Project Report

INDE 6360 – Engineering Analytics

Classification of Quora Insincere Questions Detect toxic content to Enhancement of Website Content

By

Rahul Krishna Gunneri

Rishika Reddy Muthyala

Pranodh Reddy Gummadi

Table of Contents

Abstract

The Internet has opened up a new horizon of opportunities. It has answers to all questions, and even if the answer is not found there are sites where you can ask any question, and people from all over the world answer it. Quora is one such question-answer website. Here anyone can ask a question from any domain, which is then answered by others. People misuse this boon and create havoc by asking questions that are not to be asked in a public forum. In this project, we propose to develop a system that recognizes these ”insincere” questions so that appropriate actions can be taken against them. The questions are classified as insincere if they have a non-neutral tone, are disparaging, inflammatory or the questions are in negative sexual aspects and if it is not grounded in reality. In this project, a model of text classification will be developed using Natural Language Processing. Among enormous questions data provided by Quora, this study takes into account 16000 questions as primary data. This data contains 93% of sincere questions and 7% are insincere. Under-sampling is applied for the data to balance the sincere and insincere questions. Furthermore, seven different models were performed and compared with each other for loss and accuracy in classifying the Quora-insincere questions.

1. Introduction

People can use Quora as a platform to learn from one another. According to Wikipedia, Quora started in 2009 from then it started growing rapidly. Now the platform receives more than 300 million unique visits every month, Quora is estimated to be worth $2 billion, and at an average of 614.3 million visits (in 3 months). As a popular website, it's important to maintain good content and content policies to be the best platform. One of the difficulties in such a way to maintain good content is to remove insincere questions. Insincere questions are referred to as not genuinely seeking factual answers. Just like,

1. Political or religious accusations.

2. Questions that tell you the only answer they want, "Don't you agree with me".

3. Questions answered by the same user - they didn't need us.

4. Questions hidden by authors to avoid responsibility.

This data which is classified as insincere and genuine data was released by Quora 4 years ago and is defined as "Quora Insincere Questions Classification-Detect toxic content to improve online conversations."

In this study, we took a small part of the data provided by Quora. The data has been trained and tested using different neural network models for better classification of insincere questions.

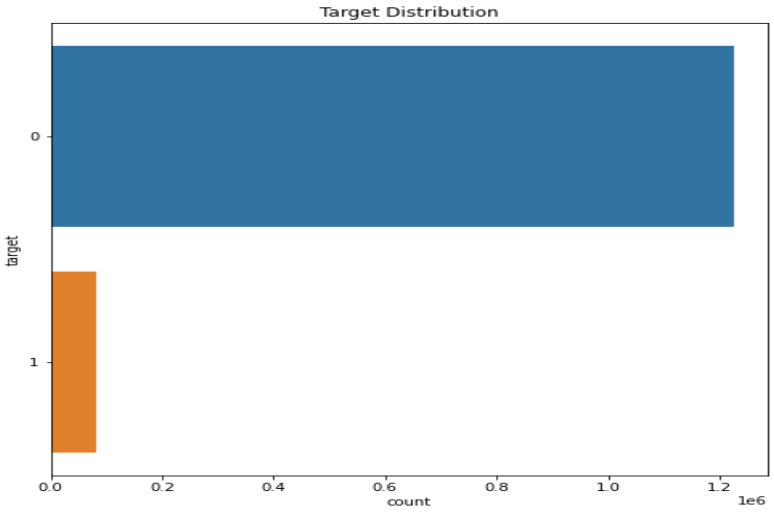
2. Data

We got the “Quora Insincere Questions Classification” dataset from Kaggle.com, This data is made available by Quora itself. This data contains 1.3 million data.

Data is collected from Kaggle which is a legit resource and ensured the quality of data is good. Initially, we have taken the data, visually inspected the classes of the target value, and verified if there is any missing or inconsistent data in the table. Data quality is ensured by other statistical analyses.

Our data contains 3 constraints i.e., qid, question_text, and target.

- qid - unique question identifier
- question_text - Quora question text
- target - a question labeled "insincere" has a value of 1, otherwise 0

