STA221 final project report: Quora Insincere Question Classification

Jue Wang, Jinpeng Gao

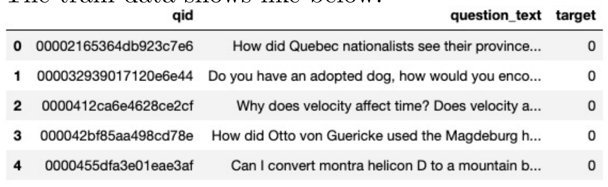
May 2020

1 Introduction

There are plenty of online platforms, such as Quora and StackExchange, that allow people to share their knowledge as well as to learn from each other. However, one issue that these websites now face is how to distinguish and remove the toxic and insincere questions and comments. This Kaggle challenge aims to develop a model that can identify insincere questions. Some typical insincere questions include questions having a non-natural tone, grounded on false premises, disparaging or inflammatory, or using sexual content for shock values. By tackling those insincere questions and toxic contents, we can contribute to keep the platform a safe and comfortable place for people to share information.

2 Data

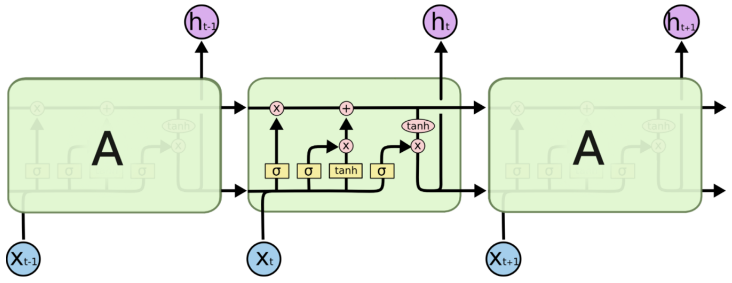
The train data includes about 1.3M rows and 3 columns (including question-id, question-text and target which indicates whether the question is insincere or not). The insincere label(label 1) accounts for about 6% and the data is highly imbalanced. Therefore, we use F1 score to evaluate the performance of classifiers.
Since this is a Kaggle competition, the test data does not have ground-truth labels. We decided to abandon the test data and randomly do the train-test-split to create our own validation data from the train data.
The train data shows like below:



	qid	question_text	target
0	00002165364db923c7e6	How did Quebec nationalists see their province...	0
1	000032939017120e6e44	Do you have an adopted dog, how would you enco...	0
2	0000412ca6e4628ce2cf	Why does velocity affect time? Does velocity a...	0
3	000042bf85aa498cd78e	How did Otto von Guericke used the Magdeburg h...	0
4	0000455dfa3e01eae3af	Can I convert montra helicon D to a mountain b...	0

3 Methodology

We specifically study three classification methods, the logistic regression, the Naive Bayesian and the bidirectional Long-Short term memory, which is a type of recurrent neural network. We choose these three algorithms because we expect logistic regression to be a simple baseline, and Naive Bayesian is a traditional machine learning algorithm to do document classification due to its simplicity and computational efficiency. On the other hand, we include a deep learning algorithm because we also want to touch on some hot topics, and explore how much they can improve from traditional ways.
Logistic regression is a statistical model that uses a logistic function to model a binary dependent variable by estimating probabilities. Naive Bayes classifiers are a family of simple 'probabilistic classifier' based on applying Bayes theorem with strong naive independence assumptions between the features.
Furthermore, we built a specific RNN model (LSTM) for the project. Following is a graphical illustration of one Long-Short term memory network, where x_t is the input at time t, h_t is the output at time t, and A is the memory cell. All recurrent neural networks have the form of a chain of memory cells (hidden layer) of neural network. In standard RNNs, this hidden layer will have a very simple structure, such as a single tanh layer. In LSTM, each memory cell has three "regulators", usually called gates, to control information inside the LSTM unit: an input gate, an output gate and a forget gate. The forget gate $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$ controls whether to include the information from previous cell c_{t-1}. The input gate $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$ controls whether to include the input x_t. After the two regulators value are determined, the memory cell can be updated as $C_t = f_t * C_{t-1} + i_t * tanh(W_c \cdot [h_{t-1}, x_t) + b_C)$. Finally, the output will be filtered by the output gate: $o_t = (W_o[h_{t-1}, x_t] + b_o)$, and the final output is $h_t = o_t * tanh(C_t)$.



The repeating module in an LSTM contains four interacting layers.

There are a variety of extensions of the RNN network. One of them is Bidirectional LSTM, which combines the LSTMs with Bidirectional Recurrent Neural Networks (BRNN). BRNNs connect two hidden layers of opposite directions to the same output. With such architecture, the output layer can get simultaneously information from both the positive time direction (forward states) and the reverse time direction (backward states), and thus can have a better understanding of the context.

