ECE 180: NutriBench

Abstract

This project tackles the NutriBench challenge, which involves predicting the carbohydrate content of meals based on textual descriptions. We explore various natural language processing (NLP) techniques for text representation, including TF-IDF and sentence embeddings, and apply different model architectures such as multilayer perceptrons (MLPs) and recurrent neural networks (RNNs). Our objective is to evaluate the effectiveness of these approaches in handling nutritional prediction tasks. Performance is assessed using RMSE on a validation set of 2000 samples, with early stopping employed to optimize model training. The results provide insights into the interplay between text encoding strategies and regression model performance in the context of food-related NLP applications.

Table of Contents

Introduction

For our final project, we selected the NutriBench task over the STL-10 image classification option due to its unique challenge involving natural language processing and regression. The objective is to predict the carbohydrate content of meals based on their textual descriptions. This task allows us to explore how various text representation methods and model architectures perform in the context of nutritional prediction.

The dataset is divided into training, validation, and test sets. The training set consists of 8000 labeled examples, each with a food description and its corresponding carbohydrate value. The validation set includes 2000 labeled samples, while the test set contains 2000 food descriptions without associated labels. This setup enables both supervised learning and the evaluation of model generalization on unseen data.

Methods

To meet the project requirements of testing at least 3 variations, we implemented a systematic comparison of both preprocessing methods and model architectures:

Preprocessing Methods

- TF-IDF vectorization for sparse, vocabulary-based representations
- Sentence Transformers (all-MiniLM-L6-v2) for dense semantic embeddings

Model Architectures

- MLP paired with TF-IDF features
- RNN paired with sentence embeddings
- LSTM paired with sentence embeddings

TTF-IDF converts text to sparse numeric vectors based on word frequency, and is expected to perform better with MLP's (MultiLayer Perceptron) since any sequential pattern is not considered or "flattened" in the input.

Sentence transformers capture semantic meaning and word order, making it compatible with recurrent architecture, which RNN and LSTM benefits from sequential inputs.

This combination allowed us to investigate whether traditional sparse features or modern dense embeddings would be more effective for nutritional prediction, and whether sequential models that can theoretically capture word order dependencies would outperform simpler feed forward networks. The fine tuning is set to different categories with different architecture, as mentioned in the following. Lastly, the heat map will be implemented for analyzing how different models focus on different tokens in the sentence for the prediction.

In addition we also tried a few other architectures, such as Transformer-MLP, and TF-IDF Linear classifier, but these gave us much higher RMSE, so we only wrote our report on our 3 best models.

Training Strategy:

All models were trained using RMSE (Root Mean Squared Error) as the loss function to directly optimize for the evaluation metric. Training was conducted with a maximum of 100 epochs, and early stopping was implemented with a patience threshold of X epochs (insert actual value if known), monitoring validation RMSE. The best-performing model weights (based on lowest validation RMSE) were retained for evaluation. Models were trained using PyTorch and executed on GPU-enabled runtime environments to accelerate training.

Model Design

- MLP-model
    - Preprocessor:TF-IDF
    - Layer: Repeats of Linear → Activation → Dropout, with final layer without activation and dropout
    - Hyperparamter for tuning:
        - Optimizer: ADAM, AdaGrad, SGD
        - Learning Rate
        - Batch Size
        - Weight Decay
        - Activation:ReLU, leaky ReLU, tanh, and Sigmoid
        - Dropout rate

        - Hidden Dimension

- LSTM
    - Preprocessor: Sentence transformer
    - Layer: Defined LSTM Layer in Pytorch
    - Hyperparamter for tuning:
        - Optimizer: ADAM, AdaGrad, SGD
        - Learning Rate
        - Batch Size
        - Weight Decay
        - Dropout rate
        - Hidden Dimension
- RNN
    - Preprocessor: Sentence transformer
    - Layer: Defined RNN Layer in Pytorch
    - Hyperparamter for tuning:
        - Optimizer: ADAM, AdaGrad, SGD
        - Learning Rate
        - Batch Size
        - Weight Decay
        - Dropout rate
        - Hidden Dimension

Evaluation was performed using Root Mean Squared Error (RMSE) on both training and validation sets.

