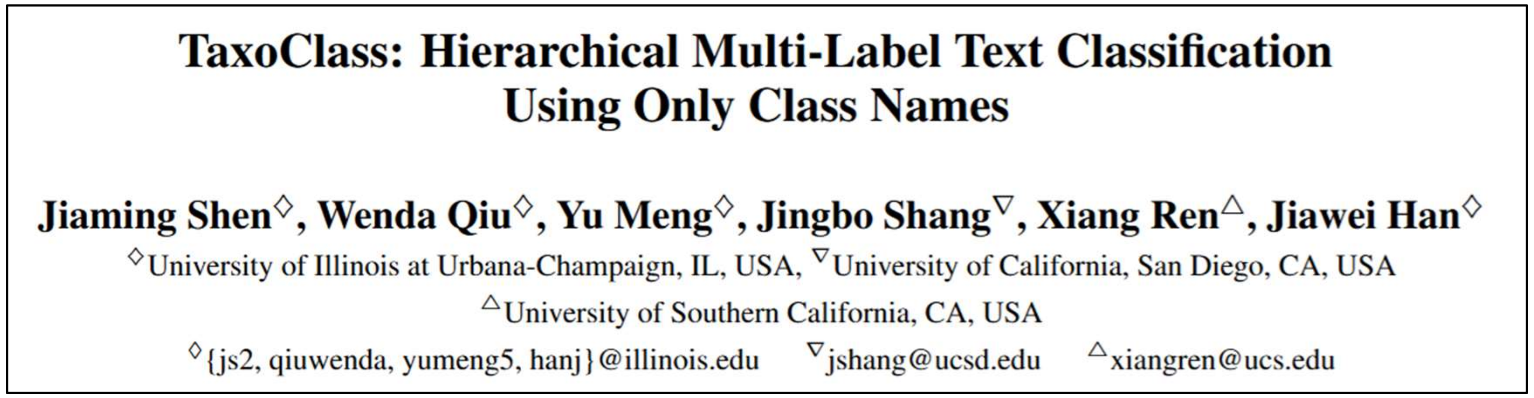


TaxoClass: Hierarchical Multi-Label Text Classification Using Only Class Names

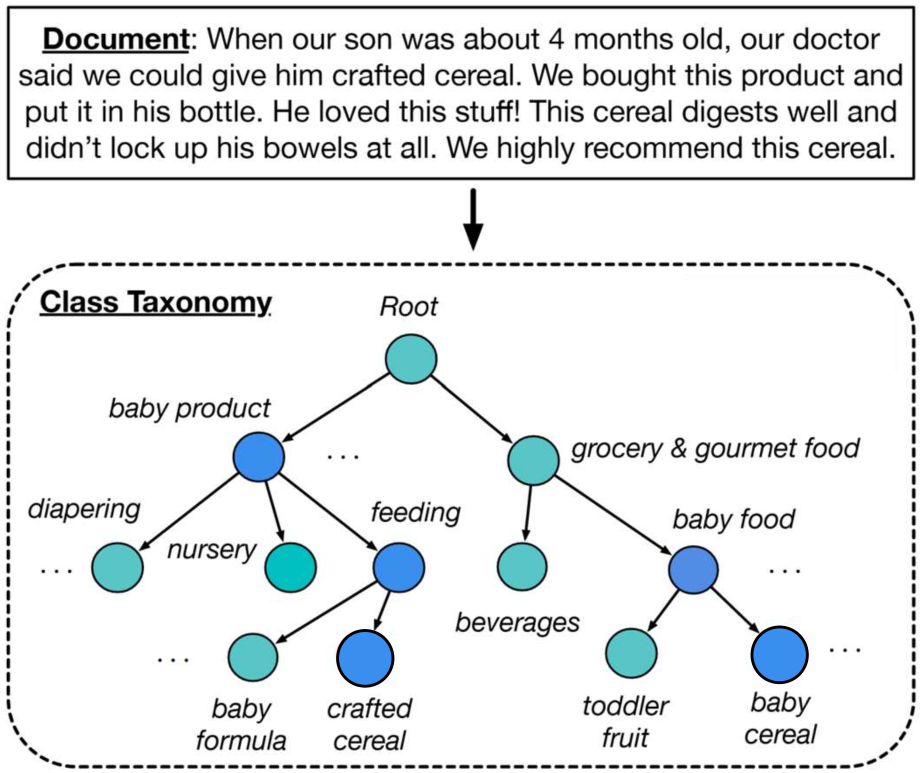
Jiaming Shen[◇], Wenda Qiu[◇], Yu Meng[◇], Jingbo Shang[▽], Xiang Ren[△], Jiawei Han[◇]
[◇]University of Illinois at Urbana-Champaign, IL, USA, [▽]University of California, San Diego, CA, USA
[△]University of Southern California, CA, USA
[◇]{js2, qiuwenda, yumeng5, hanj}@illinois.edu [▽]jshang@ucsd.edu [△]xiangren@ucs.edu

