Capstone Project

Machine Learning Engineer Nanodegree

Liang Ma

Feb. 20, 2018

Definition

Project Overview

In the internet world, those offensive, rude, or threatening written comments during online conversation may much likely stop people from free discussion and expression. Such toxic comments have severely affected the healthy development of online community. To detect these toxic comments and classify them into different types of toxicity is a prerequisite for any actions taken to prevent such negative behaviours.

Hence, this project is aiming to establish certain model which is capable to accurately and generally identify different types of toxic comments as the solid base for any solution to this problem.

The language of comments targeted in the project is English and the applied datasets are labeled sample comments extracted from real Wikipedia's talk page edits. The datasets include two individual sets as a train set and a test set. Both are imported from a Kaggle competition — Toxic Comment Classification Challenge[1].

Problem Statement

The project attempts to build a classifier which can not only identify toxic ones from arbitrary comments, but categorise them into different types of toxicity in order to facilitate the further actions against certain behaviours.

To achieve the goal, two aspects of techniques are involved in the project: 1. natural language processing or NLP — transforming the raw text of comments to some kinds of features which can be fed to machine learning algorithms; 2. supervised machine learning algorithms that are used to train on the labeled datasets to build the generalised models.

In detail, some tasks will be accomplished in the project:

1. Exploratory data analysis on the datasets

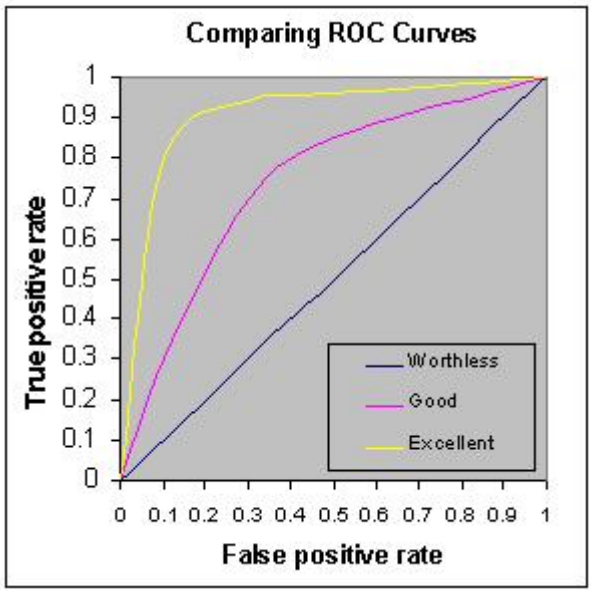
2. Text preprocessing - vectorisation
3. Text preprocessing - tokenisation
4. By using the vectorised features, a linear model is trained as a baseline model
5. By using the tokenised features, a recurrent neural network is trained as another baseline model
6. Adding GloVe word vectors to the embedding layer of the neural network to improve the performance
7. Building an ensemble model based on two baseline models as the refinement solution

Metrics

The metric being used for model evaluation in the project is the average AUC of ROC score for each column.

The AUC of ROC score measures the accuracy by the area under the ROC curve (= Receiver Operating Characteristic curve). The ROC curve is formed by computing two metrics from the confusion matrix — true positive rate and false positive rate — and then combining into one single metric. As shown in figure 1, an area of 1 represents a perfect test; an area of .5 represents a worthless test. A rough guide for classifying the accuracy of a diagnostic test is the traditional academic point system:

Fig. 1



- .90-1 = excellent (A)
- .80-.90 = good (B)
- .70-.80 = fair (C)
- .60-.70 = poor (D)
- .50-.60 = fail (F)

The AUC of ROC metric is a decent measurement of the accuracy for a probabilistic, binary classifier such as logistic regression, especially suitable for such skew datasets as in this project.

