Task 1: Lexical Complexity Prediction (LCP)

- Homepage: https://sites.google.com/view/lcpsharedtask2021
- Subtask: Task 1, predicting the complexity score of single words
- Task Type: Regression

Description

Given a sentence and a word in this sentence, you need to calculate a score between 0 and 1 to represent the lexical complexity of this word.

- Input: A sentence (`str`) and a word (`str`)
- Output: A score (`float`)

Sample data

Example 1

- Input:
  - Sentence: `Behold, there came up out of the river seven cattle, sleek and fat, and they fed in the marsh grass.`
  - Word: `river`
- Output: 0.0
- Explanation: `river` is a word that is easy to understand

Example 2

- Input:
  - Sentence: `Then go in merrily with the king to the banquet.`
  - Word: `banquet`
- Output: 0.4305555555555556
- Explanation: `banquet` might be a complex word for some people

Data link

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

Evaluation metrics

The overall score in the dataset is used to evaluate. The mean absolute error (MAE) of all predicted overall scores and ground truth scores is adopted as the evaluation metric.

Requirements

Implementation

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

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 MAE of the overall 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. MAE of the overall score on 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.