Language Modeling

Team 5: Steven, Decker, Yash, Jincen
UCSC NLP 243

Abstract

Designing and training a language model on the Penn Treebank with a semi unsupervised learning using RNN, LSTM, GRU, and hyper-tuning.

1 Introduction

The project is training a language mode without any explicit labeling of the tokens. We are given glove embeddings for the Penn Treebank dataset. We are calculating perplexity for our given loss to predict each token in the sequence.

1.1 Dataset Overview

The Penn Treebank provides detailed annotations for a large corpus of English text. These annotations include part-of-speech (POS) tags, syntactic structure, and more. The depth and accuracy of these annotations make it a valuable resource for training and evaluating various NLP models.

- **Text Source:** The bulk of the text in the Penn Treebank comes from the Wall Street Journal (WSJ). It contains approximately 2,499 stories from WSJ from the years 1987 to 1989.
- **Word Tokens and Sentences:** The WSJ portion contains about 1 million words. The entire Treebank is typically reported to contain over 4.5 million words of American English.
- **Average Sentence Length:** The average sentence length in the WSJ part of the Treebank is around 20-25 words. However, this can vary significantly given the nature of news text, which includes both short sentences and more complex structures.
- **Annotations:** The annotations in the Treebank include part-of-speech (POS) tags, syntactic trees, and some semantic information. The POS tags follow a specific set developed for the Treebank, which has become a standard in NLP tasks.
- **Other Components:** Besides the WSJ corpus, the Penn Treebank also includes annotated texts from other sources, like transcribed spoken phone conversations from the Switchboard corpus and Brown Corpus, adding variety in style and formality.

2 Models

The allowed models are any neural networks excluding pretrained transformer-based models. A starter code using Recurrent Neural Network (RNN) was provided and recommended to use attention and possibly transformers.

2.1 RNN

Tested various hyperparameters with an RNN that had frozen and tuneable glove embeddings.

2.2 LSTM

Used base RNN model architecture and converted it into LSTM while testing bi-directionality.

2.3 GRU

Used base RNN model architecture and converted it into GRU

3 Experiments

3.1 Basic Metrics

- Step loss and accuracy:
- Perplexity Score:

3.2 Data Split

All models implement the same three-part data split structure for consistency.

- **Training set (85%)**: Used to train the model from train file

- **Validation set (7.5%)**: Used for hyperparameter tuning and early stopping.
- **Test set (6.5%)**: Used to evaluate the model's performance on unseen data on the test file

3.3 Training Configurations

General model hyperparameters for each model for consistency when training.

- Embedding Dimension: 300
- Batch Size: 32
- Hidden Size: 512
- Bidirectional: True
- Number of Layers: 2
- Dropout Probability: 0.5
- Tunable Pre-Trained Embedding: True
- Learning Rate: 0.001

3.4 Model Architecture

1. **Embedding Layer**: Maps token indices to dense vectors.
   - Input: Token indices
   - Output: Embeddings of size E (embedding dimension)
2. **Dropout Layer**: Applies dropout for regularization.
   - Dropout probability: p (dropout_p)
3. **GRU Layer**: Processes sequence data with temporal dependencies.
   - Input size: E (or $2E$ if using concatenated embeddings)
   - Hidden size per direction: H (hidden_size)
   - Number of layers: L (num_layers)
   - Bidirectional: Outputs concatenated forward and backward states if true.
4. **Fully Connected Layers**: Projects GRU output to the desired output size.

   $$\text{FC1:} \quad \mathbb{R}^{H \times N_D} \rightarrow \mathbb{R}^{E}$$
   $$\text{FC2:} \quad \mathbb{R}^{E} \rightarrow \mathbb{R}^{\text{vocab\_size}}$$

   where N_D is the number of directions (1 for unidirectional, 2 for bidirectional).
5. **Output**: Log softmax applied to the final layer's output.

4 Results

4.1 RNN

Epoch

Epoch	Loss	Accuracy	Perplexity
15	4.7523	23.02%	115.9
20	4.7827	22.70%	119.4
30	4.8421	22.47%	126.7

Table 1: Training Data Metrics

Epoch	Loss	Accuracy	Perplexity
15	5.1783	21.86%	177.4
20	5.1676	21.72%	175.5
30	5.1970	21.60%	180.7

Table 2: Validation Data Metrics

Epoch	Loss	Accuracy	Perplexity
15	5.1094	21.40%	165.6
20	5.1098	21.33%	165.6
30	5.1395	21.18%	170.6

Table 3: Test Data Metrics

4.2 LSTM/GRU

LSTM

Epoch	Loss	Perplexity
0	6.4171	612.23
1	5.6037	271.43
2	5.3745	215.83
3	5.2251	185.88
4	5.1205	167.42
19	4.5150	91.38

Table 4: LSTM Performance Metrics over Epochs

Data Type	Loss	Accuracy	Perplexity
Training	4.01	29.15%	55.07
Validation	4.71	25.60%	111.35
Test	4.67	25.01%	106.52

