Data collection protocol:

- **Model choice and justification**

Our model will be a Bidirectional Long Short-Term Memory Recurrent Neural Network (BiLSTM RNN for short). RNNs, particularly BiLSTMs, are reported to perform best in text-classification according to [1]. Furthermore, in a Kaggle competition launched by Conversation AI, RNN models reached an accuracy score of 98%. Since our focus is on increasing classifiers' robustness against adversarial examples, we chose as a baseline model a classifier that is performing as well as possible on standard datasets. Another motivation for using this model is the availability of open-source examples on tasks like ours (e.g., [2]), and the abundance of libraries facilitating the implementation of such models (Keras, Tensorflow, PyTorch to only name a few). This is criteria is particularly relevant given our restrained time constraints.

- **Adversarial training**

In order to make the model resistant to adversarial attacks, we plan to explore the following methods:

- **Algorithms generating adversarial examples:** the aim of this type of algorithms is to introduce systematic modifications to text data to make it undetectable by computer models but still identifiable by human classifiers. We chose to focus on 'black-box' algorithms, which operate without knowledge of the architecture (and therefore strengths and weaknesses) of the model they are attacking. DeepWordBug is one such algorithm [3]. We plan to look up more algorithms if time permits and/or we face issues using this one.
- **Introducing a limited number of syntactic changes:** in this scenario, we will generate the adversarial examples ourselves by applying 4 syntactic modifications to toxic words. These modifications are adding space between letters, adding random punctuation in the middle of a word, repeating and swapping letters. These methods were found to fool Google's Perspective AI, a state-of-the-art model in toxicity detection with current industrial applications, which shows the effectiveness of these attacks [4]. [4] also proposes augmenting the training set with such examples as a possible fix to the classifier(s)' weakness, despite the computational cost of this approach.
- **Using human attackers:** [5] proposes a strategy called 'build it, break it, fix it' inspired by security testing of software products. Following this method, an initial classifier M1 is built, achieving as high a performance as possible on a regular dataset. Human attackers are then asked to try to break the model (by giving it a toxic sentence it would not detect). The examples generated by the attackers are added to the training dataset, resulting in a new model M2. In the next round, participants should try to fool both M1 and M2. This strategy goes on for a predefined number of rounds and is reported to improve the classifier's ability to detect linguistically advanced toxic expressions (using metaphors or requiring cultural knowledge for instance). We would use the team members as attackers if we were to implement this strategy. However, since our focus is on syntactically modified examples only, and due to time limitations, this method would be our last resort.

The methods above are in preferential order. If time permits, we might try multiple methods and compare their results. We will report on all our findings either way.

- **Training plan**

As explained in our data collection strategy, our input would be Conversation AI's publicly available dataset made of edit discussion comments from Wikipedia. We plan to use 80% of said data for training, 10% for validation, and 10% for testing. This training strategy is conforming with the guidelines given in the Kaggle competition where the dataset was presented. At an initial stage, we plan to train the model on a regular dataset to match its reported accuracy. We will then assess how the model, without additional training, performs on an adversarial dataset, obtained through one of the methods mentioned in the previous section. This initial accuracy would be used as a baseline to measure any subsequent improvement in the model's resistance to adversarial attacks. We would then retrain our model using a portion of the adversarial dataset and reassess its accuracy. This process will be repeated a predefined number of rounds and all results/observations will be reported adequately.

- **Evaluation**

In all runs of the program, we will use accuracy as a measure of performance. More specifically, we plan to use ROC-AUC measure as it provides a more robust estimation of accuracy. This measure was also proposed as an evaluation metric in Kaggle's competition. Overall, we hope to present an adversarial training method that increases the model's ability to detect adversarial toxic language and mention any observations on the training strategy and/or model we make along the way.

Resources

[1] Kohli, M., Kuehler, E., & Palowitch, J. (2018). Paying attention to toxic comments online.

[2] Pukar Acharya, Toxic Comments Classsification, (2018), GitHub repository, https://github.com/iampukar/toxic-comments-classification/

[3] Gao, J., Lanchantin, J., Soffa & Qi, Y. (2018). Black-box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers

[4] Hosseini, H., Kannan, S., Zhang, B., & Poovendran, R. (2017, February 27). Deceiving Google's Perspective API Built for Detecting Toxic Comments. arXiv:1702.08138

[5] Dinan, E., Humeau, S., Zhang, B., Chintagunta, B., & Weston, J. (2019, August 17). Build it Break it Fix it for Dialogue Safety: Robustness from Adversarial Human Attack. arXiv:1908.06083