4 Implementation

We explored the frequencies of single words, bigrams and trigrams to see the disparity between two different labels. We also conducted feature engineering to create some new features and explored the impact of the newly created eight features on the response variables. The features are number of words, unique words, characters, stop words, punctuation, upper words, title words and mean word length. The process of new feature selections are attached in appendix.
For Logistic Regression and Naive Bayesian implementation, we first transformed the question text into Tf-idf matrix using TfidfVectorizer and then attached the new features created. Since calculating TF-idf matrix for entire training dataset takes a huge amount of memory, we randomly selected 100,000 question texts to run Logistic Regression and Naive Bayes model. We also run BiLSTM models on the same dataset for comparisons. We build the Logistic Regression with the scikit-learn packages. Since we got over 3000 features, we added L1 regularization to the loss function and by tuning the penalty coefficient, we obtained the F1 score on the validation set. For the Naive Bayes model, we selected Gaussian as the assumed distribution and calculated the F1 score on the same validation set.
When implementing the LSTM algorithm, we first created an embedding layer. We implemented two different approaches: unpretrained embedding layer and pretrained embedding layers. For the unpretrained embedding, we applied the Embedding layer in Keras, with input_dim= 80000, output_dim= 100, and input_length= 300, which means we only considered the top 80000 commonly used words, the first 300 words in each question text, and each word is projected to a 100-dimension vector. For the pretrained embedding layers, we applied the four different methods: Word2Vec (GoogleNews-vectors-negative300), Glove(glove.840B.300d), paragram embeddings(paragram-300-s1999) and FastText Embedding (wiki-news-300d-1M).
We built the algorithm using the Keras LSTM architecture. Since BiLSTMs are easily getting over-fitted, we will use dropout to control the number of parameters. We let dropout=0.3 to randomly dropout some input units and recurrent_dropout = 0.2 to randomly dropout some recurrent cells. We allow "return_sequence" parameter to be true to return a 80-dimension output at every input state because we may get a significant signal at any input state. We then apply a global max pooling layer for each dimension, which reduce the 80*300 matrix to a 128 array. We apply another dropout layer with probability equals to 0.2. Following the BiLSTM structure, we add another two layers neural network. The first layer has neural units and the activation function is ReLU. The output will be project to a one dimension between 0 and 1 by the second layer, which has a sigmoid activation function. We use "binary_crossentropy" loss function and "Adam" optimizer to update the units. We let the batch size be 512 and set the learning rate to be 0.001. We use the same structure for all different embedding implementations and compare their performance on the full dataset.

5 Results

We run 100k data on the seven models and calculated the F1 score on the same validation set to compare the performance of different models. The baseline models, the Logistic Regression and Naive Bayes, have the pretty low F1 scores. On the other hand, the performance of deep learning approaches improve from below 0.5 to about 0.59 (Glove Embedding). Model with pretrained embedding layers performed very similarly with the model of unpretrained embedding layer. The reason may be only a small portion of the original data was trained. The results are summarized in the table below.

Model(baseline)	F1 score
LogisticRegression	0.498353
GaussianNB	0.166821
Model(BiLSTM)	F1 score
non-pretrained embedding	0.559462
Word2Vec	0.58
Glove	0.5891
FastText	0.5818
Paragram	0.5849

Following are the performance of 5 diffferent BiLSTMs implementations on the full dataset.

Model(BiLSTM)	F1 score
non-pretrained embedding	0.64504
Word2Vec	0.6595
Glove	0.6715
FastText	0.6514
Paragram	0.6692

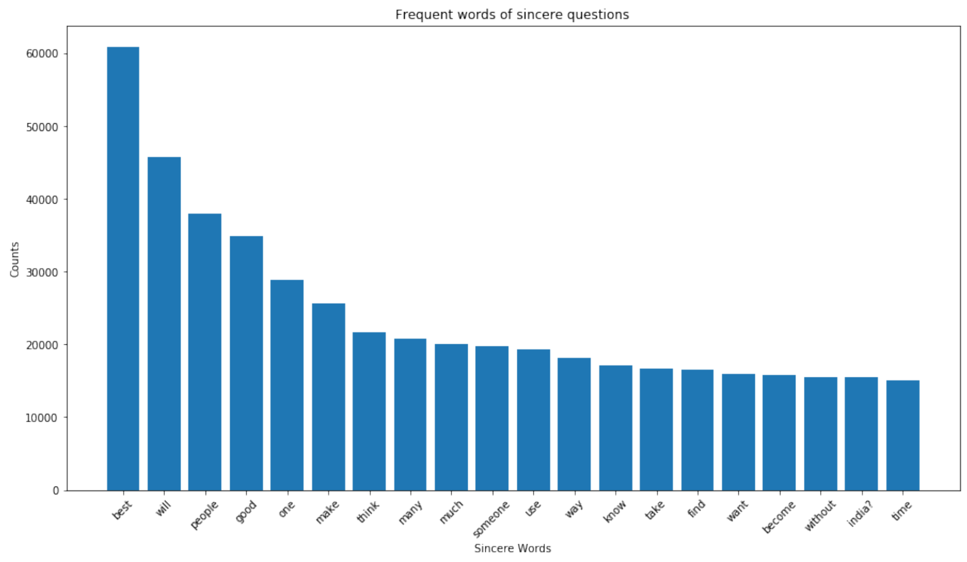
The Glove embedding has the highest validation F1 score after four epochs, while the non-pretrained embedding has the lowest F1 score. A reasonable explaination is that the self-training embedding layers could not fully represent the words in the dataset. Even though the pretrained Glove method gives the best F1 score and shows improvement from nonpretrained embedding, the difference is not significant. One possible reason is that these pretrained embeddings are not trained against text in the same context that the Quora questions are generated, so the number of common words between our text and their text would be low. Another reason is that we can use more complicated models to combine with Attention algorithm or do some cross validations to better tune the model. On the other hand, these results do not tell the full truth since there are some overfitting in these models. The specific training and validation errors are presented in appendix.

