Several of the top scoring groups have, or recommend, some form of attention based neural network, specifically an attention weighted average layer [5][6]. This may be successful due to the complexity of the task since attention mechanisms allow greater functional complexity of the model[7]. Successful groups also do not utilize a single embedding technique, but instead combine multiple embedding techniques together such as GloVe and Paragram into a single embedding matrix[5][8]. This use of multiple embeddings is most likely helpful as it encodes additional, different information about each word specific to the corpus the word embeddings were created from. While these approaches perform well, based on the F1 scores there is still room for improvement by fine tuning their models.

One area in which these groups fall short, and where we can improve upon, is error analysis. The high performing groups have drawn attention to the errors committed by the

classifiers, but have not addressed the correction of them at the embedding level. For example, some of the embedding models rely heavily on the presence of profanity to determine whether a question is insincere[8]. This causes the model to overlook insincere questions that are eloquently worded and therefore resemble the word structure of a sincere question. Addressing this at the embedding level by attempting different preprocessing methods to reduce the overfitting on profanity-ridden questions is a good start for our group to improve the results.

3) Methods

Tokenization

Pre-processing of the *Quora* question text consisted of: contraction conversion, punctuation removal, converting all numbers to a number sign, lowercasing all words except for those that contain all caps, and correcting the spelling errors that are typical in internet forums. Cascading stemming was used with a combination of Porter, Lancaster, and Snowball stemmers. TF-IDF was used for the baseline model while word embeddings were used for all the neural methods.

Embedding

The competition had the option of using three different pre-trained embeddings: GloVe, Paragram, and Word2Vec. Several combination methods were tested using these embeddings, such as using Principal Component Analysis to project several embeddings into a reasonable number of dimensions, using a weighted summation or average of different embeddings, and simply using non-pretrained embeddings. The mix that worked the best was simply concatenating the GloVe and the Paragram embeddings. This, in turn, created a V x 600 dimensional matrix, where V is the vocabulary size. A scaled random vector was created for both Paragram and GloVe to address words that did not show up in the pretrained embedding.

Model Averaging

Determining how to set up training and validation was a pivotal point of the project process. Originally, we decided to use an 80-20 split of the training data to create a validation set. With such a vast amount of data to choose from, it was not believed that cross validation was necessary. As we began submitting models for scoring on the test set, the scores were far from the range that the models were achieving on the validation set. This prompted some reevaluation of the process, and ultimately led to the model averaging method used for the duration of the process.

This method involved dividing our training data into five folds, training on each fold, and using each fold to make predictions on the test set. This resulted in five sets of predictions on the test set which were averaged together to form our final predictions. A threshold value (usually 0.5) was then applied to determine the final class for each question. This technique helped prevent overfitting issues that plagued the original validation set method. A graphic of this approach can be found in Figure 1 below.

All submissions to *Kaggle* were evaluated on the holdout test set using a macro averaged F1-score being the evaluation metric. The results are public and we were able to compare our scores to others in the competition throughout the project.

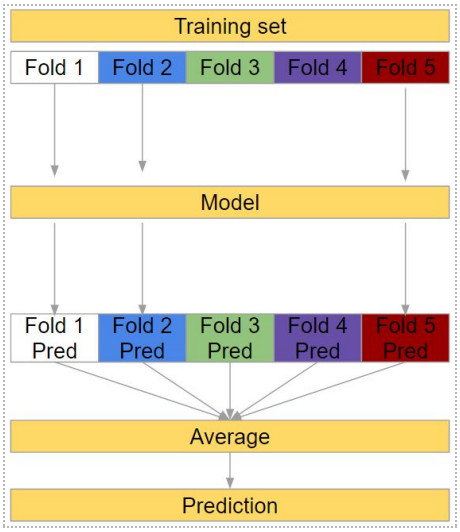

Figure 1: *Model Averaging*

Dataset & Competition

The dataset provided on *Kaggle* contains 1.3 million train and 375 thousand test questions [9]. The training set has a large class imbalance with only around 80,000 (~%6) of the questions belonging to the insincere class as shown in Figure 2. *Quora* provided the dataset to *Kaggle* with class labels double annotated by their moderators.

The *Kaggle* competition concluded in February of 2019. Submissions were required to be run on *Kaggle Kernels* which provided 30 GPU hours per week for competitors to attempt submission. The use of *Kaggle Kernels* prevented those in the competition from utilizing outside data sources and limited computation time for a single submission to two hours.

