CS 229 Winter 2019 - Project Final Report
Insincere Questions Classification on Quora using pre-trained word embeddings

Team members:
1. Michael Lanier, *mlanier@stanford.edu*
2. Nagarjuna Rao Chakka, nagarjun@stanford.edu, SUID - 06402176
3. Sibi Shanmugaraj, *sibiyes@stanford.edu*, SUID – 06407840

Abstract:

The primary motivation for this project is to attempt to formulate an NLP problem as a generalized machine learning problem and solve them using various machine learning techniques. There are various challenges involved in using language element such as text as the features for a machine learning problem. There are various language models that have come up such as word-2-vec, BERT etc… We would like to attempt to solve a common problem in which the primary features are the raw text elements involved and use various language models to formulate/featurize them as a machine learning problem and solve them using various generalized ML techniques. We also like to benchmarks in terms of data size and algorithmic choice for identifying insincere/sarcastic language in this specific context. Data availability is an issue in these problem spaces. Language based phenomenon(insincere/hateful/tone) are difficult to detect, and we believe there is value in understanding how much data is needed to train models that can effectively model them. In this project we have attempted to solve the problem of classifying insincere question in the online Q&A platform Quora by using word embeddings to featurize the text elements. We were able to observe that these embeddings provide a good set of features that are able to capture trends in the data with respect to the binary classes we want to classify and help in building predictive models that have good ROC-AUC performance even with an unbalanced class distribution.

Method:

We have a classification problem in our hands with an unbalanced dataset. The positive examples account for little more than 6 % of the total examples. We intend on featurizing the text elements using word embeddings and possibly augment them with other features such as named entities and build a classification model. The model uses manually tagged output (by experts) which had to be kept in mind.

Data Preparation:

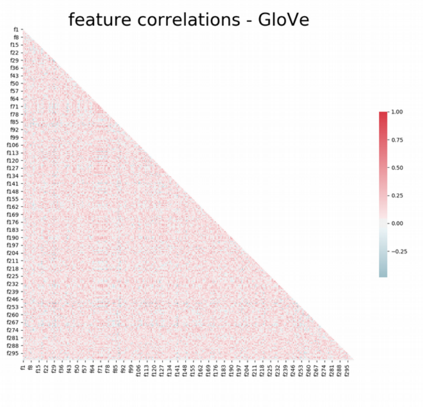
The data available for this task is the text element of the question and a flag indicating if it is an insincere question. We need to featurize them using pre-trained word embeddings. We used 2 different approaches to vectorize the text using embeddings. The first one is based on embedding obtained using word2vec method. We tried 4 different word embeddings for this class, namely, GloVe, GoogleNews word vectors, Wikinews word vectors and Paragram word vectors. All these word embeddings are word2vec based and encode word tokens using a vector of 300 elements and vary in the text corpus that they are trained on. For every question (row) we generated a feature mapping that basically averages out the the embedding for each token in the text. If a token is not available in the corpus we just ignore them in computing the average vector. After the mapping process we get a feature space that has 300 features that are floating point elements.

A second approach to vectorizing the text is based on the BERT method. The primary difference of BERT from word2vec is that they provide embeddings for a word that are much more dynamically informed by the words around them. This will let us compare the performance difference between BERT class of word embeddings and the word2vec class of word embeddings and see if BERT embeddings provide any improvement over word2vec based embeddings. There are many possible combinations one could use to vectorize based on BERT. For this case we chose to use the average of the values in the last 4 layers of the network. [It has to be mentioned that extracting BERT embeddings is a computationally expensive process and hence we were not able to extract BERT embeddings for the entire dataset we had. The models based on BERT were relatively data deficient and are built and tested from 100k rows of translated data]. BERT gives us a vector with 768 features.

These embedded features will help us run various modeling approaches to perform the classification task at hand.

Preliminary analysis:

Since we have a feature space that has relatively large number of columns (300 or 768), we look at the cross correlation between the features to ensure that there aren't many features that are highly correlated with each other. On the left we have the cross correlation heat map for the features obtained using the GloVe word embeddings. We can see that features mostly have a small degree of positive correlation between each other. Negatively correlated feature combinations are very minimal and the positive correlation observed between feature pairs are very small. The cross correlation between features is slightly lesser on other word2vec embeddings. For features obtained using BERT, a similar behavior was observed. The cross correlation between features was mostly near zero in that case.



