Automatic Quora Insincere Question Prediction

Laad, Sheetal; Navarro Garaiz, Esteban; Pearl, Adrian; Silva, Elliot

***Abstract*—For our final project, we experimented with various data preparation methods, feature extraction techniques and classification models to detect toxic and misleading questions on Quora for an on-going Kaggle competition. For feature extraction, we used count vectorization, TFIDF, and the GloVe word embedding. We focused on four classification algorithms: Bernoulli Naive Bayes, Logistic Regression, Convolutional Neural Networks, and Gated Recurrent Units. Because the metric used in the competition is F1-score, the harmonic average of precision and recall, we used F-score as the metric for model evaluation. At code freeze, our max F-score was 0.667, putting us at the 47th percentile on the leaderboard (the top-ranking F-score was .707). Future work should include implementing a topic model, integrating additional embeddings, and experimenting with other network architectures.**

I. Business Understanding

The mission statement of Quora is to "share and grow the world's knowledge" [1] by allowing users from around the globe to ask questions. These questions can be answered by those with firsthand expertise on the subject matter. Over 200 million users visit Quora on a monthly basis, [2] and thousands of questions are asked every day. [3]

One of Quora's main goals is to deliver as high quality content as possible by removing questions which are offensive, founded upon false premises, or aim to promote a user's own views rather than sincerely seek knowledge on a topic. For the sake of parsimony, any question which violates any of these standards is referred to as "insincere". Currently, Quora uses a combination of both machine learning and manual review to assess question sincerity [4] . By employing an automatic and scalable method to detect toxic and misleading content, Quora can more effectively remove inappropriate content without the need for manual review, and thus uphold its policy of "Be Nice, Be Respectful" [4]. This project aimed to develop a classification model to flag such insincere questions automatically.

A data science solution is directly applicable to this type of problem. Specifically, we are working from a dataset of Quora questions, pre-labeled as sincere or insincere. We are employing text classification techniques to treat the words (or sequences of words) as features representing the question, and the label of sincere/insincere as the target variable. This is a classic supervised learning problem. The model relies on the underlying assumption that certain words or phrases tend to be strong indicators of the sincerity of the question, and our work explores the quality of predictions you can make under this assumption.

II. Data Understanding

Our dataset was provided by Quora through the Kaggle competition. It includes two datasets: a training set (which we split into training and validation sets using sklearn's train_test_split function) with 1.3M instances, pre-labeled by Quora as sincere or insincere, and an unlabeled test set with 56K instances. Other than the label, the only feature in either dataset was the question text, which we then tokenized to create additional features, discussed in the following section.

The test set similarly contains a Quora question, except the labels for these are not provided to us per the competition's rules. In order to asses our success at predicting these labels, we train our model on the training dataset to make predictions on these test questions, then submit on Kaggle to receive an F-score evaluating our predictions. Table I shows examples of sincere (target=0) and insincere (target=1) questions in our training set.

TABLE I
EXAMPLE OF SINCERE (0) AND INSINCERE (1) QUESTIONS IN THE TRAINING DATA SET

How did Quebec nationalists see their province as a nation in the 1960s?	0
Do you have an adopted dog, how would you encourage people to adopt and not shop?	0
Why does velocity affect time? Does velocity affect space geometry?	0
Has the United States become the largest dictatorship in the world?	1
If blacks support school choice and mandatory sentencing for criminals why don't they vote Republican?	1
Which races have the smallest penis?	1

III. Data Preparation

There were several pre-processing steps that our group explored to prepare the question text for classification. We viewed these steps as part of our feature selection process, which was in turn part of our larger model selection process. Thus, we had no single pre-processing strategy. Rather, the specific steps could vary from model to model with the goal of independently observing the effect of each of them.

Some of the techniques explored were as follows: substituting contractions for their full form, removing punctuation, converting numerical digits to '#', fixing misspellings in specific words, word stemming and lemmatization, and removing stopwords.