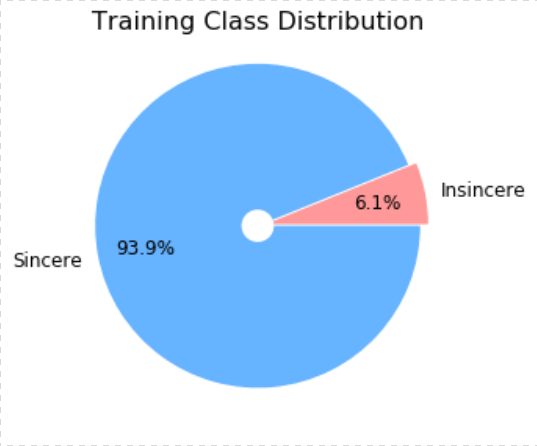

Figure 2: *Imbalanced Training Class Distribution*

4) Baseline

A Logistic Regression (LR) model was used as a baseline. TF-IDF feature vectors were used for encoding with 1-3 ngrams. This exploded the dimensionality of the model and LSA was used to project the dimensions down to 100 components due to memory constraints. The F1-score for this model on the test data was 0.59080.

Since the model uses TF-IDF, it isn't able to quantify the relation between words and the context relations. For example, one misclassified question "What is our current immigration policy concerning workers that want to come to the USA?" was wrongly predicted as insincere likely due to the words "immigration", "USA", and "policy" being present, as those words are commonly found in insincere questions. This provided a good focus area for our neural methods to improve upon.

5) Neural Network Solutions

With a solid baseline established, the next step was to generate models to increase the F1-score. We tried a variety of different neural methods, but ultimately there are 5 models we implemented that highlight our progression throughout this process.

All of these methods take the previously mentioned design of the combined GloVe and Paragram embeddings as input, and output a probability of each question belonging to the insincere class via the sigmoid function. Three epochs were used for all these models and we kept our classification threshold constant at 0.5 for each model. The learning rate was 0.001 for each model, and we used a binary cross entropy loss function with the Adam optimization algorithm. The batch size that struck the best balance between efficiency and model-fit time was 512.

Feedforward Network

The first neural method used was a standard feedforward neural network. We used 256 units in the hidden layer and a rectified linear unit after the linear layer to add some nonlinearity to the model. The *Kaggle* test set F1-score of this model was 0.59260.

Bi-LSTM Network

After the feedforward model, we progressed to a standard bidirectional long-short term memory model. This is a recurrent method that places an emphasis on important past words that are otherwise irrelevant in standard recurrent methods, due to the vanishing gradient. The model is also able to take in text from both directions and therefore gather more contextual information, rather than merely

advancing in a forward fashion. The number of layers for each direction was kept at one, as this did not improve the performance. Max and average pooling was added to the output, followed by a dropout layer, to reduce overfitting. This model saw a sizeable increase in F1-score, bringing it up to 0.64518.

Once this model was completed, we had to research other methods to solve this problem, since we had only been exposed to basic recurrent models at this point which we had maxed out performance wise. This involved reading through various inspirations in the Kaggle competition discussion board and finding research papers with successful methods.

LSTM-CNN Network

The first such model we decided to implement was a bidirectional LSTM network where the output was fed into a convolutional layer. A max pooling layer was used on this, followed by a fully-connected layer where the predictions were stored. The logic in using this method was to increase the context window of the LSTM. Convolutional layers take kernels as an input, and we believed that by using this it would allow the network to better understand the important parts of the questions related to classification, while at the same time reducing the dimensionality by max-pooling. This method achieved an F1-Score of 0.65708.

Weighted Attention Network

When the results from the Bi-LSTM/CNN network were not as high as expected, we decided to focus on other ways to capture meaning from the *Quora* questions. One method that has shown to achieve success is a hierarchical attention network[10]. The structure of this network has three separate sections. The first takes the input and places it in a single layer multi-layered perceptron. The second section involves taking that output and feeding it through a softmax function. This gives an output of the normalized word importance for the sentence. The third and final section consists of computing the sentence vector as a weighted sum of the word importance annotations. This is done for all the sentences in the data to yield a matrix containing these weighted sums as output. This attention network was used with both LSTM and GRU model outputs and the results were combined. Figure 3 provides an image of the best performing model architecture using the model averaging method discussed earlier. An F1-score of 0.67992 was achieved on the final *Kaggle* test set.