K-means clustering:

Given that we have translated our text elements into numerical features, we can cluster them using k-mean clustering and see if we can get some clusters that isolate the positive examples, which in our case are the insincere questions. Clustering was performed using k-means approach on the word2vec based features and the BERT based features. We can see that the data that results from vectorization using the word embeddings does form clusters that to an extent isolates the binary classes in the problem.

In the tables below we have the results of the k-means clustering performed using features obtained from Glove word embeddings [Left] and BERT embeddings [Right]. We show the number of clusters and the **postive_rate [precision]** (*fraction of positive examples in the cluster*) and **total_positive_percent [recall]** (*fraction of all positive examples in the problem*) for the best cluster in terms of positive rate. We can see that as the number of clusters increase, a cluster develops which, includes significant fraction of the positive examples. For example for GloVe embeddings, for a 15 clusters k-means, there is a cluster that has almost 40% of instances corresponding to positive examples in them, incorporating 38% of all the positive examples. Thus we can see that there is some trend in the features obtained using the the word embedding. A similar trend was observed on the features developed using the other word2vec word embeddings as well. Similarly for features obtained using the BERT embeddings we see a pretty similar trend. The results for the clustering performed on BERT embedded features are shown in the second table. The clusters obtained can themselves be used as a simple prediction model where the probability of belong to positive class can be assigned as the positive_rate corresponding to the cluster with nearest cluster center for each examples.

Num clusters	**pos_rate (precision)**	**total_pos_percent (recall)**
4	0.142857	0.000054
6	0.205569	0.529099
8	0.218020	0.483786
10	0.249464	0.461881
12	0.294310	0.448513
15	0.396516	0.382530

Clustering results for GloVe embedded features

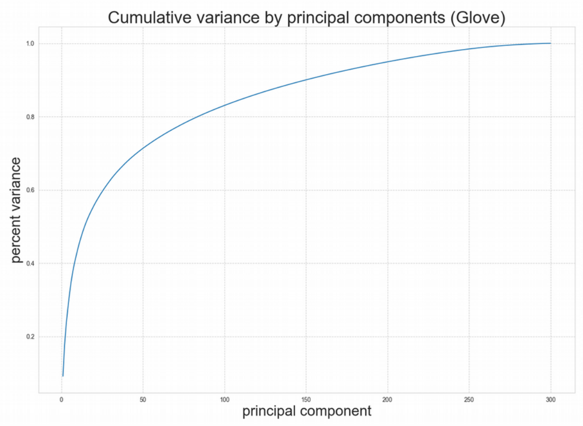
Num clusters	**pos_rate (precision)**	**total_pos_percent (recall)**
4	0.163753	0.734255
8	0.227096	0.571554
12	0.272639	0.541987
15	0.319149	0.459237
20	0.375439	0.374388
25	0.399398	0.371239

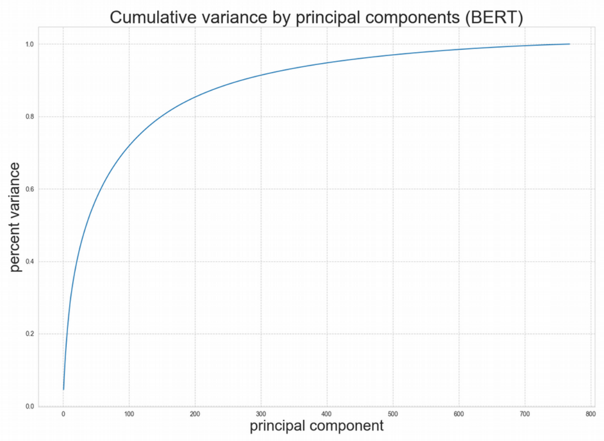
Clustering results for BERT embedded features

Principal Component Analysis:

Principal component analysis was applied to understand the variance explained by the data and their spatial orientation. On the GloVe embedded dataset, the first 100 principal components (out of 300) explained around 81.8% of the total variance and for the BERT embedded features the first 100 principal components explained around 71.8% of the total variance. In out modeling approaches the effect of applying feature reduction by using PCA, by transforming the original feature space into the another feature space composed of the top ‘k’ principal components was studied as well.

