Word Meaning Comparison

(A3: Part B)

Words often have multiple meanings associated with them and the surrounding context often determines the specific meaning of the word which is used. The goal of this assignment is to develop deep neural models that can identify whether a particular word used in a sentence pair has the same meaning in both sentences or has a different meaning in each of the sentences.

For example,

S1: *We often used to play on the **bank** of the river*
S2: *We lived along the **bank** of the Ganges.*
S3: *He cashed a check at the **bank***

S1 and S2 use the same meaning of the word *bank* (river bed) while S1 and S3 use different meanings of the word *bank* (river bed vs. financial institution)

We call this task *Word Meaning Comparison* and frame it as a classification problem. The NLP model must classify the input sentence pair (*X*) and the word (*W)* into a label **T** if W has the same meaning in both sentences of *X* and **F** if *W* has different meanings.

X, W	Ground Truth Label
(S1, S2), bank	**T**
(S1, S3), bank	**F**
(S2, S3), bank	**F**

Instructions:

In Part B of the assignment, you need to make use of the latest advances in pre-trained language models to improve performance on this task.

However, considering the large number of pre-trained models, we don't want you to spend time just experimenting with different architectures as it will require a lot of computation resources and add little learning value. Therefore, we will restrict the pretrained models you can use to the following four: 1. bert-base-(cased/uncased) (link), 2. DistilBERT (link), 3. alberta-xlarge (link) and 4. T5v1.1-small (link). We are only allowing the specific versions of the models that can run on limited GPU memory.

You can use the all-popular HuggingFace library to either 1. import these models into the code you developed for Part A and use the appropriate tokenizer, optimizer or 2. directly use the training pipelines available in HuggingFace (link).

Moreover, in this part you are allowed to use additional data, apart from the training data provided. For example, you may find that training the model on data from NLI task (SNLI (link), MNLI (link) datasets) before fine tuning on this task may result in stronger performance. You are free to explore different combinations.

Data:

You can download the training and validation data from data link. The *.data.txt correspond to the input file and *.gold.txt correspond to the final labels. Each line in the input file corresponds to one example. It has five columns - word, pos tag, location in sentence1-location in sentence2, sentence1, sentence2

You are not allowed to use the validation data for training the model. You will have to report your final models' validation performance in a Google form. These will be available for the entire class to view and compare with their own. You can make multiple submissions to the form. The last one before the deadline will be considered final. We should get the same score on re-training your model (discussed further under ***Submission***).

Evaluation:

The performance on the task will be evaluated using binary accuracy.

The testing notebook contains the format of the test file, expected format of the final predictions and code for computing the performance. You have to report the score computed with this as the final validation performance.

Submission:

You can either train your model on HPC or Colab. However, the final trained model must be stored in Google Drive and the final testing code must be a Colab notebook of the form USERID_A3_B_test.ipynb (using the same format of Part A). You must give ***EDITOR*** access to keshavkolluru@gmail.com, vishalsaley114@gmail.com and kartikeya.badola@gmail.com. We will float a Google form where you will have to paste the shareable links for the notebook as well as the trained model.

If you decide to use Colab notebook for training, you can follow the same steps as in Part A.

However if you are using HPC for training, you have to submit the code on Moodle. We will follow these steps for training the model:

1. *bash install.sh*

This should install all the libraries required for training your model using pip install. It should also download the training data using wget.

2. *bash run-train.sh*

This should train the model and package everything needed for testing into a single zip file (trained model, vocabulary, etc). We will manually upload this to Google Drive, change the link in your testing notebook and ensure that we are able to replicate your reported validation scores.

We will compute late days using the following formula:
math.ceil(max(Training notebook/Moodle submission, Testing notebook, Trained model) last modified time - 11:59 PM assignment deadline)

We will run the final model on an unseen test set and use the generated predictions to evaluate the final performance. Your assignment will be graded based on the relative ranking in the class.

In order to ensure integrity, we will randomly re-train your model with the original training/validation split and verify that the model achieves the same performance as reported in the validation scores Google form. Some minor discrepancies are acceptable due to randomness innate in Pytorch. However, significant differences may be treated as model plagiarism (such as copying model weights from another student) and will be dealt with accordingly.

We will allow a maximum runtime of 5 hours for training and 30 mins for testing on a K80 GPU (12 GB). If you find this insufficient, you can raise a ***private post*** on Piazza explaining your case. We may grant case-by-case exceptions.

What is allowed? What is not?

General

1. The assignment is to be done individually.
2. You must use Python for this assignment.
3. You must not discuss this assignment with anyone outside the class. Make sure you mention the names in your write-up in case you discuss with anyone from within the class outside your team. Please read the academic integrity guidelines on the course home page and follow them carefully.
4. We will run plagiarism detection software. Any team found guilty will be awarded a suitable penalty as per IIT rules.
5. Your code will be automatically evaluated. You get a significant penalty if it does not conform to output guidelines. Make sure it satisfies the format checker before you submit it.

Specific:

6. Apart from HuggingFace, you are not allowed to use any other libraries or frameworks available like Allennlp/Fairseq/PytorchLightning, etc.
7. You are allowed to only use those pre-trained models described in previous sections.
8. You are strictly prohibited from using existing GitHub repositories that you may find online for related tasks.
9. You are allowed to use additional tools for text preprocessing or adding additional features. Please confirm them on Piazza first. You are automatically allowed to use those which were allowed in Part A.
10. You are not allowed to re-submit your Part A submission, as you must make some use of contextual embeddings in your final architecture.

All the best!