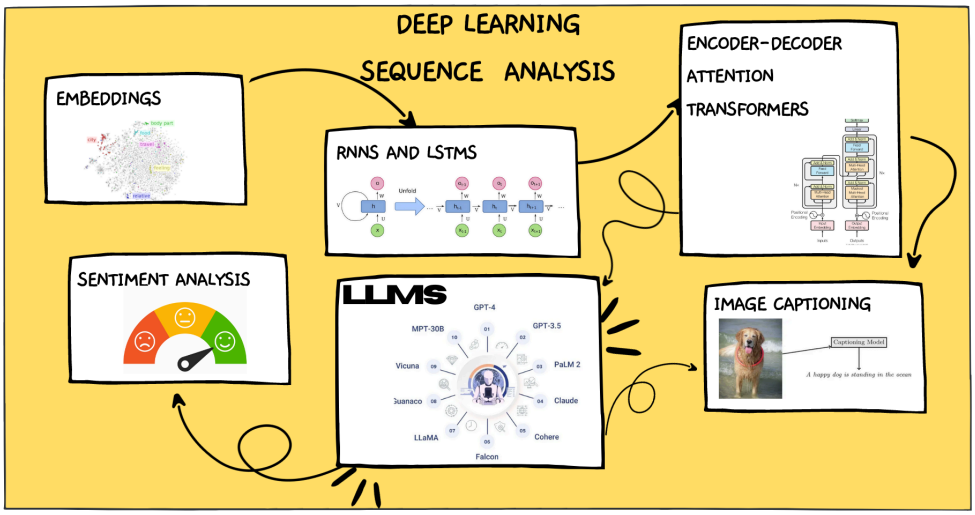

Coursework assignment

Deep Learning for Sequence Analysis

INM706

COURSE 2023-2024

GENERAL INFORMATION

This coursework extends your knowledge gained from tutorials and lectures, focusing on the design, training, and evaluation of Deep Learning (DL) based models for sequence based tasks.. By the end of this coursework, participants should demonstrate proficiency in designing, training, and evaluating different models for various tasks where sequences are used. It leverages various concepts covered in the module, including:

- Basic n-gram language models: bigrams, trigrams, etc., with smoothing methods like KneserNey smoothing.
- Neural Network Language Models (e.g., skipgrams).
- Recurrent Neural Networks (RNNs) encompassing Language Models and Classifiers, such as LSTM and GRU.
- Sequence analysis using ConvNets and RNNs for tasks like video classification, image tagging, and action classification.
- Implementation of dialogue systems (chatbots).
- Exploration of Transformers for applications like text generation and chatbots.
- LLMs (large language models), RAG (Retrieval Augmented generation)

By the end of this coursework, you will have honed your skills in applying DL concepts to real-world problems. Your chosen problem will serve as a practical case study to showcase your proficiency in utilizing PyTorch for model development and evaluation. The ultimate goal is to present a well-crafted solution that reflects your understanding and application of the learned concepts.

Read the whole document and ask as soon as possible anything that is unclear. The sections are as follow:

- Marks and Deadline
- About the coursework - general information on the courseworks purpose
- Implementation details and deliverables - contains important information about the expected code structure
- Written report - expected structure and content of your report
- Oral presentation
- Final submission - a summary of everything that needs to be on moodle
- The final sections refer to referencing, plagiarism and extenuating circumstances please read them carefully

1.Marks and Deadlines:

The coursework consists of two components:

- ***written report (WR)*** **weighing 70% of the total mark**
- ***oral presentation (OP) or Viva*** **weighing 30% of the total mark**

Each component is evaluated up to 100%, and you need to get 50% in *each* component to pass the module. Otherwise, you will need to resit the component you have failed.

You cannot do your oral presentation without having presented (or passed if resitting the OP only) your written report.

Report submission deadline (WR) Sunday, 5th May 2024 17:00 GMT

a. You must submit your WR in PDF format online using the Submission Area: Written Report in Moodle.

b. Caution: The system will not admit any submissions after the deadline. That is, if you press the submit button at 17:00:01, it won't be accepted.

Oral Presentation Date (OP) Thursday, 16th May 2024 *- the time will be scheduled*

c. Please submit your slides to the Submission Area Oral Presentation (Viva) in Moodle.

d. Your specific slot will be announced later and the timetable will be uploaded to Moodle

IMPORTANT:

In this coursework, it is imperative that the submitted code and the accompanying report go hand in hand, forming a cohesive and comprehensive submission. The code should be structured and documented in a manner that aligns seamlessly with the report's content. Each section of the report should reference and explain the relevant portions of the code, providing clarity on the implementation details, methodologies, and key findings. It is essential that both components, the code, and the report, complement each other to convey a complete and coherent narrative of the undertaken project.

2. About the coursework

In your project, you will leverage the knowledge acquired from the module to tackle a diverse range of challenges based on sequential data. This involves:

- Architecting Language Models:
    - Design and refine models tailored to language-based tasks.
    - Integrate specialized layers for handling sequential data effectively.
    - Implement discriminators, generators, and classifiers for language-related applications.