Analysis

Data Exploration

Through exploring on the structure of the datasets, some basic characteristics about the data are discovered as following:

1. The train dataset contains 159K rows of comment samples and each row is corresponding one unique comment. There are total eight columns of features: column ‘id’ is about the unique ID of each sample; ‘comment_text’ is of the text itself of comments; the remaining six columns are of the binary values (0, 1) for the six labels that we’ll try to predict:

- ❖ toxic
- ❖ severe_toxic
- ❖ obscene
- ❖ threat
- ❖ insult
- ❖ identity_hate

Here are two samples of the raw data from the train dataset: one is non-toxic comment, another is of certain types of toxicity.

Warning: some samples may contain profane, vulgar, offensive expressions.

Sample 1:

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	0000997932d777bf	Explanation\nWhy the edits made under my usern...	0	0	0	0	0	0

The full text of the comment is:

```
Explanation
Why the edits made under my username Hardcore Metallica Fan
were reverted? They weren't vandalisms, just closure on some
GAs after I voted at New York Dolls FAC. And please don't
remove the template from the talk page since I'm retired now.
89.205.38.27
```

Sample 2:

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
6	0002bcb3da6cb337	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0

2. The test dataset contains 153K rows of comment samples. Each row has an unique ID for an unique comment. There are two columns of features in the test dataset: ‘id’ and ‘comment_text’, whose representation is same as the columns in the train dataset.

Here is a sample of the raw data from the test dataset:

Sample:

	id	comment_text
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...

The full text of the comment is:

```
Yo bitch Ja Rule is more succesful then you'll ever be whats
up with you and hating you sad mofuckas...i should bitch slap
ur pethedic white faces and get you to kiss my ass you guys
sicken me. Ja rule is about pride in da music man. dont diss
that shit on him. and nothin is wrong bein like tupac he was
a brother too...fuckin white boys get things right next
time.,
```

3. Both train and test datasets are quite neat. No abnormalities and no ‘null’ values.

However, as we randomly look at the raw text of several comment samples and go further to statistically explore the length of the text of comments, we’ve found that the length of the text varies a lot:

minimum length: 1 character

maximum length: 13493 characters

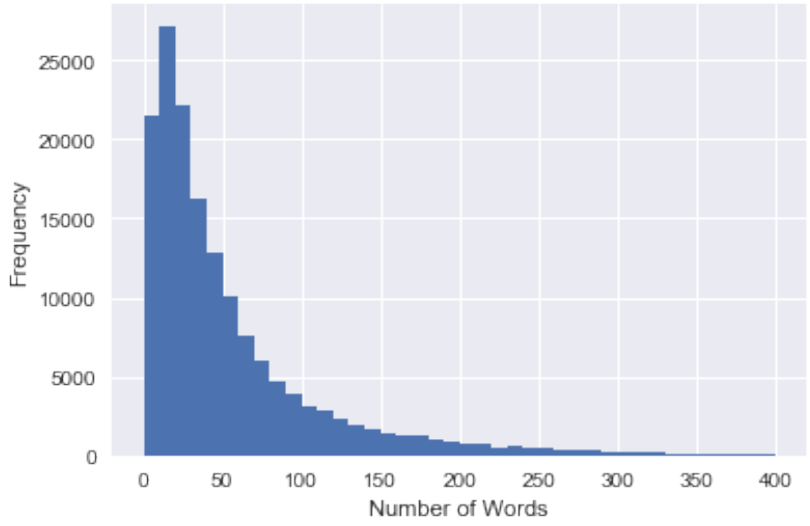
mean of length: 382 characters

This characteristic will be taken into the consideration during the later feature engineering and text preprocessing steps.

Exploratory Visualization

First, I’ve explored the distribution of the number of words in each comment. This feature will be decisive for the parameter setting in the tokenisation process. The distribution plot is shown as figure 2:

