Jigsaw Unintended Bias in Toxicity Classification

Detect toxicity across a diverse range of conversations

I.ABSTRACT

Toxic comment classification has become an active research field with many recently proposed approaches. However, while these approaches address some of the task's challenges others still remain unsolved and directions for further research are needed. To this end, we compare different machine learning, deep learning and shallow approaches on a new, large comment dataset and propose an ensemble that outperforms all individual models.

II.INRODUCTION

Social networking sites are the source of most of the recent trend. Almost every human is one way or another attached and affected by these sites. Leading platforms gives people freedom to express themselves through posting of comments and various media. Although that is good in an idea world, where no one is expected to abuse, such freedom but in real world often exactly opposite is observed. Such abuse of freedom often leads to hate spreading, racial slurring or verbal assault. These dangers refrain many people from sharing their opinion or sharing any media for that matter which is harmful for leading platform as well as our society.

The Conversation AI team, a research initiative founded by Jigsaw and Google (both part of Alphabet), builds technology to protect voices in conversation. Their idea was to build an application that could detect and remove or limit verbal abuse which crosses the 'terms of use' of a particular site. Conversational AI uses machine learning, which provides a distinct advantage over other technical solutions, and detect comments that contain toxic content. Our solution makes use of natural language processing tools for preprocessing the data and deep learning approaches were used to train a model that could detect the toxicity of the comments.

III.RELATED WORK

Task definitions

Toxic comment classification is not clearly distinguishable from its related tasks. Besides looking at toxicity of online comments (Wulczyn et al.,2017;Georgakopouloset al.,2018), related research includes the investigation of hate speech (Badjatiya et al.,2017;Bur-nap and Williams,2016;Davidson et al.,2017;Gamb¨ack and Sikdar,2017;Njagi et al.,2015;Schmidt and Wiegand,2017;Vigna et al.,2017;Warner and Hirschberg,2012), online harassment (Yin and Davison,2009;Golbeck et al.,2017), abusive language (Mehdad and Tetreault,2016;Park and Fung,2017), Each field uses different definitions for their classification, still similar methods can often be applied to different tasks. In our work we focus on toxic comment detection.

Multi-class approaches

Besides traditional binary classification tasks, related work considers different aspects of toxic language, such as "racism"(Greevy and Smeaton,2004;Waseem,2016;Kwok and Wang,2013) and "sexism" (Waseem and Hovy,2016;Jha andMamidi,2017), or the severity of toxicity (David-son et al.,2017;Sharma et al.,2018). These tasks are framed as multi-class problems, where each sample is labelled with exactly one class out of a set of multiple classes. The great majority of related research considers

only multi-class problems. This is remarkable, considering that in real-world scenarios toxic comment classification can often be seen as a multi-label problem, with user comments fulfilling different predefined criteria at the same time. We therefore investigate both a multi-label dataset containing six different forms of toxic language and a multi-class dataset containing three mutually exclusive classes of toxic language. Shallow classification and neural networks. Toxic comment identification is a supervised classification task and approached by either methods including manual feature engineering (Burnap and Williams, 2015 ; Mehdad and Tetreault,2016;Waseem,2016;Davidson et al.,2017;Nobataet al.,2016;Robinson et al.,2018) or the use of(deep) neural networks (Ptaszynski et al.,2017;Pavlopoulos et al.,2017;Badjatiya et al.,2017;Park and Fung,2017;Gamb¨ackand Sikdar,2017). While in the first case manually selected features are combined into input vectors and directly used for classification, neural network approaches are supposed to automatically learn abstract features above these input features. Neural network approaches appear to be more effective for learning , while feature-based approaches preserve some sort of explain ability. We focus in this paper on baselines using deep neural networks (e.g. CNN and Bi-LSTM) and shallow learners, such as Logistic Regression approaches on word n-grams

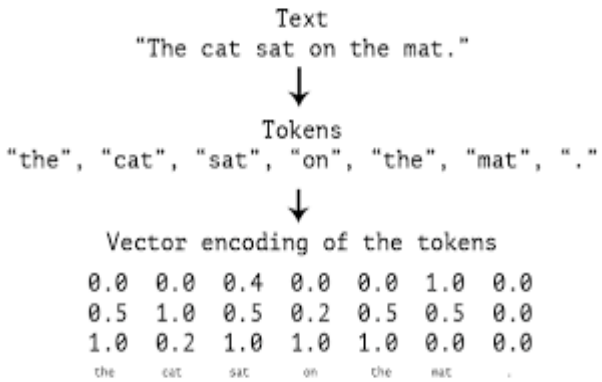
IV.METHODOLOGY

Methodologies of our approach can be summarized in three section: word segmentation, word2vec, and recurrent neural network

Word Segmentation

Text data is a perfect example of unstructured data. To efficiently translate this unstructured data into machine-interpretable information, we separate chunks of continuous text data into a list of words, then encode them into numerical vectors. We then encode each uniqueword to its numerical representation. As shown in Figure 1, mapping this tokenization to segmented text data essentially returns the text as a list of numeric factors, which represent involved words in the vocabulary. This brief explanation serves to provide foundations to the \word2vec" method we will employ.

