Detection of Question Sincerity on Online Forums

Katherine Bosshart
Graduate Student, Georgetown University
Email: keb289@georgetown.edu

Siyao Peng
Graduate Student, Georgetown University
Email: sp1288@georgetown.edu

Research was performed to determine whether sincerity of questions posted on online question and answer (Q&A) forums could be systematically and accurately identified. Utilizing a training set of questions posted on Quora and labeled as sincere or insincere, the data was augmented and cleaned. From there, three neural networks were developed and trained, and tested on a held-out portion of the dataset. Based on sensitivity, precision, and F1 score, it was determined that an Attention Model was able to successfully flag insincere questions in the training data.

1. Introduction

1.1 Overview and Problem Statement

Researchers sought to develop a method to correctly and systematically identify insincere questions posted on online Q&A forums, specifically focusing on the website Quora. For the purposes of this research, insincerity was defined to include questions founded on false premises, including those containing inappropriate content or are intended to make a statement, rather than gather meaningful answers - these types of illegitimate questions pose a challenge to Q&A forums as they can be divisive or offensive to users. Researchers sought to tackle this challenge by leveraging Quora questions to create a scalable solution to the problem of insincere questions online.

1.2 Related Work

Similar research on the identification of insincere questions on online forums has been performed as part of a Kaggle competition titled “Quora Insincere Questions Classification: Detect toxic content to improve online conversations” [4]. Proprietary models also exist across Q&A sites for internal screening and site maintenance purposes. “To date, Quora has employed both machine learning and manual review to address this problem” [4].

2. Datasets

2.1 Data Source

Data for this research was gathered from from the Quora Insincere Questions Classification Kaggle competition [4]. This dataset contains roughly 1.3 million questions sourced from Quora, each containing a Question Identification number, and labeled as either sincere or insincere.

2.2 Exploratory Data Analysis (EDA)

Initial exploration of the dataset was performed using summary statistics and visualizations. High-level exploration showed the training dataset contained approximately six percent insincere questions, suggesting an uneven class distribution. As researchers drilled down, it was determined some labels do not agree with intuition - for example, the question "Have you licked the skin of a corpse?" was labeled sincere. Kaggle warned, “The ground-truth labels contain some amount of noise: they are not guaranteed to be perfect” [4]. However, these findings suggest Quora favors labeling questions sincere, potentially to avoid over-censoring.

Additional EDA was performed to investigate the differences between the classes. Box plots in Figure 2.2.1 show key observations including: Insincere questions tend to be longer in word count, and have more punctuation, and more uppercased letters than sincere questions. Sincere questions frequently included the words 'best', 'good', and 'would' whereas 'Trump', 'women', and 'white' were frequent in insincere questions. This suggests sincere questions tend

to seek recommendations, whereas insincere questions may be racially or politically charged.

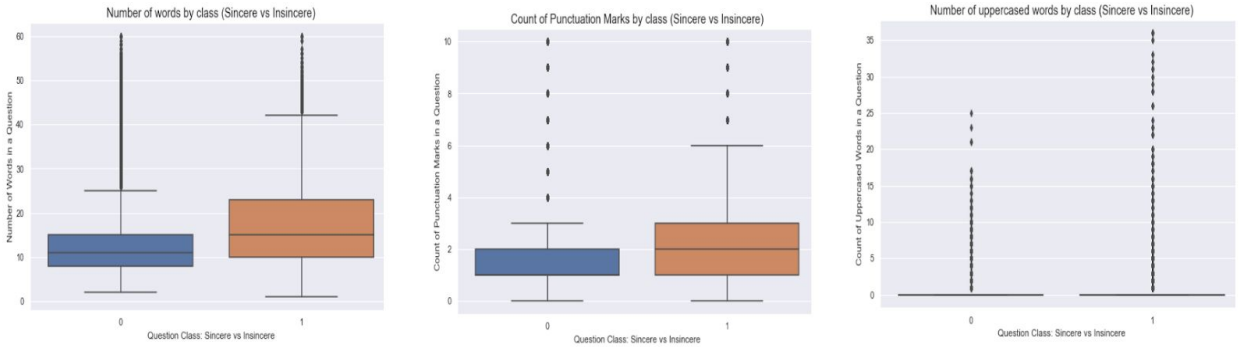

Figure 2.2 - Exploratory Data Analysis: Investigation of Class Differences

Based on EDA, it was determined that simple features of the sentences were not enough to separate the classes - more complex models would be required.