Stemming and lemmatization are text normalization techniques that help reduce the numbers of features generated by reducing the inflectional forms of each word into a common base or root. Stemming and lemmatization differ in that stemming cuts off the end or the beginning of the word, while lemmatization takes into consideration the morphological analysis of the words. For example, 'studies' and 'studying' would stem to 'studi' and 'study' (respectively), but would lemmatize to 'study'.

Stop words are extremely common words which appear frequently in language, but generally add little value in determining sentiment. Examples of stop words include: 'a', 'as', 'be', 'that', 'which'.

The idea behind pre-processing text is homogenizing token-form, so that the model can understand different

topics referring to the same thing as one. For example, 'Trump!', 'trump', 'Trump's' would be used as different words in the model without pre-processing, when all three are referring to the last name 'Trump'. Merging them into a single representation allows the model to learn the intentions and context surrounding tokens much better. Not pre-processing these means losing a lot of information. Specific forms of the token, such as 'Trump!!!!' may appear so few times in the model that they stop being meaningful altogether.

Our dataset is highly unbalanced; it contains a 94%/6% split for sincere/insincere questions. Since unbalanced datasets often yield suboptimal prediction results when training a model on them, we explored downsampling, but found that it did not improve performance.

IV. Model Evaluation

For our classification task, we decided to focus on Naive Bayes, Logistic Regression, Convolutional Neural Networks, and Gated Recurrent Units. As stated above, our goal was to attain the best F-score, so our model performances were compared based on their F-score. Because we were limited on the number of results we could submit to Kaggle to receive the F-score for the test set, we evaluated our models based on F-scores predicting the validation set. We found that, while the F-score was consistently higher on the validation set than on the test set for the models we evaluated on both, the relative performance of the models remained the same. Therefore, we assumed a model performing relatively better on the validation set would also perform better on the test set.

A. Baseline Model

For a simple baseline model, we removed stopwords, lemmatized the remaining words and kept only those with length at least 4. This helped us identify the key words in a question, the ones that distinguish it. After cleaning, we created a corpus from the text in questions labeled as insincere. We removed words that appear either too frequently or too infrequently - words that had below 150 instances, as well as words that appeared in more than 20% of the insincere questions. This left a list of 662 words that could be meaningfully predictive of the sincerity of a question.

We iterated over every question in the training set and flagged the ones that have 5 or more of the words in the set as insincere. This tags 7% of the questions as insincere, which is similar to our class prevalence of 6.2%. The baseline model achieved an F-score of 0.3225, which is significantly better than what it would do at random, even if made to predict a 94%/6% split.

B. Traditional Model

The first phase of our model evaluation focused on traditional text classification techniques that we learned in class.

1) Feature Extraction: :

Text Normalization: As mentioned above, we experimented with common techniques such as stemming and lemmatization using the SnowballStemmer and WordNetLemmatizer functions from the nltk package. We found that lemmatization slightly improved model performance. Additionally,

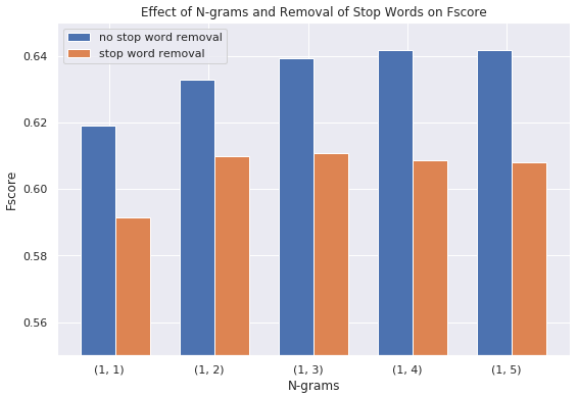

Fig. 1. Effect of N-grams and Removal of Stop Words on F-score.
Feature extraction: lemmatized text, Count Vectorizer, binary = True, default parameters for the remainder.
Model information: Logistic Regression, C = 1, default parameters for the remainder

