Assignment 3: How to Train a M.S Student

Cal Blanco

February 2025

1 Part 1

1.1 1. Objective Function

Let x_{ij}^t be a binary variable that equals 1 if there is a transition from tag i to tag j at timestep t, and 0 otherwise. Let w_{ij}^t be the weight (or score) for this transition at timestep t.

The objective function to maximize is:

$$\max_x \sum_{t=0}^{T} \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}} w_{ij}^t x_{ij}^t \tag{1}$$

where $\mathcal{S}$ is the set of all tags, including special START and STOP tags. The START tag can only appear at $t = 0$, and the STOP tag can only appear at $t = T$.

2. Timestep Constraints

At each timestep, exactly one transition must occur. This can be expressed as:

$$\sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}} x_{ij}^t = 1 \quad \forall t \in \{0, 1, ..., T\} \tag{2}$$

To express this using inequality constraints ($\leq$), we can write:

$$\sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}} x_{ij}^t \leq 1 \quad \forall t \in \{0, 1, ..., T\} \tag{3}$$

$$-\sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}} x_{ij}^t \leq -1 \quad \forall t \in \{0, 1, ..., T\} \tag{4}$$

3. Path Validity Constraints

To ensure a valid path, we need to enforce that if we transition to tag j at time t, then the next transition at time $t + 1$ must start from tag j. This can be expressed as:

$$\sum_{i \in \mathcal{S}} x_{ij}^{t} = \sum_{k \in \mathcal{S}} x_{jk}^{t+1} \quad \forall j \in \mathcal{S}, \forall t \in \{0, 1, ..., T-1\} \tag{5}$$

Using inequality constraints:

$$\sum_{i \in \mathcal{S}} x_{ij}^{t} - \sum_{k \in \mathcal{S}} x_{jk}^{t+1} \leq 0 \quad \forall j \in \mathcal{S}, \forall t \in \{0, 1, ..., T-1\} \tag{6}$$

$$\sum_{k \in \mathcal{S}} x_{jk}^{t+1} - \sum_{i \in \mathcal{S}} x_{ij}^{t} \leq 0 \quad \forall j \in \mathcal{S}, \forall t \in \{0, 1, ..., T-1\} \tag{7}$$

Additionally, we need to add constraints for the START and STOP tags:

$$\sum_{j \in \mathcal{S}} x_{\text{START},j}^{0} = 1 \tag{8}$$

$$\sum_{i \in \mathcal{S}} x_{i,\text{STOP}}^{T} = 1 \tag{9}$$

4. BIO Tagging Constraints

For BIO tagging, to enforce that tag I cannot follow tag O, we add the constraint:

$$x_{O,I}^{t} = 0 \quad \forall t \in \{0, 1, ..., T-1\} \tag{10}$$

As an inequality constraint:

$$x_{O,I}^{t} \leq 0 \quad \forall t \in \{0, 1, ..., T-1\} \tag{11}$$

Since $x_{O,I}^{t}$ is a binary variable, this effectively forces it to be 0, preventing an I-tag from following an O-tag.

2 Part 2

1. Basic Dependency Parsing ILP

Let $w = \langle w_0 = \$, w_1, w_2, \ldots, w_n \rangle$ be a sentence, where \$ is the root symbol. We define binary variables:

$$x_{ij} = \begin{cases} 1 & \text{if } w_i \text{ is the parent of } w_j \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

for $i \in \{0, 1, \ldots, n\}$ and $j \in \{1, \ldots, n\}$. Note that j begins at 1 since the root has no parent.

Objective Function:

$$\max_{x} \sum_{i=0}^{n} \sum_{j=1}^{n} \text{score}(w_i, w_j) \cdot x_{ij} \tag{13}$$

Constraints:

$$\sum_{i=0}^{n} x_{ij} = 1 \quad \forall j \in \{1, \ldots, n\} \tag{14}$$

$$\sum_{j=1}^{n} x_{0j} \geq 1 \quad \text{(root has at least one child)} \tag{15}$$

For the standard form of the ILP, we have:

$$\mathbf{x} = [x_{01}, x_{02}, \ldots, x_{0n}, x_{10}, x_{11}, \ldots, x_{nn}]^T \tag{16}$$

$$\mathbf{c} = [\text{score}(w_0, w_1), \text{score}(w_0, w_2), \ldots, \text{score}(w_n, w_n)]^T \tag{17}$$

Matrix $\mathbf{A}$ has $n+1$ rows (one for each constraint), and $N = (n+1)n$ columns:

- The first n rows enforce that each word has exactly one parent:

$$\sum_{i=0}^{n} x_{ij} = 1 \quad \Rightarrow \quad \sum_{i=0}^{n} x_{ij} \leq 1 \text{ and } -\sum_{i=0}^{n} x_{ij} \leq -1 \tag{18}$$

- The last row enforces that the root has at least one child:

$$-\sum_{j=1}^{n} x_{0j} \leq -1 \tag{19}$$

Vector $\mathbf{b} = [1, 1, \ldots, 1, -1]^T$ of size $M = 2n + 1$.