2.3 Data Preprocessing

2.3.1 Data Augmentation

EDA showed there were considerably more questions labeled sincere than insincere, suggesting data augmentation is required to correct this imbalance. “An increasingly popular and critical technique in modern machine learning is data augmentation, the strategy of artificially augmenting existing labeled training datasets by creating transformed copies of the data points.” [5]. Here, data augmentation was performed on the insincere questions by tokenizing into sentences, shuffling, and rejoining to generate new texts. Synonym replacement was also performed by randomly choosing non-stop words from insincere question text and replacing with similar words. Related research [2, 5, 6] shows that this is an effective method: Zhang et al. had success utilizing synonym replacement to perform text augmentation in their paper entitled “Character-level Convolutional Networks for Text Classification” [2, 6]. The Snorkel package in Python offers an automated method to do this, “implement[ing] a simple text data augmentation strategy — randomly replacing a word with a synonym… express[ed] as a *transformation function*” [5].

After augmentation, insincere questions accounted for 15% of the total data. To further improve the balancing of classes, undersampling was used to remove a portion of the sincere questions. The training dataset was reduced to roughly 735,000 questions with the random removal of 40,0000 sincere questions. The resulting cleaned dataset was more balanced with insincere questions accounting for 23% of the data.

2.3.2 Word Embeddings

Additional data preprocessing was performed using the Global Vectors for Word Representation (GloVe) word embedding to map question text to numerical counterparts for input into models. Previous research showed GloVe had the most success, and typically performs well on similar problems [3]. Here, word embeddings were identified for 37% of the unique words within the training dataset, and 87% of all text. Investigation of words without embeddings showed they fell within one of four categories:

1. Punctuation - e.g. 'why?', 'it?'
2. Contractions - e.g. "aren't", "let's", "she'll":
3. Special Characters - e.g. θ, √
4. Non-American English Words - e.g. ‘centre’, ‘favourite’

After mapping to standard words, word embedding coverage was enhanced to 82% of unique words and 99% of all question text, suggesting clean data ready for use in modeling.

3. Methodology

To identify the best model for detecting insincere questions, three models were developed including an Long-Short Term Memory (LSTM) model, Word-Level 1D Convolutional Neural Network (CNN) model, and a Bidirectional LSTM and Attention model. Each was trained with a batch size of 256 on the 80% of word

embeddings data, holding out the remaining 20% of the data for validation. Such a large batch size was determined to be appropriate given the size of the dataset to train the models faster, although this will make the models converge slower.