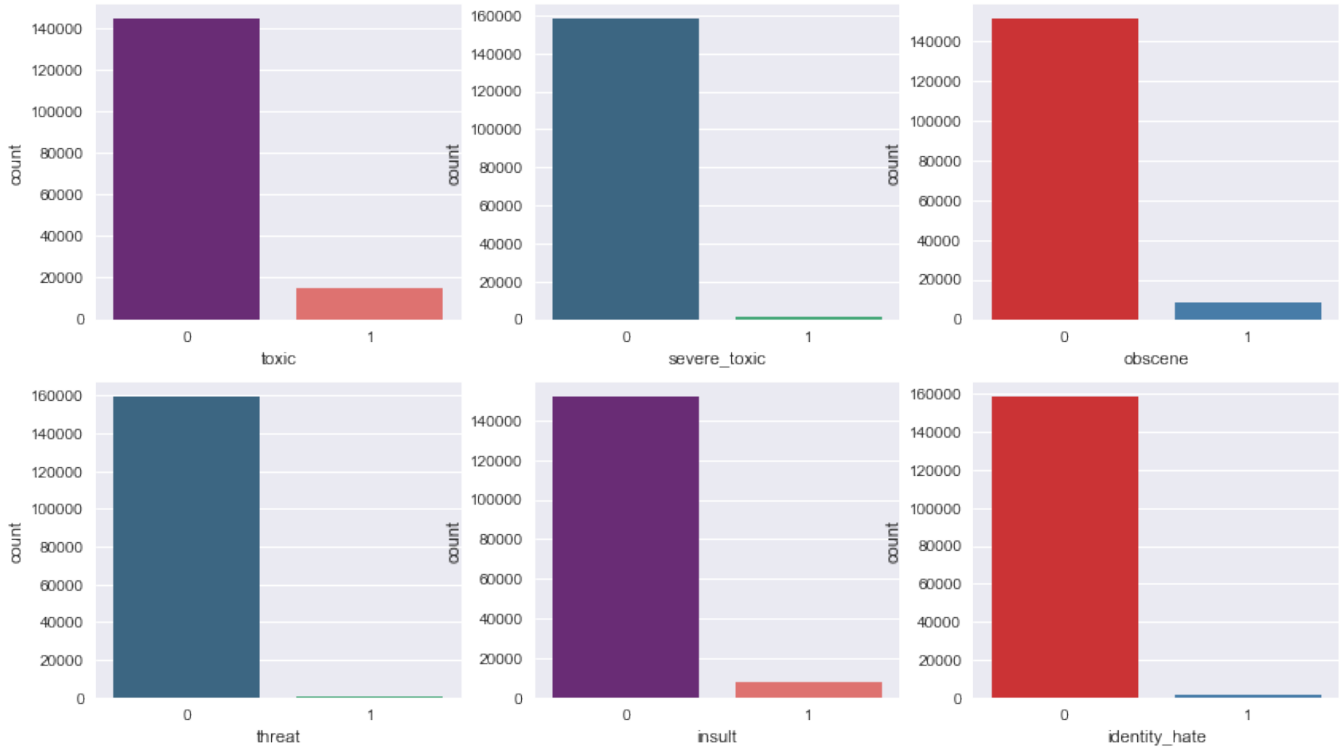
Fig. 2



As we can see, most of the comment length is about 30+ and only a few portion of comments contain more than 200 words. That is the justification for the parameter of 'maxlen' in tokenisation set to 200.

Next, I've explored the value distribution of six target variables as in figure 3:

Fig. 3



From the diagram we can see that the binary values of each target variable have shown extreme imbalance where positive data stands for much ratio. Such skew suggests me of the following benchmark setting method.

Algorithms and Techniques

1. Logistic Regression

Logistic regression is a classification algorithm which models the probability of the default class for binary classification problems.

Logistic regression is a linear regression method, but the predictions must be transformed into a binary values (0 or 1) in order to actually make a probability prediction. This transformation is realised based on the logistic function or the sigmoid function as:

$$1 / (1 + e \text{ ^ } -value)$$

The function can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

Rooting from the sigmoid function, the logistic regression equation can be stated as:

$$p(x) = e \text{ ^ } (b0 + b1*x) / (1 + e \text{ ^ } (b0 + b1*x))$$

Where b0 is the intercept term and b1 is the coefficient for the single input value x. The predicted output of p(x) is a numeric value between 0 and 1. By setting a threshold (e.g. 0.5), the output values more than or equal the threshold will be transformed to 1, while the values less than threshold are transformed to 0.