2. Projectivity Constraints

For projectivity, we add constraints to ensure no crossing dependencies. If w_i is the parent of w_j, then for any word w_k where k is between i and j, the parent of w_k must also be between i and j (inclusive).

For $i \neq j$ and for all k such that $\min(i, j) < k < \max(i, j)$:

$$x_{ij} \leq 1 - \sum_{l \notin [\min(i,j), \max(i,j)]} x_{lk} \tag{20}$$

This constraint states that if $x_{ij} = 1$ (i.e., w_i is the parent of w_j), then for any word w_k between w_i and w_j, the sum of x_{lk} for all l outside the range $[i, j]$ must be 0.

Equivalently:

$$x_{ij} + \sum_{l \notin [\min(i,j), \max(i,j)]} x_{lk} \leq 1 \quad \forall i, j \in \{0, 1, \ldots, n\}, i \neq j, \forall k : \min(i, j) < k < \max(i, j) \tag{21}$$

This adds $O(n^3)$ constraints in the worst case.

3. Acyclicity Constraints (Bonus)

To ensure the dependency structure is a tree (acyclic), we can use a flow-based approach. We introduce additional continuous variables f_{ij}^k representing the flow from word i to word j for commodity k.

We can enforce that:

$$f_{ij}^k \leq n \cdot x_{ij} \quad \forall i, j, k \tag{22}$$

$$\sum_{i=0}^{n} f_{ij}^j = 1 \quad \forall j \in \{1, \ldots, n\} \tag{23}$$

$$\sum_{l=1}^{n} f_{jl}^k - \sum_{i=0}^{n} f_{ij}^k = \begin{cases} -1 & \text{if } j = k \\ 0 & \text{otherwise} \end{cases} \quad \forall j, k \in \{1, \ldots, n\} \tag{24}$$

This ensures that:

- Flow can only exist on edges that are in the dependency structure
- For each node j, there is a unit of flow of commodity j from the root to j
- Flow conservation ensures that the structure is connected and acyclic

Alternatively, we can use the single-commodity flow formulation:

Introduce variables $y_{ij} \geq 0$ representing the flow from i to j, with the constraints:

$$y_{ij} \leq n \cdot x_{ij} \tag{25}$$

$$\sum_{i=0}^{n} y_{ij} - \sum_{k=1}^{n} y_{jk} = 1 \quad \forall j \in \{1, \ldots, n\} \tag{26}$$

This ensures that each node receives one more unit of flow than it sends out, and the total flow is conserved, guaranteeing a tree structure.

3 Part 3

3.1 Introduction

In this project, I tackled the challenging task of abstractive text summarization using sequence-to-sequence models with attention mechanisms. The goal was to develop a system capable of generating concise summaries that capture the essential information from longer news articles, while maintaining readability and coherence.

My journey began with significant technical hurdles, as the provided starter code was severely outdated and plagued with dependency issues. The code base, originally designed for machine translation tasks, required substantial modifications to work with modern PyTorch versions and to address the specific

challenges of text summarization. Rather than being a helpful starting point, resolving these compatibility issues consumed valuable development time.

Despite these initial setbacks, I persevered and implemented a functional seq2seq model with Bahdanau attention. The implementation process involved careful consideration of architectural choices, hyperparameter tuning, and optimization strategies to achieve the best possible performance within the constraints of the assignment.

3.2 Dataset

The CNN/Daily Mail dataset served as the foundation for this project. This extensive collection contains over 300,000 news articles, each paired with human-written highlights that function as multi-sentence summaries. The dataset presents a realistic challenge for abstractive summarization systems, as it requires models to understand complex news articles and generate coherent summaries that capture key information.

Working with this dataset presented several challenges:

- **Length variation**: Articles varied significantly in length, requiring careful handling of sequence padding and truncation
- **Vocabulary size**: The news domain includes specialized terminology and named entities, creating challenges for vocabulary management
- **Abstractive nature**: Unlike extractive summarization, abstractive approaches must generate new text rather than simply selecting important sentences

For evaluation, I used the ROUGE metric suite (ROUGE-1, ROUGE-2, and ROUGE-L), which measures the overlap between generated summaries and reference summaries. The assignment required achieving minimum F-scores of 0.2, 0.04, and 0.2 respectively, which served as my initial performance targets.

