Notebook checklist FS25

General criteria

Reproducibility:

- Notebook can run from beginning to end without error
- Controlled for randomness by e.g. fixing the random seed
- Experiment tracking matches the training and results

Form:

- Correct terminology
- No explanations of NLP concepts (ChatGPT will do this)
- Documentation uses proper markdown formatting
- Documentation is complete
- Documentation is structured and formulated in an understandable way
- Documentation is short and precise
- Code cells' outputs are shown (save them in the notebook)
- Code is reasonably easy to follow (structure, variable naming etc.)
- Code is correct
- Code includes tests
- Code uses library functions to simplify code
- Code avoids unnecessary operations/code redundancies (source of errors)
- The acknowledgment of any external sources used/AI-generated content is as clear and specific as possible

Experiment tracking:

- Important metrics tracked (train loss, train accuracy, val loss, val accuracy)
- Report or view present and linked in introduction
- Report/view organized (important metrics shown, useful ordering, unimportant metrics hidden, useful smoothing settings)
- Useful naming of runs
- If many runs shown, useful ordering (e.g. by validation loss, not by date)

Notebook sections

Introduction: Executive summary of project

- Includes topic of project, description of data, description of method, experiments performed, results
- Experiment tracking link present
- Short and precise

Setup: Install libraries, imports, library settings

- Complete
- No unnecessary imports/installs

Preprocessing: Data loading, preprocessing (includes converting to tensors, batching), input format

- Preprocessing decisions match the model and task
- Correct and justified decisions on:
    - Tokenization
    - Lowercasing, stemming, lemmatizing, stopword/punctuation removal
    - Removal of unknown/other words
    - Format cleaning (e.g. html-extracted text)
    - Truncation
    - Feature selection
    - Input format: how is data passed to the model?
    - Label format: what should the model predict?
    - Train/valid/test splits
    - Batching, padding
    - Vocabulary, embedding

Model: Model selection, model architecture, classifier

- Correct and justified model architecture:
    - Pretrained model checkpoint (if applicable)
    - Network architecture (layers, dims, nonlinearities, regularizations, normalizations, classifier)
    - Loss, optimizer (either here or in the training section)

Training: Training code, hyperparameters, training runs, selection of best model, early stopping

- Correct and justified decisions on:
    - Loss, optimizer (either here or in the model section)
    - Experiment design (model/optimizer/loss variants, hyperparameters)
        - What do we want to find out?
    - Number of training runs
    - Model checkpointing
    - Early stopping
- Optional, but recommended: automated training runs (later: sweeps)

Evaluation: Code to compute results, presentation and description of results

- Metrics and confusion matrix present
- Good error analysis (e.g. analyze the examples where the model makes mistakes, try to identify patterns)
- Clean graphics, labeled axes (if used), no screenshots
- Objective description of results
- Subjective interpretation and speculation not present (belong in next section)

Interpretation: Personal interpretation of results, learnings

- Stated and justified expectations before the project
- Comparison of results to expectations
- Comparison to other architectures (for project 2, compare to project 1 as well)
- Constructive interpretation
- Speculation grounded in results/error analysis
- Take-aways/summary of the project results

Tools used:

- Tools used are stated (includes AI tools)