To deal with the multiclass classification problems as in this project, we can just apply the logistic regression algorithm to each target class as a binary classification problem, then average the evaluation of every prediction to obtain the overall performance.

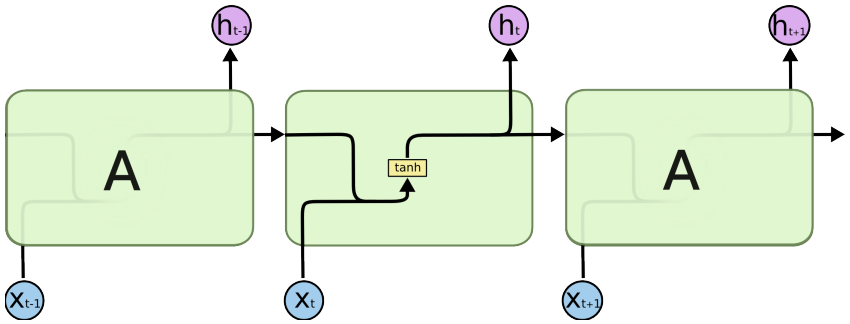
2. Bidirectional LSTM network

Long Short-Term Memory network or LSTM network is a special kind of recurrent neural network (RNN) which works, for many tasks, much better than the standard version.

Recurrent neural networks have the form of a chain of repeating modules of neural network that allows information to be passed from one step of the network to the next. This chain-like nature makes RNNs the natural architecture of neural network to use for handling sequences and lists. The typical application of RNN is just in the natural language processing domain.

Figure 4 shows RNN's repeating module with a very simple structure, such as a single tanh layer.

Fig. 4

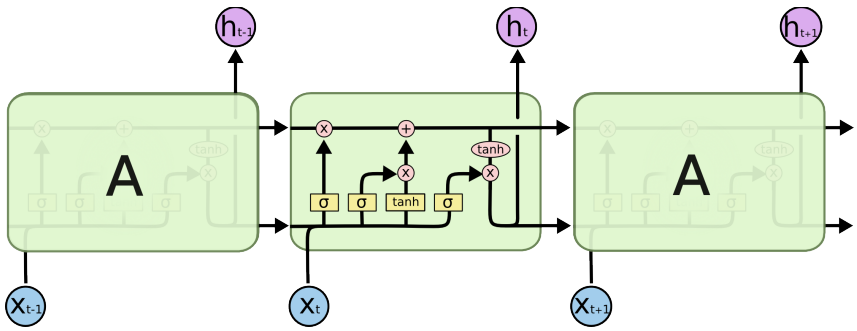


The repeating module in a standard RNN contains a single layer

However, when the gap between the two points of related information becomes large, RNNs will encounter increasingly difficulties in learning such information of long periods of time. This is called the long-term dependency problem.

LSTMs are explicitly designed to avoid such long-term dependency problem by offering a different structure in the repeating module. Instead of having a single neural network layer, there are four, interacting in a very special way:

Fig. 5



The repeating module in an LSTM contains four interacting layers

This change of structure in the repeating module is the core idea behind LSTMs which results in a lot better performance for most tasks.

Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on sequence classification problems[4]. Bidirectional LSTMs train two instead of one LSTMs on the input sequence. The first on the input sequence as-is and the second on a reversed copy of the input sequence. This can provide additional context to the network and result in faster and even fuller learning on the problem.

Benchmark

Instead of using random prediction as the benchmark for performance comparison, I would like to use all-zero values for prediction on the test dataset as the benchmark. This choice is mainly under the consideration of extreme positive skew distribution of categorical values among six target classes of the datasets.

Evaluated by the AUC of ROC metric, the accuracy rate of the benchmark is scored at 0.5000.

