CS310 Natural Language Processing - Assignment 1: Neural Text Classification
Total points: 50

Train a neural network-based text classification model on the Chinese humor detection dataset. The dataset is collected by this project: https://github.com/LaVi-Lab/CLEVA.
Submit: The modified notebook file A1.ipynb and any dependent Python files.

1. Data Processing (15 points)
The original dataset (`train.jsonl` and `test.jsonl`) properly so that they can be loaded with torch.utils.data.DataLoader.
The raw `jsonl` data looks like this:
`{"sentence": "和平说：你不挨屋儿做功课你跟这儿晃悠什么呢你", "choices": ["0", "1"], "label": [0], "id": "dev_292"}` This is a negative (0) example (not humor)
`{"sentence": "季春生说：你还别说，就剩这部分机能还正常", "choices": ["0", "1"], "label": [1], "id": "dev_78"}` This is a positive (1) example (humor)
Requirement:

a) First, use a basic tokenizer that treats each single Chinese character (字) as a token, and discard all the non-Chinese tokens (English letters, numbers, punctuations).

b) Incrementally improve your tokenizer by recognizing special patterns, such as consecutive digits, English words, and punctuations.

2. Build the Model (15 points)
Build the neural network model using `torch.nn` module.
Requirement:

a) Use the bag-of-words method provided by `nn.EmbeddingBag`.

b) Use at least 2 hidden layers when defining the fully-connected component. You can use the `torch.nn.Sequential` module for convenient implementation.

3. Train and Evaluate (10 points)
Make sure your train and evaluate code can run correctly and print the accuracy information properly.
Requirement:

a) Train with sufficient epoch numbers (for example, 10) until you observe a good performance.

b) Report the final test accuracy, precision, recall, and F-1 score.

4. Explore Word Segmentation (10 points)
Include word segmentation program to the processing step and observe how that will affect the classification performance. You can think of word segmentation as a more advanced tokenizer, which can group multiple characters (字) to a word (词). Note that the vocabulary needs be re-built once segmentation is applied, whose size will get smaller.
Requirement:

a) Install a word segmentation package, such as `jieba`: https://github.com/fxsjy/jieba. Use it to process the data and save as the new segmented data.

b) Run through the train/evaluate process on the segmented data and compare the performance (accuracy and F1) with the original data.