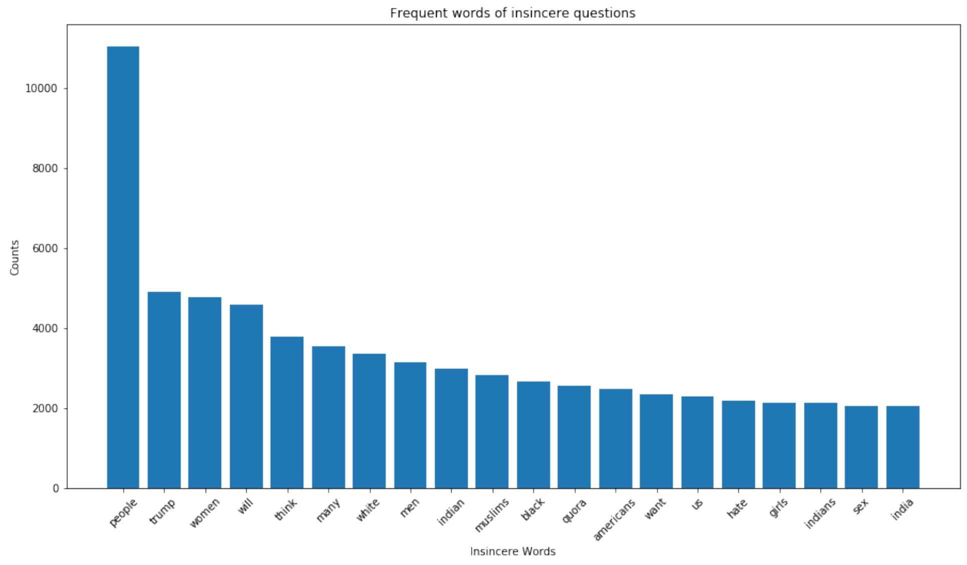
Appendices

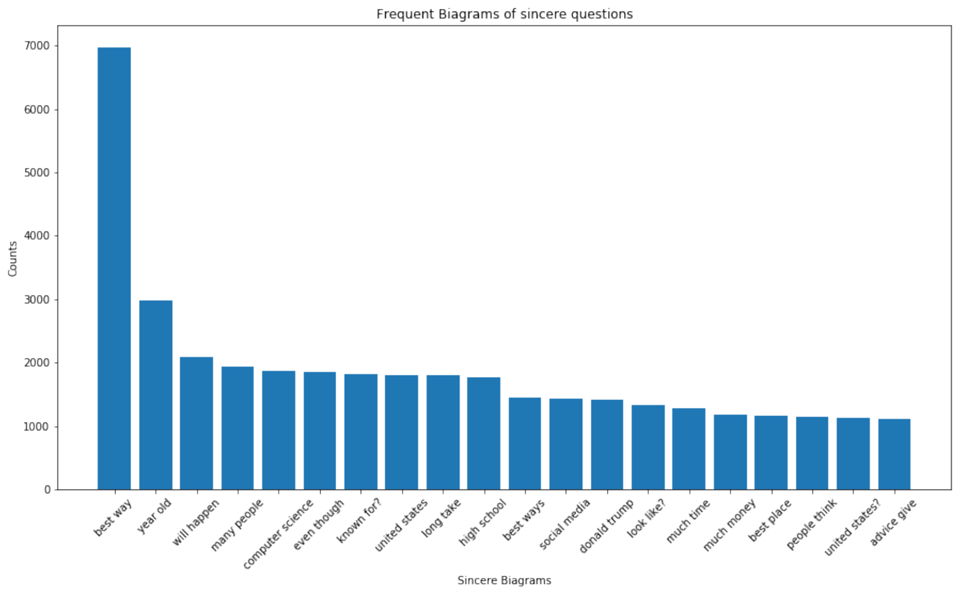
Feature selection

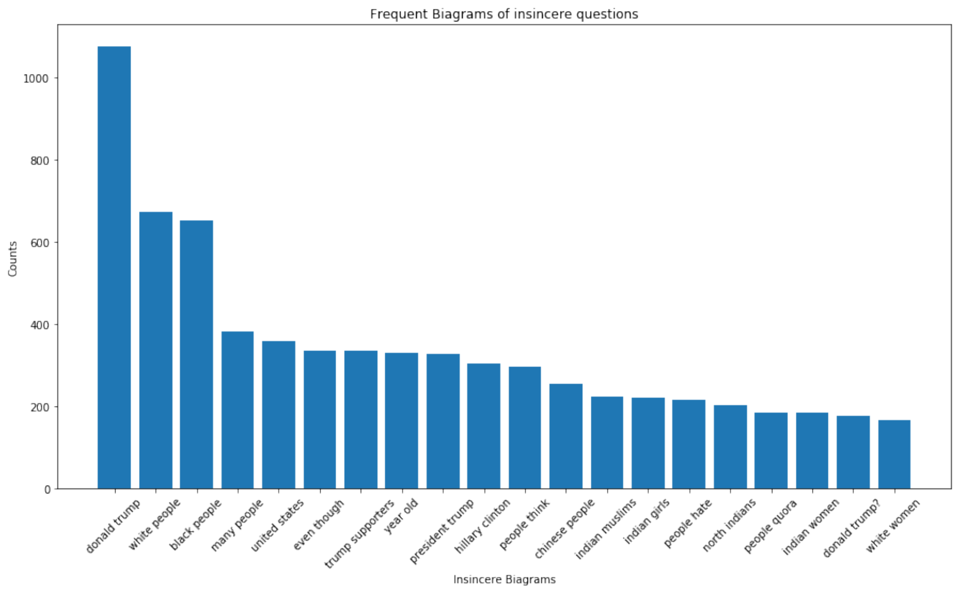
The following are some visualizations of the question text and the process of selections the new