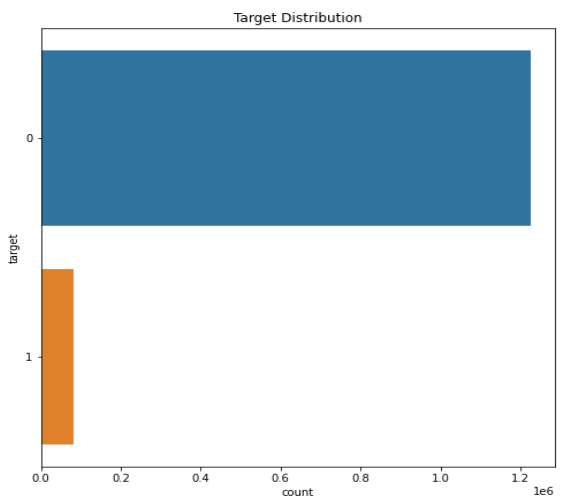


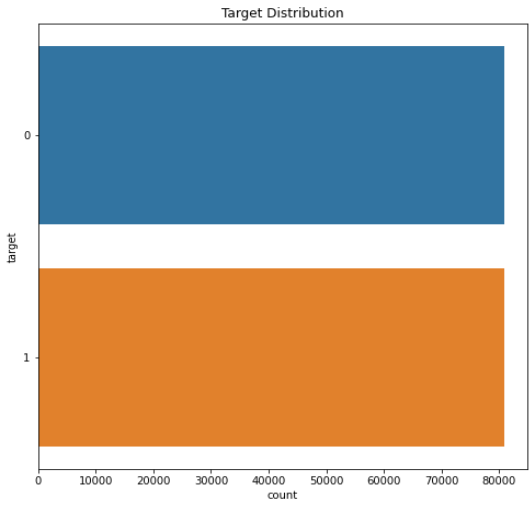
2.1 Data Pre-Processing

Pre-processing is the important step in any data science project. It holds very significance especially in text classification. Actually, Text pre-processing improves the performance of an Natural Language system. For tasks such as sentiment analysis, document categorization, document retrieval based upon user queries, and more, adding a text pre-processing layer provides more accuracy. The Pre-processing technique used are:

1. Under-Sampling
2. Tokenization
3. Stop words removal
4. Lemmatization

From the statistical analysis, we can say that the data is imbalanced as it has 93% of the data in class of ‘sincere’. Hence there’s a need to resampling of data, we under-sample the major class to the same amount of data under class of ‘insincere’. the total data boils down to 1,61,620 questions with equal weight. We under-sampled the class insincere by randomly choosing the question from the same class.



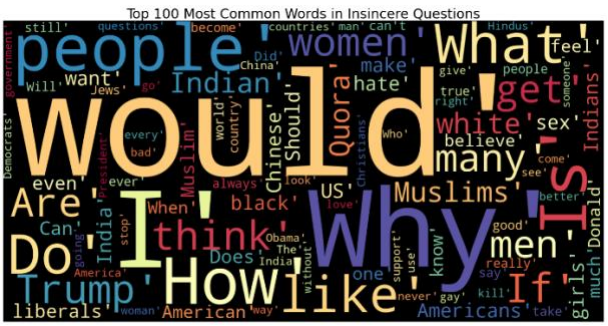


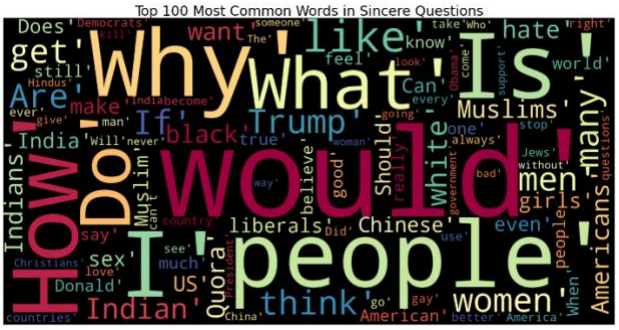
Stemming is the process of converting all words to their base form, or stem. Normally, a lookup table is used to find the word and its corresponding stem. Many search engines apply stemming for retrieving documents that match user queries. Stemming is also used at the pre-processing stage for applications such as emotion identification and text classification.

Lemmatization is a more advanced form of stemming, which involves converting all words to their corresponding root form, called “lemma”, While stemming reduces all words to their stem via a lookup table, it does not employ any knowledge of the parts of speech or the context of the word. This means stemming can’t distinguish which meaning of the word is intended in the sentence.

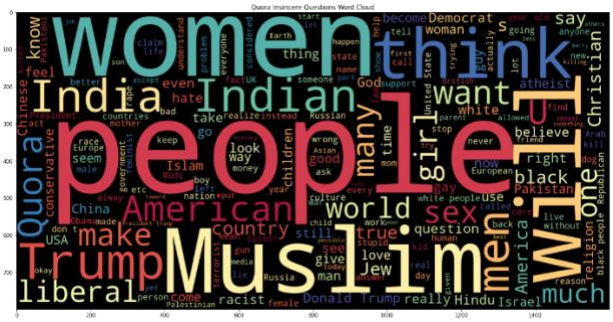
question_text	target	cleaned_text
Has the United States become the largest dicta...	1	united state become largest dictatorship world
Which babies are more sweeter to their parents...	1	baby sweeter parent dark skin baby light skin ...
If blacks support school choice and mandatory ...	1	black support school choice mandatory sentenci...
I am gay boy and I love my cousin (boy). He is...	1	gay boy love cousin boy sexy dont know hot wan...
Which races have the smallest penis?	1	race smallest penis

