MIE 524 Data Mining
Assignment 5: Applications

This assignment allows students to experiment with recommender systems and natural language models.

- Programming language: Python (Google Colab Environment)
- Due Date: Posted on Quercus

Marking scheme and requirements: Full marks will be given for (1) working, readable, reasonably efficient, documented code that achieves the assignment goals, (2) for providing appropriate answers to the questions in a Python notebook (named `MIE524_A2.ipynb`) committed, together with any associated output files, to the student's assignment repository, and (3) attendance in the post-assignment lab quiz.

Please note the plagiarism policy in the syllabus. If you borrow or modify any multiline snippets of code from the web, you are required to cite the URL in a comment above the code that you used. You do not need to cite tutorials or reference content that demonstrate how to use packages – you should certainly be making use of such content. *The use of generative AI solutions to generate code or text answers to the assignment questions is not allowed.*

What/how to submit your work:

1. All your code should be included in a notebook named `MIE524_A5.ipynb` that is provided in the cloned assignment repository.
2. Commit and push your work to your GitHub repository in order to submit it. Your last commit and push before the assignment deadline will be considered to be your submission. You can check your repository online to make sure that all required files have actually been committed and pushed to your repository.
3. Your committed notebook should include all the computed outputs by first running it from top to bottom on Google Colab and then exporting the notebook.
4. A link to create a personal repository for this assignment is posted on Quercus.

This assignment has 4 points in total and the point allocation is shown below:

- Q1: 2 points
- Q2: 2 points

Note: Partial points may be given for partial answers. Points will be deducted for missing or incomplete answers. Points could also be deducted for styling.

Q1 - Recommender Systems

The paper **Neural Collaborative Filtering**[1] (NCF) aims to improve traditional collaborative filtering by replacing the inner product with a neural architecture that can learn an arbitrary function from the data. To supercharge NCF modelling with non-linearities, the authors propose to leverage a multi-layer perceptron (MLP) to learn the user–item interaction function.

Task

a) In the lab, we have examined the Generalized Matrix Factorization framework (NCF-GMF). Design an experiment to compare the impact of changing the embedding size and the role of regularization on NCF-GMF. Evaluate the performance of at least two different configurations using MSE.

b) Implement the NCF-MLP framework, as shown in Figure 2 from the paper, using PyTorch by completing the `my_NCF_MLP` class with **2** linear layers and the **relu** activation function. Split the provided `MovieLens` dataset into train and validation sets. Train the model on the provided train set using the `Adam` optimizer and the MSE loss function for 10 epochs. Report the loss on the validation set of each epoch.

c) Design an experiment to compare the impact of changing the embedding size, hidden layers size, regularization, and possibly other parameters on the NCF-MLP you have implemented. Evaluate the performance of at least two different configurations using MSE.

d) Based on your experiments, compare your best NCF-GMF and NCF-MLP. Comment on the results.

e) Now let's assume the ratings are either 0 or 1 (the user either likes it or dislikes it). Let's transform our data such that any rating greater or equal to 3 is mapped to 1 and any rating less than 3 is mapped to 0. Compare your best NCF-GMF and NCF-MLP on this new dataset. What do you observe differently? Would you change anything about your implementation to conform with this new dataset?

Q2 - Natural Language Models

For this problem, we will perform a text classification task using a filtered version of the `arxiv_dataset`[2].

[1] `https://arxiv.org/pdf/1708.05031.pdf`

[2] `https://huggingface.co/datasets/arxiv_dataset`

Task

a) Using the Zero-Shot Text Classification[3] framework with `model="facebook/bart-large-mnli` that we saw in the lab, predict the category (`first_category_english`) of each paper using the title of the paper and compute the accuracy and F1 scores.

b) For which categories does the Zero-Shot Learning framework perform poorly? What are some possible reasons and what might you try to improve the performance?

c) Try using at least two other language models[4] of your choice (anything with `mnli` in the name of the model) and repeat part a). What do you observe differently?

d) Repeat parts a) and b) but now using the abstract. Comment on your results.

e) You may find long descriptions of each category on the arxiv website[5], can the Zero-Shot Text Classification do a better job at classifying the categories using the long descriptions as labels?

[3] `https://huggingface.co/tasks/zero-shot-classification`
[4] `https://huggingface.co/models?pipeline_tag=zero-shot-classification`
[5] `https://arxiv.org/category_taxonomy`