- Architecting Sequence-to-Sequence Models:
    - Explore cutting-edge techniques within sequence-to-sequence architectures.
    - Incorporate attention mechanisms for improved sequence modeling.

- Training and Evaluating DL Models for Language Tasks:
    - Utilize various optimizers, including nuanced applications of ADAM, for training language models.
    - Methodically document experiments, leveraging tools like the Wandb library.

- Potential Problems to Address in Language and Sequence Tasks:
    - Text Generation:
        - Develop models capable of generating coherent and contextually relevant text.
    - Named Entity Recognition:
        - Implement models for accurately identifying and classifying named entities in text.
    - Sentiment Analysis:
        - Utilize DL models to analyze and classify sentiment in textual data.
    - Machine Translation:
        - Architect sequence-to-sequence models for translating text between languages.
    - Question Answering:
        - Develop models capable of understanding and answering questions based on textual input.

In addition to language and sequence-based tasks, consider expanding the scope of your project to encompass domains such as trading data and weather, where sequential data plays a crucial role. Here are examples that illustrate the application of deep learning in these domains:

- Time Series Forecasting for Trading Data:
    - Problem: Predicting stock prices or market trends based on historical trading data.
    - Potential Solution: Employ recurrent neural networks (RNNs) or long short-term memory networks (LSTMs) to capture temporal dependencies and patterns in financial time series data.

- Weather Prediction using Sequential Data:
    - Problem: Forecasting future weather conditions based on historical meteorological data.
    - Potential Solution: Utilize recurrent neural networks (RNNs) or convolutional neural networks (CNNs) to model temporal and spatial dependencies in sequential weather data.

You will work on exactly one problem you will have selected from one of the topics on this list. If you have a different idea in mind we can discuss it, but always be aware of the time you have to implement it.
Before starting to work on your projects make sure you have a quick chat with the module leader about your idea and your implementation plan (i.e. design, train and evaluate).
You do not need to add any state of the art functionality to your report, this is left for the individual project, however make sure you make your project your own. Use your own ideas.

3.Implementation and deliverables:

During the duration of the course you will be presented best practices in structuring your code, using github and logging libraries.

Must:

- Only Pytorch framework must be used for all implementations.
- Only Wandb should be used for logging.
- Code should be committed periodically to github.

Code Deliverables are:

- Zipped folder containing the project
- Github url of the project (make sure it is accessible)
- Wandb logs - you can add them to a report and share the report - a how to file will be provided
- Extras:
    - Database url wether from a website or where it has been uploaded

The project folder should have the following structure:

- Python files with models (name them appropriately and split based on functionality)

- Important files:
    - Requirements.txt - Should have all the needed requirements for your project
    - Setup.sh - running this bash file should create a virtual environment and install all requirements
    - Train.py - running python train.py should start a train loop, and should also log results. I will run this to ensure everything was set up correctly. Parameters can be passed to this file to select nr_epochs and other relevant parameters.
    - Inference files:
        - At inference time your model should be run with a selected checkpoint to showcase the final results. Example: If it's a chatbot then I should be able to chat with it. If it's sentiment analysis I should be able to give it some input and see the results. If you choose something more complicated in terms of input (e.g. trading) make sure you write down how to test it. During the labs several methods of showing your results will be presented and you can choose which one you like best.
            - Jupyter notebook
            - Dash app
            - Streamlit app
            - Colab
            - Files needed for inference: notebook / folder to run your (streamlit/dash app), checkpoint file

4. Written Report (70%):

The report must be at most 3000 words and 15 pages long. Format for reports: pdf format, single column, standard A4 margins, standard default line spacing of 1.15, Arial 11, including all figures. It is preferable to use Latex, but Word/LibreOffice are fine too. Late submissions will score 0. You can upload work on Moodle more than once, so there is no need for last minute submission. Don't leave final submission until the last minute though.

CW reports should encompass the following sections:

1. Introduction (15 marks, Individual Component):

Provide an overview of the problem you are addressing, specify the dataset employed, mention any borrowed source code, and include a brief literature review (which can be integrated into the Introduction text or having a separate section).

2. Methodology (30 marks):

Explain the models used along with Architecture figures for clarification - if you're using parts of an existing Architecture don't forget to reference them, as well as making it clear what your

contribution is. Include all relevant equations, such as loss functions, ensuring they are presented, explained, correctly labeled/numbered, and appropriately referenced.

3. Results (25 marks):

Presenting the results should involve meticulous attention to detail, employing tables, such as the confusion matrix for classification problems, and plots. The submitted code for generating these figures is imperative, accompanied by the inclusion of Wandb logs for comprehensive transparency.

It is essential that all visual elements, tables, and accuracy metrics adhere to a precise labeling and numbering scheme. Each component should be thoroughly referenced, and the rationale behind their inclusion must be clearly explained to ensure the integrity and interpretability of the results.

In the case of multiple experiments showcased in the results, it is crucial that accompanying comments provide a comparative analysis, highlighting both commonalities and distinctions among the various trials. This approach contributes to a nuanced understanding of the outcomes and facilitates a comprehensive evaluation of the experiments conducted.