After performing the above processes, we are cleaning the statements by removing all the punctuations and making sure that the entire text is in lowercase. We are adding tokenization and removal of stop words from the text. Here we are cleaning the statements by removing all punctuation marks and making sure that the entire text is in lowercase. we are adding tokenization and removing stop words from the text. Below stop words cloud, which indicates the most common words in the insincere and sincere questions.





After removing the stop words, we collect word cloud again to see the significance of each word in the sincere and insincere classification. These words are the significant features in both insincere and sincere class.



After performing under-sampling, we split the data into train and validation. Training data size is of 16162 (i.e., 0.1 of under sampled data) and validation data size is 1454 questions.

3. Overview of the model

A neural network consists of 4 layers, the first layer is the built-in pre-processing layer. They are many models which are used for pre-processing of text classification datasets. These models are trained on huge datasets by Google. In our project, we have opted for 7 built-in pre-processing models. They are google news swivel 20 dim, google news swivel 20 dim finetuned, neural network language model with 50-dimension, neural network language model with 50 dimensions finetuned, neural network language model with 128 dimensions finetuned, universal sentence encoder and, universal sentence encoder large models.

One can use these built-in TensorFlow models by loading them using the TensorFlow hub library. Just by inputting the URL of the model. We can load these amazing complex models for our pre-processing layer.

For the 2 hidden layers we are using the activation function as ‘relu’, we are considering 256 dimensions and 64 dimensions for the second and third layers of the model. The output layer is taking sigmoid as the activation function as our problem is classification one.

A model needs a loss function and an optimizer for training. Since this is a binary classification problem and the model outputs a probability. So, we will use binary cross entropy as the loss function. usually, for a classification problem, it’s preferred to use binary cross entropy as we deal with probabilities. it measures the "distance" between probability distributions, or in our case, between the ground-truth distribution and the predictions. We are considered Adam as an optimizer. Optimizer is the algorithm that is used to change the attributes of the neural network in order to minimize the value of the loss function. Optimizer helps to achieve the results at a faster pace. Adam algorithm combines the best properties of AdaGrad and RMSProp algorithms to provide an optimal algorithm that can handle noisy problems efficiently. Because of this adam is highly rated as the default optimizer method for neural networks. Since we are fine-tuning pre-trained models, which overfit our data very easily with a high learning rate. So, we are using a low learning rate of 0.0001.

Train the model for 100 epochs in batches of 900 samples. This is 100 iterations over all samples in the question_text and target tensors. While training, we monitor the model's loss and accuracy for the validation set.

We are storing the training process in a variable called ‘history’ and use this variable to get the accuracy and loss values for each epoch. We use the ‘valid_df’ dataset to evaluate the model. In all the below pre-processing models the features are nothing the entire question. We are stopping the epochs when the model training stops when the validation loss quantity has stopped decreasing for 2 epochs.

3.1 gnews-swivel-20dim

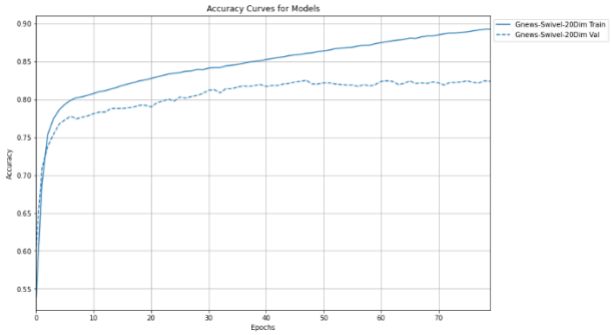
Token-based text embedding trained on English Google News corpus. This model is created using the Swivel matrix factorization method. Swivel termed as Submatrix-wise Vector Embedding Learner, this method will generate low-dimensional feature embeddings from a feature co-occurrence matrix. Swivel performs approximate factorization of the point-wise mutual information matrix via stochastic gradient descent. It uses a piecewise loss with special handling for unobserved co-occurrences, and thus makes use of all the information in the matrix. The google news swivel module takes a batch of sentences in a 1-D tensor of strings as input. The module pre-processes its input by splitting it into spaces. So, we get all the words as our features from the inputted question.