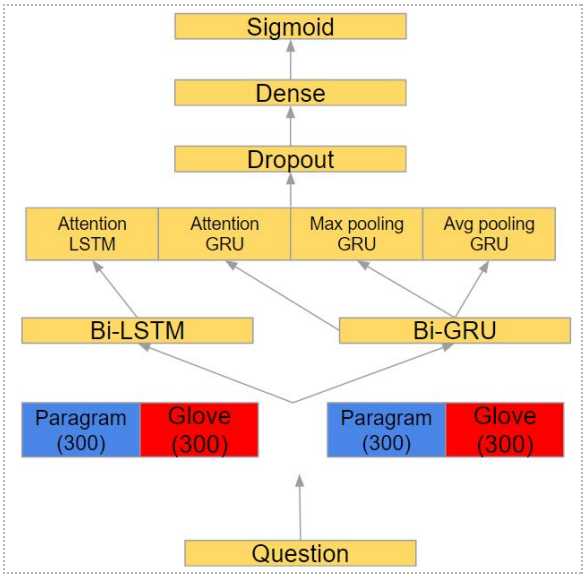

Figure 3: *Attentional Network*

Statistical Ensemble Classifier

Seeing how were were still behind others in the competition we attempted to build a model that combines successful features from the LSTM-CNN model and simple statistical features from each sentence into a single classifier. It was reasoned that the LSTM-CNN model could only be improved by adding in other general features related to each question, and others in the competition suggested similar ideas[12,13].

To create these features we brainstormed various metrics generic to each question, created feature vectors for each question, and inputted these feature vectors into a LR model to determine which of the statistical metrics were most predictive. Measures considered included the number of words in each question, number of characters, number of unique words, number of capital letters, ratio of unique words to total

words, number of nouns, number of verbs, and ratio of nouns to verbs (smoothing applied). A successful feature included counting the number of profane words in each question by comparing them to a list of profane words provided by sites like Google and other sources [11]. Additionally, the BlobText package was investigated which provides subjectivity and sentiment values for english words. Average sentiment of each sentence as well as average sentiment times subjectivity for sentence were considered, but ultimately not applied in the final model due to time constraints on the *Kaggle Kernel*. The final selected statistical features were chosen for their predictive power based on the LR model as well as their computational simplicity due to time constraints.

Once the final statistical features were chosen they were concatenated to the output of the Bi-LSTM/CNN model as shown in Figure 4 below. The results of this ensemble were the most promising, achieving an F1-score of 0.69789.

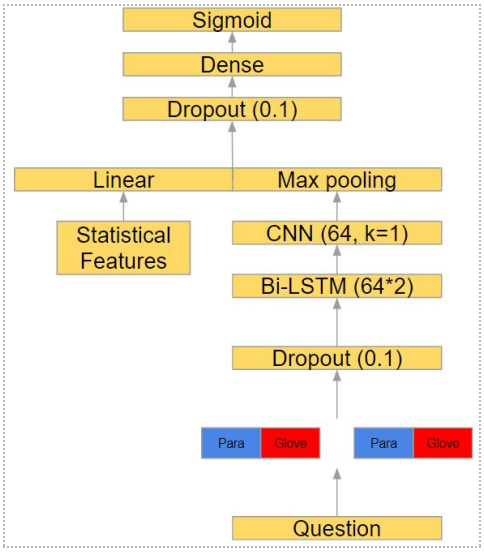

Figure 4: *Statistical Ensemble Model*

6) Discussion

Importance of Model Averaging

The most critical component for success in this competition was being able to make models generalize well to the test data. As an experiment, we tried using 25% of the training set for training, 25% for validation, and 50% for testing, and achieved far better results than we did on the true test data. This shows that there is something fundamentally different about the test data, either in the class distributions, or in the questions themselves.

Since the testing data class labels are stored on the *Kaggle* servers, the distribution of testing class labels is not known. Sometimes, upon completion of the competition, the test labels are released, but this has not occurred for the Insincere Questions Classification Competition.

The cross validation model averaging method we used was highly effective in helping our models to generalize to unseen data. Using final predictions which contained information from 5 slightly different models and training sets was key to avoiding overfitting, despite having such a large training dataset.

Error Analysis

Analyzing errors on the test data is impossible as the true class labels are not disclosed as discussed above. Throughout the training process we used hold-out sets and k-fold cross validation to judge the various model performances. These validation sets provided insight into the types of errors our models were making.