Methodology

Data Preprocessing

1. Feature transformation — Vectorisation

Vectorisation processing is applied in order to feed the text of comments to the logistic regression model. The output of the process is a sparse matrix of 30,000 TF-IDF features for each data sample. The relevant procedures are as following:

➢ Build a vectoriser by using the word 1-gram, then train the vectoriser on the raw texts from train and test datasets, and then transform the texts to 10,000 TF-IDF features for each data sample.

➢ Build another vectoriser by using the character 1-gram to 5-gram and then fit and transform the raw texts to 20,000 TF-IDF features for each data sample.

➢ Concatenate the above two TF-IDF features to build a total of 30,000 features matrix.

2. Feature transformation — Tokenisation

Tokenisation processing will produce a list of integral features that represent the sequence of words in the raw text. This integer sequence will then be fed to the LSTM network. The processing involves the following four steps:

➢ Break down the sentences of the raw text into the list of unique words.

➢ Put the words in a dictionary-like structure and give them an index each.

➢ Represent the sequence of words in the form of index.

➢ Fill the shorter sequences and trim the longer ones to the same length as the fixed number of features for each data sample.

3. New features creation

Adding some meta-features that are assumed to be helpful in improving the accuracy of the model is the goal of this preprocessing step. These new meta-features will be computed from the raw text of comments, they are:

➢ length of the comment — based on the assumption that angry people tends to write short messages

➢ proportion of capitals — based on the observation that many toxic comments being ALL CAPS

➢ number of exclamation marks — based on the observation that several toxic comments with multiple exclamation marks

➢ number of question marks — based on the assumption that angry people might not use question marks

➢ number of punctuation symbols — based on the assumption that angry people might use fewer punctuation

➢ number of unique words — based on the observation that angry comments are sometimes repeated many times

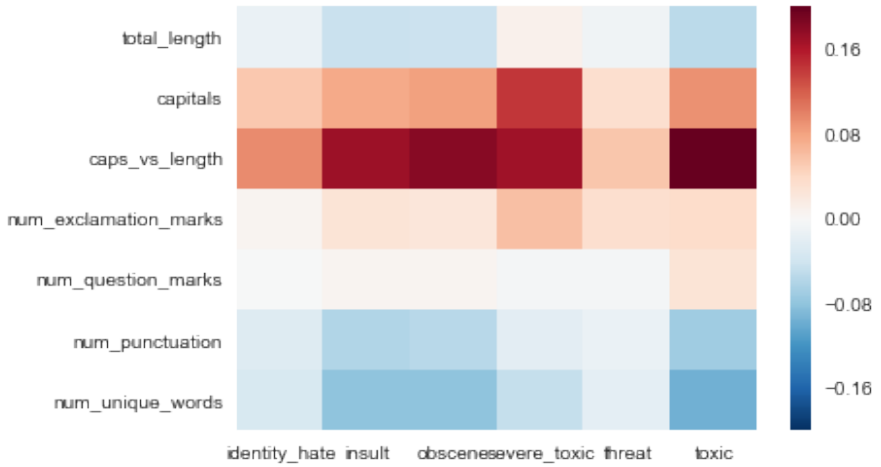
4. Feature Normalisation

The value range of the above new added features varies a lot, hence, I will scale these values within the range of [0, 1] by using a sklearn library - MinMaxScaler.

5. Feature selection

Now I would like to calculate the correlation between the added features and six target variables to determine the importance and relevance of these features. The corresponding heat map of the correlation is showed as figure 6:

Fig. 6



As we can see from the heat map, the following four features are more correlated to target classes:

- ❖ proportion of capitals
- ❖ number of exclamation marks
- ❖ number of punctuations
- ❖ number of unique words

Therefore, these four features will be chosen to concatenate to the previous transformed feature sequences.

Implementation

1. Logistic Regression

The algorithm is applied to train on the vectorised TF-IDF features with 4 extra normalised meta-features. First, an 3-fold cross validation on the train dataset has been implemented. The log loss value for each target class is calculated and stored in a list, and then the comprehensive CV score of the model is finally obtained by averaging the CV scores of six target classes.