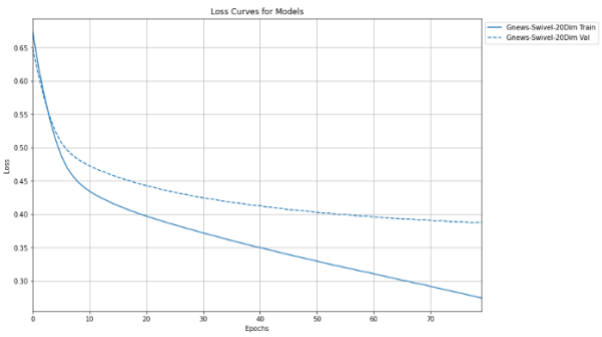
In the model, we use trainable=False, as we want to use the modules to generate embeddings that are of fixed throughout our training process. Once we generate those fixed embeddings, we can use architectural convolutional neural networks.

Model summary:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer (KerasLayer)    (None, 20)                400020

 dense (Dense)               (None, 256)               5376

 dense_1 (Dense)             (None, 64)                16448

 dense_2 (Dense)             (None, 1)                 65

=================================================================
Total params: 421,909
Trainable params: 421,909
Non-trainable params: 0
```

The accuracy kept increasing for each epoch, so the model is learning the parameters. For this model provides 89% and 82% accuracy for training and validation datasets respectively. The below graphs indicate the accuracy and loss variation per epoch for both training and validation.





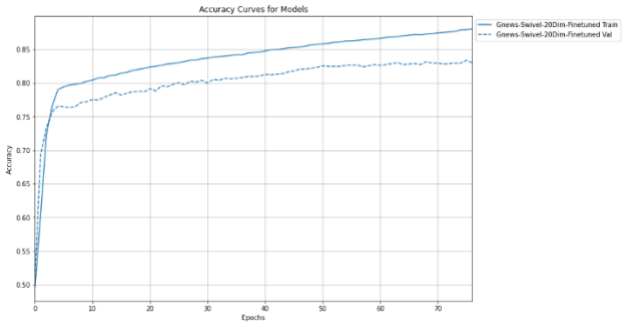
3.2 gnews-swivel-20dim-finetuned

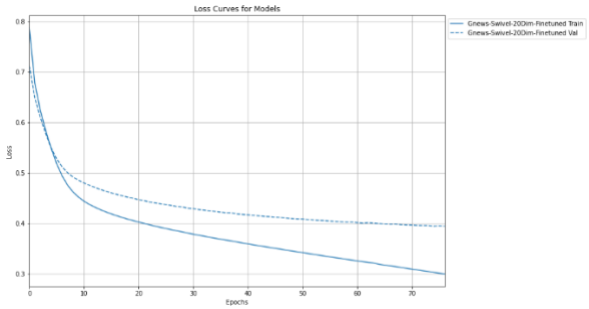
This model is just like the above mentioned google swivel 20dimension built-in model. But the only difference is finetuned by setting the trainable parameter as True, we can fine-tune all the parameters of the model.

Model summary:

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer_2 (KerasLayer)  (None, 20)                400020

 dense_6 (Dense)             (None, 256)               5376

 dense_7 (Dense)             (None, 64)                16448

 dense_8 (Dense)             (None, 1)                 65

=================================================================
Total params: 421,909
Trainable params: 421,909
Non-trainable params: 0
```

This model provides an 88% and 83% accuracy for training and validation dataset respectively. Below graphs indicate the accuracy and loss variation per epoch for both training and validation.





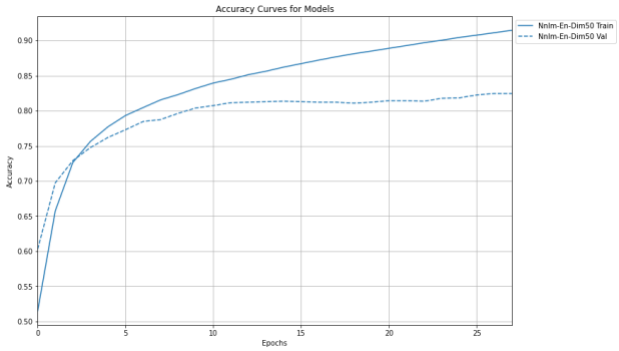
3.3 nnlm-en-dim50

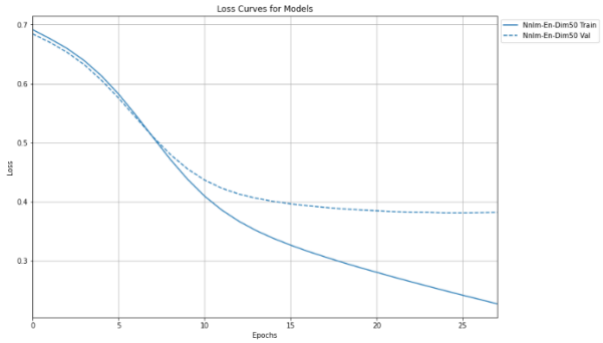
Neural-Net Language Models with pre-built Out-Of-Vocabulary. This model Maps from text to 50-dimensional embedding vectors. A small fraction of the least frequent tokens and embeddings are replaced by hash buckets. Each hash bucket is initialized using the remaining embedding vectors that hash to the same bucket. This module takes a batch of sentences in a 1-D tensor of strings as input. The module pre-processes its input by splitting into spaces.