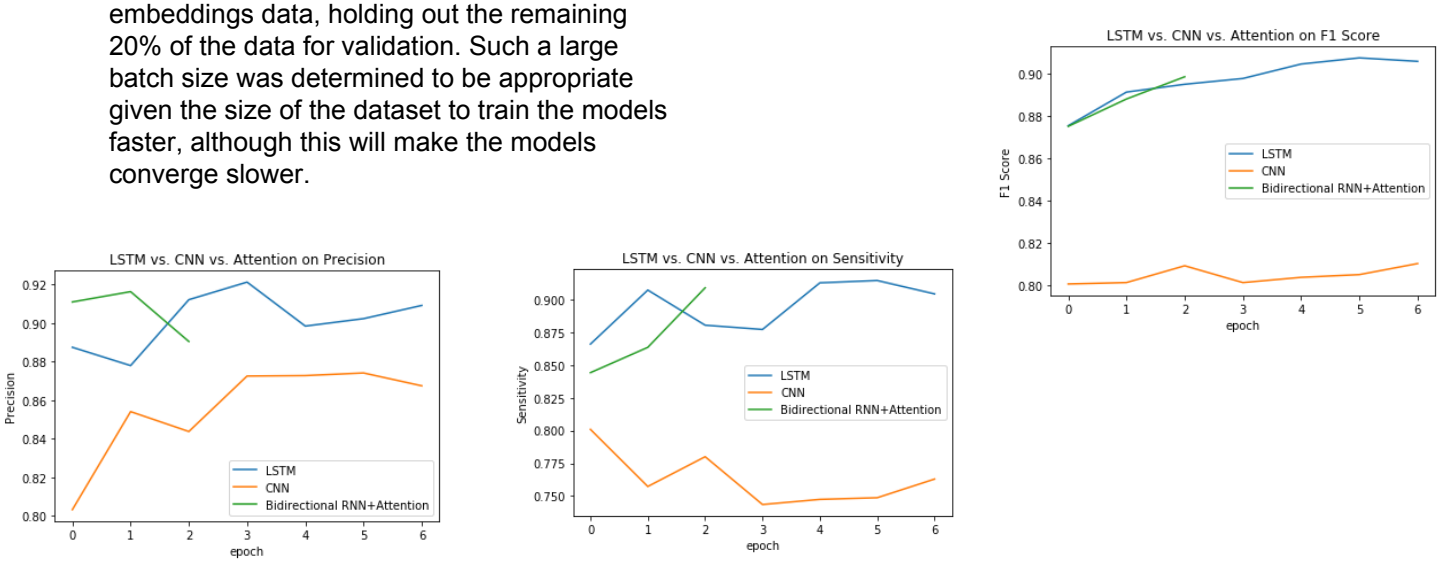

Figure 4.1 - Performance Evaluation: Analysis of Model Results

3.1 LSTM Model

An LSTM model was developed utilizing one input layer, one embedding layer, one LSTM layer with 128 neurons and one output layer with one neuron because the model required to output two labels. This provided 27,230,017 parameters *(see Appendix for model details)*. The LSTM Model was trained on the word embeddings for seven epochs with a batch size of 256.

3.2 Word-Level 1D CNN Model

A Word-Level 1D CNN model was developed with the following infrastructure: Input - > Embedding -> conv1D -> max_pooling1D -> conv1D -> max_pooling1D -> conv1D -> max_pooling1D -> flatten -> dense -> output. This model provided 27,049,889 parameters *(see Appendix for model details)*. Just as with the LSTM Model, the CNN Model was trained on the word embeddings for seven epochs with a batch size of 256.

3.3 Bidirectional LSTM and Attention Model

A Bidirectional LSTM and Attention model was developed with the following structure: Input -> embedding - > bidirectional LSTM -> self attention -> flatten -> dense -> output. This provided 27,256,594 parameters *(see Appendix for model details)*. Due to resource limitations and computational ability, the Attention model was run with a batch size of 256 for only three epochs, while the other models were trained for seven.

4. Results

4.1 Performance Evaluation

Performance of the three models was evaluated using the held-out validation dataset (20% of the total cleaned and augmented labelled dataset) by comparing the model predictions against the sincerity labels included within the testing dataset. To meet the goal of discovering insincere questions, knowing that the training dataset is imbalanced, it was determined that precision, sensitivity, and F1 score were the most critical metrics: “F1 is best if you have an uneven class distribution,” as with the labeled question data used here [1]. Plots of all three of these metrics are shown in Figure 4.1 above.

5. Discussion of Results

Figure 4.1 illustrates the Attention model performs the best on classification with higher sensitivity but lower precision than LSTM. This shows Attention classifies more sincere questions as insincere to capture more insincere questions.

Overall, the CNN model was determined to be not very suitable for complex text classification, performing the worst of the three models trained on the dataset.

5.1.1 Precision

First, the models were evaluated on precision - how many questions the model flagged as sincere that were actually sincere. As discovered through EDA, Quora appears to favor incorrectly labelling questions as sincere at the cost of potentially allowing some insincere questions to be considered sincere, suggesting precision is an important measure. When evaluated on precision *(see Figure 4.1 above)*, the LSTM model performed the best, although the Attention model has room for improvement if trained for longer. On this measure, the CNN model appears to perform more comparably to the other models. All three of the models, when done training, ended with precision over 86%, suggesting that on average the models correctly labelled at least 86 of every 100 insincere questions [1].

5.1.2 Sensitivity

Each of the models was also compared on sensitivity, or how often the model correctly labelled the sincere questions. The plot of the LSTM, CNN, and Attention models on sensitivity shows that the LSTM and Attention model performed similarly again *(see Figure 4.1)*. Even with only training the Attention model for three epochs, it still performs better than the other two models. The CNN model is out-performed, and evidence suggests that this model may have been over-trained.

5.1.3 F1 Score

Lastly, the models were compared on a weighted average of the sensitivity and precision - the F1 Score. The plot of the LSTM, CNN, and Attention models on F1 Score *(see Figure 4.1)* shows the LSTM and Attention model performed similarly, with both greatly out-performing the CNN model.

From the combination of both the F1 score and sensitivity plots, evidence suggests the Attention model correctly captures the most insincere questions and has the potential for further enhancements to performance if trained longer for additional epochs.

6. Conclusion

Evidence suggests the Attention model is a good approach to systematically and accurately identifying sincerity of questions posted on online question and answer (Q&A) forums. This model has success in classifying questions in a manner consistent with the labels provided by Quora.

6.1 Next Steps

Follow-on research could be performed to further enhance model performance including investigating alternative methods for labeling data, word embeddings, performing data augmentation, and further model training.

6.1.1 Hand-Labeled Data

Initial exploration suggested data labels potentially incorrectly flagged questions as sincere. Hand-labeling could be performed to improve data accuracy and potentially enhance results.

6.1.2 Word Embeddings

To perfect the model, different word embeddings should be tested beyond GloVe, but due to time limit and computation capabilities researchers did not try others. Additional research could be done to determine whether changing the word embedding has an impact on the success of the models.

