[Team 17] Detecting Insincere Questions on Q/A Platforms

Chintan Gandhi (cagandhi), Dip Patel (dpatel27), Faraaz Kakiwala (fmkakiwa)

***Abstract*—The primary goal of this project is to develop a model which can flag insincere questions on Q/A platforms. This is a relevant challenge as there is a lot of false information on Q/A platforms as well as verbal attacks which tend to belittle certain communities or races. Due to the humongous number of questions posted on platforms such as Quora, manually filtering out such insincere questions is out of scope. Such questions need to be weeded out so that users can feel safe while sharing knowledge. We treat the problem as a binary classification problem and propose an embedding based recurrent architecture to classify the question as insincere or not. We leverage a pre-trained GloVe model to generate meaningful word embeddings which are passed to a Bidirectional LSTM block followed by fully connected layers with the final layer outputting a probability which is 1 for an insincere question and 0 otherwise. On the highly imbalanced dataset of "Quora Insincere Questions Classification", we were able to achieve a validation F1 score of 0.67 vastly improving upon the baseline F1 score of 0.5.**

I. Model Training and Selection

We model the problem of classifying a question as insincere or not as a binary text classification problem. Deep learning approaches have proven to be extremely effective in the domain of text classification. LSTM [5] networks and their variants such as Bidirectional LSTM have shown to perform well in sequence based learning tasks. Furthermore, word embedding techniques such as Global Vectors for Word Representations (GloVe) [3] have proven to be powerful in learning word analogies and have been used to obtain vector representations in lower-dimensional spaces for words in the sentences. Since the classification problem is more intricate than just looking for a set of stopwords, we exploit the sequential nature of the text in both forward and backward directions through the use of a Bidirectional Recurrent Neural Networks [8].

We adhere to the approach described in [2] which combines the ideas described above into a single approach. To exploit the sequential nature of a tweet and learn the temporal correlation among the words, we use a deep learning architecture known to perform better on textual data. Firstly, we pre-process the question text to remove links and punctuations from the raw questions. Secondly, we use a pre-trained GloVe embedding layer to obtain feature representations for the words. Thirdly, we pad or truncate the sentences to fix their length to the number of timesteps to process in a bidirectional LSTM cell block. The outputs from both the forward and backward layer LSTM cells are combined. The BiLSTM cell block is followed by several fully connected layers and a sigmoid activation function which provides us the probability score of the sentence which is either 1 (insincere question) or 0 (sincere question).

The toolbox and libraries along with the specific functions used are listed below:

Libraries	**Function Names**
numpy	load, save, squeeze, random.uniform
pandas	read_csv
nltk	WordNetLemmatizer, word_tokenize
scikit-learn	classification_report, confusion_matrix, train_test_split, resample
re	compile
matplotlib.pyplot	plot
tqdm	tqdm
keras preprocessing	Tokenizer, pad_sequences
keras optimizers	Adam
keras models	Sequential, Model, load_weights
keras layers	LSTM, Bidirectional, Dense, Embedding, Flatten
keras callbacks	ModelCheckpoint

TABLE I: Toolboxes used

A. The Model

1) *Data pre-processing*
The raw questions in the dataset can have several discrepancies like the presence of hyperlinks. Since a hyperlink is used only to navigate users to other sources and are not required to identify the question as insincere or not, we need to filter them out. Similarly, all the punctuation symbols also need to be removed as we can learn the tone of the sentence from prepositions and conjunctions. Furthermore, we apply lemmatization to convert the inflectional form of all words to their root form. We do not remove any stopwords as removing them would distort the sentence and it would be difficult to learn the sentence structure itself. Each pre-processing step is illustrated by the example given below:

Original sentence:
Why is Bannon trying to help Swedish democratic party (extreme right wing party)?
https://www.dn.se/nyheter/varlden/bannon-weve-studied-the-sweden-democrats-for-a-while/

After removal of URLs:
Why is Bannon trying to help Swedish democratic party (extreme right wing party)?