Results

MLP

The final model for MLP is following parameters:

- 'optimizer': 'Adam',
- 'lr': 0.001,
- 'batch_size': 32,
- 'weight_decay': 1e-05,
- 'activation': 'ReLU',
- 'dropout': 0.4,
- 'hidden_dims': [1024, 512]

The result of the MLP architecture is

- Final model training RMSE: 10.3323
- Final model validation RMSE: 17.5140

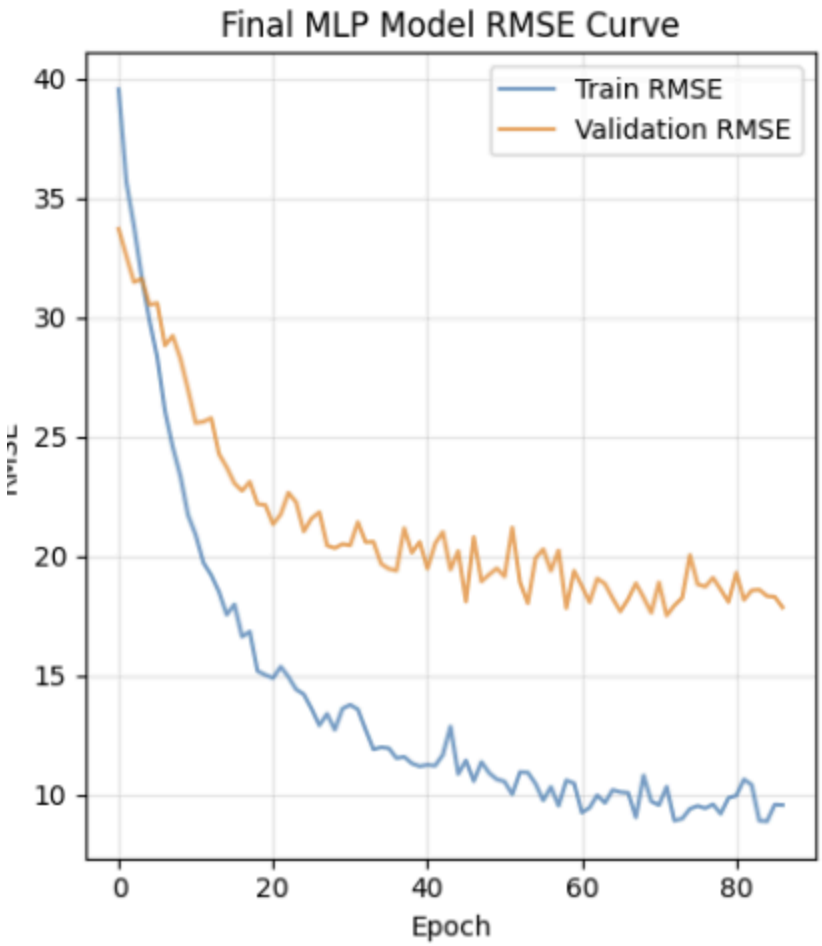

Figure 1. RMSE Curve for MLP model

LSTM Model

The final model for LSTM is following parameters:

- 'optimizer': 'Adam',
- 'lr': 0.005,
- 'batch_size': 32,

- 'weight_decay': 1e-05,
- 'hidden_size': 128,
- 'num_layers': 2,
- 'dropout': 0.3