Model Summary:

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer_1 (KerasLayer)  (None, 50)                48190600

 dense_3 (Dense)             (None, 256)               13056

 dense_4 (Dense)             (None, 64)                16448

 dense_5 (Dense)             (None, 1)                 65

=================================================================
Total params: 48,220,169
Trainable params: 48,220,169
Non-trainable params: 0
```

This model provides an accuracy of 91.5% and 82% for training and validation data respectively. Below graphs indicate the accuracy and loss variation per epoch for both training and validation.





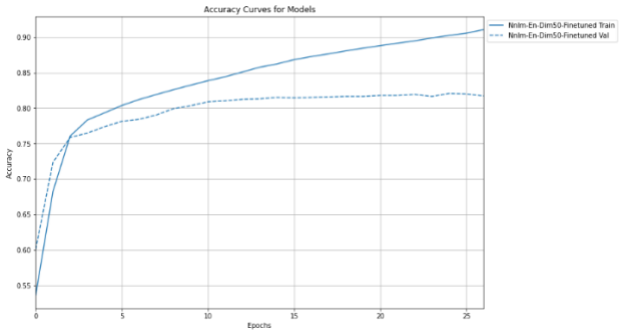
3.4 nnlm-en-dim50-finetuned

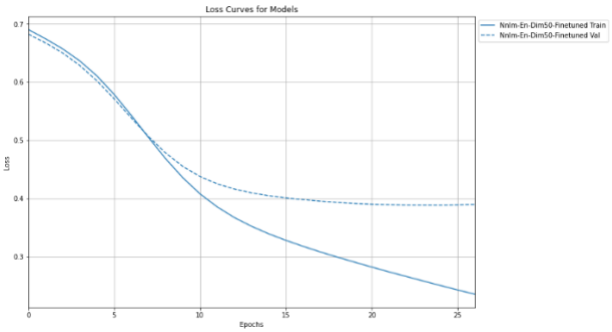
This model is just like the above-mentioned Neural Network Language Model with a 50-dimension built-in model. But the only difference is finetuned by setting the trainable parameter as True, in order to fine-tune all the parameters of the model.

Model Summary:

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer_3 (KerasLayer)  (None, 50)                48190600

 dense_9 (Dense)             (None, 256)               13056

 dense_10 (Dense)            (None, 64)                16448

 dense_11 (Dense)            (None, 1)                 65

=================================================================
Total params: 48,220,169
Trainable params: 48,220,169
Non-trainable params: 0
```

This model provides an accuracy of 91% and 82% for training and validation data, respectively. Below graphs indicate the accuracy and loss variation per epoch for both training and validation.





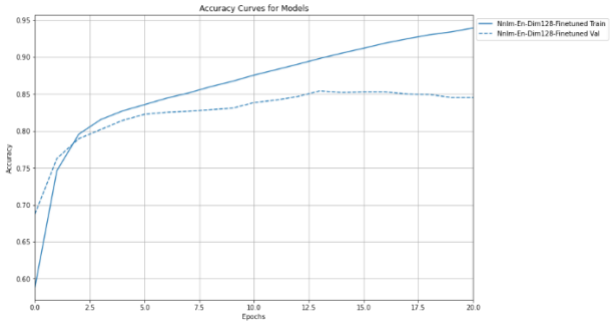
3.5 nnlm-en-dim128-finetuned

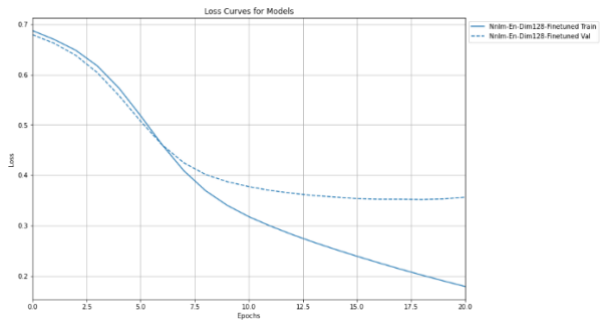
This model is more complex than the above-mentioned nnlm models, as it has more neurons, A larger model with an embedding dimension of 128 instead of the smaller 50. And It's finetuned by setting the trainable parameter as True, we can fine-tune all the parameters of the model.