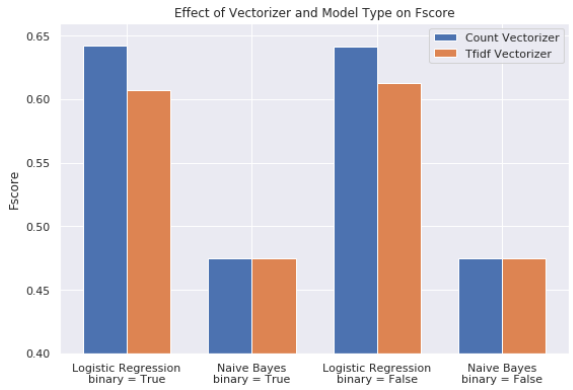

Fig. 2. Effect of Vectorizer and Model Type on F-score.
Feature extraction: lemmatized text, no stop word removal, N-grams = (1,4), default parameters for the remainder.
Model information: C = 1, default parameters for the remainder

we tested the effect of removing stop words from the question's text on model performance. As seen in Figure 1, removing stop words consistently resulted in lower F-scores.

Word Vectorization: After cleaning the text, we experimented with the CountVectorizer and TFIDFVectorizer functions from the sklearn package for extracting features. Count Vectorizer enumerates the word frequencies in every question, while TFIDFVectorizer values increase proportionally to count, but are offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently. The binary parameter within the vectorizer accepts True of False values. If binary is True in CountVectorizer, all non-zero term counts are set to 1. This is useful for discrete probabilistic models that model binary events rather than integer counts. As seen in Figure 2, CountVectorizer performed significantly better than TFIDFVectorizer for Logistic Regression and performed equally for Naive Bayes. Changing either vectorizer from binary to non-binary doesn't significantly affect F-scores.

N-grams Representation: The bag-of-words approach is a common text mining strategy, which treats every word as a term, discarding word order. Because word order is important in understanding a sentiment, preserving some information about word order could provide additional insight [5]. A common representation tactic is called n-grams, in which words are in combination with their adjacent set of words. Using a bag of n-grams up to three means that we use as features the individual words, adjacent word pairs, and adjacent word triplets. The disadvantage of using n-grams is that they greatly increase the size of the feature set, so it is important to find the optimal n-gram value. As seen in Figure 1 , model performance begins to plateau at n-grams up to four.

2) Modelling & Hyper-parameterization: The Bernoulli Naive Bayes classifier is a relatively simple algorithm. It assumes features to describe the data points are independent and boolean. In this case, the features would be words, the instances are the

questions and the boolean determines whether a word in the corpus is present in the question. Since our data is heavily structured under lexicographic rules, which is particularly true for questions, the independence assumption for the features is unrealistic, and thus the classifier does not perform well. Figure 2 confirms the poor performance of Bernoulli Naive Bayes, as all Logistic Regression models show significantly better performance. As a result, we focused on optimizing Logistic Regression models.

Classification Threshold: Within each model, we conducted an exhaustive search for the classification threshold value that would optimize model performance. A probability score below the classification threshold was classified as sincere, while a probability score above the threshold was classified as insincere. Adjusting threshold values from the default 0.5 greatly improved F-scores. Threshold values that maximized F-scores varied between models. Figure 3 shows model performance on the validation set for all thresholds. F-score appears to peak early and then abruptly decline. For our best model, the classification threshold that maximized F-score was approximately 0.1542.

Regularization: In order to prevent overfitting, we tested our model's performance on different levels of regularization (C-values). Figure 4 below shows that a C-value of 1 resulted in the highest F-score.

3) Best Traditional Model Results: Our highest performing model used binary count vectorizer, after lemmatizing the text with no removal of stop-words, and using n-grams up to 4. We then used these features on a Logistic Regression model with regularization parameter C=1. The best classification threshold for this model was 0.1673, resulting in a validation F-score of 0.6425. This model achieved a test F-score of 0.637 on Kaggle.

