Advanced Natural Language Processing

Assignment 3: Character-level Language Modelling with LSTM

Lucia Welther, Matrikelnummer: 835106

January 6, 2026

1 LSTM Model

The LSTM model was implemented with three layers as specified:

1. **Embedding Layer**: Maps character indices (0-99 from `string.printable`) to dense vectors of dimension 128.
2. **LSTM Layer**: Two stacked LSTM layers process the embedded input, maintaining hidden and cell states of dimension 128.
3. **Decoder Layer**: A linear layer projects the hidden state back to the vocabulary space (100 dimensions), producing logits for the next character.

The hidden and cell states are initialized to zero vectors at $t = 0$ with shape $[\text{n\_layers}, \text{batch\_size}, \text{hidden\_size}] = [2, 1, 128]$.

1.1 Default Training Configuration

The default training parameters are:

- **n_epochs**: 3000 total training iterations
- **print_every**: 100 (sample generation frequency)
- **plot_every**: 10 (loss recording frequency)
- **hidden_size**: 128 (LSTM hidden dimension)
- **n_layers**: 2 (stacked LSTM depth)
- **Learning rate**: 0.005 (Adam optimizer)

1.2 Training Process and Results

During training, the model learns to predict the next character in Dickens' novels, one character at a time. The training loop:

1. Samples a random 200-character chunk from the training corpus
2. Creates input/target pairs offset by one position (e.g., input "ab" predicts target "bc")
3. Computes cross-entropy loss between predicted character distribution and ground truth
4. Backpropagates gradients and updates weights via Adam optimizer

1.3 Observed Learning Progression

The training loss decreases from an initial ≈ 2.56 at epoch 100 to ≈ 1.75 by epoch 3000, indicating the model is learning meaningful character dependencies. Key observations:

- **Early epochs (0-500)**: Loss remains high ($\approx$ 2.0-2.5). Generated text is mostly random character sequences with occasional dictionary words like “the”, “and”, “Mr.”. The model has not yet learned English structure.
- **Mid epochs (500-1500)**: Loss gradually decreases to $\approx$ 1.8-1.9. Generated text shows emerging word boundaries and common English patterns. Proper nouns (“Mr.”, “She”) appear more frequently. Sentence-like structures begin to form, though still largely incoherent.
- **Later epochs (1500-3000)**: Loss stabilizes around 1.6-1.8. Generated text exhibits increasingly coherent word sequences and grammatical structures. The model learns to produce proper spacing, punctuation, and realistic word sequences like “I have”, “the way”, “little more”. Some samples show proto-dialogue structure with quotation marks.

The model demonstrates the characteristic learning trajectory of language models: first learning character-level statistics and spacing, then progressing to word-level patterns, and finally developing rudimentary syntactic awareness. By epoch 2500-3000, generated samples approximate Dickens’ prose style with recognizable Victorian-era vocabulary and sentence structure, though semantic coherence remains limited.

1.4 Why Character-Level vs. Word-Level?

A character-level model, unlike word-level models, does not require a predefined vocabulary. It can generate any character sequence, including misspellings, novel word formations, and punctuation handling. This flexibility comes at the cost of longer sequences (needing to predict many more timesteps) and slower training, as evidenced by the 25+ minute training time for 3000 epochs on CPU.

2