Introduction

- Rapid growth of online platforms and forums propagates abusive language and toxic speech
- An individual risks being harassed by other users when participating in online discussions
- Problem: deep learning models that detects types of toxic comments (clean, toxic, obscene, insult, identity hate, severe toxic, and threat)

Dataset

- Google Jigsaw’s Kaggle dataset: “Toxic Comment Classification Challenge” (published in 2017)
- Randomly select 20% from test set to be dev. set
- Separated into train and test sets, both containing approximately 160,000 comments and labels
- Randomly select 20% from test set to be dev. set

Approach

- Existing approaches include “classical methods” such as regression and SVM and deep learning models like CNN and RNN variants
- Text classification problem
- Apply LSTM, GRU, and VDCNN

Conclusion

- Deep learning models are accurate, but have high computational cost
- Adopting a cascading allows us to utilize the efficiency of “classical methods” while drawing from Bi-LSTM model’s accuracy when it is needed
- Lack of clean training data and lack of testing on diverse datasets
- Limited computing power for char-based models
- Future work: refining cascading model, combining deep learning architectures, and explore feature extraction mechanisms for SVM

Problem: Computational Inefficiency

- Deep models are expensive
- Thus, they are difficult to use on a large scale
- We evaluate a forward pass with one document for 10 trials of 100,000 runs each and calculate the mean and standard deviation of run time
- Our best model (Bi-LSTM with attention) is 14 times slower than logistic regression
- Since the baseline performs quite well with 96.7% test accuracy, there is less incentive to adopt a deep learning approach

Potential Solution: Cascading Model

- We propose a cascading model which combines a series of models, optimizing for accuracy and speed in average case.
- Use intermediate steps and confidence scores at each step
- Each subsequent step has higher computation cost
- We test a small cascading model composed of a logistic regression as the first step and Bi-LSTM as the second

Performance of Cascading Model:

- Accuracy is 0.973, higher than that of baseline
- Average latency of 5.18 ms, about 2.5 times slower than the baseline but still 6 times faster than Bi-LSTM
- Only 31% of the comments required the use of the Bi-LSTM

Experimental Results

<table>
<thead>
<tr>
<th>Model</th>
<th>F1 Score</th>
<th>Test Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear regression (Baseline)</td>
<td>0.44</td>
<td>0.967</td>
</tr>
<tr>
<td>Bi-GRU</td>
<td>0.57</td>
<td>0.971</td>
</tr>
<tr>
<td>Bi-GRU (FastText)</td>
<td>0.61</td>
<td>0.974</td>
</tr>
<tr>
<td>Bi-GRU (FastText + Attention)</td>
<td>0.66</td>
<td>0.987</td>
</tr>
<tr>
<td>Bi-LSTM</td>
<td>0.60</td>
<td>0.975</td>
</tr>
<tr>
<td>Bi-LSTM (FastText)</td>
<td>0.62</td>
<td>0.980</td>
</tr>
<tr>
<td>Bi-LSTM (FastText + Attention)</td>
<td>0.66</td>
<td>0.989</td>
</tr>
<tr>
<td>VDCNN-9</td>
<td>0.62</td>
<td>0.975</td>
</tr>
<tr>
<td>VDCNN-17</td>
<td>0.62</td>
<td>0.978</td>
</tr>
</tbody>
</table>

Inference Speed Analysis

- All deep learning models that we tested are able to outperform the baseline
- Bi-LSTM with FastText embeddings and attention produced the highest F1 score and test accuracy
- Using pretrained FastText embeddings leads to a systematic increase in performance, for the following reasons:
  1. FastText is trained on a large corpus (16B tokens), as opposed to our training set (160,000 comments)
  2. FastText generates subword embeddings whereas tokens like “sucklol” would otherwise be treated as unknown
- Using scaled dot product attention also leads to a systematic increase in performance
- Deeper VDCNN appears to produce higher accuracy, but the F1 score remained the same for the two depths

References