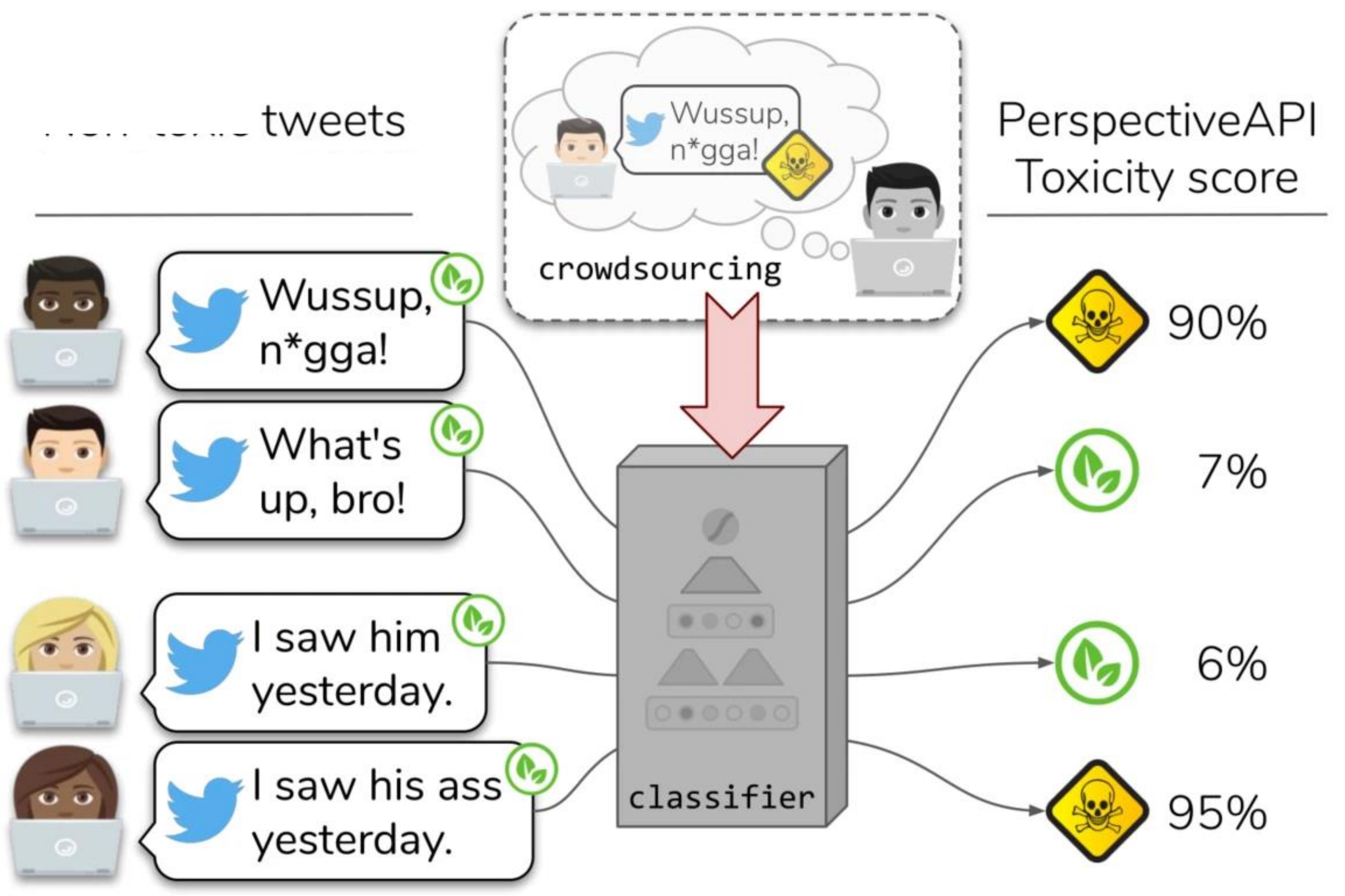


Research Motivation

Automated detection using a computer algorithm is one of the potential alternatives to human moderators against hate speech spread on social network. This project **aimed at understanding the mechanism** behind this automation by building a deep-learning detector with the dataset from the open web. Thus, it was duplication work fundamentally. It attempted to state-of-industry **RNN approach** for classification task on a Twitter dataset.