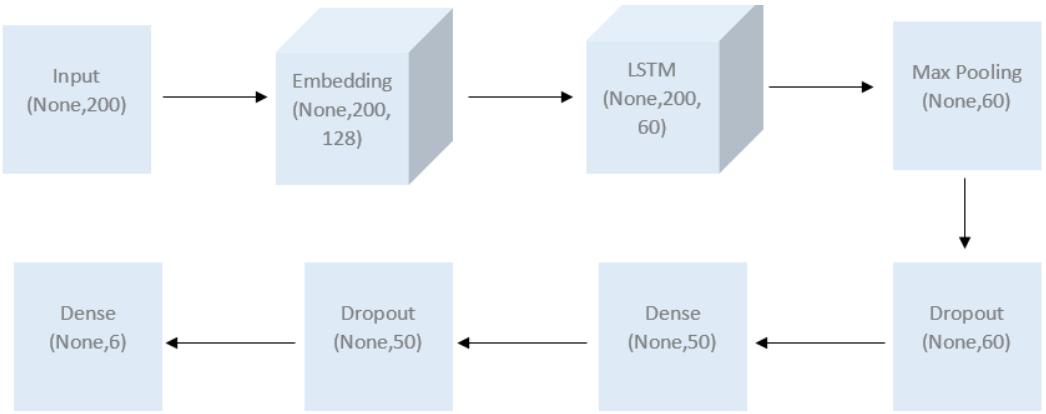
Next, the established classifier is trained on the train dataset and then predict the probability of being each class for the test dataset. The prediction result will then be saved to an CSV file for submission to the Kaggle competition for scoring.

As for the parameter setting for the algorithm, all by using the default values except for the ‘solver’ parameter which is set to ‘sag’ other than the default value ‘liblinear’. The reason for setting ‘sag’ is that the algorithm could run faster on such a large dataset with 30,004 features.

2. Bidirectional LSTM

The architecture of the bidirectional LSTM network is designed as shown in figure 7:

Fig. 7



There are total eight layers designed in the network that starts with defining an Input layer that accepts a list of encoded sentences that has a dimension of 204.

Next, we pass it to an Embedding layer, where we project the words to a defined vector space depending on the distance of the surrounding words in a sentence. Embedding allows us to reduce model size and most importantly the huge dimensions we have to deal with, in the case of using one-hot encoding to represent the words in our sentence. The output of the Embedding layer is just a list of the coordinates of the words in this vector space.

Next, we feed this Tensor into the bidirectional LSTM layer. We set the LSTM to produce an output that has a dimension of 50 and want it to return the whole unrolled sequence of results.

Before we could pass the output to a normal layer, we need to reshape the 3D tensor into a 2D one. We reshape carefully to avoid throwing away data that is important to us, and ideally we want the resulting data to be a good representative of the original data. Therefore, we use a Global Max Pooling layer which is traditionally used in CNN problems to reduce the dimensionality of image data. In simple terms, we go through each patch of data, and we take the maximum values of each patch. These collection of maximum values will be a new set of down-sized data we can use.

With a 2D Tensor in our hands, we pass it to a Dropout layer which indiscriminately "disable" some nodes so that the nodes in the next layer is forced to handle the representation of the missing data and the whole network could result in better generalization.

After a drop out layer, we connect the output to a densely connected layer and the output passes through a RELU function.

Finally, we feed the output into a Sigmoid layer. The reason why sigmoid is used is because we are trying to achieve a binary classification(1,0) for each of the 6 labels, and the sigmoid function will squash the output between the bounds of 0 and 1.

Some important parameters used in my LSTM model are listed as following:

- maxlen = 204 (the dimension of the input layer)
- embed_size = 50
- dropout rate = 0.1 (for both dropout layers)
- units = 50 (for both LSTM layer and the first full connected layer)
- epochs = 2
- batch_size = 32 (for training the model)
- batch_size = 1024 (for predicting the test dataset)

Refinement

Two refinement techniques have been experimented for the final solution. One is to make an ensemble model based on previous two baseline algorithms.