4. Conclusions (15 marks, Individual Component):

Sum up the framework you've worked on in the earlier sections, tying it back to the Introduction. Capture the main takeaways concisely and clearly. This part should showcase your understanding of the topic.

5. Reflections (15 marks): Reflect on the learning outcomes of the coursework, detailing encountered challenges, deviations from the initial plan, and insights gained. Discuss what you would have done differently.

For pairs working on the coursework, each student should choose one of the individual components (Introduction or Conclusions) to work on independently. These components will be assessed separately, resulting in different final marks. Equal contribution is expected for all other sections.

5. Oral Presentation (30%):

- The Oral Presentation will last 15 minutes.
- You are required to give a presentation of your work in which you go over the methodology and the results, this should be 12 minutes long, and 3 minutes will be used to ask additional questions.
- Slides must be submitted by the deadline above in Moodle’s submission area.

You cannot do your oral presentation without having presented your written report.

When choosing a problem, consider these constraints:

Time:
- Plan effectively for timely submission.
- Avoid overly ambitious or overly simplistic projects.
- Be prepared for the possibility of unforeseen challenges during implementation or evaluation.

Results:
- Aim for meaningful results; surpassing benchmarks is not mandatory.
- Avoid outcomes like a 1% overall accuracy or an F1 score of 125.

Dataset:
- Optimize time by using complete, pre-labeled datasets.
- Consider benchmark datasets or open-source alternatives discussed in the course.

6. Final submission and deliverables:

Submission is through Moodle, and no other method of submission will be accepted.

You should submit the following files:

- Report (pdf) - check the Written Report section above for details on the content
- Code deliverables check Implementation and deliverables section above for a detailed outline of the needed files.
- Presentation - will be online, you must be presented during your assigned slot on the given conference link.

Coding & Referencing:

This assignment primarily involves coding. If you use code or any other materials authored by someone else, it is essential to provide proper citation. Failure to appropriately cite work constitutes **Academic Misconduct** and will result in appropriate consequences. Superficial modifications to the code do not establish it as your own. Refer to the addendum to the guidelines or consult the module leader for further clarification.

Extenuating Circumstances:

If unforeseen medical or personal circumstances prevent you from submitting your coursework on time, promptly contact the Programmes Office. Fill out an Extenuating Circumstances form and provide strong, genuine evidence, such as medical certificates or legal statements, to support your case.

Plagiarism:

Copying the work of others, whether from another team or a third party, with or without permission, will result in zero marks, and disciplinary action will be taken. The same consequences apply if you allow others to copy your work. Refer to the addendum to the guidelines or consult the module leader for additional clarification.

Feedback

In the labs we can check your progress and give formative feedback.
Evaluative feedback and marks on your coursework will be released after the presentations.

1. Main Points

The primary objective of the coursework is to demonstrate the knowledge you have gathered. Focus on choosing a problem that is sufficiently challenging to showcase understanding but not overly trivial. Avoid simplistic tasks, such as a plug-in classifier, that may not adequately demonstrate your knowledge.

2. Dataset Selection

- Start by identifying a labeled open-source dataset.
- Minimize or eliminate time spent on labeling; the emphasis is on models rather than data engineering.
- No strict preferences for datasets; feel free to use any open-source dataset, including those provided in the module - always provide link or access to the dataset.

3. Problem Definition

- Clearly define the problem you intend to address and explain why it is interesting.
- Set realistic objectives, considering the limited time available.

4. Model Selection

- Choose an open-source model suitable for the identified problem.
- Provide a rationale for why the selected model is a good fit.
- Don’t just use the model out of the box - come-up with additions, provide the reasoning for the changes.

5. Accuracy Metrics

- Define the accuracy metrics you plan to use.
- Explain why these metrics are appropriate for the specific problem.

6. Collaboration and Approval

- Prior to starting the coursework, discuss your plan with the instructor to receive feedback.

7. Pair Work Considerations

- If working in pairs, each student must select a different section for work.
- Grades will differ based on individual contributions.

8. Reporting Guidelines

- Present important details clearly and readably.
- Enumerate tables, equations, figures, etc., and refer to them correctly in the text.
- Discuss key concepts rather than restating or rewording published papers.

9. Conciseness and Focus

Exclude discussions on basic and widely-known concepts, such as ConvNet and MLP layers, learning rate, weight decay, batch, etc. Avoid in-depth discussions on older models like (Fast) R-CNN. However make sure you reference them and specify why they are relevant to YOUR project.

- Succinctly discuss basic classifiers like ResNet, focusing on their specific methodology.
- Present mathematical ideas, such as optimizer functionality and loss functions, succinctly with equations and explanations.

10. Code Quality

- Ensure all code submitted is meaningful.
- Models must train effectively, and results should reflect the intended metrics. Your grade will suffer if your model was not able to learn anything.

11. Models that are too basic

- Avoid the use of overly basic models like MLE and NNLM from Weeks 2 and 3.

12. Evaluation Criteria

- Hyperparameters and model discussions are essential, but code quality and meaningful results take precedence.
- Avoid extreme values in metrics (e.g., 100% or 0%) that may indicate bugs; strive for realistic performance.