NLP 244: Assignment 2 - Mathematical Entity Extraction

1 Objective

In this homework you will creating a system for information extraction of mathematical text. In particular, you will be extracting **entities** (definitions, theorems, proofs, and examples) and **identifiers** (names and references) from undergraduate/graduate math textbooks.

Here is an example of some text which contains a definition:("least element").

> By $S \subset \mathbb{N}$ having a *least element*, we mean that there exists an $x \in S$, such that for every $y \in S$, we have $x \leq y$.

Here is an example of a theorem:

> "**Corollary 7.2.** *Let E be a finitely generated module and E a submodule. By lemma 5, then E is finitely generated.*"

"Corollary 7.2" is the name of the theorem, "finitely generated module" is a reference to an object, "submodule" is another reference, and finally "finitely generated" is another reference.

Here is a short description of each entity type:

- `definition`: text which define some new concept or object.
- `theorem`: text which make a rigorous, provable claim.
- `proof`: text which proves (or suggests a proof) to a theorem
- `example`: an example of a defined object or application of a theorem.
- `name`: the name of a newly defined object or a theorem (e.g., "abelian group", "Theorem 1.2")
- `reference`: a reference to a name which may have been defined previously.

Names and references ("identifiers") can only appear within other entities. Also, the annotated training data ignores entities found within proofs. If your model finds names and references within a proof, you can filter them out in postprocessing to improve your F1 score.

For a complete description of how the data is annotated and what each tag means, refer to section 4.2.1 of `math-atlas-paper.pdf` (included in the .zip). You can also look at the training data to see examples of each tag.

2 Instructions

You are given a small annotated dataset with a train and validation split. You are also given the file IDs of the test data (but without any annotations). Additionally, you are given a set of unannotated MMD documents for inference time.

- Input: raw MMD text
- Training data: Triples of (start character index, end character index, annotation type)

2.1 Required: Part 1 - Baseline Model

Your first goal is to solve the task using few-shot prompting. For each token, your model should output one or more tags. Here is an example prompt and output:

```
-- Instruction
Perform BIO tagging on the following snippet of text. Assign one or more labels
for each token. You can choose from "definition", "theorem", "proof", "example",
"name", or "reference".

-- Input
Here's some stuff about mathy nonsense...
By \(S\subset\mathbb{N}\) having a /least element/, we mean that there exists
an \(x \in S\), such that for every \(y \in S\), we have \(x \leq y\).
Here's some more stuff...

-- Output
[O] [O] [O] ...
[definition] ... [definition, name] [definition, name] [definition] ...
[O] [O] [O] ...
```

Note that the text "least element" is both part of the definition and the name of the object being defined, so it gets both tags.

2.2 Optional: Part 2 - Training

Next, your objective is to fine-tune an LLM on the provided training data, and report results on the validation and test datasets. The input to your model should be a section of text (e.g., a chunk of 1024 tokens). The output (after post-processing) should be a list of the annotations found within that snippet of text.

Stage	Description
Input	MMD text
Model output	???
Final output	List of annotations in the same format as the training data

Note that this output format is very flexible, and allows for several different approaches to solving the task. For example, one approach is to make your model a BIO tagger, whose output may look something like this:

```
**Corollary   B-theorem, B-name
7.2.**        I-theorem, I-name
Let           I-theorem
E             I-theorem
be            I-theorem
a             I-theorem
finitely      I-theorem
generated     I-theorem
module        I-theorem
...
Here's        O
some          O
random        O
text          O
```

Note that annotation types may overlap, and your model should account for this. One option is doing multi-label classification for each token. Another is merging multiple classes into one. For example, `theorem-definition` would indicate an overlap of a theorem and definition. Or, you can train an ensemble model which handles each tag separately, and then merge all the outputs at the end.

Here are some popular LLMs you may consider using as a base model:

- Llama 3 (1B/3B/8B)
- Qwen 2/2.5 (1.5B/3B/7B)
- Mistral (7B)

You are not restricted to this set - you may use any pretrained decoder-only model found on the huggingface hub, within reason (e.g., don't use a model smaller than BERT, and don't use a massive 500B parameter model). Also, consider using the instruction-tuned variants of the models, which often perform considerably better across most tasks. Finally, you may also consider using an LLM which has been pretrained on large amounts of math data.

2.2.1 Inference

Once you train your best model, you are to run your model on the provided unannotated documents, and submit the extracted annotations in a pandas DataFrame saved to JSON in the following format:

Fields: `fileid`, `start`, `end`, `tag`

2.3 Optional: Kaggle Leaderboard (Extra Credit)

Additionally, the top 5 highest token-level F1 scores on the combined test/validation data will get extra credit. You can submit your test file predictions on Kaggle to get your scores.

Place	Bonus %
1	+5%
2	+4%
3	+3%
4	+2%
5	+1%

2.4 Required: Part 3 - Analysis

For the prompting method, and optionally for the trained model (if you trained a model), perform an error analysis on what kind of mistakes your model makes. Look at the outputs on the unannotated documents. Consider mistakes such as class confusion (e.g., does it think definitions are actually theorems?). Here are some questions you should answer in your analysis:

1. What kind of mistakes is your model making? (cite examples)
2. Why do you think the model makes those mistakes?
3. How would you go about improving your model? (Be more creative than "more training data"!)
   (a) Does your solution introduce any new potential problems?

3 Submission

****** Please do not submit any big files (>50 MB), such as model files or the training corpus. Thank you. ******

Submit the following files:

1. All source code, along with instructions to
   (a) run training (using your best model/hyperparameters)
   (b) run inference on a particular MMD file and save output
2. Your test validation/test set predictions.
3. Your results for inference on the unannotated MMD files
4. A report, containing
   (a) A detailed description of your model, training procedure, and data processing. Remember, your report should make your work reproducable.
   (b) Your token-level F1 scores on the validation set for your baseline & experimental models (highlight your best model score).
   (c) Your error analysis for part 3, including your responses to the questions listed.