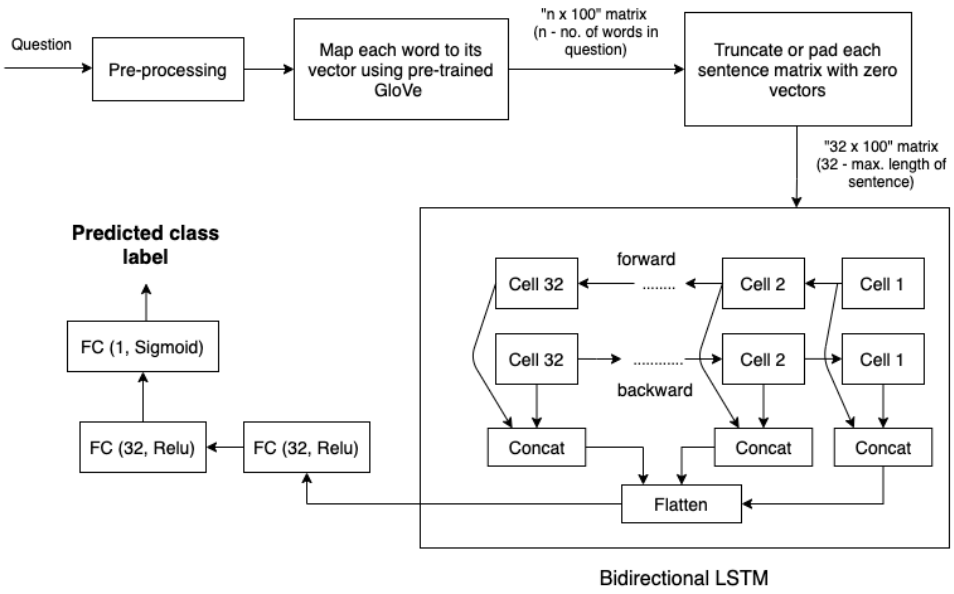

Fig. 1: Model Diagram

After removal of punctuations:
Why is Bannon trying to help Swedish democratic party extreme right wing party

After Lemmatization:
Why be Bannon try to help Swedish democratic party extreme right wing party

2) *Vector Embeddings*
GloVe [3] embeddings are used to represent words into vectors. This method transforms every word into a vector of length 100. The vectors are computed on the basis of probabilities defined from a global word-word co-occurence matrix. Thus, the words that are similar are projected closer to each other. As the number of input states in BiLSTM is pre-defined, we pad the output with zeros or truncate to convert the dimension from N $\times$ 100, where N is the number of words in the sentence, to 32 $\times$ 100. To overcome the issue of overfitting and resource constraints, we have used a pre-trained GloVe model from Wiki 2014 corpus that contains 6 billion tokens and generates 100-dimensional vectors for words.

3) *Bidirectional LSTM*
The BiLSTM [4] is a special variant of RNN. It consists of 2 layers of LSTM: a forward layer and a backward layer. Both these layers process the text in forward and backward sequences respectively. Thus, at any timestep *t*, we have information about previous words in hidden state of cell in forward layer and information about future words in hidden state of cell in backward layer. These vectors are then concatenated and passed forward so that the model has enough information about the semantics and structure of the sentence.

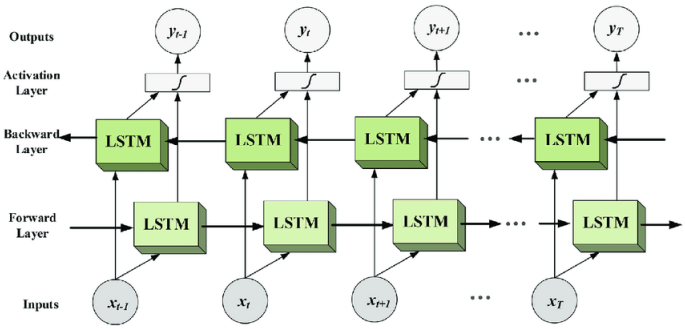

Fig. 2: Basic Structure of the BiLSTM Network [7]