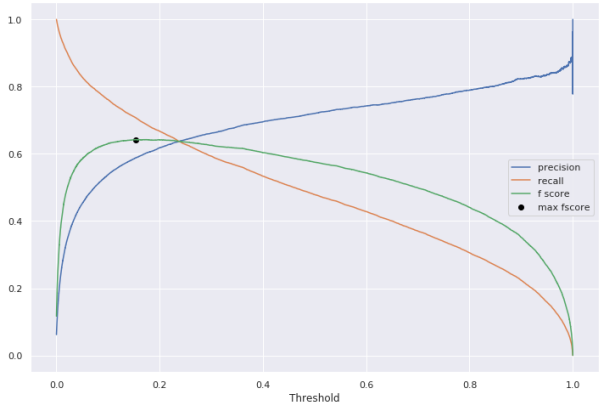

Fig. 3. Effect of Classification Threshold on F-score, Precision, Recall for Best Traditional Model

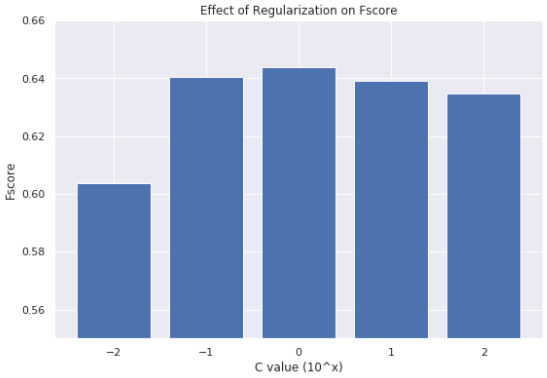

Fig. 4. Effect of Regularization on F-score.
Feature extraction: lemmatized text, Count Vectorizer, binary = True, default parameters for the remainder.
Model information: Logistic Regression, default parameters for the remainder

C. Advanced Techniques

1) Feature Extraction - Word Embeddings: After optimizing the performance using traditional text classification algorithms and feature extraction techniques, we set about exploring the potential benefits of using distributed word representations, commonly referred to as embeddings. In a typical tokenized

document representation, a given word w is mapped to a single integer i, and its presence or absence in a document is denoted by a 1 or 0 in the document vector at index i. The main problem with this format is that it gives no indication of word relationships or semantics. Other than looking for correlations of features across the corpus of documents, the model has no way of inferring whether two features represent synonyms, antonyms, or words with some more nuanced relationship such as superlatives.

Embeddings attack this problem by mapping each word onto a latent space, giving us a way to directly compare similar words. In this way, the dimensionality of each word's representation becomes larger, but the dimensionality of document vectors can be dramatically reduced because they no longer need to include zeros for every absent word. Recent research has shown that well-made embeddings can give significant performance boosts in various text-related tasks, including text classification [6]. We decided to test out the Logistic Regression model but replace the tokenized representation with the popular GloVe embedding of 300 dimensions, pre-trained on 8.4 billion tokens [7].

2) Using Embeddings with Logistic Regression: When using a word embedding in place of a tokenized representation, the first question to answer is how to combine embedding word representations, corresponding to a document's words, into a single vector, representing the document as a whole. Our proposed solution was to take different convex combinations of word embeddings to form a single document embedding of the same dimensionality. A first attempt created document embeddings by taking an average of the vector representations of the words in the document. This lead to significantly worse results than our traditional model and a maximum F-score of about 0.558.

The next attempt used two steps of preprocessing. First, tokenizing as before by using TFIDF counts, but then, for each document, utilizing normalized TFIDF scores to form a weighted combination of the embeddings corresponding to the words in the document. The motivation behind this approach was that it would significantly deemphasize unimportant words, and create document vector representations based mostly on words that were more indicative of the overall subject, meaning, and sentiment of the document. However, this approach performed worse than simple averaging, producing a maximum F-score of just below 0.550. After adjusting various parameters but not improving the result, we decided that using embeddings in combination with a simple Logistic Regression model was not a promising area for further investigation.