You are encouraged to use this paper as a key reference of this project.

Final project:
Hierarchical Multi-Label Text Classification

Task: Hierarchical Multi-Label Text Classification

- Your task is to perform product review classification *without using any labeled data*.
- Each review is associated with multiple product categories that are organized in a hierarchical taxonomy.



Document: When our son was about 4 months old, our doctor said we could give him crafted cereal. We bought this product and put it in his bottle. He loved this stuff! This cereal digests well and didn't lock up his bowels at all. We highly recommend this cereal.
Class Taxonomy
Root
baby product
grocery & gourmet food
diapering
nursery
feeding
baby food
beverages
baby formula
crafted cereal
toddler fruit
baby cereal

You are given:

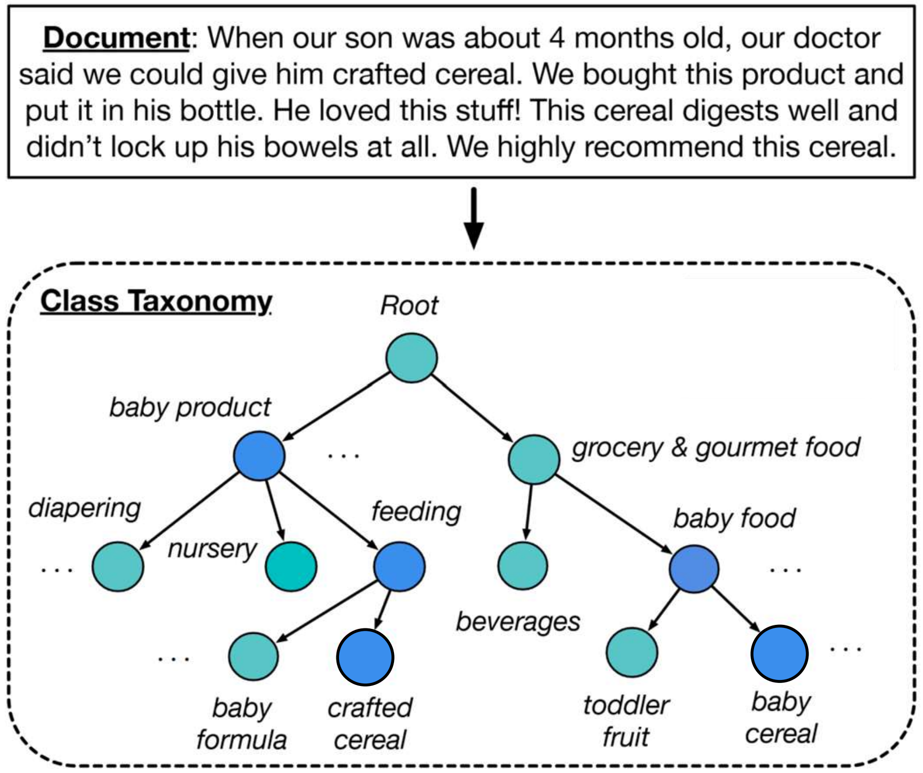
- 29,487 reviews (training), 19,658 reviews (test)
- 531 product classes and their hierarchy

The document (review) is related to total of five classes:
baby product, feeding, crafted cereal,
baby food, baby cereal

"grocery & gourmet food" is not answer since the review is about a specific product (*crafted cereal*), not the broader grocery category.