features, which have some significant difference over the the sincere and insincere questions.

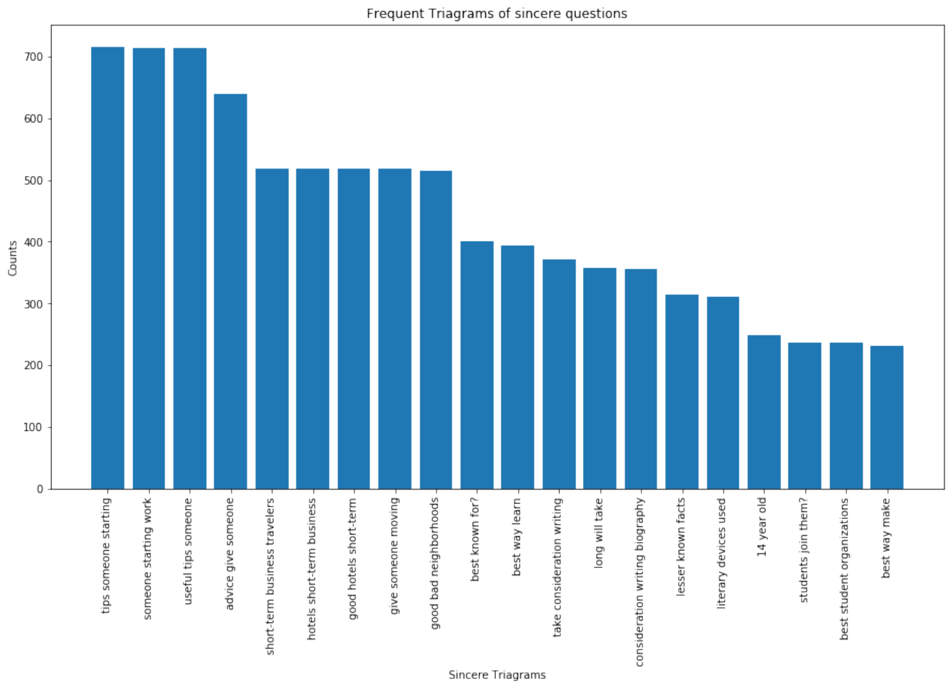
The following figures are the counts of some words with highest frequencies sincere and insincere questions separately.