Ensemble is a very powerful technique to increase accuracy for classification problems by remarkably reducing overfit and smoothing separation between classes[5]. The most basic and convenient way to ensemble is to ensemble already existing model predictions. Some simple arithmetical methods like averaging the predictions of individual algorithms will be usually very effective. Thus, I take the mean of the predictions from logistic regression model and LSTM network to form a new prediction for the test dataset. As the result, the ensemble model makes an apparent improvement on the performance of prediction.

Another refinement method is to include the GloVe — Global Vectors for Word Representation — into the embedding layer of LSTM network. Learning such vector space representations of words have succeeded in capturing fine-grained semantic and syntactic regularities[6]. Therefore, this technique shows the outstanding effectiveness in this emotion-related NLP problem of the project. The performance of LSTM network has been improved significantly even though I've used just a small GloVe word vectors with 50 dimensions.

Thus, the final refinement solution is achieved by taking two steps: adding the GloVe word vectors into LSTM network for learning and then making an ensemble model based on the improved LSTM network and previous logistic regression model.

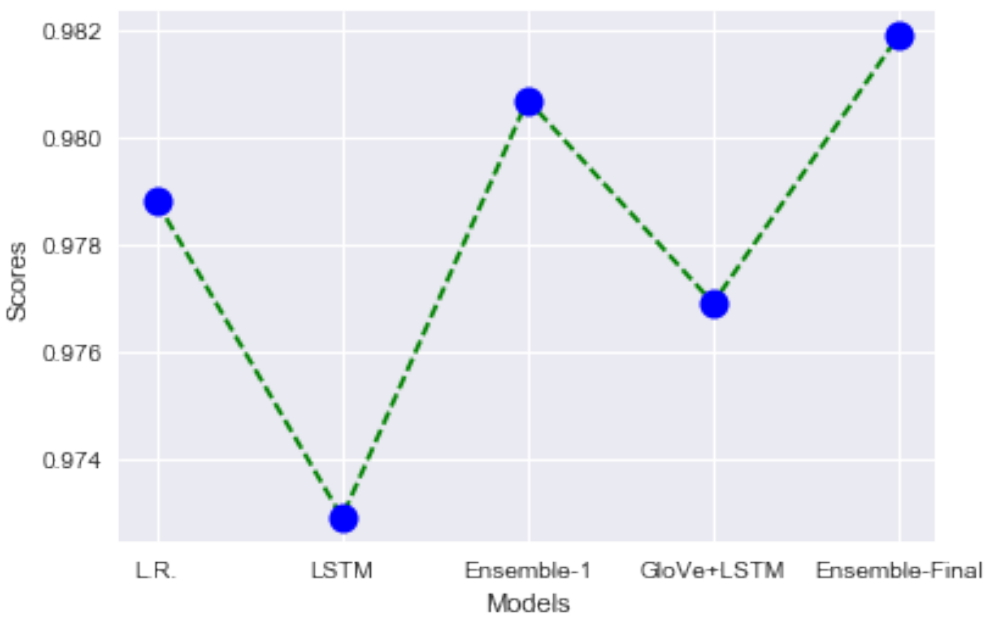
Results

Model Evaluation and Validation

All established models are evaluated by the metric for prediction on the unseen test dataset. In this way, the generalisation of the models is well tested.

The evaluation scores of all established models are shown in figure 8:

Fig. 8



We can see that the two ensemble models rank top two scores and logistic regression algorithm with TF-IDF features stands at the third position. LSTM network without using GloVe word vectors represents a relatively unsatisfactory performance. The final ensemble model achieves the highest score at 0.9819.