Figure 1: Tokenization Example



word2vec

Created by a team of researchers from Google in 2013, word2vec is a group of models to produce word embeddings that retains the context of words[11]. Word2vec models are shallow, two-layer neural networks constructed based on the idea that similar words would appear in similar positions in the context. Representing this intuition is Cosine Similarity. Similarities of the given vectors are measured in their inner product space by the cosine angle between them.

$$\cos(\theta) = \frac{u \cdot v}{||u||||v||}$$

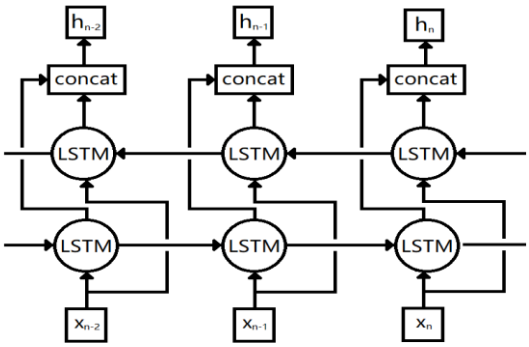
Which in turn, means that larger dot product: u. v = ||u||||v||cos(thetha), indicates more similarity.Word2vec introduced two approaches in calculating the word embedding so that similar word vectors have higher dot product.

Bidirectional Long Short-term Memory

Bidirectional Recurrent Neural Network was first introduced by Schuster et al in 1997.

It increases the amount of input for a neural network. As the name suggests, the model not only takes in information in previous states, it processes data from both past and future states, which further enhances the neural network's ability to understand the context of the input. Comparing to a standard LSTM layer, a bidirectional LSTM adds another set of LSTM cells to process inputs in a reversed sequence. Outputs in both sets of cells are concatenated and feed to the next layer.

Figure 2: Bidirectional LSTM



V.EXPERIMENTAL RESULTS

We have split the data into train set and validation set with 80:20 proportion. We used two methods for training a model with gradually increasing complexity of model. In first phase we decided to use simple classifying algorithms which are computationally light and can serve as milestone for more complex methods.

METHOD 1: Logistic Regression

The Logistic Regression (LR) algorithm is widely used for binary classification tasks Also it is supervised learning algorithm.

Data Processing

The data consist of text comments identifier and its toxicity level that many users has recorded. Text comments contained may symbols, emoji, dates, codes (E.g. UTC) and escape characters like '\n', '\t other than normal English words. We removed white spaces which are more than one space bar input. We used regular expression to remove most of the special characters, escape characters and numbers from comments.

Comments may have many words which do not belong to English vocabulary as they are generally used abuse words by native speakers of that language. As social media is globally accessible, discarding words which does not belong to English would not be a good idea. Thus we did not assess the data for foreign words.

RESULT:

toxic comments

```
Test accuracy is 0.9216666666666666
```

severe_toxicity comments

```
Test accuracy is 0.9883333333333333
```

obscene comments

```
Test accuracy is 0.9516666666666667
```

threat comments

```
Test accuracy is 0.9966666666666667
```

insult comments

```
Test accuracy is 0.9516666666666667
```

identity_attack comments

```
Test accuracy is 0.9933333333333333
```

METHOD 2: Bidirectional LSTM

Data Processing

Word Embedding :

We used advanced text representation using word embedding. Using keras package, the input data is prepared by tokenizing the text, converting words to index and by replacing the index with the pre-trained word vectors. This result in a 2D array representation of input text or comments which will be used to train a Neural Nets. Using the index generated on train, the data for validation and test are similarly converted to a 2D array which will be

used for tuning and cross-validation respectively.

Approch:

Here we use a RNN approach named bidirectional LSTM (bidirectional Long-Short-Term-Memory Network) Our LSTM model takes a sequence of words as input. An embedding layer transforms one-hot-encoded words to dense vector representations and a spatial dropout, which randomly masks 2% of the input words, makes the network more robust. To process the sequence of word embeddings , we use an LSTM layer with 128 units, followed by a maxpooling and averge pooling. Both the pooling outputs are concated into one layer. 2 hidden layers are added with relu activation function.At last we have a dense output layer with sigmoid activation function to predict the output which is our target. Bidirectional RNNs can compensate certain errors on long range dependencies. In contrast to the standard LSTM model, the bidirectional LSTM model uses two LSTM layers that process the input sequence in opposite directions. Thereby, the input sequence is processed with correct and reverse order of words.

RESULT:

```
Epoch 1/1
 - 778s - loss: 0.5262 - dense_7_loss: 0.4196 - dense_8_loss: 0.1066 - dense_7_acc: 0.6937
- dense_8_acc: 0.8546
Epoch 1/1
 - 778s - loss: 0.5076 - dense_7_loss: 0.4055 - dense_8_loss: 0.1020 - dense_7_acc: 0.6954
- dense_8_acc: 0.8549
Epoch 1/1
 - 776s - loss: 0.5018 - dense_7_loss: 0.4007 - dense_8_loss: 0.1011 - dense_7_acc: 0.6957
- dense_8_acc: 0.8550
Epoch 1/1
 - 774s - loss: 0.4981 - dense_7_loss: 0.3975 - dense_8_loss: 0.1006 - dense_7_acc: 0.6959
- dense_8_acc: 0.8550

Complete. Exited with code 0.
```

VI.CONCLUSION

We successfully employed word2vec embedding and recurrent neural network in building a toxic comment classiffcation model and achieved high accuracy with relatively low cost.Gated recurrent units layer proves to be more effcient at training but performs slightly worse than the baseline model with LSTM layer. Comparison studies show that pre-trained embedding vectors obtained from larger corpora do not necessarily improve the performance of our model because they require domain-specific vocabulary to perform accurately enough.

On a closing note, Conversation AI team's intention and effort in building an open source tool to monitor and control online toxicity is commendable. Researchers and discussion platform moderators have already found numerous ways to apply this tool in very creative manners. We hope that with the collaborative help from the machine learning community,the team can continuously improve the performance of Perspective API and help maintaina toxic-free environment for our online discussions.

VII.REFERENCES

EMBEDDING FILES:

https://www.kaggle.com/yekenot/fasttext-crawl-300d-2m
https://www.kaggle.com/takuok/glove840b300dtxt

DATA SET:

https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification/data