Below we can see the plot showing the total variance explained by top ‘k’ principal components for GloVe and BERT embedded features





Modeling results:

We began by building a naive bayes classifier to get a baseline estimate and followed it up with other predictive algorithms. Naive Bayes method gave us a baseline performance of AUC-ROC = **0.8646.** This was followed up with other predictive modeling algorithm. Predictive modeling using logistic regression, SVMs, neural networks and random forests are ran and the results obtained are shown below. Since the data set is heavily unbalanced (positive examples account for 6% of the total examples), accuracy might not be a good metric. The AUC value of the ROC curve and the precision-recall values should give us a better sense of the model fit. Below we have tabulated results which shows the ROC-AUC and precision-recall values for various prediction cutoffs (in case of logitsic regression and neural network models) and for various regularization parameter values for SVMs. A 0.80 to 0.20 ratio of train to test data split was used.

Logistic Regression:

The logistic regression algorithm was applied to the dataset obtained using the word2vec, BERT and the PCA reduced versions of the same. The ROC-AUC performance of these data variants are shown below. We can see that the features obtained using BERT embeddings have a better AUC score but they come at the cost of a slightly increased variance. We can also observe the fact that the PCA transformation on the data to a lower dimensional space didn't help in improving the model performance (We show the results for PCA with features transformation with top 100 principal components). We can further look at the precision-recall scores for varying prediction threshold for probability of positive class to make predictions and get a sense of the model performance. Below is the precision/Recall performance for logistic regression model with L2- regulaization. We can also observe that the logistic regression model on the BERT dataset gives a better precision-recall performance in comparison with the model built using the GloVe embeddings. (*values are precision/recall for various prediction probability cutoff*)

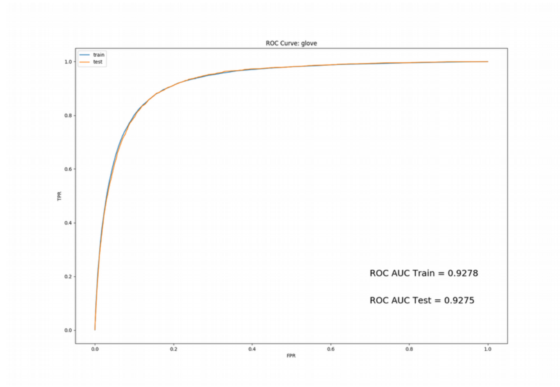
	Glove		BERT	
Features	All	PCA 100	All	PCA 100
AUC Train	0.928	0.919	0.955	0.939
AUC Test	0.927	0.919	0.937	0.932

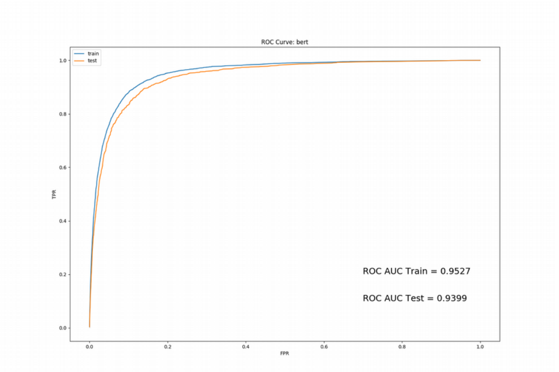
logitsic regression ROC AUC score

Embedding	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Glove	0.33/0.80	0.45/0.60	0.52/0.46	0.57/0.36	0.61/0.28	0.62/0.20	0.65/0.15	0.67/0.09	0.72/0.05
BERT	0.37/0.81	0.49/0.70	0.55/0.60	0.59/0.50	0.63/0.43	0.67/0.35	0.70/0.25	0.74/0.16	0.77/0.08

Precision and recall scores for Logistic regression by prediction cutoff probabilities for GloVe and BERT

Below we have the ROC plots for logistic regression models built using GloVe (left) feature set and BERT (right) feature set on both train and test data set. We can see the slight variance associated with BERT feature set by the small gap between the curves for train and test data set.





NOTE: In the subsequent discussions we just mention the ROC-AUC score and not show the ROC plot for brevity.

