ILP and Summarization

March 8, 2025

Jou-Yi Lee
jlee1039@ucsc.edu
University of California, Santa Cruz

Abstract

In Problem 1, we develop an I formulation for sequence labeling under the BIO tagging scheme. The ILP model uses binary transition variables to represent shifts between labels, and incorporates constraints to ensure valid sequence paths, including special start and stop tokens. In Problem 2, the ILP framework is extended to dependency parsing. We design an ILP model that guarantees each word (except the designated root) receives exactly one parent, while also enforcing projectivity constraints. Furthermore, we discuss additional constraints for ensuring acyclicity within the dependency tree structure. Finally, Problem 3 involves the implementation of an abstractive summarization system for the CNN/Daily Mail dataset using a sequence-to-sequence model with attention. Enhancements such as pointer-generator modules and copy mechanisms are explored to improve the ROUGE scores over the baseline. Together, these approaches demonstrate the versatility of ILP models and neural methods in addressing complex NLP tasks.

1 Consolidated Explanation for Problem 1

In this problem, we create an ILP for sequence labeling by defining binary decision variables and adding linear constraints to ensure that the selected transitions form a valid tag sequence under the BIO tagging scheme.

1.1 Decision Variables

Let x_t^{ij} be a binary variable for each timestep $t = 0, \ldots, T$ and for each possible transition from tag i to tag j, including transitions from the `START` token and to the `STOP` token. That is,

$$x_t^{ij} = \begin{cases} 1, & \text{if at timestep } t \text{ a transition from tag } i \text{ to tag } j \text{ is selec} \\ 0, & \text{otherwise.} \end{cases}$$

1.2 Objective Function

We define the objective function to maximize the total score of the selected transitions:

$$\max \sum_{t=0}^{T} \sum_{i \in \mathcal{T}} \sum_{j \in \mathcal{T}} \text{score}(i, j, t) \cdot x_t^{ij},$$

where $\mathcal{T}$ is the set of all tags (including `START` and `STOP`), and $\text{score}(i, j, t)$ is the score associated with transitioning from tag i to tag j at timestep t.

1.3 Constraints

1.3.1 Exactly One Transition per Timestep

At each timestep, exactly one transition must be selected. We express the equality constraint as two linear inequalities:

$$\sum_{i \in \mathcal{T}} \sum_{j \in \mathcal{T}} x_t^{ij} \leq 1, \quad \forall t = 0, \ldots, T,$$

and

$$-\sum_{i \in \mathcal{T}} \sum_{j \in \mathcal{T}} x_t^{ij} \leq -1, \quad \forall t = 0, \ldots, T.$$

1.3.2 Valid Path Constraint

To ensure that the transitions form a valid path, the destination tag at timestep t must match the source tag at timestep $t + 1$. This is expressed

as:

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

1.3.3 BIO Tagging Constraints

To enforce the BIO tagging rules, we add constraints that prevent illegal transitions. For example, to ensure that tag `I` cannot follow tag `O`, we impose:

$$x_t^{\mathtt{O},\mathtt{I}} = 0, \quad \forall t = 0, \ldots, T-1.$$

Furthermore, to enforce that the first transition from `START` goes to `B`, we add:

$$x_0^{\mathtt{START},\mathtt{B}} = 1.$$

1.4 Optimal Sequence Example

An optimal sequence obtained from the ILP solver using the above formulation is:

```
Time step 0:  START → B
Time step 1:  B → B
Time step 2:  B → I
Time step 3:  I → O
Time step 4:  O → STOP
```

This result confirms that:

- The sequence correctly starts with a transition from `START` to `B`.
- Each step follows continuously with the destination of one timestep becoming the source of the next.
- The BIO constraints are satisfied, ensuring that no illegal transitions occur (e.g., no `O` is followed by `I`).

2 Problem 2: Dependency Parsing with an ILP

2.1 Key Points

- **Dependency Tree Variables:** I define binary variables $x[i, j]$ for $i \in \{0, 1, \ldots, n\}$ and $j \in \{1, \ldots, n\}$ such that $x[i, j] = 1$ if word w_i is the parent of w_j. Self-loops (i.e., $x[j, j]$) are not allowed.
- **One Parent per Non-root Word:** Every non-root word receives exactly one parent.
- **Objective Function:** We maximize the sum of scores over all dependency edges.

2.2 Mapping to Standard ILP Form

Decision Variables as a Vector x. I collect all binary variables $x_{i,j}$ into a single vector $x \in \{0, 1\}^N$, where

$$N = (n + 1) \times n.$$

Objective Function (c). I define a vector $c \in R^N$ whose components store the score of each possible edge. Concretely,

$$c_{(i,j)} = \text{score}(i, j),$$

and objective is

$$\max c^\top x \;=\; \max \sum_{i=0}^{n} \sum_{j=1}^{n} \text{score}(i, j)\, x_{i,j}.$$

Constraints (A and b). All constraints can be written in the form $Ax \leq b$.

- **One Parent per Non-root Word:**

$$\sum_{i=0}^{n} x_{i,j} = 1 \quad \Longrightarrow \quad \begin{cases} \sum_{i=0}^{n} x_{i,j} \;\leq\; 1, \\ -\sum_{i=0}^{n} x_{i,j} \;\leq\; -1, \end{cases} \qquad \forall j = 1, \ldots, n.$$

  Each such equality can be replaced by two linear inequalities.