In general, this was not surprising, as any averaging scheme will lose most of the information contained in the geometric space and, therefore, the information contained in sequences of words.

3) Modeling - Neural Networks: As our attempts to use embeddings in conjunction with Logistic Regression were unsuccessful, the final phase of our modeling explored pairing the embedding's representation with Neural Networks. While our team had little prior exposure to this class of models, the

wealth of recent research showing that Neural Networks tend to outperform more traditional models for text classification tasks [8] motivated some experimentation with them.

The first important distinction between Neural Networks and Logistic Regression is that Logistic Regression averages the word vectors pertaining to each document, while Neural Network concatenates into a 2-dimensional array. Using Neural Networks results in much lower information loss and allows the model to reason about every word vector rather than trying to distill all of the document's meaning into a single vector.

Our first attempt used a 1-dimensional convolutional layer that processed five input vectors at a time. By processing groups of adjacent words, the convolutional layer served a purpose that is somewhat analogous to an n-gram vectorization, but with the added benefits of using the embedding and the network's ability to learn optimal feature weights. The convolutional layer was followed by a max-pooling operation - to make subsequent steps more computationally tractable - and a pair of dense layers in which a dot product is taken between a weight vector and the entirety of that layer input all at once to produce the layer output, rather than performing this operation for small chunks of the input. After tweaking model parameters, we achieved our best validation F-score of 0.647 by using three convolutional layers, each followed by a max pooling operation and a 10% dropout layer to add regularization. As before, the convolutional stages were followed by two dense layers using softmax loss.

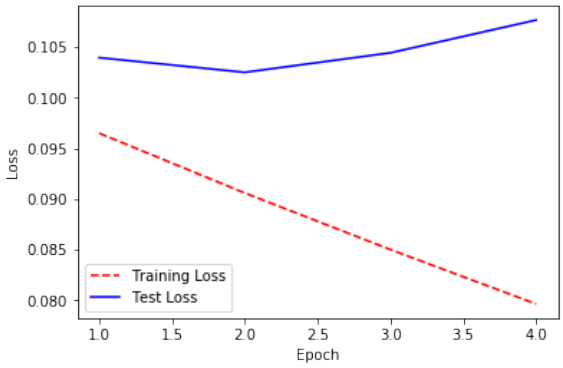

Fig. 5. Training and validation loss for a convolutional network model after each epoch of training

Figure 5 shows the average loss calculated on both the training and validation sets after each of four epochs of training, using one of our earlier convolutional networks. It is clear from the increase in validation loss that the model substantially overfits the training set, and adding the dropout layers helped fix this problem.

Our best model to date was the result of transitioning from a convolutional network architecture to using a Gated Recurrent Unit (GRU). Our team had virtually no prior exposure to this more complex architecture, but background research suggested that the GRU was a promising candidate to improve upon our previous work. The GRU is able to more intelligently reason about how long sequences of words evolve, thanks to the use of a hidden state that is partially maintained and partially updated when analyzing each word vector. Our best GRU model achieved our overall best F-score at code freeze of 0.667 by using a bidirectional GRU layer and max pooling, followed by three dense layers, each with 30% dropout. One feature that appears to be a strength of this model is its performance is nearly invariant in the neighborhood of its optimum threshold.