Conclusion

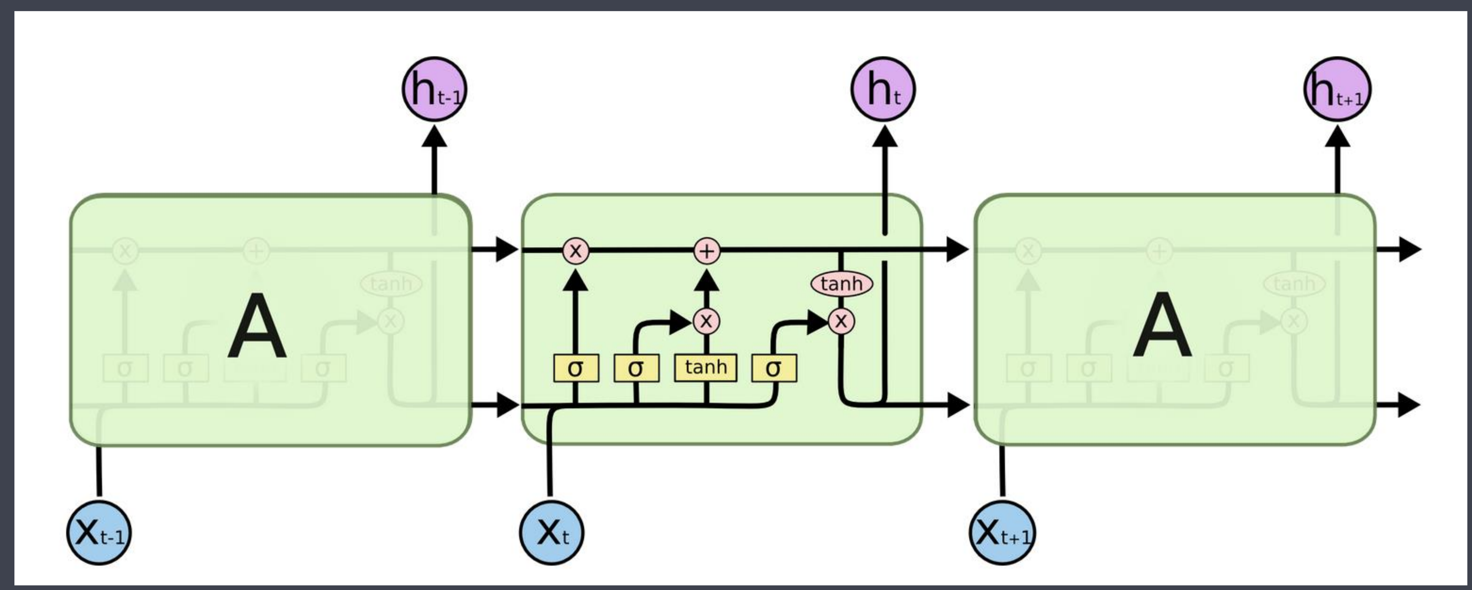
As a work of reproducibility, this project managed to adapt a simple LSTM model on a different dataset labelled by Davidson. The project achieved an accuracy rate of 84.68%, which is a state-of-the-art result. The implementation process provided the author with an opportunity to dive deeper into the theory behind text classification and the rationale behind automated hate speech detection.

Automated Hate Speech Detection with LSTM Classifier

Team Project by Hu, Xiaoyan for the *Course: Natural Language Processing with Deep Learning, Hertie School*

Proposed Method

- **Dataset:**
  25k tweets containing terms collected by Davidson et al. (2017) from the lexicon and had them manually coded by CrowdFlower (CF) workers.
- **Model:**
  - Baseline: Multinomial Naive Bayer
  - Deep Learning: Long Short-Term Memory network (LSTM)
    - Implementation framework: PyTorch



[SIMPLE ILLUSTRATION OF THE LSTM MODEL FROM COLAH'S BLOG]

- **Results:**
  79.22% for baseline model;
  84.68% for LSTM model on the validation dataset.