New challenges

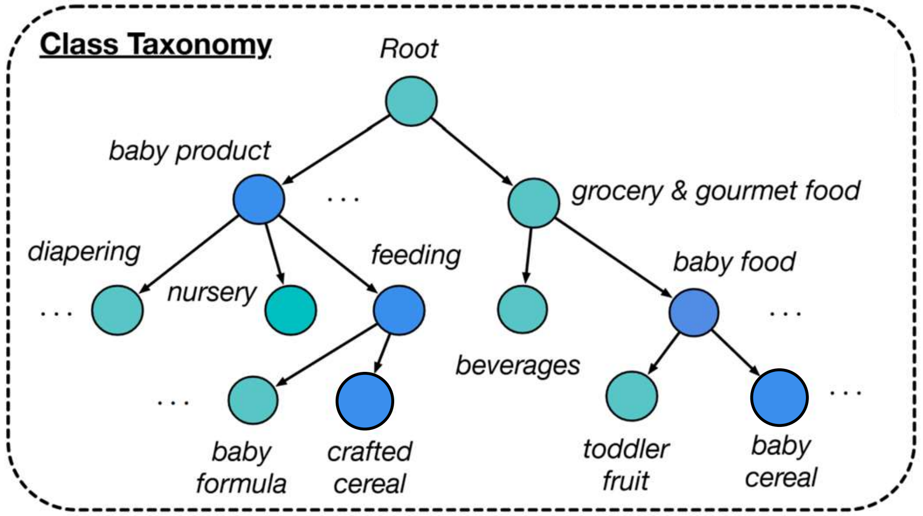
1. We have multiple labels for each document:
   - In the given dataset, each document is associated with **at least two and at most three labels.**


Document: When our son was about 4 months old, our doctor said we could give him crafted cereal. We bought this product and put it in his bottle. He loved this stuff! This cereal digests well and didn't lock up his bowels at all. We highly recommend this cereal.

*How can we determine **which classes are most strongly associated** with a given document?*

