Natural Language Processing – HW3

In this assignment you will classify tweets to emojis using LSTMs. Each tweet will be classified to one of the following emojis:

0: ❤️ 1: 😂 2: 💕 3: 🔥 4: 😊

Instructions

- Assignment will be submitted in singles or pairs.
- **Basic Part**: Use LSTMs to reach 30% accuracy on the eval set.
- **Competitive Part**: Train the best model you can. Write your predictions on the test set to "competitive.csv" at the root dir. File should be written in the same format as the other files. Make sure it is exactly in the same format, without an index column.
- You should write readable, modular code. Document your methods. Code is 10% of grading.
- In the assignment's directory you will find the following files:
  - data – a directory with all your data files. The file contains comment that direct to a solution that we recommend you try to follow.
  - modeling.py – Used for modeling your NN. The file contains comments that direct to solution that we recommend you try to follow.
  - train.py – Used for training and analysis. The file contains a basic training and eval procedure, including reporting to Weights & Biases. The only line missing is the line that performs gradient

accumulation, which allows us to increase the batch size without actually running a batch of examples into the model. This is done by performing an optimization step only every n iterations. You can read about this here.
At submission, running this script should load the data, train your model for the competitive part and outputs the predictions to the "competitive.csv" file. The code should run in less than 30 minutes.
    - consts.py – used for global constants and paths.
    - utils.py – used for utility functions.
    - dataloading.py – Used for loading data and preprocessing.
- Tuning – Here are some directions for hyperparams\modeling choices:
    - Learning rate, accumulation steps (batch size)
    - LSTM params: number of layers, bidirectionality, hidden_dim, dropout.
    - Backbone model: RNN/LSTM/GRU
    - Vocab preprocessing: Should words that appear once be represented in the vocab? Lemmatization.
    - Embedding: initializing the embedding weights using a pretrained embedding.
- Write a report of up to 2 pages containing
    - What hyperparameters did you tune?
    - What models/hyperparameters/modeling choices did you try?
    - How did you select the hyperparams for the competitive part?
    - Any other results\plots you think are interesting.

- Submission until Thursday the 18.7 at 23:59 pm. You should submit a zip file with the code and the report in the following format:
    - hw1_id1_id2.zip
        - train.py
        - dataloading.py
        - modeling.py
        - utils.py
        - consts.py
        - competitive.csv
        - report.docx/pdf
- Grading
    - 80% base part
    - 10% code
    - 10% competitive results