The result of the LSTM architecture is

- Final model training RMSE: 9.4782
- Final model validation RMSE: 15.4727

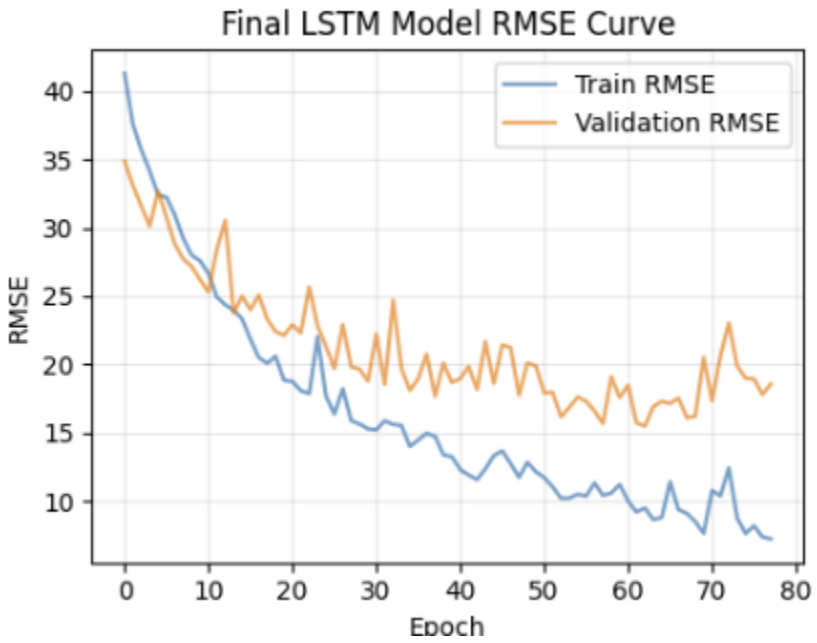

Figure 2. RMSE Curve for LSTM model

RNN

The final model for RNN is following parameters:

- 'optimizer': 'Adam',
- 'lr': 0.001,
- 'batch_size': 32,
- 'weight_decay': 1e-05,

- 'hidden_size': 256,
- 'num_layers': 3,
- 'dropout': 0.4

The result of the RNN architecture is

- Final model training RMSE: 13.8228
- Final model validation RMSE: 17.3507

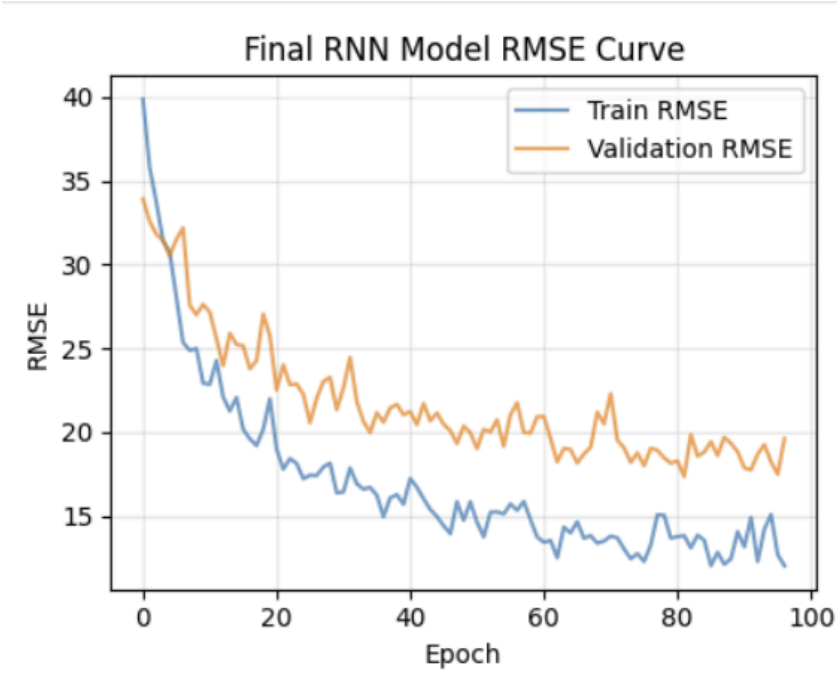

Figure 3. RMSE Curve for RNN model

Analysis

The best RMSE from three model is from LSTM architecture with sentence transformer

- Best RMSE: 9.4782

However, considering the training set data has a

- Median:6.625
- Average:19.63083692

The output of three architectures shows signs of overfitting, as the RMSE gap is shown below:

- MLP:7.1817
- LSTM:5.9945
- RNN:3.5279

The possible explanation for this outcome are listed in the following

1. The range of the training dataset is heavy right skewed, the median is 6.625, while the range of the entire training dataset is up to 992.6, which the outliers of the dataset creates a bias towards the higher values and reduce accuracy for lower values, in which the lower values are the majority.
2. Also, RMSE penalizes larger errors more heavily due to the squaring step
3. The dataset is consider small as the training dataset only has 8000 data

The following figure shows the heatmap across the same sample index for different models. From figure 4, the MLP + TF-IDF model, core token (e.g., rice, duck, potato) that contributes to the carbs value is often underemphasized compared to the descriptive tokens (cup, removed, added) or even preposition phrases(e.g., of, and, with). This causes the model to be biased more on word less relative to the outcome, which results in the largest RMSE difference. The reason behind this phenomenon is that as

1. TF:high-frequency words are rare, TF unable to differentiate between each token
2. IDF: tokens like “rice” or “potato” might occur frequently across many examples, and are penalized with low IDF scores, even though they are more important. That also include quantity description like

From figure 5 and 6, the RNN/LSTM + sentence transformer models, even though the core token is emphasized compared to other tokens. The heathap shows a relatively average importance, which still allows the architectures to recognize each core token, with an important descriptive token. However, due to the sequential characteristics, the model will still overvalue word position at similar locations across the set, causing some sort of overfitting, especially, the input is short-phases input.