New challenges

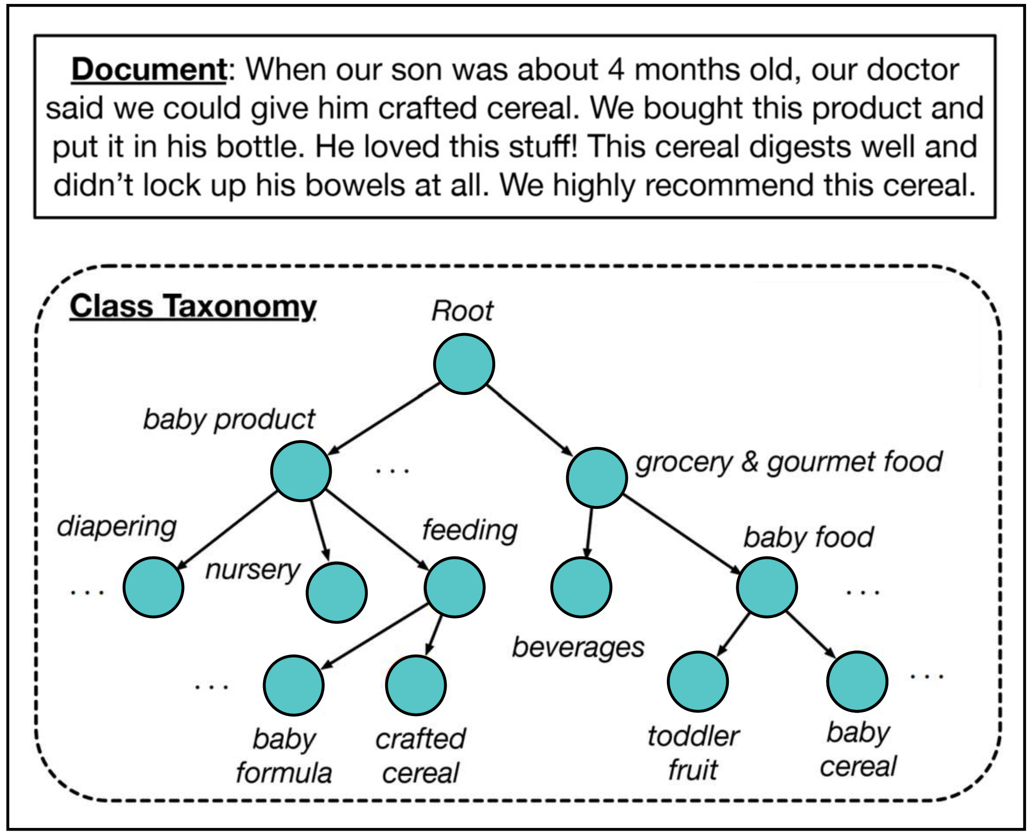
2. We have a hierarchy of classes:
   - Unlike PA#3, where all classes were flat, here they are **organized in a hierarchical taxonomy**.
     - Each node represents a category, and child nodes inherit *semantic relationships* from their parents.



- When generating ***silver labels***, constructing ***label embeddings***, or applying ***pseudo-labeling***, these hierarchical relationships need to be reflected to ensure consistency across related classes.

Good news! We can use Large Language Models (LLMs)

- LLMs have strong language understanding and can help address the scarcity of labels.
- One intuitive approach is to simply **ask the LLM** to find the most relevant classes for each document:



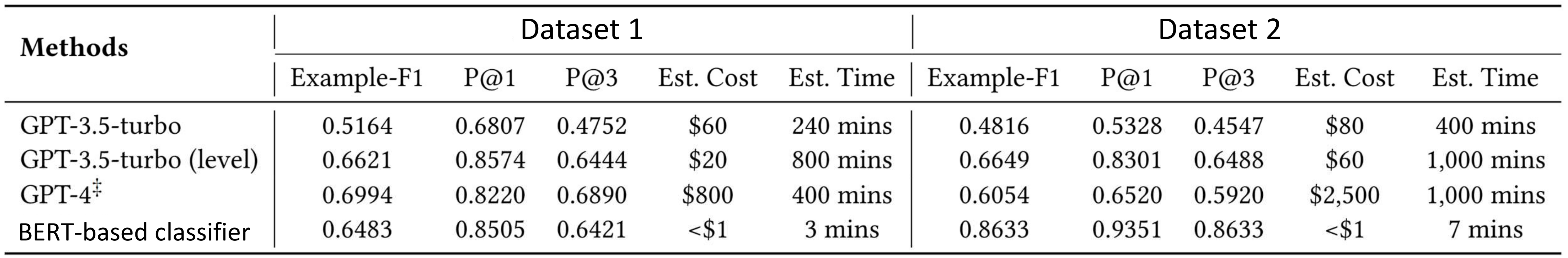
Document: When our son was about 4 months old, our doctor said we could give him crafted cereal. We bought this product and put it in his bottle. He loved this stuff! This cereal digests well and didn't lock up his bowels at all. We highly recommend this cereal.

Prompt: "find relevant classes for this document"

Crafted cereal, baby food…

LLMs empower you only with proper usage

- LLMs are not a free, all-powerful solution.
  - You may spend a lot of money and time, yet achieve limited performance if used naively.



