Contents



1 IN5550 Obligatory 1 common issues

Results reporting



Common issues

- Did not organize the results in tables and used screenshots of terminal output or numbers scattered through text instead
- Did not number tables
- Tables that exceed the text margin
- Used abbreviations or new terms for experiment set ups but never introduced/explained what they mean
- Lengthy (8 pages) and detailed explanations of code (how it was implemented)
- Did not report the exact IDs of word embedding models
  - e.g., I compared a word2vec model and a fastText model.
- Limited interpretation of the results, reporting only numbers

Results reporting

Do's and dont's

Do's

- Put your results in numbered tables with clear column and row names and titles
- Discuss and interpret your results by clearly referring to the relevant table
- Clearly introduce any used abbreviations/terms at their first appearance
- What you experiment, what you observe, and what you can infer from the results.
- Store large files in the project work directory on Fox (`/cluster/work/projects/ec403/`).

Dont's

- Report results directly in text or in screenshots
- Use unnumbered tables, cryptic columns and rows
- Put detailed explanations of code in the report (they should be in code comments)
- Store files (models, datasets, etc) in your home directory on Fox (which we cannot access, of course)

Data preprocessing



Common issues

- Always lower-casing the input text before feeding it to the word embedding model
  - Not all word embeddings use lower-cased tokens!! For example, models 10 and 222 from `https://vectors.nlpl.eu/repository/` and on Fox preserve the words case
- Did not specify the creation of the train/dev/test splits or mention the split twice in the report but with different splits

Do's and dont's

Do's

- Make sure the words are in the format expected by the word embedding model you are loading.
- Always look at the data (distribution), be creative about pre-processing and cleaning (e.g., remove digits), and remember to do this for the test set as well.

Dont's

- Use the same preprocessing method regardless of what the word embedding model takes as input.

Model Building and Training

Common issues

- Did not use early-stopping!
- Requesting the default 4 CPU cores instead of asking 16 or 32 for much faster training

Do's and dont's

Do's

- Use a training stopping criterion (early-stopping)
- Plot the training and validation loss
- Request more resources when necessary, to save training time

Dont's

- Tune the number of epochs as a hyper-parameter
- Decompress the datasets manually; `pandas` can read and write compressed files
- Rebuild wheels if there are good available implementations
  - e.g., implementing own BoW vectorizer instead of using CountVectorizer() from scikit-learn

Code cleanliness

Common issues

- No repository organization
- Classes and functions duplicated in several different scripts across the repository

Do's and dont's

Do's

- Use `.py` scripts instead of Jupyter notebooks
- Write clean code (well-organized, good names, documentation etc.)

Dont's

- Duplicate code
- Automatically generate `.py` scripts from notebooks without cleaning it up
- Forget to provide a link for your GitHub repository or model path.

Evaluation



Common issues

- Evaluation script is in a different folder with the data loading and model scripts which are used in the evaluation script.
- Evaluation script doesn't work with the final model or no instruction is provided.
- Evaluation script was deleted and only the training script is provided for training and testing.
- Did not add the data preprocessing method (which was used on the training data) in the evaluation script.

Evaluation script

Do's and dont's

Do's

- Make sure it can be run on a different set than the one it was trained on.
- Load everything that's needed (model, training vocabulary, etc.)
- You can add extra modules such as the data pre-processing logic used for the training data.

Dont's

- Remove the evaluation script.
- Train the model again at test time.
- Put the evaluation script and the modules it needs at different folders.

More issues

Some more issues

- Not giving permission to all the teachers (you can find us at `https://github.uio.no/orgs/in5550/teams/2026`)
- Forking the whole course GitHub repository for this assignment
- No name on the report
- Putting excuses/disclaimers in your report
- Not addressing explicit tasks from the obligatory text

More issues



Some more issues

- Not giving permission to all the teachers (you can find us at `https://github.uio.no/orgs/in5550/teams/2026`)
- Forking the whole course GitHub repository for this assignment
- No name on the report
- Putting excuses/disclaimers in your report
- Not addressing explicit tasks from the obligatory text
- Inconsistency in results reporting between different parts of the report (eg. one section says split the data into train/dev/test, while another section says train/dev)
- Please always declare the full composition of your team when submitting the assignment in Devilry (all the team members!)