<CS410 Techreview : Toxic text classification, based on Machine learning approaches>

In 2016, a national referendum resulted in Britain's decision to leave the EU, and Donald Trump was elected as a president of the United States. The two incidents were a far cry from what many experts had predicted beforehand, giving a big shock to people around the world. However, both incidents were the result of public opinion at the time, which could be predicted by the voter reactions in cyberspace. These days, public opinion is expressed mainly through cyberspace. Although online communication platforms can vary from Facebook, Instagram, Twitter and portal websites, 'how public opinion can be identified in a concrete form' has been a matter of interest.

Online comments have been also a hot topic in Korea, which is my native country. This is because there have been many attempts to sway public opinion through comments. For instance, Korean NIS tried to censor comments against the government, and political parties hired people to make favorable comments that would be helpful in the next election.

Moreover, I'm currently making a mobile application with an anonymous bulletin board menu. Since malicious comments do not suit the purpose of using the application, I want to implement a function to filter them. In relation to this, 'toxic comment classification challenge' was held at Kaggle, 2018. (Link : https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/overview/timeline) I think the dataset and approaches to predict the toxic comments would be helpful for my application. Therefore, in this tech review, I will briefly cover the methodologies that achieved high performance at the competition, and discussions that are made among the participants.

Winners of the competition summarized their approach as four subjects below.
Overall, they said that they were able to get optimal parameters by searching the solution space exhaustively. (with 6 GPUs) They also used 8-fold OOF (Out of Fold) and stacking models.

1. Use FastText and Glove as pre-trained embeddings

   They used FastText and Glove embeddings (which were trained against Wikipedia and Twitter, ..., etc.) as pre-trained word embeddings.

2. Use machine translations as train/test-time augmentation (TTA)

   They used machine translation, to augment both the train and test data. Their datasets expanded to French, German, and Spanish translations which were translated back to English. In fact, to prevent the possibility of information leaks (which would leads to overfitting of the model), they made sure that the extended data were on the same side as the original. Using TTA improved their performance a lot.

3. Use rough-bore pseudo-labelling (PL)

They used semi-supervised learning method to minimize a gap between the training data and the test data. Labeling the test data using their best-performing ensemble model and adding the data to the train set improved overall performance of the model.

4. Stacking and Cross-Validation

   They used LightGBM for stacking the models, and they found the parameters of LightGBM with Bayesian optimization. (used with strong L1 regularization)

   For cross-validation, they used 3 metrics : accuracy, log loss, and AUC.

Using the same dataset which was opened for the Kaggle competition, the paper below applied several machine learning methodologies to the dataset and compared their performance.

- Kevin Khieu, Neha Narwal, CS224N : Detecting and Classifying Toxic Comments
  https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1184/reports/6837517.pdf

In the paper, the authors applied several supervised deep learning algorithms to the dataset.
Since the Kaggle dataset provided labels with 'toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate', the authors were able to define their task as a binary classification problem ('toxic' for 1, non-toxic for 0) or a multiple classification problem. Among multiple algorithms, LSTM which was applied to word-level preprocessed dataset achieved highest F1-score to solve the binary classification problem.

(cf. According to Nguyen and Minh-Le.N (2017), LSTM and DeepCNN models would be able to achieve high performances on the task of toxicity classification.)

The Kaggle dataset seems to be useful to train toxic comment classifier for a service that I am currently working with. The tips from the competition winners would be helpful for me to quickly find best toxic comment classifying models for my service. Also, discussions with the competition and the papers regarding this subject would be able to broaden scope of my ideas.

<References>

1. Toxic comment classification challenge, Kaggle, 2018, https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/overview/timeline
2. Kevin Khieu, Neha Narwal, CS224N : Detecting and Classifying Toxic Comments, https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1184/reports/6837517.pdf
3. 1st place overview, Toxic comment classification challenge, Kaggle, 2018, https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/overview/timeline
4. Huy Nguyen & Minh-Le.N A Deep Neural Architecture for Sentence-level Sentiment Classification in Twitter Social Networking, 2017, https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/discussion/52557