The overview of our approach (**Fig. 1**) can be summarized by the steps given below:

1) Apply pre-processing techniques on the question text as discussed before.
2) Generate 100-dimensional GloVe embeddings for every word in the question text.
3) Truncate or pad the sentence till its length becomes 32.
4) Pass these 32 words as input to Bidirectional LSTM cells corresponding to 32 timesteps.
5) Pass the LSTM output through 2 fully connected layers of 32 neurons each.
6) Apply sigmoid activation at last layer to receive final probability of question being insincere.

B. Baseline

We implemented a traditional machine learning based approach as described in [2] for our baseline and compare the results. For the baseline approach, the text is pre-processed

in a similar fashion to our approach with the added function of removing the stop-words. Vectors for each of the question sentence are then generated with their length being the size of the vocabulary. The entry in the vector corresponding to each word is the TF-IDF score which reflects the relative importance of the word in the all the training sentences. These vectors are then fed as input to a Support Vector Machine with a Linear kernel and the metrics are calculated on a validation dataset.

II. Experimental Section

A. Metrics

The metrics we use to check model performance are F1 score and Recall. Since the dataset is highly imbalanced, accuracy is not a useful measure and may represent classifier bias towards the majority class. We also observe that the model should be able to maximize recall since not flagging insincere questions carries a greater cost than flagging sincere questions.

B. Model Selection

For all our experiments related to hyper-parameter tuning, we take into account validation loss and F1 scores. Initially, the model architecture had a fixed number 128 units for various RNN cell structures. The models were trained for 10 epochs with a batch size of 64 and default Adam optimizer. The experiment was carried out on three different types of RNN cells namely: GRU, LSTM and Bidirectional LSTM.

RNN Architecture	Val. Loss	Val. F1-score
GRU	0.2876	0.6195
LSTM	0.2522	0.6406
Bi-LSTM	**0.2446**	**0.6481**

TABLE II: Architecture Selection

From the table, we see that the Bidirectional LSTM outperforms other RNN cells with the least validation loss and the highest validation F1 score. We carried out further experiments on hyperparameters with BiLSTM as our cell structure. The parameters that we chose for further hyperparameter tuning were as follows:

- Batch Size
- Number of Bi-LSTM units
- Learning Rate

With the same parameters as mentioned in the above paragraph, various batch sizes were tested to find the best value for our model.

Batch Size	Val. Loss	Val. F1-Score
32	0.2534	0.6235
64	0.2446	0.6481
128	**0.2118**	**0.6757**
256	0.2634	0.6503

TABLE III: Batch Size Tuning

The batch size of 128 gave the highest F1-score and the lowest loss among all the other choices. Hence, we fix the batch size to 128 for our future experiments. Next, the parameter for number of units in a Bi-LSTM cell was tuned.

No. of Bi-LSTM units	Val. Loss	Val. F1-Score
64	0.2551	0.6298
128	**0.2118**	**0.6757**
256	0.2358	0.6648

TABLE IV: Bi-LSTM Unit Tuning

The existing model which consisted of 128 units yielded the best performance on F1-Score and loss and no improvement was found for other configurations. All the above experiments were performed with Adam optimizer for a default learning rate of 0.001. We wanted to check how much does the learning process change with the change in learning rate.

Learning Rate	Val. Loss	Val. F1-Score
0.005	0.2634	0.6503
0.001	**0.2118**	**0.6757**
0.0005	0.2426	0.66
0.0001	0.2394	0.6606

TABLE V: Learning Rate Tuning

Decreasing or increasing the learning rate yielded no improvement to the validation F1-Score or the loss. The default value for the learning rate in Adam optimizer achieved the best results.

C. Performance and Comparison to Baseline

The training and validation loss and f1 score can be seen in the graphs below. The best values for both the metrics were obtained on the 8th epoch of our training. The training was continued for 10 epochs but no further improvement was found.