Support Vector Machines:

SVM algorithm when applied to the dataset variants had a very comparable performance. Given that we do not get the prediction probabilities directly from an SVM model we can't directly come up with an ROC-AUC score to compare with the logistic regression model. However we can compute prediction probabilities using 'Platt scaling' approach. The ROC-AUC computed using those values, resulted in value lesser than the one obtained using logistic regression model. Over to the right we have the results from SVM with RBF kernel on the GloVe embedded dataset for varying regularization parameter values. We can see that as we increase the regularization parameter, we can see an increase in recall rate. Computationally SVMs took much longer to complete than other algorithms.

Embedding	C = 10	C = 100	C = 200	C = 500	C= 1000
Glove	0.69/0.09	0.70/0.22	0.71/0.25	0.72/0.29	0.72/0.32

Precision and recall scores for SVM (RBF kernal) by regularization parameter

Neural Networks:

Neural Network Models were tried on the datasets generated and their performance were studied between the word2vec embedded feature sets and the BERT embedded feature sets. It can be observed that the BERT based models result in a better bias performance (lower bias) but the variance of the models increase. We need significant regularization to reduce the variance of the resulting model. The models built using the word2vec embedded features (GloVe, GoogleNews etc…) show very low variance (The ROC_AUC values between train dataset and the test dataset are close to each other) even without high regularization.

Below is the precision/Recall performance for neural network models with 1 hidden layers with ReLu activation function for the hidden layers and sigmoid activation for the output layer. For the GloVe based model a regularization value of 0.0001 is used and for the BERT model, a regularization value of 0.001 is used. The training is run for 100 epochs as the model appeared to converge before that. Below we can see the precision-recall results for the models. For the GloVe feature set we show a model with 64 hidden nodes and for the BERT feature set we show one model with 64 hidden nodes and another one with 300 hidden nodes.

Embedding	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Glove (64)	0.34/0.88	0.44/0.79	0.51/0.70	0.57/0.61	0.63/0.52	0.66/0.41	0.73/0.30	0.77/0.16	0.86/0.04
BERT (64)	0.31/0.88	0.38/0.81	0.45/0.74	0.50/0.67	0.54/0.60	0.60/0.50	0.66/0.37	0.72/0.21	0.81/0.06
BERT (300)	0.35/0.84	0.45/0.72	0.52/0.63	0.58/0.54	0.63/0.44	0.69/0.34	0.79/0.10	0.80/0.04	1.0/0.004

Precision and recall scores for Neural Networks by prediction cutoff probabilities for GloVe and BERT

The train and test set ROC-AUC score for the above models are Glove(64) = **0.953/0.947;** BERT (64) **= 0.956/0.938;** BERT (300) **= 0.958/0.940** respectively. We can see that the ROC-AUC score are pretty close to each other, but the GloVe model has the best score on the test set. If we study the precision-recall scores, the BERT (300) model has the best precision performance and the BERT (64) has the best recall performance. By comparing the 2 BERT models we can see that for the same training data set and regularization setting, increased number of nodes gives better precision but loses some recall, whereas the network with lesser number of nodes does better on recall with a loss in precision. The GloVe (64) model is a middle ground between the other two. Depending on the use case we can choose the model that works best or tune them to better suit our use case.

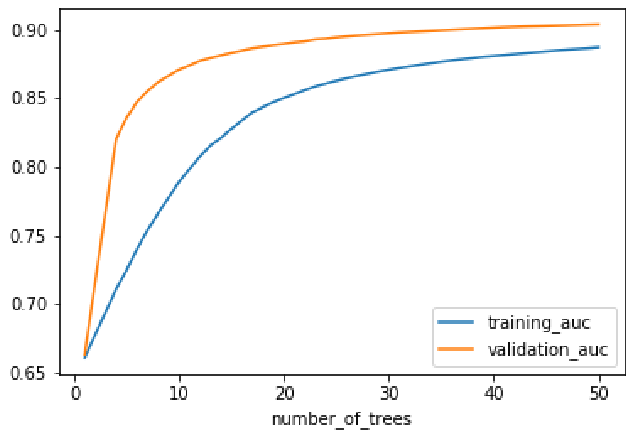
Different activation functions were tried for the hidden layers namely, Tanh, ReLu and Leaky ReLu. The performance weren't that much different. Model variants with an extra hidden layers are tried and the performance is pretty much the same (based of ROC-AUC). Dropouts helped with networks with 2 layers and by trial and error it was found that a dropout factor of 0.2 works best by a small degree. For leaky ReLu, a leakage factor of 0.01 works best. Regularization didn't affect the model performance for GloVe feature sets by a significant extent. But for the BERT feature set, regularization in very critical as it was possible to get an ROC-AUC score of close to 1.0 on the train set but with a large model variance. Hence stricter regularization was needed to bring down the model variance. The table above shows the variation of train and test dataset ROC-AUC score for varying regularization parameter values for a single layer 300 node network on BERT feature set. We can see that with a regularization value of 0.00001 the ROC-AUC for the train dataset is almost 1.0 but the ROC-AUC score on the test set drops to 0.879.