6.1.3 Data Augmentation

Different data augmentation methods could be attempted to investigate whether improved results may be achieved. In “Data Augmentation in NLP,” Ma cites successful paper entitled “Data Augmentation for Visual Question Answering,” where Kafle et al. developed an approach for generating entire sentences rather than using techniques to replace individual words, like those used here [2]. This strategy could be utilized to determine whether it leads to improved classification of question sincerity.

6.1.4 Model Training

In order to make the training process faster, batch size was set to a size of 256 for each model. One downside is the model will converge slower, so if computation capability allows, a reduced batch size of 32 could be tried. Additionally, due to resource limitations, the LSTM and CNN models were trained for seven epochs, and the Attention model was only trained for three. There models could be trained for longer and re-evaluated for performance.

References

[1] Ghoneim, S. “Accuracy, Recall, Precision, F-Score & Specificity, which to optimize on?”. https://towardsdatascience.com/accuracy-recall-precision-f-score-specificity-which-to-optimize-on-867d3f11124, April 2, 2019

[2] Ma, E. “Data Augmentation in NLP: Introduction to Text Augmentation” https://towardsdatascience.com/data-augmentation-in-nlp-2801a34dfc28.

[3] Pennington, J., Socher R., and Manning, C. D. “GloVe: Global Vectors for Word Representation”. https://nlp.stanford.edu/projects/glove/, August 2014.

[4] “Quora Insincere Questions Classification: Detect toxic content to improve online conversations“ https://www.kaggle.com/c/quora-insincere-questions-classification/overview.

[5] “Programmatically Building and Managing Training Data with Snorkel” https://www.snorkel.org/get-started/.

[6] Zhang, X., Zhao, J., and LeCun, Y. “Character-level Convolutional Networks for Text Classification” https://arxiv.org/pdf/1509.01626.pdf, 2015. PDF file.

Appendix

Appendix A: Code Used in Analysis

The annotated Python code developed for this research can be found at: https://github.com/jenniferpeng007/590project

Appendix B: Supplementary Plots and Figures

LSTM Model Details

Layer (type)	Output Shape	Param #
input_21 (InputLayer)	(None, 60)	0
embedding_19 (Embedding)	(None, 60, 300)	27000000
lstm_17 (LSTM)	(None, 128)	219648
dense_24 (Dense)	(None, 64)	8256
dropout_11 (Dropout)	(None, 64)	0
dense_25 (Dense)	(None, 32)	2080
dense_26 (Dense)	(None, 1)	33

Total params: 27,230,017
Trainable params: 230,017
Non-trainable params: 27,000,000

Word-Level 1D CNN Model Details

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_46 (InputLayer)        (None, 60)                0
_________________________________________________________________
embedding_43 (Embedding)     (None, 60, 300)           27000000
_________________________________________________________________
bidirectional_20 (Bidirectio (None, 60, 128)           186880
_________________________________________________________________
seq_self_attention_16 (SeqSe (None, 60, 128)           8257
_________________________________________________________________
flatten_8 (Flatten)          (None, 7680)              0
_________________________________________________________________
dense_53 (Dense)             (None, 8)                 61448
_________________________________________________________________
dense_54 (Dense)             (None, 1)                 9
=================================================================
Total params: 27,256,594
Trainable params: 256,594
Non-trainable params: 27,000,000
```

Bidirectional LSTM and Attention Model Details

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
_________________________________________________________________
dropout_15 (Dropout)         (None, 60, 300)           0
_________________________________________________________________
conv1d_5 (Conv1D)            (None, 58, 32)            28832
_________________________________________________________________
max_pooling1d_5 (MaxPooling1 (None, 29, 32)            0
_________________________________________________________________
dropout_16 (Dropout)         (None, 29, 32)            0
_________________________________________________________________
conv1d_6 (Conv1D)            (None, 27, 32)            3104
_________________________________________________________________
max_pooling1d_6 (MaxPooling1 (None, 13, 32)            0
_________________________________________________________________
conv1d_7 (Conv1D)            (None, 11, 16)            1552
_________________________________________________________________
max_pooling1d_7 (MaxPooling1 (None, 5, 16)             0
_________________________________________________________________
flatten_2 (Flatten)          (None, 80)                0
_________________________________________________________________
dense_29 (Dense)             (None, 200)               16200
_________________________________________________________________
dropout_17 (Dropout)         (None, 200)               0
_________________________________________________________________
dense_30 (Dense)             (None, 1)                 201
=================================================================
Total params: 27,049,889
Trainable params: 49,889
Non-trainable params: 27,000,000
```