Homework 2: Slot Tagging for Natural Language Utterances

Mudit Arora

1 Introduction

The goal of this homework is to train [GBC16]deep learning model to tag slots in natural language utterances of users addressing a virtual personal assistant. The tagged slots and the associated values are traditionally used to accomplish user requests.

ID	utterances	IOB Slot tags
0	is this a foreign film	O O O O O
1	star of twilight	O O B_movie
2	how many movies were filmed in asia	O O O O O O B_location

Table 1: Training Set example

For example, if a user asks to learn about movies of a specific director, we would need to identify the name of the director (in addition to the user's intention to find out about movies, as we did in homework 1), and issue a query to the backend knowledge graph to get information for formulating a response back to the user.

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

These are two relations that are invoked by this utterance:

- director = "Woody Allen"
- release_year = "recently"

The correct labels for this utterance would then be:

Word	Label
show	O
me	O
movies	O
directed	O
by	O
Woody	B_director
Allen	I_director
recently.	B_release_year

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 Bidirectional Long Short-Term Memory (BiLSTM) which is a a type of recurrent neural network (RNN) that's designed to learn long-term dependencies in sequence data.

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

Table 2: Test Set example

There are 26 different unique tags in the training set which the model will learn and predict the core relations for the utterances.

2 Models

For this problem, we used BiLSTM model with various embeddings, including GloVe, CountVectorizer, and n-grams for implementing a sequence tagging model for slot filling in natural language understanding.

2.1 Embedding Layer

The EmbeddingLayer class supports three types of embeddings:

- GloVe Embeddings: These pre-trained embeddings capture semantic word relationships, useful for capturing general word meanings. GloVe embeddings were chosen for this model to leverage pre-existing semantic knowledge, enhancing generalization, especially in domains where labeled data is sparse.
- CountVectorizer: This embedding uses term frequency-based representation, suitable for simpler applications or bag-of-words features.
- n-Gram Embeddings: Generates embeddings based on combinations of words (n-grams). Useful for capturing context-dependent meanings, especially in multi-word phrases.

The embeddings are loaded and converted to a vocabulary-to-index map, where words or n-grams not found in the pre-trained embeddings are assigned to <UNK> (Unknown).

2.2 Dataset Preparation

The SlotTaggingDataset class prepares the data for training:

- Sentences are tokenized and mapped to embedding indices.
- Tags are converted into indices for model compatibility.
- The data loader uses padding for batch processing.

2.3 Model Architecture

The model architecture is based on a bidirectional LSTM (BiLSTM) layer that captures both forward and backward dependencies in a sequence, followed by a fully connected layer for classification. Key components include:

- Embedding Layer: Uses the pre-trained embeddings (e.g., GloVe).
- BiLSTM Layer: A bidirectional LSTM with two layers, set to a hidden dimension of 512. This layer captures dependencies in both directions, which is helpful for slot filling.
- Fully Connected Layer: The output layer has a size equal to the number of unique tags and is used to generate predictions for each token in the input.
- Hidden Layers: Two hidden layers with ReLU actiavtion and [IS15]batch normalization.
- [Sri+14]Dropout: Preventing overfitting.

Tunable Hyperparameters:

- Embedding dimension: 100 (for GloVe embeddings).
- Hidden dimension: 512, chosen to balance model complexity and learning capacity.
- Number of layers: 3, allowing for deeper sequence representation while avoiding overfitting.
- Learning rate: Set to 0.001 with Adam optimizer (tunable)
- Batch size: Set to 32 (can experiment with different sizes)
- Dropout rate: Set to 0.4 (tunable)
- Number of epochs: Set to 5 (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.

3 Experiments

Hyperparameter Selection:

- We experimented with different values for the hidden dimension (e.g., 256 vs. 512 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.
- We also experimented with different epochs (e.g., 5 vs. 10 vs. 20), and batch size also (e.g., 32 vs 64)

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

Evaluation Methods:

- We calculated the training loss and F1 score to compare predicted and true tags for non-padded tokens.
- While in the evaluation mode, the model masked padding tokens, and accumulated predictions across batches to calculate the final F1 score for the epoch.

4 Results

The model with the following configuration performed the best:

- Epoch: 5
- 256 units hidden dimension
- Dropout rate: 0.4
- Learning rate: 0.001

4.1 Comparison of Different Hyper-parameters

These are few of the comparisons, where we changes the hidden dimension and dropout rate only, and not the epoch and learning rate.

Epoch	Loss	F1 Score
1	1.0519	0.8297
2	0.4271	0.8382
3	0.2885	0.8347
4	0.2094	0.8299
5	0.1574	0.8212
Best F1 Score	**-**	**0.8382**

Table 3: Hidden dimension: 256, Dropout Rate: 0.3

Epoch	Loss	F1 Score
1	1.0382	0.8427
2	0.4681	0.8282
3	0.3082	0.8460
4	0.2357	0.8426
5	0.1975	0.8450
Best F1 Score	**-**	**0.8460**

Table 4: Hidden dimension: 1024, Dropout Rate: 0.4

Epoch	Loss	F1 Score
1	0.9189	0.8870
2	0.4202	0.8268
3	0.2995	0.8150
4	0.2235	0.8422
5	0.1722	0.8441
Best F1 Score	**-**	**0.8870**

Table 5: Hidden dimension: 512, Dropout Rate: 0.4

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).

[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.