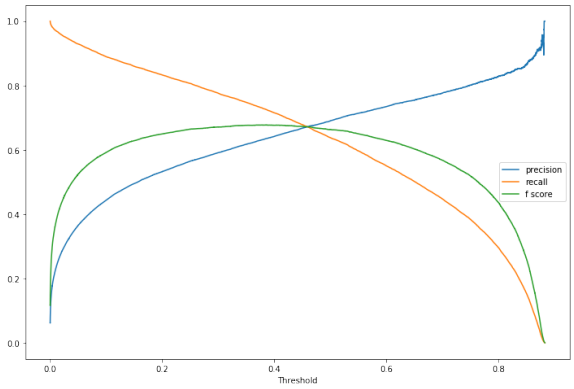

Fig. 6. F-score at every threshold for our best Neural Network model

As Figure 6 shows, the F-score curve is nearly flat, so we would not expect a slight change in the threshold to have significant performance impact. This is in contrast to the traditional model, which peaks more abruptly and begins to decline faster as the threshold increases.

Because there are many decisions to make in choosing a Neural Network architecture, our team decided to experiment with a select few of them. We noticed that two hidden layers worked better than one, as this makes the network deeper and allows for interdependencies between word embeddings to be represented more. In addition, we took the bias-variance tradeoff between complexity of the model and overfitting the training set into consideration, by increasing the number of neurons per layer from 16 to 32, and increasing the dropout probability from .1 to .3. Dropout is a form of regularization wherein certain neurons are "shut off" during an iteration of training the weights. Shutting off certain neurons simplifies the model and prevents certain neurons from becoming too important. Finally, we found that the Sigmoid activation function between layers performed better than the Rectified Linear Unit ("ReLu") activation function.

TABLE II
VALIDATION CONFUSION MATRIX AND F-SCORE.
BASELINE (.322) BEST LR (.642) BEST NN (.675)

0.890	0.048	0.908	0.030	0.910	0.027
0.041	0.021	0.018	0.044	0.017	0.046

TABLE III
EXAMPLE OF FALSE NEGATIVE (FN) AND FALSE POSITIVE (FP) QUESTIONS THAT OUR NEURAL NETWORK GENERATED IN THE TRAINING DATA SET

Why do Nepalese look like Chinese despite being culturally closer to Indians?	FP
Why do most Arab Muslims dislike Sufis?	FP
Is Barack Obama going to follow in the footsteps of his colleagues, the Clintons, and embark on a speaking tour that will net him millions of dollars?	FP
Why did American people vote Biff Tannen as the President of the United States?	FN
Can evolution be put in a bottle and taken to Mars?	FN
What is the main reason of underdevelopment and poverty in Greece making them very dependent on bailout aid from EU member states even though they received plenty of support already?	FN

Table III shows sample false negative and false positive questions that our model generated. For the first two false positives, these discuss race, which is likely why they were classified as insincere. Many Quora users might post hateful comments about Muslims, or racist comments about people who "look Chinese", so the presence of these words triggered the misclassification, despite being sincere questions about the dynamics between religions, or similarities between ethnicities. Furthermore, we see many questions where people troll President Obama or the Clintons, which is likely why the third question was predicted as insincere as well. The first false negative is based on a false premise, which our learning model would have no way of knowing. The second false negative is completely nonsensical, but mentions topics such as Mars and evolution that many people may have sincere questions about.

Finally, the last false negative question introduces the user's own opinion ("plenty of support"), around sincere and intellectual phrases, which mislead the model.

V. Deployment

By integrating this model into its system, we can help Quora get one step closer to its goals of creating a safe and comfortable environment for its users to share and gain knowledge. The best way to integrate this model would be to put it as a instant filter when a user clicks 'submit'. Those questions that are flagged as insincere wouldn't be allowed to be submitted, and Quora could use the model, like their current model, in combination with a human review system. One possible monitoring strategy early on in deployment could be to extract a daily sample of questions to be classified manually, and compare these ground truth labels to the model predictions. The goal would be to make sure that the model's performance remains near the level it achieved on the test set when it was created.