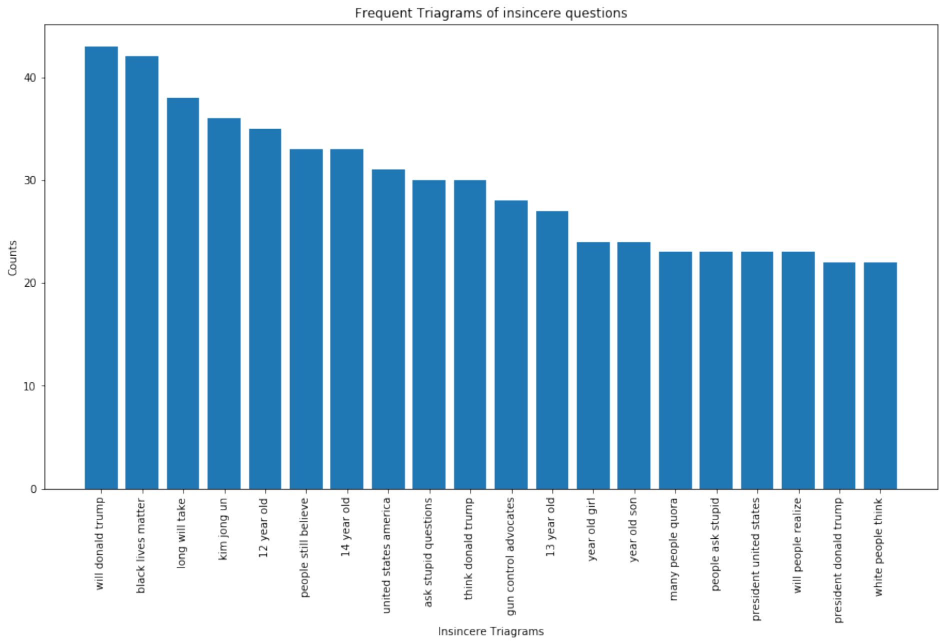




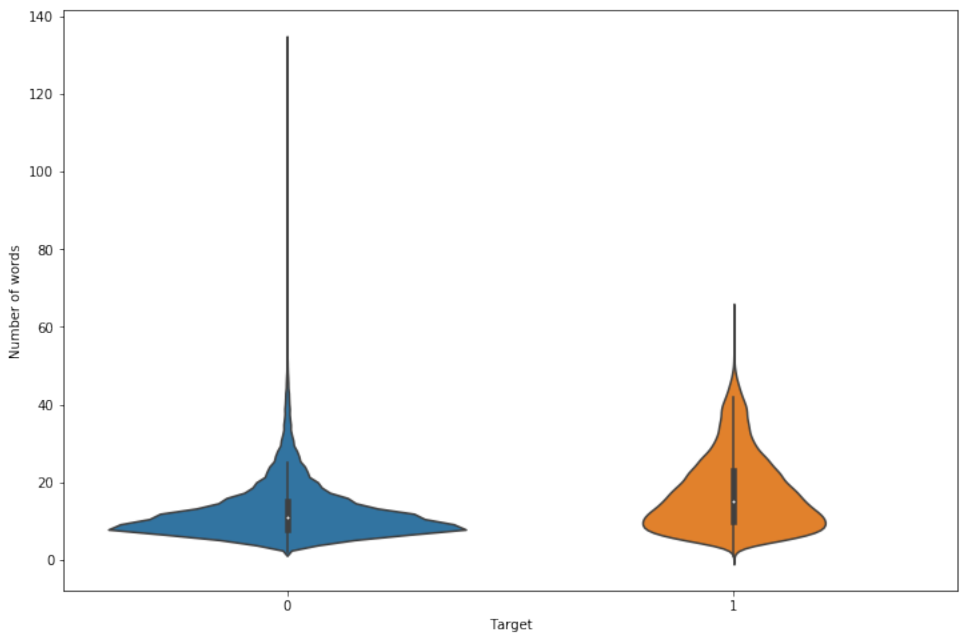

Frequent Biagrams of sincere questions
Counts
Sincere Biagrams
best way
year old
will happen
many people
computer science
even though
known for?
united states
long take
high school
best ways
social media
donald trump
look like?
much time
much money
best place
people think
united states?
advice give

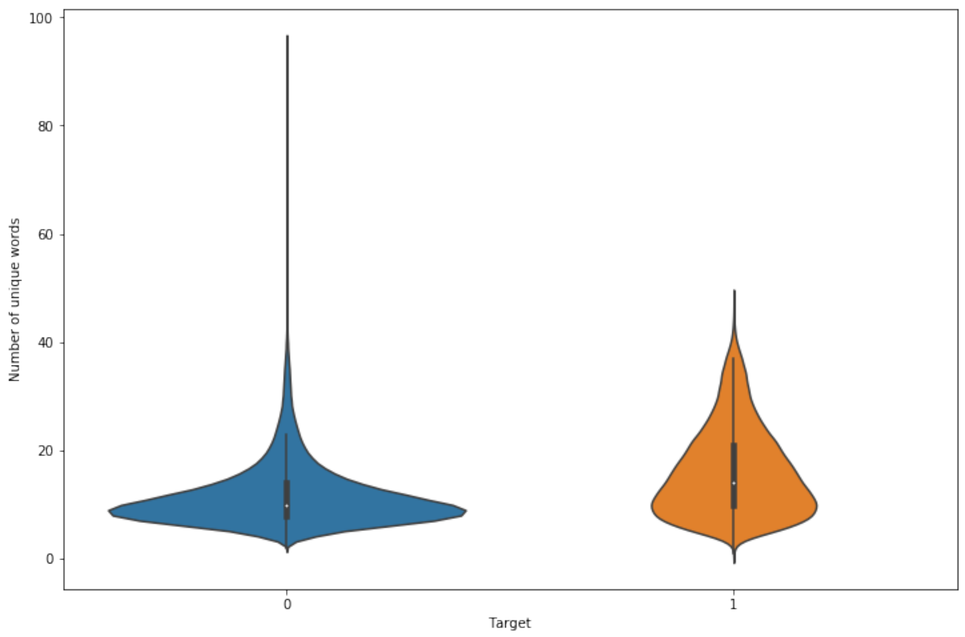

Frequent Biagrams of insincere questions
Counts
Insincere Biagrams
donald trump
white people
black people
many people
united states
even though
trump supporters
year old
president trump
hillary clinton
people think
chinese people
indian muslims
indian girls
people hate
north indians
people quora
indian women
donald trump?
white women