Reg	0.00001	0.0001	0.0005	0.001
Train	0.994	0.980	0.965	0.958
Test	0.879	0.917	0.937	0.940

ROC-AUC for NN on BERT by regularization parameter

Random Forests (Ensembles):

Random forests algorithm was applied to the task at hand and their performance was studied, Compared to the logistic regression and the neural network models the performance of the random forests model was rather poor. We used gini coefficient as the split condition. In terms of ROC AUC performance, we were able to get to a value close to 0.9 with 50 trees in the model. But interestingly enough, we observed that the ROC-AUC score of the test data set is higher than the ROC-AUC of the training data. This points to the model underfitting the training data. The gap between the training data and test data ROC-AUC somewhat reduces as we increase the number of trees. But the fact that we observe the trend as described above, points to this class of modeling algorithm underfitting the data at hand. Over to the right we have the ROC-AUC scores for the training and the test data set by the number of trees in the ensemble. The random forests algorithm typically address the issue of variance by building multiple deep decision trees that are uncorrelated to each other (one of the random element of the algorithm), but are bound by the bias of the independent trees. It appears that the independent trees are limited by the bias performance they could achieve and hence this class of modeling algorithm falls short in comparison with the others that we discussed before. We have 300 features that are numerical in this problem. And it appears that such a feature space didn't suit the random forests algorithm well enough. It could be the fact that multiple features contribute towards the successful prediction of an example. With the randomness element of the random forests, it could well be that in each of the independent tree we exclude some of the significant features that are contributing towards the successful prediction of the example, and hence affect the bias performance of the model. Or it could generally be the case that a multi-dimensional linear hyperplane could be a good separator of the classes and throwing in an algorithm like random forests that iteratively divides the feature space might not be a good candidate in here. The random forests were tried only on the word2vec based features (GloVe, GoogleNews, WikiNews). Since they struggled on a 300 feature numerical feature set, we didn't try it on a 768 feature numerical feature set in BERT.

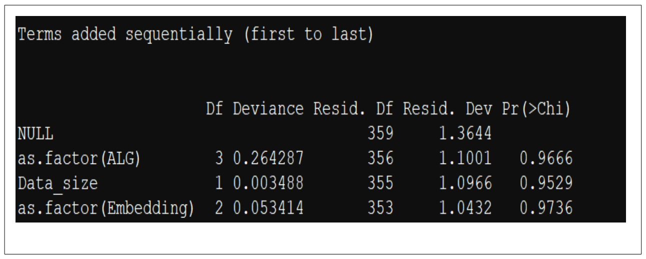


Modeling results conclusion:

Looking at the results above we can see that, between the neural network and logistic regression models, the neural network model is better than the logistic regression by a small degree. We can also see the possibility of improving upon the neural network models on BERT with more data, since we are able to see a good bias performance but a poor return on the variability. Also with neural network models we can see the flexibility to tune the model towards a higher precision or a recall or settle somewhere in a middle ground. SVMs which do not give a probabilistic output (though we can get one using Platt Scaling), give a good precision, but the recall metric is not encouraging. Also the downside of SVMs is the run time. As we increase the regularization parameter the run time increases a lot. The run-time is the fastest for logistic regression and is relatively low for neural networks (100 epochs in our case) in comparison with the SVMs. We tried RBF kernal for SVM as the results from clustering showed trends that would make this a good choice. The linear kernels will be very similar to the logistic regression model and the computational time involved made us hold off on using any sort of polynomial kernels. The ensemble model based on random forests didn't perform well enough in comparison with logistic regression and the neural networks. We could also observe that the application of PCA based feature reduction didn't help in improving the model performance. In fact they underperformed in comparison with the equivalent model when applied on the original dataset.

