CS4375 Assignment 2

FULL\_GITHUB\_REPOSITORY\_URL

|  |  |  |
| --- | --- | --- |
|  | Gregory Rodriguez GXR230010 |  |

# Introduction and Data (5pt)

**TODO:** Briefly describe the project and your main experiments and results, including mentioning the data you use.

**TODO:** Briefly describe task and data (e.g. how many examples are in the training, development, and test sets.), it is best to report all the statistics, including counts, in a table. including how you treat casing, tokenization, and anything else that you did to the raw data before computing features from it.

**####**

The project involves the implementation and evaluation of neural network models for sentiment analysis on Yelp reviews. The main experiments revolve around employing two types of neural networks: a Feedforward Neural Network (FFNN) and a Recurrent Neural Network (RNN). The data provided consists of train, validation, and test sets of Yelp reviews, each associated with a rating ranging from 1 to 5. The task is to predict the sentiment rating based on the review text. No further preprocessing of the data is required for this assignment.

TABLE

DATASETS # OF EXAMPLES

# Implementations (45pt)

## FFNN (20pt)

**TODO:** Explain briefly how you implemented filled in the incomplete code for FFNN.py in the form of screenshot (and explanations) in the report. Provide any other libraries/tools that are used; tutorials/materials that you referred to; or how you were doing debugging. Try to understand what other part of the code is doing, and write your understandings here (e.g. optimizers, initializations, stopping, etc.)

###

In the FFNN implementation (ffnn.py), the provided code is completed by implementing the forward computation method. This method involves obtaining representations for the input, hidden layer, and output layer, followed by applying softmax to obtain the probability distribution. Libraries such as PyTorch are utilized for neural network operations, including modules for linear transformations, activation functions (ReLU), softmax, and loss computation (NLLLoss).

In completing the code, references to PyTorch documentation and related tutorials were used for understanding and debugging. The code follows a structured approach, with data loading, vectorization, model instantiation, training loop, and evaluation loop.

## RNN (25pt)

**TODO:** Explain briefly how you implemented filled in the incomplete code for RNN.py in the form of code-snippet screenshot (and explanations) in the report. Provide any other libraries/tools that are used; and tutorials/materials that you referred to. Try to understand what other part of the code is doing, and write your understandings here (especially parts that is functioning differently as compared to FFNN).

**###**

The RNN implementation (rnn.py) involves similar steps to the FFNN, with additional considerations for processing sequential data. The RNN module from PyTorch is employed to process sequences of word embeddings, with a linear layer for output prediction. Adam optimizer is used for optimization.

Similar to FFNN, PyTorch documentation and tutorials were referred to for understanding and debugging. Notably, word embeddings are utilized for input representation, and the RNN model iterates over each word in the input sequence to compute representations.

# Experiments and Results (25pt)

**Evaluations (5pt) TODO:** Explain how you evaluate the models. What metric is used – you can refer to the current implementation.

The models are evaluated based on accuracy metrics, comparing predicted labels with ground truth labels. The provided implementation computes loss during training and utilizes accuracy metrics during evaluation.

**Results (20pt) TODO:** Apart from the default hyperparameters, try multiple variations (between 1-2 for FFNN and RNN each) of models by changing hidden unit sizes.

**TODO:** Summarize the performance of your system and Put the results into tables or diagrams and include your observations and analysis.

**TODO:** (Extra Bonus), try other variations of the model by changing the default code and report resutls, e.g. you can try different number of layers, initializations, etc.

Multiple variations of the models are experimented with by adjusting hyperparameters, particularly hidden unit sizes. Performance results are summarized and presented in tables or diagrams. Observations and analyses of the results are included, highlighting the impact of hyperparameter variations on model performance. Additional experiments beyond default hyperparameters, such as varying the number of layers or initialization techniques, may also be conducted and reported.

# Analysis (20pt)

**TODO:**

* (10pt) Plot the learning curve of your best system. The curve should include the training loss and development set accuracy by epoch.
* (10pt) Error analysis. List some (one or more than one) error examples and provide some analysis. How might you improve the system?
* (Extra Bonus) other analysis and discussions.

# Conclusion and Others (5pt)

**TODO:**

* Individual member contribution.
* Feedback for the assignment. e.g., time spent, difficulty, and how we can improve.

2