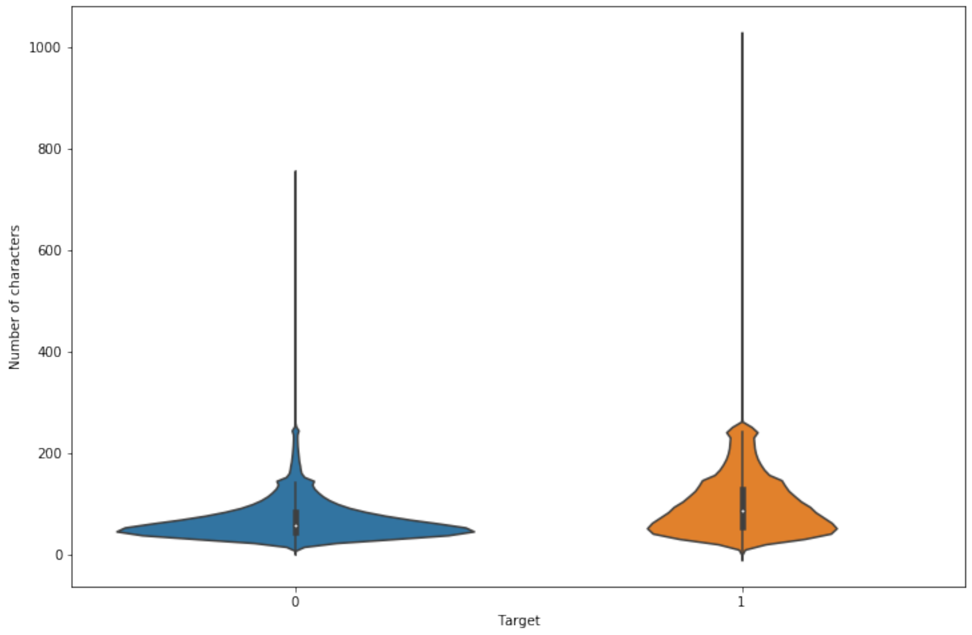


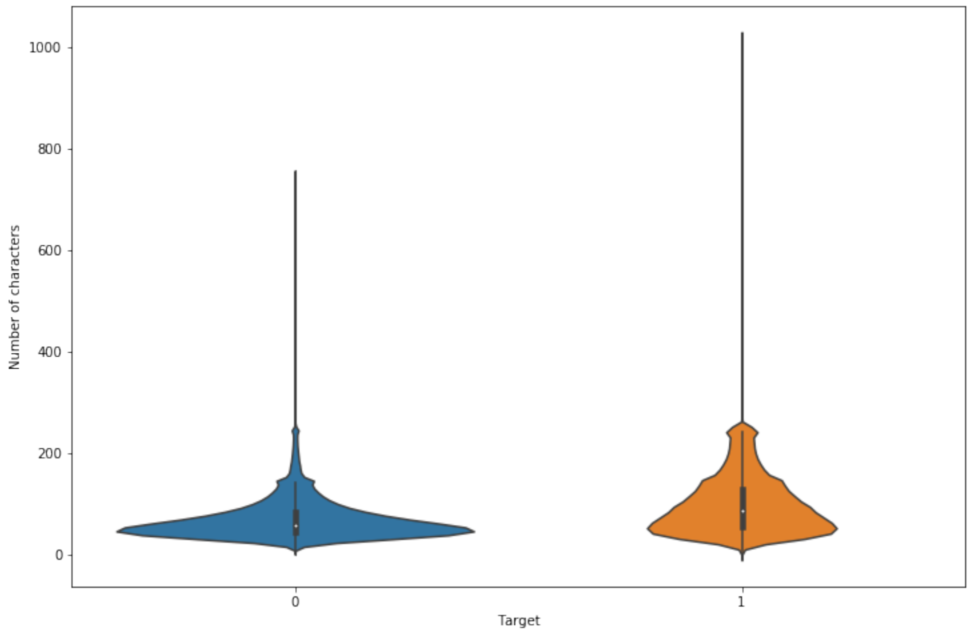


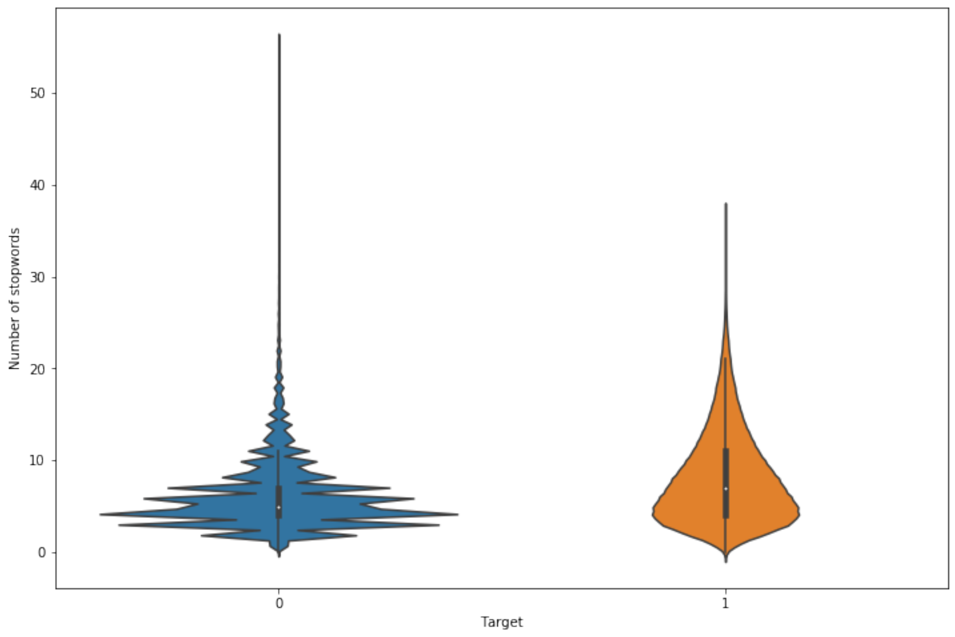
The following figures are violin plots of certain features of both the sincere (target=0) and insincere (target=0) questions. The plots illustrate that these features are significantly different for the two targets.