3.3 Model Description

3.3.1 Model Architecture

The summarization model is an abstractive sequence-to-sequence architecture with attention mechanism. The model consists of three main components:

- **Encoder**: Processes the input article text
- **Attention**: Bahdanau attention mechanism to focus on relevant parts of the input
- **Decoder**: Generates the summary text with attention context

The model was implemented with the following parameters:

- Embedding dimension: 256

- Hidden size: 512
- Vocabulary: Custom built using a FastTokenizer
- Training epochs: 5
- Batch size: 64
- Optimizer: Adam
- Loss function: CrossEntropyLoss (ignoring padding tokens)

The training process utilized mixed precision training with gradient accumulation steps to optimize memory usage and training speed. The model was trained on multiple GPUs (CUDA devices 1, 2, and 3).

3.3.2 Notable Implementation Details

The implementation includes several optimizations:

- Gradient accumulation (4 steps) to effectively increase batch size
- Mixed precision training using PyTorch's autocast and GradScaler
- Parallel data loading with multiple workers (num_workers=4)
- Pin memory for faster data transfer to GPU
- Model checkpointing based on validation loss

3.4 Notable Predictions

The model's predictions show interesting patterns and limitations:

- **Repetition issues**: Many predictions contain repeated phrases like ", and are, and are" or repetitive commas, suggesting the model struggles with sequence generation control.
- **Partial coherence**: Some predictions start with reasonable content but degrade into repetition:

  > "amnesty international says it's 'not a better' the report says 'the most hated countries' is a factor in the country."

- **Entity recognition**: The model correctly identifies entities in some cases:

  > "anne frank anne died of ¡unk¿ in a nazi concentration camp. the ¡unk¿ concentration camp became a more virulent risk."

- **Grammar issues**: Many predictions have grammatical errors or incomplete sentences:

"trey ¡unk¿, a teacher at the age of 16. she is a high school freshman with down syndrome."

- **Token limitations**: The frequent appearance of "¡unk¿" tokens indicates vocabulary limitations.

3.5 Performance Evaluation

Metric	Recall	Precision	F1-Score
ROUGE-1	0.4473	0.2350	0.2971
ROUGE-2	0.0886	0.0584	0.0664
ROUGE-L	0.4230	0.2223	0.2810

Table 1: ROUGE scores for the summarization model on validation data

The ROUGE scores indicate that the model performs moderately well on unigram matching (ROUGE-1) with a recall of 0.4473, suggesting it captures a significant portion of the reference summary content. However, the lower ROUGE-2 scores (F1 of 0.0664) indicate that the model struggles with capturing bigram sequences correctly, which aligns with the observed repetition issues in the predictions.

The ROUGE-L score (F1 of 0.2810) measures the longest common subsequence, providing an assessment of sentence-level structure similarity. The gap between recall and precision across all metrics suggests the model tends to generate longer summaries than the reference, potentially including irrelevant information.

3.6 Conclusions

In this project, I implemented an abstractive summarization system for the CNN/Daily Mail dataset, overcoming significant technical challenges along the way. The journey began with frustrating dependency issues in the provided starter code, which required substantial refactoring to work with modern libraries.

My implementation focused on a sequence-to-sequence architecture with Bahdanau attention, using an embedding dimension of 256 and hidden size of 512. The model was trained for 5 epochs with a batch size of 64, utilizing Adam optimization and mixed precision training across multiple GPUs to improve efficiency.

3.6.1 Notes about performance

1. **Repetition patterns**: The model frequently generated repetitive phrases and comma sequences
2. **Vocabulary limitations**: Many predictions contained `<unk>` tokens, indicating out-of-vocabulary words

3. **Grammatical inconsistencies**: While some summaries started coherently, they often degraded into grammatically incorrect sequences

The gap between recall and precision scores suggests the model tends to generate longer summaries than the reference, potentially including irrelevant information. This indicates room for improvement in controlling generation length and focusing on the most salient information.

Future work could address these limitations through:

- Implementing a pointer-generator network to handle out-of-vocabulary words
- Adding a coverage mechanism to reduce repetition
- Exploring more sophisticated encoder architectures while still avoiding pre-trained models like BERT
- Implementing beam search with length penalties to improve generation quality

Overall, this project demonstrated both the potential and challenges of neural abstractive summarization. While the implemented model successfully met the baseline requirements, the analysis of its outputs provides valuable insights for further improvements in text generation quality and coherence.