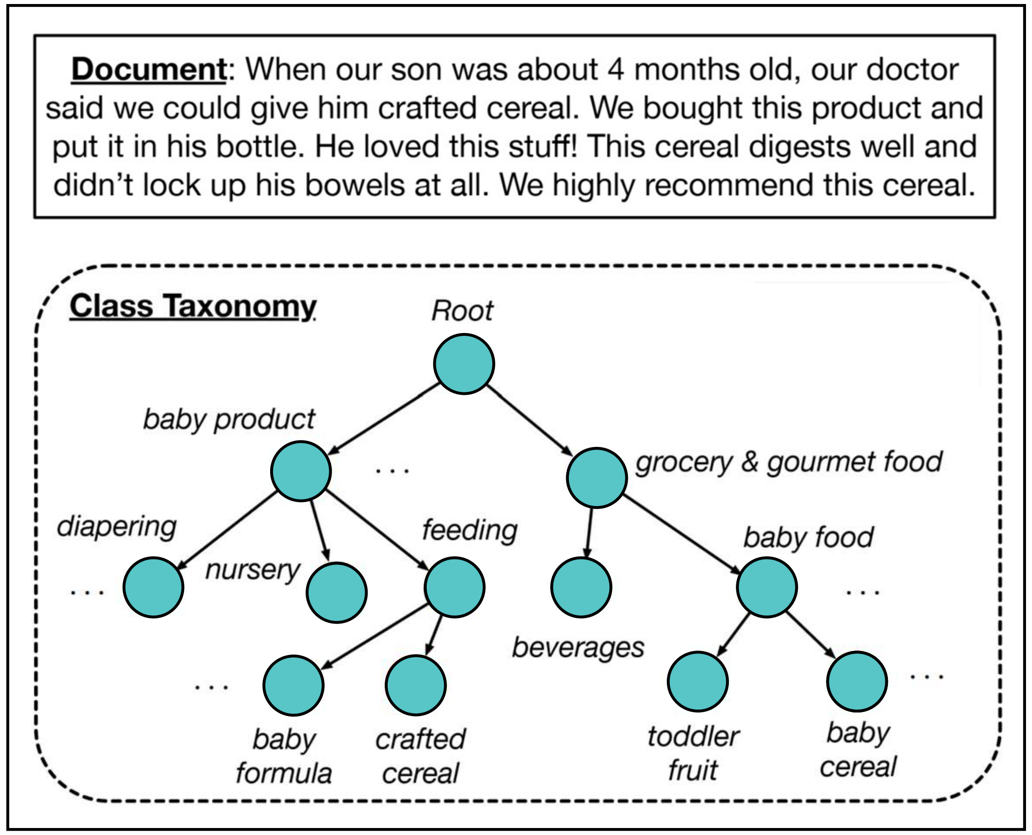
Methods	Dataset 1					Dataset 2				
	Example-F1	P@1	P@3	Est. Cost	Est. Time	Example-F1	P@1	P@3	Est. Cost	Est. Time
GPT-3.5-turbo	0.5164	0.6807	0.4752	\$60	240 mins	0.4816	0.5328	0.4547	\$80	400 mins
GPT-3.5-turbo (level)	0.6621	0.8574	0.6444	\$20	800 mins	0.6649	0.8301	0.6488	\$60	1,000 mins
GPT-4[‡]	0.6994	0.8220	0.6890	\$800	400 mins	0.6054	0.6520	0.5920	\$2,500	1,000 mins
BERT-based classifier	0.6483	0.8505	0.6421	<\$1	3 mins	0.8633	0.9351	0.8633	<\$1	7 mins

BERT-based classifier is trained with the help of LLMs.
Costs and times shown above refer to inference only.

- **Why limited?**
  - There are too many classes, and hierarchical relations exist among them.
    - When the prompt becomes too long, the LLM cannot process it efficiently.
  - The information for each class is very limited. We only have the category name.

How can we do better?

1. **Generate more contexts** for each class
   - Class information is often too limited (only category names).
   - We can enrich class information by generating additional contexts (e.g., key terms).



