Final Project Proposal - Marie Huynh

I. Introduction

I plan on using my CS224N (Natural Language Processing with Deep Learning) final project as my final project in BIODS53.
This final project will be a collaboration with two other students. We aim to set up a **single multi-task model that can perform three separate linguistic tasks**:

- Sentiment Analysis
- Paraphrase Detection
- Semantic Similarity

We aim to leverage multi-task learning to reduce the number of required model parameters while maintaining high performance across multiple tasks. This goal is critical for building lightweight language models intended for mobile and wearable devices with limited computing, battery, memory, and processing. Instead of storing multiple models, we can store a single multi-task model that addresses multiple tasks.

We haven't started the project and we just created the repository on github. We will be strating working on the project this week.

II. Data

- We will be using some preprocessed data given to us:
  - the provided **Stanford Sentiment Treebank (SST) dataset** for sentiment analysis training data,
  - the **Quora dataset** for paraphrase detection training data
  - the **SemEval dataset** for semantic textual similarity training data.
- We will not collect new data, but we may leverage other public datasets reported in the GLUE platform for additional training data, that we may have to preprocess to input in our model.
- The inputs to the model will be one or two input sentences (depending on the task) and the task label. The output will be a single categorical value, where the possible categories depend on the chosen task.

III. Methods

- We will first implement the BERT model as **a baseline model**.
- We will execute **three strategies** to improve our multi-task results beyond baselines. We believe the combination of these strategies may yield competitive multi-task results to any single paper.
- First, we'll explore **gradient surgery for multi-task learning** from Finn et al (source : https://doi.org/10.48550/arxiv.2001.06782). This approach projects competing gradients from one task onto the normal plane of another task, preventing the competing gradient components from being applied to the network and impairing optimization. In our implementation, we will identify whether multiple task gradients are conflicting (check if their cosine similarity is negative), remove the conflicting

components of the gradients, and pass these updated gradients to our optimizer of choice.

- ➢ Next, we'll evaluate **the SMART fine-tuning framework** developed by Jiang et al (source : https://aclanthology.org/2020.acl-main.197/).
- ➢ Finally, we'll evaluate **the training scheduling method and the PALS (projected attention layers)** described in the paper (source : https://arxiv.org/abs/1902.02671) to see if it yields improved results.
- ➢ Hopefully, the combination of the above approaches yields improved (or competitive) accuracy for our multi-task model over our baseline single-task models, but with a reduced parameter count.

IV. Software Engineering Practise

- ➢ I believe that using good software engineering practice from the start in this project can make it more reproducible and more efficient.
- ➢ First, we will build a **solid design document** to enhance the expectations and objectives for each of the collaborators, to set priorities (goals) and nongoals, to improve the planning of the different milestones, and to keep as a reference for future development of the project.
- ➢ We will be using **source control (Git/Github)** to collaborate.
  - We will make sure to emphasize code review and review requests for pull requests in order to limit mistakes and improve communication among us.
- ➢ We will be writing some **unit tests, automated testing** on various functions :
  - If we end up leveraging other types of public data and preprocess it, we will test preprocessing functions.
  - We can do some unit tests on our algorithms, …
  - …
- ➢ This project involves writing a significant amount of code and generating several logs (losses logs, testing out different hyperparameters) and graphs to compare different approaches and hyperparameters. Thus, we will pay strict attention to :
  - **code documentation** (docstrings, good variable names, …)
  - **code organization** (good split of the files, build functions, no copy-paste, …)
  - **data organization and good graph organization** : make sure all logs allow us to retrieve the hyperparameters tested, make sure the graphs are well labelled and named in our project to be efficient and make our work reproducible.
- ➢ We will be using diverse python packages. Thus, we will make sure our project is reproducible and include **dependencies management.**
- ➢ We will make sure the code has **good instructions to download the project and run it from scratch** for a new person.

V. Link to Project Handout

- http://web.stanford.edu/class/cs224n/project/default-final-project-bert-handout.pdf