Some of the errors our models exhibited appear to be entirely subjective and even humans would have a hard time labeling. An example of this can be seen in “Why are women the weaker sex?”. At first glance it appears the question is insincere, as it is based on a bias assumption against women being somehow weaker than men. However, this example is unintuitively labeled “sincere” in the training data. This may be because the asker is genuinely interested in why they have heard this saying, however it seems difficult to know for sure simply from the question along without more context that may have been afforded to the annotators.

Given these subjective labels, the limits of the gold standard labels need to be considered

when interpreting model performance. A perfect F1-score on this dataset may not be possible without the full context the annotators were most likely supplied.

Aside from the subjectivity of the questions, the clearest error our best model (the statistical ensemble model) showed was a tendency to incorrectly classify “sincere” questions as “insincere” when they possessed a profane word. This is directly the result of the profane word count feature which tended to dominate some of the question classifications. .

Final Results

A table of our final results for each model on the test set is provided in Table 1 below. While we were successful in dramatically improving upon our baseline solution to the problem, the current model falls short of the top performer in the competition. From what they have published on *Kaggle*, the structure of their model is not dissimilar to the statistical ensemble model described in this paper[13]. Aside from having a different embedding combination strategy and different statistical features, the improvements in their performance is most likely to superior tuning. Other aspects on which we can improve our present model is discussed in the future improvements section below

Table 1: *Final Results*

Type	Classifier	F1 (%)
Gold Standard	Stat Ensemble	71.323
Our Model	Stat Ensemble	69.789
Our Model	Attentional	67.992
Our Model	RNN-CNN	65.708
Our Model	Bi-LSTM	64.518
Our Model	FeedForward	59.260
Baseline	Logistic Reg.	59.080

7) Conclusions

Ineffective approaches

As with any data science project there are a variety of reasonable things attempted which do not yield good results. As mentioned, trusting our validation set score was not effective in estimating our final score on the test set. We also tried weighting our loss functions based on the class proportions in the training set. This also did not improve our performance, likely due to the same issues mentioned earlier regarding differences in the test set. TextBlob sentiment and subjectivity were also added as features to the embedding matrix for each word but this did not result in performance improvements and was removed due to timing constraints. Correcting spelling using an edit distance approach based off of words in the given word2vec embeddings met a similar fate.

Future Improvements

All models would be improved with more tuning. We contemplated using a more methodical tuning approach such a parameterized search approach like GridSearch or a more formalized Baysian optimization approach using packages like Hyperopt, but did not have time to implement them. A critical tuning parameter that we did not have time to vary is the threshold value of the Sigmoid function for the final prediction. Charging this value as well as changing the way the model predictions were combined (e.g., combining class predictions instead of class prediction probabilities) would further improve our performance.

Implementing a cyclical learning rate would further improve our performance. Our current approach relies on just a few epochs per model fit and implementing a cyclical learning rate would help this low epoch approach end at an improved performance level.

Based on our error analysis the refinement of the profane word count feature, either through scaling, elimination or some other variant would improve our model performance. The addition of other statistical feature may also aid our performance. Ensembling the statistical features with the attentional model may be another avenue of improvement for work in the future.

References

1) https://kb.iu.edu/d/afhc
2) https://slate.com/technology/2007/12/why-yahoo-answers-is-a-librarian-s-worst-nightmare.html
3) https://medium.com/product-vs-product/quora-v

s-yahoo-answers-cb0845790d33
4) https://www.kaggle.com/c/quora-insincere-questions-classification/data
5) https://www.kaggle.com/wowfattie/3rd-place/comments
6) https://users.soe.ucsc.edu/~slgabbar/resources/quora_kaggle_report.pdf
7) http://akosiorek.github.io/ml/2017/10/14/visual-attention.html
8) http://web.stanford.edu/class/cs224n/reports/custom/15763730.pdf
9) https://www.kaggle.com/c/quora-insincere-questions-classification
10) https://www.cs.cmu.edu/~./hovy/papers/16HLT-hierarchical-attention-networks.pdf
11) https://www.freewebheaders.com/full-list-of-bad-words-banned-by-google/
12) https://www.kaggle.com/c/quora-insincere-questions-classification/discussion/79824
13) https://www.kaggle.com/c/quora-insincere-questions-classification/discussion/80568