VI. Risks and Ethical Considerations

The two types of risk associated with any change to Quora's current system for sincerity detection are derived from either increasing the false positive rate (sincere questions labeled as insincere), or increasing the false negative rate (insincere questions not flagged as insincere). The risks of a false negative are apparent, and the root of this competition - questions which are insincere may not be filtered out - thus reducing the quality of the platform. False positives also have harmful effects - users are prevented from asking sincere questions to which they want to know the answer. In particular, there is the potential for discrimination if certain demographics use language or slang which has been deemed as strongly indicating insincerity.

VII. Conclusion & Future Work

Figure 6 shows the results of our best traditional model and best Neural Network.
As suspected, our Neural Network model performed significantly better than the traditional model which used Logistic Regression. While we had limited bandwidth to experiment with other techniques, below are few that our group would consider for future work.

Latent Dirichlet Allocation (LDA) is a type of topic model, which is a class of statistical models for discovering the abstract "topics" that occur in a collection of documents. A topic is not a defined instance, neither by its scope nor by its semantic relationships. A topic is identified by the model automatically using the likelihood of term co-occurrence.

While a word may occur in various topics, with different associated probabilities, it will be linked to a different set of neighboring words within each topic. In LDA, each document can be imagined as a mixture of several topics, assigned to the document by the allocation. After experimenting with several configurations of the model, we ended up with a 6-topic model, which captures a good portion of the information. The vector of probabilities for insincere given a topic is: $(0.013, 0.026, 0.087, 0.027, 0.161, 0.046)$. So a question predicted to be topic one has $\frac{1}{6}$ the chance of being insincere - given the class prevalence - and a

question labeled as topic five has close to three times the chance. Further work could attack the problem with an ensemble method incorporating the LDA to the Logistic Regression model or including it as another layer in the Neural Network.

Other popular pre-trained embeddings that we could use are are word2vec, fasttext and Paragram. Because Neural Networks and embeddings were new to us, we did not have the bandwidth to explore the other embeddings. A future iteration of this project would explore how the other embeddings could improve our model, perhaps using a combination of the four.

VIII. Contributions

As a group, we all implemented the traditional model from class (CountVectorizer, TFIDFVectorizer, Logistic Regression, Naive Bayes), and began initial exploration of choices for data cleaning, feature selection, model hyperparameters, etc. From there, we each added the following unique elements to the project:

A. Sheetal

- Grid search for best vectorizer, model, and hyperparameter tuning for traditional model
- Explored optimal data cleaning techniques for traditional model

B. Elliot

- Adapted best model to a modularized form, so we can work on specific sections and integrate together
- Hyperparameter tuning for Neural Networks

C. Esteban

- First look into word embeddings and cleaning techniques for their use.
- Created the Baseline Model for evaluation and implemented LDA topic model.
- Copy editing of final report and formatting into LaTeX

D. Adrian

- Tested use of GloVe embeddings with Logistic Regression
- Implemented Convolutional Neural Network model, tested use of dropout and different network configurations
- Implemented initial GRU model

References

[1] Quora. Our Mission. https://blog.quora.com/Our-Mission.

[2] Quora. How many people use Quora? https://www.quora.com/How-many-people-use-Quora-7.

[3] Quora. How many questions are asked on Quora each day? https://www.quora.com/How-many-questions-are-asked-on-Quora-each-day.

[4] Kaggle. Quora Insincere Question Classification https://www.kaggle.com/c/quora-insincere-questions-classification.

[5] Provost, Foster; Fawcett, Tom. *Data Science for Business*. O'Reilly Media, 2013.

[6] Mokolov, Chen, et al. *Efficient Estimation of Word Representations in Vector Space* https://arxiv.org/pdf/1301.3781.pdf

[7] GloVe: Global Vectors for Word Representation https://nlp.stanford.edu/projects/glove/

[8] Y. Bengio, R. Ducharme, P. Vincent, C. Jauvin. A Neural Probabilistic Language Model. *Journal of Machine Learning Research.*, Montreal, Quebec, Canada, pp. 1137-1155, Febraury 2003.

-