Table 5: LSTM Accuracy Metrics

Val Sequence Predictions

Example 1
Actual:

```
aer banknote berlitz calloway centrust
    ↪ cluett fromstein gitano guterman
    ↪ hydro-quebec ipo kia memotec mlx
    ↪ nahb punts rake regatta rubens
    ↪ sim snack-food ssangyong swapo
    ↪ wachter </s>
```

Predicted:

```
the <unk> a <unk> and and ... and snack-
    ↪ food ... snack-food ... advancers
    ↪  ... advancers ...
```

Example 2
Actual:

```
pierre <unk> N years old will join the
    ↪ board as a nonexecutive director
    ↪ nov. N </s>
```

Predicted:

```
the <unk> a years old was retire the
    ↪ chairman of chairman director
    ↪ director </s> N </s> advancers
    ↪ ... advancers ...
```

BiLSTM

Epoch	Loss	Perplexity
0	4.5652	96.09
1	1.7432	5.716
2	1.1339	3.108
3	0.8386	2.313
4	0.6586	1.932
5	0.5354	1.708
Overall Performance		
Training	0.0720	1.075
Validation	0.1872	1.206
Test	0.1072	1.113

Table 6: BiLSTM Performance Metrics over Epochs

Test Sequence Predictions

Example 1
Actual:

```
no it was n't black monday </s>
```

Predicted:

```
no it was n't black monday </s>
    ↪ alternative ... alternative ...
```

Example 2
Actual:

```
but while the new york stock exchange
    ↪ did n't fall apart friday as the
    ↪ dow jones industrial average
    ↪ plunged N points most of it in
    ↪ the final hour it barely managed
    ↪ to stay this side of chaos </s>
```

Predicted:

```
but while the new york stock exchange
    ↪ did n't fall apart friday as the
    ↪ dow jones industrial average
    ↪ plunged N points most of it in
    ↪ the final hour it barely managed
    ↪ to stay this side of anxiety </s>
    ↪  alternative ... alternative ...
```

GRU

Epoch	Loss	Perplexity
0	4.0637	58.188
1	1.1313	3.0998
2	0.5903	1.8045
3	0.3702	1.4480
4	0.2618	1.2993
5	0.2007	1.2222
6	0.1654	1.1798
7	0.1439	1.1548
Overall Performance		
Type	**Loss**	**Perplexity**
Training	1.9318×10^{-5}	1.00002
Validation	0.0078511	1.00788
Test	0.0012214	1.00122

Table 7: BiGRU Performance Metrics over Epochs

5 Inference

5.1 RNN

RNN performed the worst as it could not get context from previous words and did not perform well for longer sequences. RNN was evaluated with perplexity ignoring the padding. When running my prediction model I ignored any words generated past the EOS token. The model continued

to predict the same word up to the batch size and therefore was not included in my perplexity score. Otherwise, it will be a lot higher. Running a smaller batch size will reduce these padding generations, which was what I observed when calculating the perplexity score from the batch output.

Increasing the epoch made the model overfit more, and had to monitor step size at each epoch. This was more prevalent with BiLSTM and GRU as they learned way faster than an RNN. All of them achieved similar scores as the base model parameters +/- 2%. The extra layer model took way too long to learn and was stopped manually to save computational time.

5.2 LSTM/GRU

LSTM?GRU performed a lot better at perplexity, keeping more long-term memory for sequences. GRU performed similarly to the base model of LSTM. The accuracy only increased by 5% but perplexity reduced by 50. However, there were issues when Bi-Directionality was turned on making the model learn the predictions insanely well. It might be how the tags were basically given to each other and the hidden layer was able to retain the whole sequence prediction.

Using other hyperparameters did not affect the overall accuracy, but may caused training to be longer or shorter. Batch size was important for handling different length sequences. Embed dimension allowed for weight to be calculated taking longer to train. Layers made the model too complex and slowed training significantly. Dropout adjusted the rate of convergence and required manual stopping to prevent overfitting.

6 Appendix

Data Type	Loss	Accuracy	Perplexity
Training	5.0580	20.74%	157.3
Validation	5.3518	20.18%	211.0
Test	5.2758	19.81%	195.5

Table 8: Performance Metrics with Batchsize 16

Data Type	Loss	Accuracy	Perplexity
Training	4.6384	23.50%	103.4
Validation	5.2118	21.83%	183.4
Test	5.1346	21.75%	169.8

Table 9: Performance Metrics with Embedding Dimension 600

Data Type	Loss	Accuracy	Perplexity
Training	4.6791	23.42%	107.7
Validation	5.2206	21.86%	185.0
Test	5.1671	21.42%	175.4

Table 10: Performance Metrics with Dropout = 0.3

Data Type	Loss	Accuracy	Perplexity
Training	6.0019	14.28%	404.2
Validation	6.0861	14.07%	439.7
Test	5.9974	14.06%	402.4

Table 11: Performance Metrics with 4 Layers