Analysis of Data Size on model performance:

Another experiment that we were interested in running was to study the effect of training data size on model fit. Sometimes in problem spaces like this, the data availability might be an issue or it might take time to collect more data. So it would be interesting to see if the model performance increase with more data.



```
Terms added sequentially (first to last)


                      Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                                   359     1.3644
as.factor(ALG)         3 0.264287      356     1.1001   0.9666
Data_size              1 0.003488      355     1.0966   0.9529
as.factor(Embedding)   2 0.053414      353     1.0432   0.9736
```

We ran 300 plus models varying with differing amounts of data, algorithms, and embeddings. For algorithms we examined deep learning, gradient boosted trees, logistic regression, and a naïve Bayes classifier. We used blocks of incrementing blocks of 5600 records for 30 models per embedding per algorithm. These were evaluated on 13118 held out records (1% of the total available data). We found neither embedding, data size, nor algorithm choice to be statistically significant predictor of model accuracy compared to a baseline Naïve Bayes, Glove embedded model. We found neither embedding, data size, nor algorithm choice to be statistically significant predictor of estimated model area under the receiver operator curve compared to a baseline Naïve Bayes, Glove embedded model. The Chi-square analysis of variance results are shown here. The results here were based only on the word2vec based features. We didn't run a similar study of BERT based features as we didn't translate the entire raw data into BERT embedded features due to computational constraints.

Other Attempts:

We did try to attempt LSTM where instead of an aggregated embedding for every example in the data we could use the embeddings for each token as a sequence and run them through an LSTM algorithm and study the performance of the resulting model. We needed more data processing and engineering work to be able to do that from where we were at that point. It would be an other interesting thing to pursue and see how the memory and related dependencies affect the model performance.

Future Work:

Below are some of the possible future directions we can take from where we are right now.

- Get BERT embeddings on the entire dataset and study how it affects the model performance. We saw the BERT models achieving a good bias performance but falling behind on variance. May be with more training data, the variance issue could be addressed a bit. We can also explore different ways of extracting the features from a BERT network. In the work we have done, the average of the last 4 layers of the BERT network were used to compute embeddings for an example. May be a different extraction method can be much more suited to this use case. It would be interesting to study the performance of the modeling approaches under various different BERT feature extraction procedures.
- Pursue LSTM and other recurrent neural network approaches. Studying how the dependencies influence the predictive ability of the models for this use case could be interesting.
- Combine other features outside of the word embeddings and see if they improve the predictive ability as an aggregate. For example entity recognition and features extracted from that could be a different set of features to augment the features from word embeddings.

References:

[1] Xiangxin Zhu; Carl Vondrick; Charless C. Fowlkes; Deva Ramanan, "Do We Need More Training Data?" [Online]. Available: http://www.cs.columbia.edu/~vondrick/bigdata.pdf
[2] Chris McCormick, "BERT Word Embeddings Tutorial" [Online]. Available: https://mccormickml.com/2019/05/14/BERT-word-embeddings-tutorial
[3] Michele Banko, Eric Brill, "Scaling to very very large corpora for natural language disambiguation", Proceeding
ACL '01 Proceedings of the 39th Annual Meeting on Association for Computational Linguistics. Pages 26-33, July 2001

Code link: https://drive.google.com/drive/folders/1YbKIbq0bRvnIS0EbgyckhtB-S1yNEAc6

Work Division:
- **Michael Lanier –** Baseline Naive Bayes, Logistic Regression, Random Forests, GBMs, analysis of data size and embeddings on model fit, LSTM.
- **Sibi Shanmugaraj –** Data Preperation using embeddings, Exploratory analysis, Clustering, Logistic Regression, SVM, Neural Networks and PCA.
- **Nagarjuna Rao Chakka –** attempt on entity recognition for additional features to augment word embeddings.