Fig. 3: Model Loss

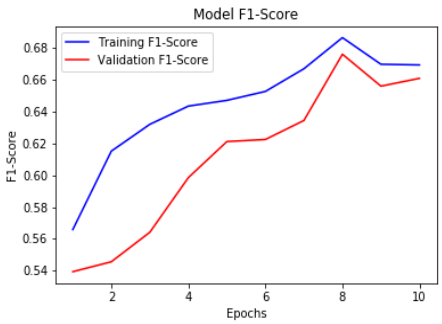

Fig. 4: Model F1-Score

The images below show some questions from the results of the classification obtained from our final model. Here *class 0* denotes a sincere question and *class 1* denotes an insincere question.

```
What osi layer uses frames? The data link layer or physical layer?
Do you ever go back and delete some of your old answers?
What if Edward Oxford assassinated Queen Victoria, would World War I never happen?
Is SolidWorks helpful for embedded engineers?
Can small noise or signature help fix issues of dangers of natural language processing?
```

Fig. 5: Class 0 Questions

```
Why do Iranians let Islam keep their country so backward?
How did Hitler maintain power?
Why did Prophet Muhammad (P.B.U.H.) eat dog shit?
Why are Israeli men short?
Why are most Europeans racists?
```

Fig. 6: Class 1 Questions

We do not have access to the test data labels and hence, we compute confusion matrix and classification report for our validation data. The baseline model achieved a F1-score of 0.5058 and the final approach achieved a F1-score of 0.67. The recall and precision values for class 1 have greatly increased from our baseline model.

	Precision	**Recall**	**F1-Score**	**Support**
0	0.99	0.90	0.94	122532
1	0.36	0.82	0.50	16162
Accuracy			0.90	138694
Macro Avg	0.67	0.86	0.72	138694
Weighted Avg	0.95	0.90	0.92	138694

TABLE VI: Classification Report for Baseline Approach

	Precision	**Recall**	**F1-Score**	**Support**
0	0.99	0.89	0.93	122532
1	0.53	0.91	0.67	16162
Accuracy			0.89	138694
Macro Avg	0.75	0.90	0.80	138694
Weighted Avg	0.93	0.89	0.90	138694

TABLE VII: Classification Report for Final Approach

From the confusion matrix tables below, we observe that our final model has much more True positives and much less False negatives than the baseline approach. From the confusion matrix and F1 scores, it is evident that our final model performs way better than the baseline approach.

		Predicted	
		0	1
Actual	0	110744(TN)	11787(FP)
	1	2880(FN)	13282(TP)

TABLE VIII: Confusion Matrix for Baseline Approach

		Predicted	
		0	1
Actual	0	108891(TN)	13461(FP)
	1	1518(FN)	14644(TP)

TABLE IX: Confusion Matrix for Final Approach

REFERENCES

[1] Quora Insincere Questions Classification Challenge on Kaggle. [https://www.kaggle.com/c/quora-insincere-questions-classification]

[2] Badjatiya, Pinkesh, Shashank Gupta, Manish Gupta, and Vasudeva Varma. "Deep learning for hate speech detection in tweets." In Proceedings of the 26th International Conference on World Wide Web Companion, pp. 759-760. 2017.

[3] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global vectors for word representation." In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532-1543. 2014.

[4] Zhiheng Huang, Wei Xu and Kai Yu. "Bidirectional LSTM-CRF Models for Sequence Tagging". ArXiv:1508.01991v1, 9 Aug 2015

[5] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." Neural computation 9, no. 8 (1997): 1735-1780.

[6] Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014).

[7] Yildirim, Özal. "A novel wavelet sequence based on deep bidirectional LSTM network model for ECG signal classification." Computers in biology and medicine 96 (2018): 189-202.

[8] Schuster, Mike, and Kuldip K. Paliwal. "Bidirectional recurrent neural networks." IEEE transactions on Signal Processing 45, no. 11 (1997): 2673-2681.