Model Summary:

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer_4 (KerasLayer)  (None, 128)               124642688

 dense_12 (Dense)            (None, 256)               33024

 dense_13 (Dense)            (None, 64)                16448

 dense_14 (Dense)            (None, 1)                 65

=================================================================
Total params: 124,692,225
Trainable params: 124,692,225
Non-trainable params: 0
```

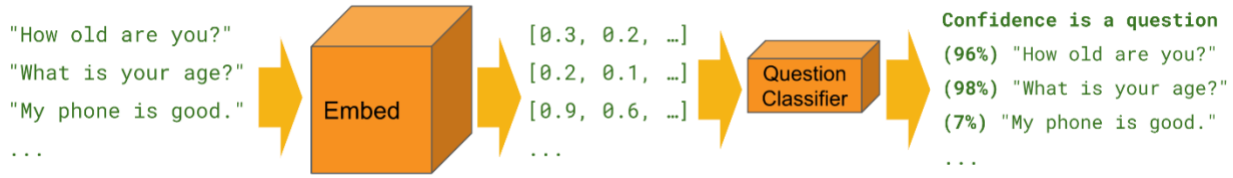
This model provides an accuracy of 93% and 85% for training and validation data respectively. The below graphs indicate the accuracy and loss variation per epoch for both training and validation.





3.6 universal-sentence-encoder

Universal Sentence Encoder encodes text into high-dimensional vectors. This model is trained and optimized for greater-than-word length text, such as sentences, phrases, or short paragraphs. It is trained on a variety of data sources and a variety of tasks with the aim of dynamically accommodating a wide variety of natural language understanding tasks. The input is variable-length English text and the output is a 512-dimensional vector. This model has been trained with a deep averaging network (DAN) encoder.

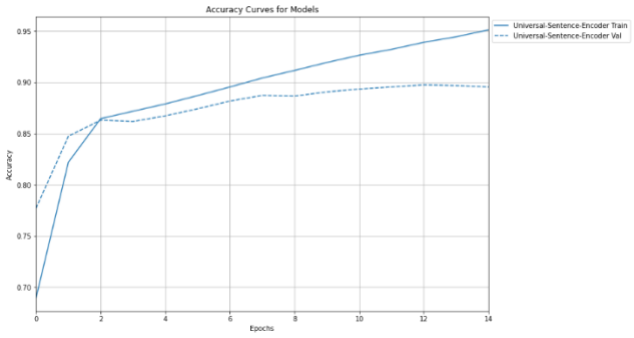


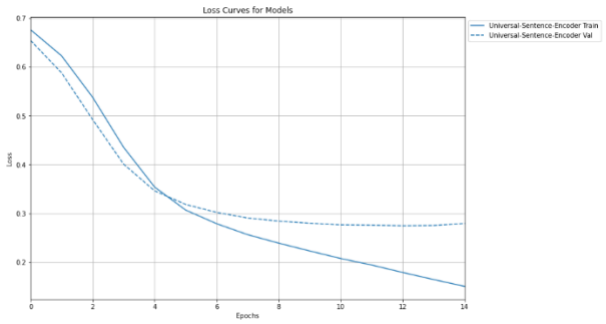
This encoder differs from word-level embedding models in that we train on a number of natural language prediction tasks that require modeling the meaning of word sequences rather than just individual words.

Model Summary:

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer_5 (KerasLayer)  (None, 512)               256797824

 dense_15 (Dense)            (None, 256)               131328

 dense_16 (Dense)            (None, 64)                16448

 dense_17 (Dense)            (None, 1)                 65

=================================================================
Total params: 256,945,665
Trainable params: 256,945,665
Non-trainable params: 0
```

This model provides an accuracy of 95% and 89% for training and validation data respectively. Below graphs indicate the accuracy and loss variation per epoch for both training and validation.