Document: When our son was about 4 months old, our doctor said we could give him crafted cereal. We bought this product and put it in his bottle. He loved this stuff! This cereal digests well and didn't lock up his bowels at all. We highly recommend this cereal.

Example instruction:

[Target Class] is a product class in Amazon and is the subclass of [Parent Class].
Please generate 10 additional key terms about the [Target Class] that are relevant to [Target Class] but irrelevant to [Sibling Classes].
Please split the additional key terms using commas.

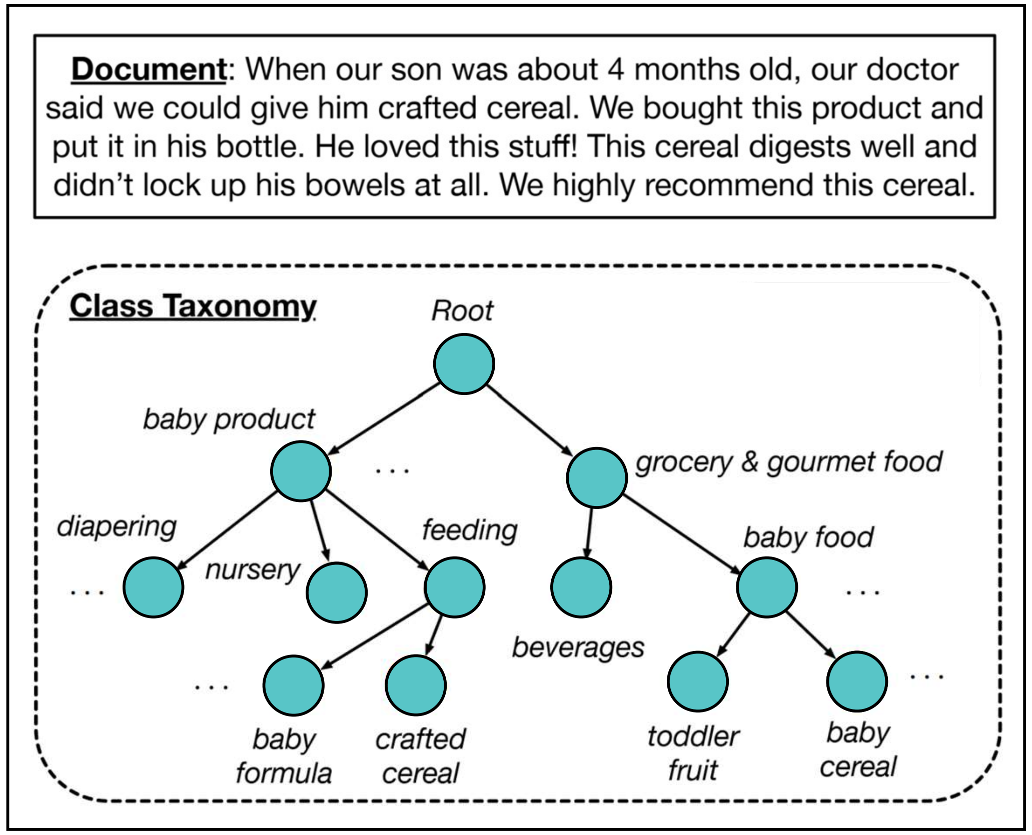
multigrain cereal, iron-fortified cereal, stage 1 baby food, …

Results of target class: crafted cereal

How can we do better?

2. Simplify the problem of LLMs

- First, narrow down the most probable candidate classes using reasonable heuristics or models.
- Then, ask the LLM to identify the most relevant ones among the reduced set.



Document: When our son was about 4 months old, our doctor said we could give him crafted cereal. We bought this product and put it in his bottle. He loved this stuff! This cereal digests well and didn't lock up his bowels at all. We highly recommend this cereal.

Example instruction:

You will be provided with an Amazon product review, and select its product types from the following candidates:

baby product, feeding, nursery, crafted cereal, …

feeding, crafted cereal, …

How can we do better?

