Homework 1: Relation Extraction from Natural Language using PyTorch

Mudit Arora

1 Introduction

The goal of this homework is to train [GBC16]deep learning model to determine knowledge graph relations that are invoked in user utterances to a conversational system. We are given a training set of utterances paired with a set of relations, that we can use to train the model to predict the corresponding relations for a given utterances.

ID	UTTERANCES	CORE RELATIONS
0	is this a foreign film	movie.starring.country
1	star of twilight	movie.starring.actor
2	how many movies were filmed in asia	movie.locations

Table 1: Training Set example

Deep learning is a subpart of Machine Learning that is based on Artificial Neural Network in which multiple layers of processing are used to extract progressively higher level feature from data. It is an unsupervised learning in which the model learns from the data provided without the human supervision.

For example:
Show me movies directed by Wooden Allen recently.

These are two relations that are invoked by this utterance:

- movie.directed_by
- movie.initial_release_date

The goal is to use PyTorch to develop and train a deep neural network model and output the associated set of relations when given a new utterance.

For this particular Homework problem, we'll be using Multilayer Perceptron (MLP) which is a type of feed-forward neural network consisting of fully connected neurons with an activation function. It is widely used to distinguish data that in not linearly separable.

ID	UTTERANCES
0	star of thor
1	find a comedy
2	find films from china

Table 2: Test Set example

There are 19 different unique labels in the training set which the model will learn and predict the core relations for the utterances.

2 Models

For this current model, we used TF-IDF (Term Frequency-Inverse Document Frequency) method to process the text data. TF-IDF helps to convert textual data (i.e., utterances) into numerical vectors, which serve as inputs for the machine learning model. We chose TF-IDF due to its simplicity and effectiveness in capturing important features (terms) while reducing the weight of commonly occurring terms that may not hold much significance in the classification task.

2.1 Multilayer Perceptron (MLP)

The model consists of two hidden layers, each with 512 units. The input layer size corresponds to the dimensionlity of the TF-IDF vectors (i.e. 5000), while the output layer size is determined by the number of unique labels (19 in total) in the multi-label classification task.

Model Structure:

- Input layer: Takes the TF-IDF vector representation of the utterances.
- Hidden Layers: Two hidden layers with ReLU actiavtion and [IS15]batch normalization.
- [Sri+14]Dropout: Preventing overfitting.

- Output Layer: A sigmoid activation function is applied to generate multi-label predictions.

Tunable Hyperparameters:

- Hidden layer sizes: 512 units in each layer (can be adjusted)
- Learning rate: Set to 0.001 with Adam optimizer (tunable)
- Batch size: Set to 32 (can experiment with different sizes)
- Dropout rate: Set to 0.3 (tunable)
- Number of epochs: Set to 100 (tunable)

Training Process:
The model was trained using *binary cross-entropy loss*, appropriate for multi-label classification tasks where each label is considered independently. The optimizer used is *Adam*, which is well-suited for non-convex optimization problems such as neural networks.

Original Citations:

- TF-IDF: Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. Communications of the ACM, 18(11), 613-620
- [KB14]Adam Optimizer: Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980
- MLP: Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533-536

3 Experiments

The data set was split into 80% training and 20% validation sets using train_test_split. The test set was kept separate for final evaluation. No special techniques were used to handle missing values since the NaN values in the relations column were replaced with an empty string.

Hyperparameter Selection:

- We experimented with different values for the hidden layer sizes (e.g., 512 vs. 1024 units), dropout rates (e.g., 0.3 vs. 0.4), and learning rates (e.g., 0.001 vs. 0.0001) to find the best performing configuration.
- The weight decay in the Adam optimizer was set to 1e-5 to mitigate overfitting.

Handling Data Imbalance:
The class imbalance was addressed using thresholding for predictions. Instead of using default threshold of 0.5, we used 0.4 for binary conversion which gave better result.

Evaluation Methods:

- For evaluation, we used accuracy and binary cross-entropy loss on both the training and validation sets.
- We printed all the unique labels from the test set, the model was not able to identify some labels so it just didn't fill any value, the lesser the number of empty labels, the better the result was one of our criteria to judge.

4 Results

The model with the following configuration performed the best:

- 512 units per hidden layer
- Dropout rate: 0.3
- Learning rate: 0.001

4.1 Training Set

- The training loss gradually decreased as the epochs progressed. This meant that the model became better at predicting the correct labels for the utterances in the training set.
- The training accuracy increased as the training progressed, which showed that the model's predictions aligned more closely with the true labels in the training data.

4.2 Testing Set

- The validation loss spikes in between indicating potential overfitting.
- The validation accuracy increased as well, reaching upto 98%.

4.3 Validation Set

- The actual accuracy with the correct labels came out to be 79.8% due to unseen data's complexity and distribution.

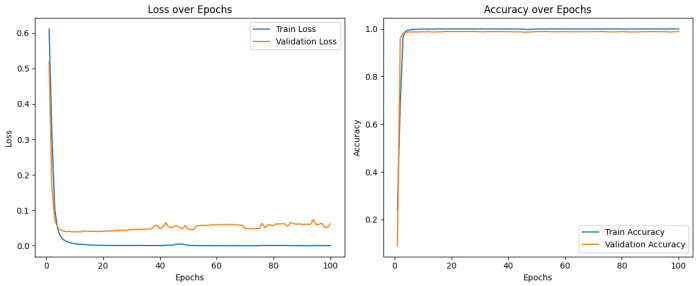

Figure 1: Training vs Validation Graph plots

References

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[IS15] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[KB14] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[Sri+14] Nitish Srivastava et al. "Dropout: A simple way to prevent neural networks from overfitting". In: *Journal of machine learning research* 15.1 (2014), pp. 1929–1958.