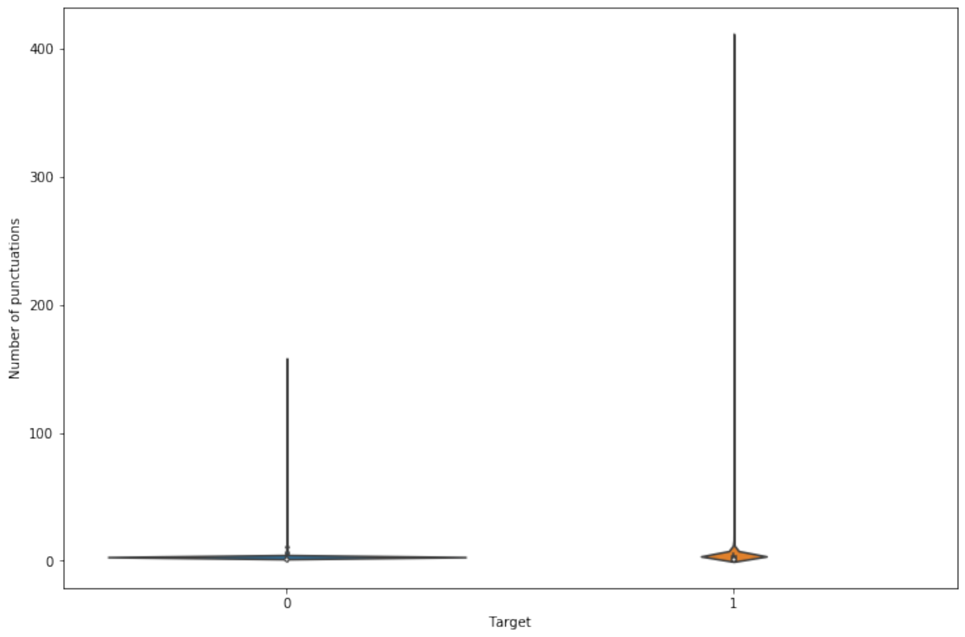



Number of characters
1000
800
600
400
200
0
0
1
Target

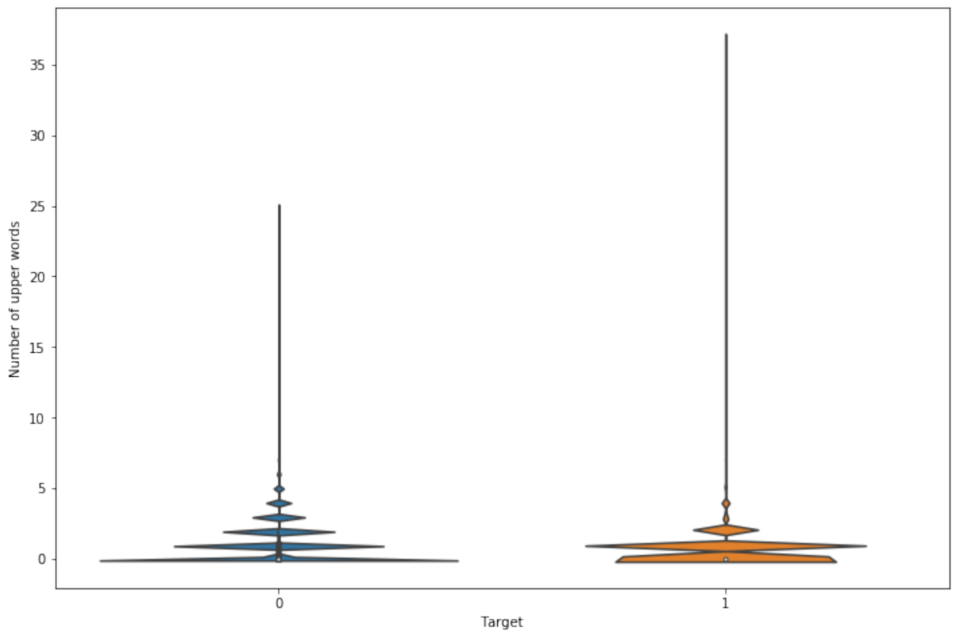

Number of characters
1000
800
600
400
200
0
0
1
Target



Number of stopwords
50
40
30
20
10
0
0
1
Target

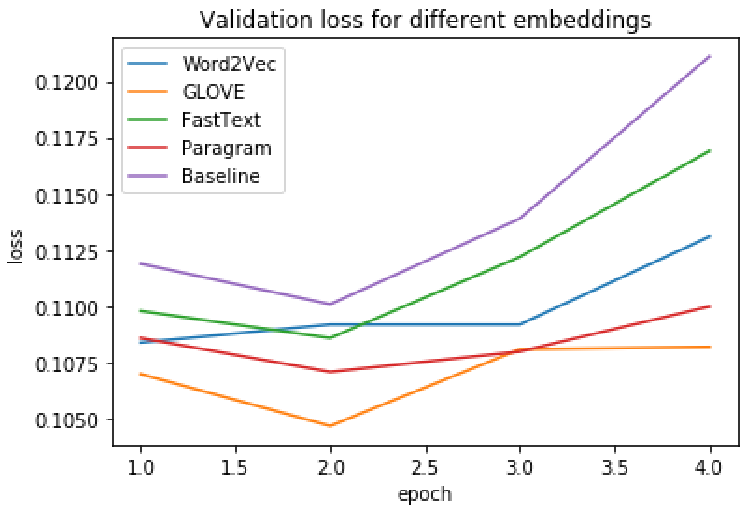


Number of punctuations
400
300
200
100
0
0
1
Target

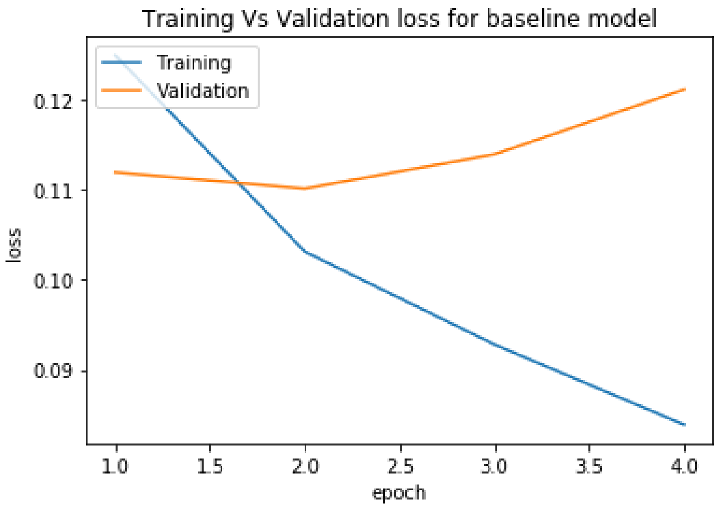


Number of upper words
35
30
25
20
15
10
5
0
0
1
Target

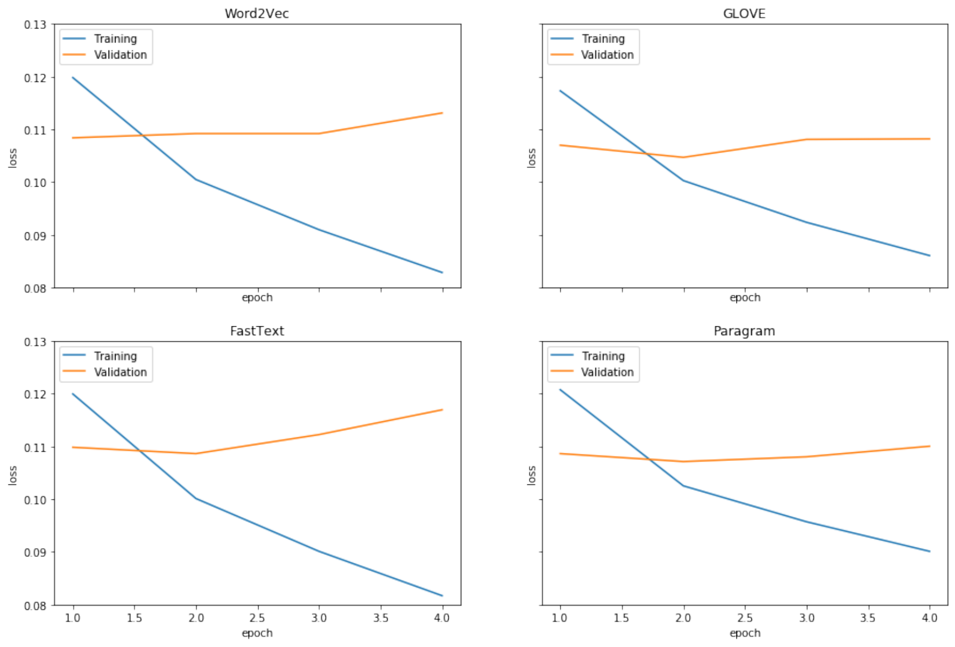
Training and validation error



The validation loss of these models has shown clearly the overfitting that after two epochs, the validation loss starts to increase. We also noticed that the Glove embedding has the lowest validation loss in the second epoch when the baseline model has the highest validation loss on the second epoch.



The training and validation loss for baseline BiLSTM model shows the overfitting in the traning process.



The plot above for four different pretrained embedding layers also shows the overfitting. With all the losses laid out, it's easy to see that the Glove has the lowest validation loss compared with other approaches. Then let's look at the F1 score on the validation set.