Task 2: Toxic Spans Detection

- Homepage: https://competitions.codalab.org/competitions/25623
- Task Type: Sequence labelling

Description

Given a sentence, you need to extract a list of indexes that point to toxic phrases in this sentence.

- Input: A sentence (`str`)
- Output: A list of indexes (`List[int]`)

Sample data

- Input: `"Your comment is ridiculous"`
- Output: `[16, 17, 18, 19, 20, 21, 22, 23, 24, 25]`
- Explanation: `ridiculous` is a toxic word. Its span in this sentence is `[16, 17, 18, 19, 20, 21, 22, 23, 24, 25]`, which is corresponding to all characters of `ridiculous`.

Data link

- Training data: link
- Test data: link
- Validation data: None

Evaluation metrics

F1 score for set.

Given a sentence, let G denote the ground truth of spans and A denote the prediction set of spans. The precision is calculated as below:

$$P = \frac{|A \cap G|}{|A|}.$$

The recall is calculated as below:

$$R = \frac{|A \cap G|}{|G|}.$$

The F1 score is calculated as below:

$$F_1 = \frac{2 \times P \times R}{P + R} = \frac{2 \times |A \cap G|}{|A| + |G|}.$$

Here, $|\cdot|$ denotes the cardinality of a set. Specially, if $|G| = 0$, we let $F_1 = 1$ if $|A|$ is also 0, otherwise $F_1 = 0$.

Finally, the F_1 score of the entire dataset is an average of F_1 score of all sentences.

Requirements

Implementation

1. Implement a model to predict the toxic spans in the test data.
2. Implement the evaluation metrics.
3. Split the training data into training and validation data.

Submittion

1. A `requirements.txt` that contains the required packages and versions to run the code.
2. A `run_prediction.py` or `run_prediction.ipynb`
   2.1. A `run_prediction.py` that outputs the F_1 score of test data by running `python run_prediction.py`.
   2.2. A `run_prediction.ipynb` of a jupyter notebook or a Google Colab notebook.
3. A set of model parameters that will be loaded in prediction.
4. Runnable training code that can be executed by `python run_train.py`
5. A `network.txt` that contains the model structure and parameter number. For PyTorch users, please consider `pytorch_lightning`. For Tensorflow users, please consider `model.summary()`.
6. A `report.pdf` that describe your problem formulation, model choice, ways to improve prediction scores, and experimental results (table/plot), etc.

Project evaluation

1. Runnable after installing `requirements.txt` by running `pip install -r requirements.txt`
2. F_1 score of test data
3. Model implementation
4. Optimizer, regularizer, and other techniques to involved in the project.

Note:

1. You are allowed to use Python >=3.7,<=3.9
2. You are allowed to use GPU from your own device or Google Colab. Also, you should not worry about the CUDA compatibility.

Important Notification

- The test data downloaded include labels. You are **NOT** allowed to incorporate test data into training. Please only use validation data to adjust hyper-parameters.
- This is an open task of SemEval. There are solutions on GitHub. Please implement the code by yourselves and do **NOT** copy/paste them.