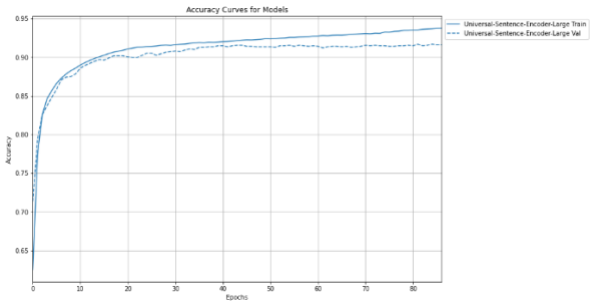
3.7 universal-sentence-encoder-large

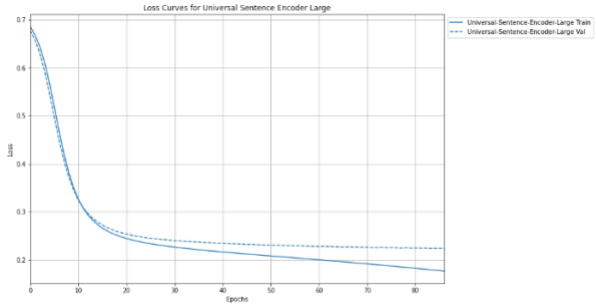
This module performs best-effort text input pre-processing; therefore it is not required to pre-process the data before applying the module.

Model Summary:

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 keras_layer_6 (KerasLayer)  (None, 512)               147354880