Results Analysis

- **Successfully implemented the deep learning model**
- **High probability of overfitting:** From graph on the right-hand side, this project observes a high probability of over-fitting in this LSTM model. While the training loss steadily decreases from 0.361 to 0.218, the validation loss gradually increases from 0.3 to 0.445. At the same time, Training accuracy improved from 86.33% to 91.87% while the validation set accuracy stagnated around 84% through the whole course.
- **Potential cause:** too small data size for LSTM model.

```
Epoch: 0
        Train Loss: 0.477 | Train Acc: 82.05%
         Val. Loss: 0.405 |  Val. Acc: 82.76%
Epoch: 1
        Train Loss: 0.376 | Train Acc: 85.51%
         Val. Loss: 0.376 |  Val. Acc: 85.70%
Epoch: 2
        Train Loss: 0.361 | Train Acc: 86.33%
         Val. Loss: 0.400 |  Val. Acc: 83.81%
Epoch: 3
        Train Loss: 0.345 | Train Acc: 87.35%
         Val. Loss: 0.383 |  Val. Acc: 85.46%
Epoch: 4
        Train Loss: 0.350 | Train Acc: 86.87%
         Val. Loss: 0.406 |  Val. Acc: 84.79%
Epoch: 5
        Train Loss: 0.327 | Train Acc: 87.93%
         Val. Loss: 0.384 |  Val. Acc: 85.90%
Epoch: 6
        Train Loss: 0.303 | Train Acc: 88.92%
         Val. Loss: 0.392 |  Val. Acc: 85.49%
Epoch: 7
        Train Loss: 0.292 | Train Acc: 89.18%
         Val. Loss: 0.556 |  Val. Acc: 80.47%
Epoch: 8
        Train Loss: 0.285 | Train Acc: 89.37%
         Val. Loss: 0.399 |  Val. Acc: 85.31%
Epoch: 9
        Train Loss: 0.272 | Train Acc: 89.90%
         Val. Loss: 0.392 |  Val. Acc: 84.63%
Epoch: 10
        Train Loss: 0.265 | Train Acc: 90.20%
         Val. Loss: 0.412 |  Val. Acc: 85.42%
Epoch: 11
        Train Loss: 0.275 | Train Acc: 89.90%
         Val. Loss: 0.417 |  Val. Acc: 85.25%
Epoch: 12
        Train Loss: 0.255 | Train Acc: 90.21%
         Val. Loss: 0.407 |  Val. Acc: 85.49%
Epoch: 13
        Train Loss: 0.248 | Train Acc: 90.74%
         Val. Loss: 0.420 |  Val. Acc: 84.79%
Epoch: 14
        Train Loss: 0.247 | Train Acc: 90.67%
         Val. Loss: 0.417 |  Val. Acc: 84.34%
Epoch: 15
        Train Loss: 0.241 | Train Acc: 91.26%
         Val. Loss: 0.441 |  Val. Acc: 84.70%
Epoch: 16
        Train Loss: 0.230 | Train Acc: 91.40%
         Val. Loss: 0.428 |  Val. Acc: 84.82%
Epoch: 17
        Train Loss: 0.236 | Train Acc: 91.04%
         Val. Loss: 0.431 |  Val. Acc: 84.95%
Epoch: 18
        Train Loss: 0.229 | Train Acc: 91.63%
         Val. Loss: 0.432 |  Val. Acc: 84.35%
Epoch: 19
        Train Loss: 0.218 | Train Acc: 91.87%
         Val. Loss: 0.445 |  Val. Acc: 84.68%
```

Reflection

- The implementation process provided the author with **an opportunity to understand** the theory behind text classification and the rationale behind automated hate speech detection.
- Although this project adopted the LSTM model, the author is aware of **the drawbacks of LSTM** as it still cannot memorize sequences that scale beyond a certain range. Moreover, the RNN models are not hardware friendly which means that it takes many computation resources. Many improvement and alternatives have been proposed, especially those that centered around "attention".
- Apart from efficiency and accuracy, we still face other challenges:
  - ☐ **Technical:** distinguish subtle contextual differences, active detection evasion etc.
  - ☐ **Non-Technical: right** balance between freedom of speech and hate speech quarantine, what do our values stand for facing hate speech.