Artificial Intelligence II

Deep Learning for Natural Language Processing

Spring Semester 2025-2026

Homework 2

Weight: 25% of Course Grade

Announced: April 1, 2026

Due: April 24, 2026, before 23:55

Description

You will develop a classifier for the task of **response clarity classification** by fine-tuning transformer-based models on a dataset of political question–answer pairs (CLARITY dataset). Each instance consists of a question and its corresponding answer, and your goal is to classify the answer into one of the following categories: *Clear Reply*, *Ambivalent*, or *Clear Non-Reply*.

The models you will use are:

- The pretrained model `bert-base-uncased`.
- The pretrained model `distilbert-base-uncased`.
- The pretrained model `microsoft/deberta-v3-base`.

In this homework, you should use the machine learning framework PyTorch and the Hugging Face Transformers library, following the official sequence classification guide and the AutoModel/AutoTokenizer guide.

You must implement the training procedure with your own explicit training loop (do not use the Hugging Face `Trainer` API).

Before you start the homework, make sure that you have studied the relevant slides of the course (PDF files *"Transformers"* and *"BERT"*), as well as any additional material on transformer-based models that you may find useful.

It is your responsibility to choose all the details of developing a good model (e.g., whether to use validation splits, regularization techniques, which optimizer to use, how to tune hyperparameters, and how to ensure that your model does not underfit or overfit). You are encouraged to experiment with larger or stronger variants of these models (e.g., BERT-large, RoBERTa-base, RoBERTa-large, or DeBERTa-v3-large) and compare their performance with the base models.

Guidelines

This task consists of the following main steps:

1. **Input construction:** Decide how you will combine the question and the answer into a single model input (e.g., concatenation with separator tokens or a structured input format). State your design explicitly.

2. **Tokenization and encoding:** Use the appropriate pretrained tokenizer for each model and prepare the inputs for transformer-based sequence classification.

3. **Model development:** Fine-tune the required pretrained transformer models for the clarity classification task.

4. **Training and optimization:** Explore training decisions such as learning rate, batch size, number of epochs, maximum sequence length, regularization, class imbalance handling, and validation strategy.

5. **Evaluation and comparison:** Compare the required models and analyze their performance. On Kaggle, the primary evaluation metric is **F1-score**, while **Macro F1-score** should be reported as an additional metric. In your report, include **accuracy**, **precision**, **recall**, and **F1-score**.

6. **Error analysis:** Analyze the failure cases of your models and discuss what makes some examples more difficult than others.

It is your responsibility to choose all details of developing an effective model, including hyperparameter choices, optimization strategy, validation setup, and ways of preventing underfitting or overfitting.

For implementation purposes, students **must use a random seed** to ensure reproducibility.

Presentation of Results: Ensure that your results are supported with clear, high-quality, and well-labeled plots that effectively illustrate your findings. For example, verify that your model does not suffer from underfitting or overfitting.

Additional Considerations: You are encouraged to mention in the appendices of your report any other approaches you explored that did not improve the model's performance but contributed to your understanding of the task.

Experiments

You must conduct a systematic comparison between the required models. Your experimental analysis should go beyond reporting final scores. In particular, you should investigate:

- the effect of hyperparameter choices,
- the effect of input formulation choices,
- the effect of model capacity on performance and generalization.
- differences between smaller and larger models,
- performance across different data subgroups (e.g., categories based on question length and answer length).

Your discussion must explain observed differences in performance rather than simply listing numerical results.

Error Analysis

A central component of this assignment is the analysis of model failures. You must perform a detailed error analysis and discuss questions such as:

- Which class is hardest to predict?
- Are there recurring linguistic patterns in incorrect predictions?
- Do stronger or larger models fail differently from smaller baseline models?

- What is the fundamental source of error: what do models fail to understand in question–answer interactions?
- What characteristics would an ideal model need in order to improve performance on this task (even if not implemented in this homework)?

You must include concrete examples of errors and organize your discussion in a meaningful way. In addition, you must describe:

- what changes you attempted in order to improve your models,
- whether these changes improved performance,
- why you believe these changes helped or failed.

The main objective is not only to obtain a strong classifier, but also to understand why models succeed or fail on this task.

Kaggle Competition

Submit your code as a **Jupyter Notebook** via the Kaggle competition. Follow these rules:

- Your team name must be your academic identification number (Αριθμός Μητρώου - sdiXXYYYYY or the one for graduate students).
- You must create and submit **three different Kaggle notebooks**, one for each required model (`bert-base-uncased`, `distilbert-base-uncased`, `microsoft/deberta-v3-base`).
- Notebook naming format (use your academic ID): `[academic-id] bert-base-uncased`, `[academic-id] distilbert-base-uncased`, `[academic-id] microsoft-deberta-v3-base`. Share all notebooks on Kaggle with the Teaching Assistant responsible for grading this assignment (Kaggle username: `despinap`).
- Each notebook must output a prediction file in csv format with clear model-specific naming:
  - `submission_bert-base-uncased.csv`
  - `submission_distilbert-base-uncased.csv`
  - `submission_microsoft-deberta-v3-base.csv`

  Do **not** submit as a file upload.
- On Kaggle, the primary metric is **F1-score**; report **Macro F1-score** as an additional metric in your analysis.
- The resulting file must follow the format specified in the provided `sample_submission.csv` file and must contain the predictions that your model makes over the test set.
- In your report, clearly state which model performed best overall and explain in detail why, based on your evaluation findings.
- **Do not share your notebook publicly.**

Report

For this project, you are asked to create a detailed report. For this reason, we provide you with a LaTeX template. You may use the Overleaf online editor. Open Overleaf, create an account if you do not already have one, and upload the provided zip file by selecting *New Project* and then *Upload Project* (the template uses the pdfLaTeX compiler). If you encounter difficulties writing in LaTeX, you may initially draft your report in another editor following the structure of the template; however, you are strongly encouraged to produce the final version in LaTeX, as tools such as Overleaf or Prism can significantly simplify the writing process.

Your report must include: a description of the models you used; your input representation choices; your training setup and hyperparameter choices; your evaluation results; a comparative analysis of the models; a detailed error analysis; and a discussion of the modifications you tried and their effect.

The emphasis of the report is not only on predictive performance, but also on: understanding model behavior; explaining performance differences; analyzing model limitations; and justifying experimental decisions.

The report must be written in English for students in the Master's program in Data Science and Information Technologies. All other students may choose their preferred language. The report must include links to your three Kaggle notebooks, which must be shared with the teaching assistant and must not be made publicly available.

Grading

Implementation: Code, Kaggle submission [**Total 70%**]

- Data handling and preprocessing: [**10%**]
- Model development: [**20%**]
- Experiments: [**30%**]
- Fine-tuning & Optimization: [**10%**]

Report: Analysis and Presentation [**Total 30%**]

- Experiments: [**10%**]
- Analysis: [**15%**]
- Plots: [**5%**]

Important grading note: Code submissions without clear model-by-model explanation and analysis in the report will **not** receive implementation credit.

Submission Guides

We expect you to:

1. Submit your **three executed Jupyter Notebooks** (one per required model, with outputs) and make them available to supervisors in **Kaggle** only.
2. Submit your report in **.pdf** format in e-class. Name your report as [**full-id**]**.pdf**.

For example: `ZZZZZZXXYYYYY.pdf`
**We won't accept code submissions from e-class, e-mails, etc.*

Support

Despina-Athanasia Pantazi is supervising this assignment. For any questions, please post them on Piazza.