Next milestone (29.07.)

Implement neural embeddings – **either** hardcode or softer version, using PyTorch

- Hardcode version:
    - No PyTorch, no ML libraries, **only numpy** (at least for the neural embeddings, can use PyTorch, etc. for GPT implementation)
        - Can use Counter and defaultdict from Collections
        - We can, but do not **have** to hardcode the optimiser (can use Adam, need to use at least SGD)
    - Measure perplexity
    - Implement early stopping (when validation error/loss diverges from training error to avoid overfitting to training set) with patience
        - Do not need to optimise for patience, but can
        - Save top k (the amount that fits reasonably on your disk) of model checkpoints (can name that file for validation score and iteration)
- Softer version:
    - Using PyTorch
    - Implement early stopping (when validation error/loss diverges from training error to avoid overfitting to training set) with patience
        - Do not have to optimise patience, but can
        - Save top k (the amount that fits reasonably on disk) of model checkpoints (can name that file for validation score and iteration)
    - Tune hyperparameters using a grid search **for each separately** and validation set (order is important: number of merges, learning rate, weights of interpolation) – do not have to do all of this to **pass**, but for 1.0
        - vocabulary size – gridsearch for max. 10 different amounts of merges
        - learning rate of optimiser
        - interpolation
    - Try versions with different optimisers
- Watch RAM during training, especially for higher batch sizes (>=32)