IS 584: Deep Learning for Text Analytics
Assignment 1

Volga Sezen

Due by 09.04.2025

In this assignment you will analyze meta reviews of submissions to a given conference to determine whether they were accepted or rejected. You will train your own network, and compare it's level of success with one that borrows embedding matrices from GloVe. Finally, with the best version, you will investigate using lemmas instead of stems.

This data was originally scraped from the Open Review database by Weizhe Yuan, Pengfei Liu and Graham Neubig as part of ASAP-Review dataset. A subset with only decisions and reviews for submissions from 2017-2020 to ICLR (International Conference on Learning Representations) is provided. metaReview column was edited such that sentences with the words "accept" or "reject" were omitted.

Also keep in mind the accept class has many subclasses that are in order of "strength": Poster/Workshop, Spotlight and Oral/Talk. Recommended label groupings are: Reject, Accept (Poster), Accept (otherwise).

Disclaimer

This is an individual assignment. Please refrain from collaboration. Upon any hurdle, you can contact your TA. It is important you adhere to academic integrity principles.

LLM's can only be used to paraphrase your writing or to help diagnose coding errors. You have to make decisions, write the code, execute it and report the outcomes yourself.

Deliverables

- An ipython notebook with versions of major packages used (pytorch, torchtext etc.), cell outputs and random seeds visible.
- A pdf report. This should include high resolution exports (dpi $\geq$ 200) of any graphic and make comparisons using tables. Besides visual elements, provide all commentary on the output in the report.

Late Submissions

Late submissions will be accepted until 13.04.2025 with a 5% grade reduction each day. (For a total of 20% at maximum)

1. Data Exploration and Preprocessing (20p)

Download the dataset from the attached json file. (Can be read with pandas.read_json) Report on the dimensionality and quality of the data. (Lengths, unique values, distributions etc.) Apply data quality checks and comment on the state of the dataset.

Generate word-level tokens, decide and implement a number of rules to filter unrelated tokens and normalize the rest with a stemmer. Document how long it took to preprocess one example on average. Show example strings and their processed versions, and highlight any unique approach.

2. Language Modeling (30p)

2.1 Initialization

Set a seed in PyTorch and any pandas function you use. Make sure to document what that seed is so your analysis is reproducible. Separate the preprocessed data into train-val-test splits with 0.7-0.15-0.15 ratios. Create necessary functions to convert word indices to tensors and create dataloaders.

Define a fully connected network with the following layers: An EmbeddingBag layer with an embedding size of 100 and a hidden layer size of 20 with relu activation. Pick a loss function and an optimizer, with an initial learning rate. (This can be tweaked in the next subsection.)

2.2 Training and Evaluation

Train your neural network for at least 10 epochs. Monitor under or overfitting with training and validation batch performance metrics. Report these values with graphs. Make sure the model does not underfit the data. (A bit of overfit is fine, but ideally utilize early stopping.)

Generate predictions for the test set and report the performance with a confusion matrix as well as central metrics like precision, recall, F-scores and cohen's kappa. Comment on the results.

2.3 Error Analysis and Interpretation

Identify misclassified examples and comment whether the meta reviews share common points that lead to failure. Also express the confidence score of the model on these examples, and highlight predictions with the least confidence.

Then, select words from your vocabulary (can use most common unique ones for each class). Plot learned embeddings of these words using PCA and T-SNE projections. Investigate whether there are identifiable clusters and relationships.

3. Pre-Trained Embeddings (35p)

Download 100 dimensional pre-trained GloVe weights. Using torchtext utilities, generate a new embedding matrix for the words in your dictionary. Copy its weights inside a freshly initialized network and finetune it on the dataset. Report on the performance and compare metric values with those observed in part 2.3.

Compare projected embedding vectors from GloVe with those observed in part 2.3. Are similar relationships kept?

4. Improved Preprocessing (15p)

Create a new version of the dataset. During preprocessing, use a lemmatizer for normalization. Also consider each words' part of speech tag for the normalization. Document how long it took to preprocess one example on average.

Pick either randomly initialized embeddings or pre-trained ones. Train an otherwise identical network on this new dataset. Report on the performance and compare performance metrics as well as preprocessing time with those observed in part 2.3 or part 3.

Look back on the misclassified examples. Have their class predictions or confidence scores improved?