 dense_18 (Dense)            (None, 256)               131328

 dense_19 (Dense)            (None, 64)                16448

 dense_20 (Dense)            (None, 1)                 65

=================================================================
Total params: 147,502,721
Trainable params: 147,841
Non-trainable params: 147,354,880
```

This model provides an accuracy of 95% and 89% for training and validation data respectively. The below graphs indicate the accuracy and loss variation per epoch for both training and validation.



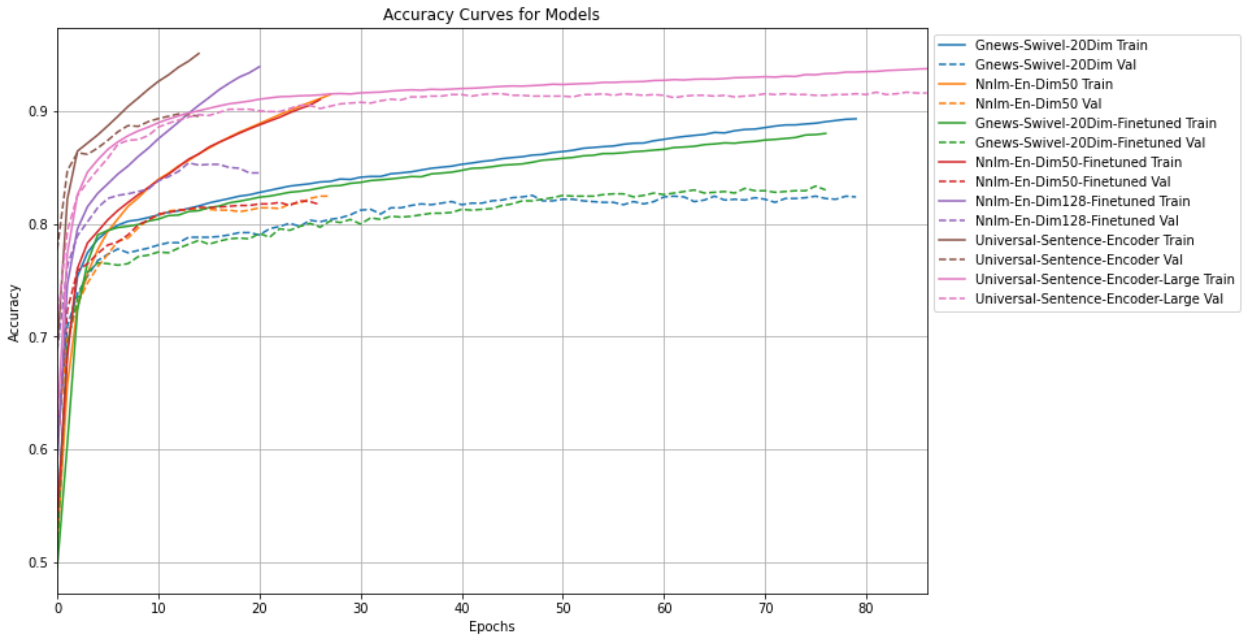


4. Results and Discussions

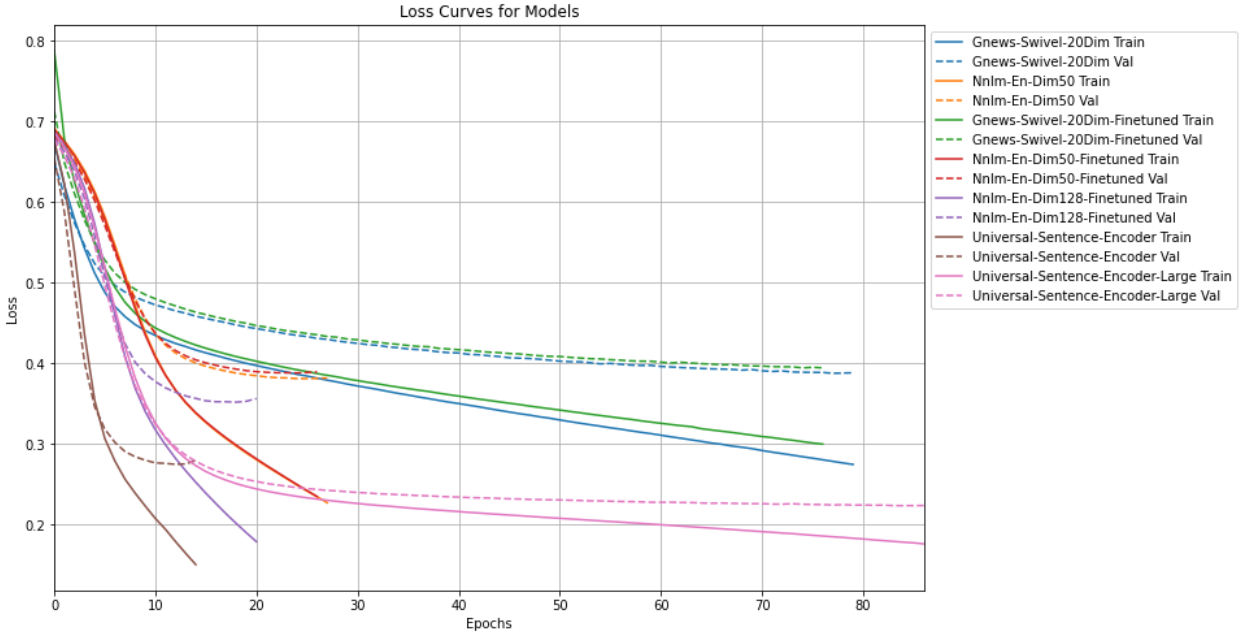
The below table indicates the minimum loss and maximum accuracy for the training and validation data:

Model	Training accuracy	validation accuracy	Training Loss	validation loss
gnews-swivel-20dim	0.8932	0.8253	0.27467	0.38777
gnews-swivel-20dim-finetuned	0.8802	0.8336	0.29994	0.39443
nnlm-en-dim50	0.9147	0.8246	0.22722	0.38114
nnlm-en-dim50-finetuned	0.9103	0.8205	0.23487	0.38823
nnlm-en-dim128-finetuned	0.9394	0.8542	0.17886	0.35195
universal-sentence-encoder	0.9511	0.8975	0.15044	0.27483
universal-sentence-encoder-large	0.9376	0.9168	0.17623	0.22359

The below graphs represent the accuracy per each epoch for training and validation sets:



The below graphs represent the loss per each epoch for training and validation sets:



For the insincere questions classification, we can see that the model using nnlm-en-dim50, nnlm-en-dim50-finetuned, nnlm-en-dim128-finetuned, universal-sentence-encoder and universal-sentence-encoder-large performs provides an training accuracy above 90%, but nnlm-en-50dim, nnlm-en-dim5-finetuned and nnlm-en-dim128-finetuned doesn't perform well on the validation set, even the loss seem to be high. We prefer universal-sentence-encoder and universal-sentence-encoder out of all other pre-processing models, universal-sentence-encoder performs well for training data rather than validation. As we always go for the model which performs better on both training and validation we opt universal-sentence-encoder-large as our pre-processing model for best results.

5. Conclusion

Since the traditional method of text classification is not appropriate for the data, this study presents seven different models to classify the insincere questions in Quora. All seven models performed accurately in classifying the data. Among all the models, the universal sentence encoder large built-in pre-processing model used in our neural network model structure provides the best results with less loss function and great accuracy for insincere question classification.

References

[1] https://amitness.com/2020/06/universal-sentence-encoder/

[2] https://www.nltk.org/data.html

[3]https://github.com/tensorflow/hub/blob/master/docs/tutorials/text_classification_with_tf_hub.ipynb

[4] https://realpython.com/python-nltk-sentiment-analysis/

[5] https://neptune.ai/blog/sentiment-analysis-python-textblob-vs-vader-vs-flair

[6] https://www.analyticsvidhya.com/blog/2021/06/rule-based-sentiment-analysis-in-python/

[7] C. Liu, Y. Sheng, Z. Wei, and Y. Yang, "Research of text classification based on improved tf-idf algorithm," in *2018 IEEE International Conference of Intelligent Robotic and Control Engineering (IRCE)*, Aug 2018, pp. 218–222.

[8] A. Wibowo Haryanto and E. K. and, "Influence of word normalization and chi-squared feature selection on support vector machine (svm) text classification," in *2018 International Seminar on Application for Technology of Information and Communication*, Sep. 2018, pp. 229–233.