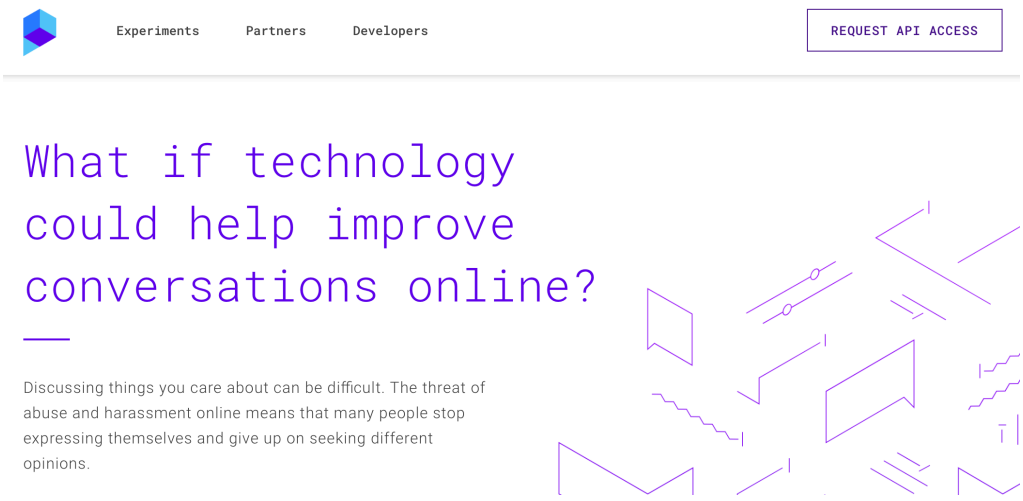
Justification

Comparing with the benchmark score of 0.5000, and even with the baseline model of logistic regression scoring at 0.9788, the final ensemble model is found much stronger and better. Moreover, as we look back to the description of the AUC of ROC metric, the scores between 0.90 ~ 1 represent an excellent performance. Hence, the score of 0.9819 means that the prediction by our final model is accurate enough to make a pretty good distinction between targeted classes and, with strong confidence, our final solution will be significant enough for solving the problem of this project.

Conclusion

Free-Form Visualisation

Fig. 9



A research initiative for toxic content founded by Jigsaw and Google

Nowadays, many AI teams around the world have built a range of publicly available models served to detect online toxic content. Figure 9 just shows one example. However, most of the current models still make errors, and they don't allow users to select which types of toxicity they are interested in finding. I hope that the model built in this project could be a qualified solution being used in the real application.

Reflection

The problem of the project falls into the domain of NLP in which the preprocessing of raw text plays a significant role in the whole solution. I've adopted two different processes along with two different learning algorithms. Obviously, the combination of Vectorisation + Logistic Regression outweighs the combination of Tokenisation + Bidirectional LSTM, even though LSTM network has been upgraded by adding GolVe word vectors. This result might suggest that a meticulous text processing method in pair with a simple linear machine learning algorithm could probably bring us a solution of not only the acceptable performance but relatively fast running time for NLP problems.

Beyond my expectation, the ensemble technique — even just by using the simplest arithmetical method to compute — has obtained the outstanding improvement from individual models. The main effect of ensemble technique is to reduce overfit in

classification problems. Therefore, we can infer that the overfit problem must have occurred in the two baseline models, especially in the LSTM network, even 2 epochs with two dropout layers is enough to overfit.

Improvement

There are a bunch of options and alternatives that can be experimented in the solution to see if they can make improvement. For example, we can try some other algorithms for the individual baseline model like NB-SVM (Naive Bayes - Support Vector Machine); we can use other layers such as GRU and SpatialDropout1D in our recurrent neural network; more different parameter settings in both text preprocessing and learning algorithms can be tested and tuned; some more complicated ensemble techniques can be tried.

It seems that there are endless possibilities for attempting, that there are always better solutions existing. To find them, is a challenge for us, but is more the beauty of machine learning problems.

References

1. Toxic Comment Classification Challenge (https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge)

2. Log Loss Wikipedia http://wiki.fast.ai/index.php/Log_Loss

3. Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/

4. How to Develop a Bidirectional LSTM For Sequence Classification in Python with Keras https://machinelearningmastery.com/develop-bidirectional-lstm-sequence-classification-python-keras/

5. Kaggle Ensembling Guide https://mlwave.com/kaggle-ensembling-guide/

6. GloVe: Global Vectors for Word Representation https://nlp.stanford.edu/pubs/glove.pdf