3. **Focus on the parts where your classifier struggles**
    - For samples that your classifier already handles well — relatively easy data — the help of LLMs may not be necessary.
    - Instead, leverage LLMs <u>selectively for the parts where your model struggles</u>, using them to provide additional or corrective information.

- And, there are many more possible directions!
    - This remains an active and evolving research area.

Task summary: product review classification

- You are provided with the following resources:
  1. **Product review data**: 29,487 reviews (training), 19,658 reviews (test)
  2. **Classes**: 531 product categories and their hierarchy
  3. **Class-related keywords**: 10 keywords per class
     - These are generated using GPTs with the below instructions

[Target Class] is a product class in Amazon and is the subclass of [Parent Class].
Please generate 10 additional key terms about the [Target Class] that are relevant to [Target Class] but irrelevant to [Sibling Classes].
Please split the additional key terms using commas.

- You can additionally use 1,000 API calls of LLMs.
  - You may use the GPT-4o mini API (approximately $1) or any other freely available LLM.

General instructions: submission & grading

- **How to submit**:
  1. Prediction Results (Kaggle):
     - Submit your prediction results on Kaggle.
  2. Code (GitHub):
     - Submit your code via GitHub. The submitted repository should include all components to reproduce results.
  3. Report & GitHub link (LMS):
     - The report should be written in English, up to 8 pages.

- **How the project is graded**:
  1. Performance (50%):
     - Your performance on Kaggle will be evaluated. A private leaderboard will be used internally for grading.
     - To get a credit, your results must be reproducible.
  2. Report quality (50%):
     - Evaluation based on the completeness, clarity, and organization of your report.

General instructions: extra credits

- **Extra credits (+ 10%)**: specifically requested by college of informatics.
    1. At least 10 GitHub commits (5%)
    2. At least 90% utilization of the allocated AWS resources (5%).
        - Information about remaining AWS usage time will be provided periodically via LMS.
        - *Full credit will be given as long as the above criteria are met.*

- **GitHub details**:
    - Your GitHub repositories will be collected by the College of Informatics and analyzed as evidence of your open-source community contributions. These will be used for university-level policy decisions.
    - You must make at least 10 commits (counts toward extra credit).

    - Follow the naming format below:
        - Repository name: https://github.com/Your_GitHub_ID/20252R0136DATA30400

    - It is recommended to keep your repository private before the deadline and make it public after submission.

General instructions: honor code

1. All submitted code and report must be your own work.
2. The use of any data other than the provided dataset is strictly prohibited.
   - If you use public pretrained models (other than BERT), make sure they are not fine-tuned on Amazon data.
3. You may use LLMs to improve the quality of your report.
   - You must carefully review and take full responsibility for all content.
4. If you used an LLM for the task, the prompt and output file (including results up to 1,000 calls) must be provided.
   - Exceeding the specified number of API calls will be considered inappropriate behavior.
   - During the grading process, we may additionally request supporting materials (e.g., OpenAI dashboard logs) to verify your LLM usage

- If you are uncertain or have any concerns, please consult the TA.

General instructions: reproducibility

- You are responsible for ensuring that your results are fully reproducible.
  - The submitted code must be executable without any modification.
  - We will clone your repository and run your code.
  - Submissions that cannot be reproduced (within a reasonable range) will receive no credit.

  - Provide a clear and detailed description of how to reproduce your results in the repository's README.md file.
    - If your implementation involves multiple steps and cannot be executed with a single Python file, include a shell script (.sh) that runs the entire process to ensure reproducibility.
    - If you use intermediate files or trained models that are too large to upload to GitHub, store them in an external file system (e.g., Google Drive) and clearly provide the download link in your README.md.

  - I have worked with this dataset for a long time and am very familiar with the impact of each component; any irregular behavior can be easily identified.

General instructions: submission format

- Make sure to follow the submission format exactly:
  - Kaggle team name (AWS account ID)
  - GitHub repository name

⚠ Do not risk losing credit due to incorrect naming or submission format.