LSTM model results in higher RNN model might be due to higher complexity, whereas, the better learning performance causes the model to have greater confidence in the wrong patterns. As a result, it biases heavily on less important relationships, resulting in poorer generalization comparing to a simpler model.

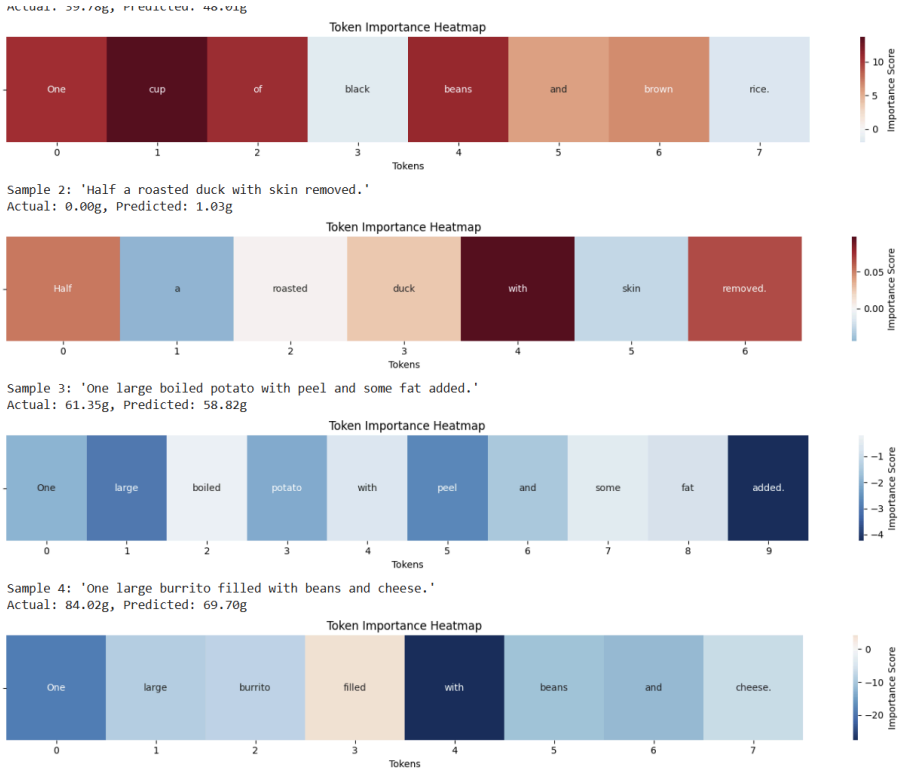

Figure 4. Token Importance Heatmap for MLP model

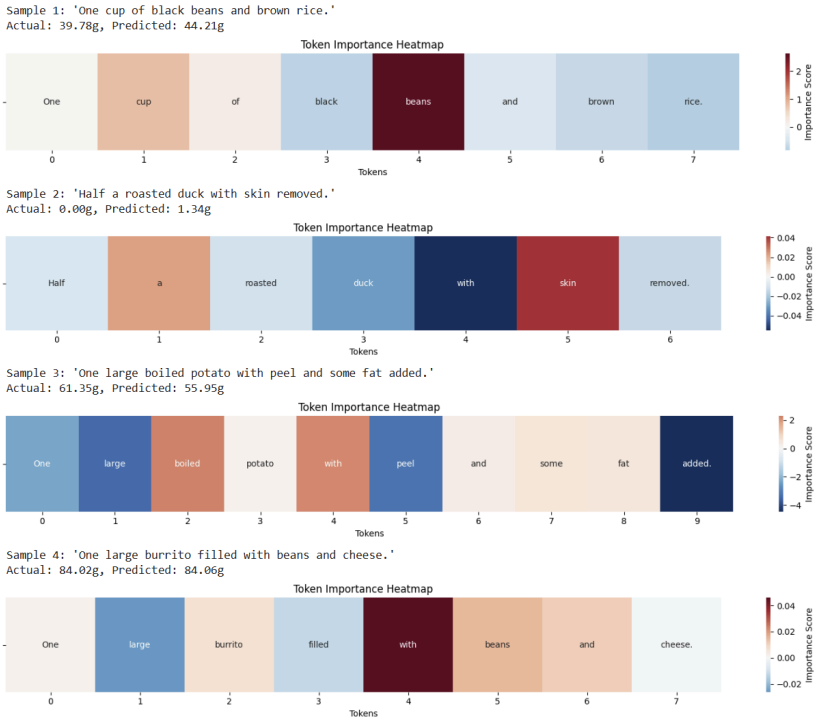

Figure 5. Token Importance Heatmap for LSTM model

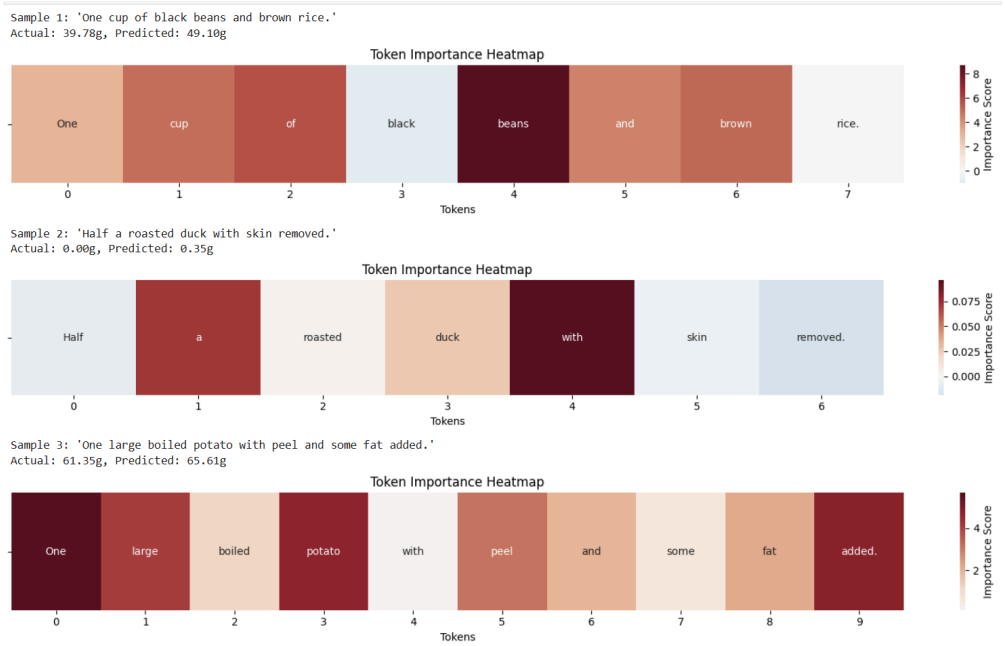

Figure 6.Figure 4. Token Importance Heatmap for RNN model

Conclusion

In this project, we explored the NutriBench regression task of predicting carbohydrate content from meal descriptions using a combination of text representation techniques and neural network architectures. By comparing traditional sparse representations (TF-IDF) with modern dense semantic embeddings (sentence transformers), and evaluating their performance across MLP, RNN, and LSTM models, we were able to assess both the strengths and limitations of each approach in a real-world nutritional prediction setting.

Our results indicate that the LSTM model paired with sentence transformer embeddings achieved the best performance with a validation RMSE of 15.4727, outperforming both the MLP and RNN alternatives. The improved performance of the LSTM may not suggest that its ability to capture sequential dependencies and long-range context is advantageous in modeling nutritional semantics, as discussed in analysis, it creates a larger RMSE gap suggesting stronger overfitting.

However, all three models exhibited signs of overfitting, with the training RMSE significantly lower than validation RMSE across the board. This overfitting likely stems from the highly skewed distribution of carbohydrate values in the training set and the limitations of RMSE in evaluating performance when large outliers are present. Additionally, our heatmap analysis revealed that TF-IDF-based models tend to misattribute importance to function words over nutritional keywords, while transformer-based embeddings allowed sequential models to better identify relevant tokens, though at the cost of introducing positional biases.

Future work could focus on mitigating data imbalance through advanced normalization or custom loss functions, incorporating nutritional ontologies to guide embedding attention, or exploring transformer-based sequence-to-regression models for improved generalization.

Nonetheless, this project highlights the importance of model-embedding alignment in NLP regression tasks and sets a foundation for further investigation into nutrition-focused language modeling.