Home Assignment 02

Text Mining — Due by **June 20th, 2025 at 23:59 CET**

Sandipan Sikdar, Jonas Wallat, Huyen Nguyen — Summer Semester 2025

Submission Details

- Please submit a single compressed folder (.zip) containing one PDF file with all text responses and one Jupyter Notebook for the coding tasks. Please keep all generated outputs in the Jupyter Notebook so we can grade your submission without rerunning your code.
- For each answer/code snippet, clearly state what task you are referencing. If you are not confident in an answer, try nonetheless, as it is possible to receive partial points.
- You are encouraged to work in groups of up to three students. Add all team members' names and matriculation numbers at the top of the PDF file **and** your notebook.
- You may use online resources for help, but copying online resources is prohibited. Cite or link any resources you use.
- If you have any questions, create a post in the forum.

1 CNN Classification [30 points]

In this task, you will perform a multi-class emotion classification using a convolutional neural network. The file *emotion.csv* contains 16000 sentences along with class labels. Follow these steps:

- Use a 80/10/10 train/val/test split[1], to devide the dataset into 80% training, 10% validation, and 10% test data.
- Use pretrained GloVe embeddings or initialize embeddings randomly.
- Use PyTorch to create dataloaders, build and train a CNN model with Adam optimizer and Cross Entropy Loss.
- Compute the loss value on the validation data after each epoch and save the model that achieves the lowest validation loss.
- Evaluate the saved model on the test data. Report accuracy, F1 score, and a detailed classification report[2].

2 RNN [30 points]

In this task, you will build a Recurrent Neural Network (RNN) to perform next-word prediction on a text corpus. Given a sequence of words, your model should predict the next most likely word. You will use the "*A Song of Ice and Fire*" books a.k.a. *Game Of Thrones* (*GoTBooks.zip*)

[1] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
[2] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

to achieve this. The .zip file contains five files, one for each book in the series. Apologies for any spoilers. Depending on your computational resources, you may choose to use one or all of the files. Follow these steps:

- Load and preprocess the data. Preprocessing includes lower-casing all words, removing line breaks, punctuation, removing all alpha-numeric characters (leaving only words and numbers), collapsing multiple spaces, and removing single-character words (to prevent noise).
- Tokenize[3] the text and build a dictionary that maps the tokens to indices. Convert the corpus into sequences of fixed length n (e.g., 5), where the first n-1 words are inputs and the last word is the target. The sequences should be generated by sliding a window of size n over the tokenized text with a stride of 1, so that each new sequence overlaps with the previous one by n-1 words.
- Split the dataset into training (80%), validation (10%), and test (10%) sets.
- Build a RNN model: Use an embedding layer followed by the RNN and a fully connected layer to output predictions over the vocabulary. Train the model using Cross Entropy Loss and Adam optimizer.
- Evaluate validation loss after each epoch and save the model with the best validation performance.
- Evaluate the saved model on the test set using accuracy.
- Think of some examples, provide them to the trained model, and print out the top 3 predicted next words.

3 Sequence-to-Sequence Modeling [40 points]

We provide you with a processed version of the topical chat dataset[4]. In the file topical_chat_pairs.csv you can find chat interaction pairs containing a message and an answer to the message. Your task is to train a sequence-to-sequence model on these message and answer pairs. Note: Depending on your compute limitations, you are free to use a subset data to train your model. For decoding, use Top-k sampling. Continue generating until you have generated a pre-defined number of tokens (t) or the <EOS> token. The first token in the decoder is the <SOS> token. You are free to choose k and t.

- Split the dataset into sensible train, validation, and test splits
- Construct your sequence-to-sequence model
- Evaluate the validation loss after each epoch and plot the results (you are free to use tensorboard, weights and biases, or similar tools for this)
- Print 5-10 test set examples and answers generated by your best model

[3] https://www.nltk.org/api/nltk.tokenize.html
[4] https://www.kaggle.com/datasets/arnavsharmaas/chatbot-dataset-topical-chat