- **No Self-loops:**

$$x_{j,j} = 0 \quad \Longrightarrow \quad x_{j,j} \;\leq\; 0 \quad \text{and} \quad -x_{j,j} \;\leq\; 0.$$

- **Root Does Not Have a Parent:** We either do not define $x_{k,0}$ at all, or enforce $x_{k,0} = 0$ similarly. This adds additional rows in A if needed.
- **Projectivity (Optional):** I add the projectivity constraint, we have $O(n^2)$ more inequalities of the form:

  (if $x_{i,j} = 1$, then the parent of any w_k in the span (i, j) must

These can be linearized by combining $x_{i,j}$ with $x_{\ell,k}$ in an appropriate inequality (e.g., $x_{i,j} + x_{\ell,k} \leq 1$ under certain conditions).

All these constraints define a matrix A and a vector b. Finally, I add

$$x \geq 0,$$

and specify x to be integral (binary). This completes the standard ILP form:

$$\max c^\top x \quad \text{subject to} \quad Ax \leq b,\ x \in \{0,1\}^N.$$

2.3 Implementation

- **Sentence and Word Setup:** I set $n = 5$ to represent a sentence with 5 non-root words (i.e., $w_1, \ldots, w_5$). The word list is defined as:

  `["$","w1","w2","w3","w4","w5"]`.

- **ILP Model Initialization:** I initialize the ILP model with a maximization objective.

- **Decision Variables:** I define binary variables $x_{i,j}$, where i ranges from 0 to n (with $i = 0$ corresponding to the root) and j ranges from 1 to n.

- **Constraints:**

  1. **No Self-loops:** For each non-root word w_j, enforce $x_{j,j} = 0$.
  2. **Exactly One Parent per Non-root Word:** For every w_j (with $j \geq 1$), impose
  $$\sum_{i=0}^{n} x_{i,j} = 1.$$
  3. **Optional Projectivity Constraint:** For each potential edge from w_i to w_j (with $i < j$), if there is a span (i.e., words exist between w_i and w_j), then the number of words between them, $j-i-1$, multiplied by $x_{i,j}$ is bounded by the sum of dependency assignments for words within that span.

- **Score Definition:** A score for each potential dependency edge, score(i, j), is generated using a random value between -1 and 1.

- **Maximization Objective:** The ILP model is set to maximize the total score, ensuring that the solution corresponds to the highest-scoring dependency tree.

2.4 Result Explanation

Dependency Parsing Tree:

```
Parent of w1 is $
Parent of w2 is $
Parent of w3 is $
Parent of w4 is w5
Parent of w5 is w3
```

1. w_1, w_2, and w_3 are directly attached to the root ($).

2. w_4 is dependent on w_5, and w_5 is dependent on w_3.

3. All non-root nodes have exactly one parent, and no self-loops occur. The optional projectivity constraint (if active) ensures that the structure adheres to projectivity rules.

3 Problem 3: Summarization System for the CNN/Daily Mail Dataset

3.1 Data Loading and Preprocessing

For text preprocessing, I build separate vocabularies for the source (articles) and target (summaries) texts. During vocabulary construction, I use a minimum frequency threshold to filter out rare words and include a set of special tokens (e.g., `<PAD>`, `<SOS>`, `<EOS>`, and `<UNK>`). The resulting vocabularies are then used to transform each text sample into a sequence of integer IDs. The training set consists of 287,227 samples. The vocabulary size for the source texts is 50,004, while the target vocabulary size is 30,004.

3.2 Model Architecture Overview

The model is built upon the standard seq2seq framework augmented with an attention mechanism. The Encoder, the Attention mechanism, the Decoder, and the Seq2Seq wrapper. Below is a concise explanation of each component:

3.3 Encoder

- **Embedding and Bidirectional LSTM:** The encoder first maps input tokens to embeddings, then processes the sequence using a bidirectional LSTM.
- **Hidden State Combination:** The final hidden states from the forward and backward passes are concatenated and passed through a linear layer with a tanh activation to obtain a combined hidden state.
- **Output:** It returns the LSTM outputs (for attention) and the combined hidden state.

3.4 Attention

- **Input Preparation:** The decoder's current hidden state is unsqueezed and repeated to match the encoder output length, while the encoder outputs are permuted so that both have shape [batch_size, src_len, $\cdot$].
- **Energy Computation:** The repeated hidden state and encoder outputs are concatenated and passed through a linear layer followed by a tanh activation to compute an energy tensor.
- **Attention Scores:** Another linear layer projects the energy to scalar scores which are squeezed and normalized via softmax to yield attention weights.

3.5 Decoder

- **Token Embedding:** The decoder embeds the current input token.
- **Attention Integration:** It computes attention weights using the current hidden state and encoder outputs to generate a context vector.
- **Decoding Step:** The embedded input and the context vector are concatenated and passed to an LSTM, which updates the hidden state.
- **Output Generation:** The LSTM output, context vector, and input embedding are combined through a fully connected layer to predict the next token.

3.6 Seq2Seq Wrapper

- **End-to-End Processing:** The Seq2Seq module first encodes the input sequence and then uses the decoder to generate the target sequence token by token.
- **Teacher Forcing:** During training, teacher forcing is applied—using the ground truth token as the next input with a certain probability—to aid convergence.
- **Final Output:** It produces a sequence of token probability distributions corresponding to the predicted summary.

3.7 Training Setup and Loop

Hyperparameters:

- **INPUT_DIM** = The vocabulary sizes.
- **OUTPUT_DIM**: The source and target texts.
- **ENC_EMB_DIM** = 256 **HID_DIM** = 512
- **N_LAYERS** = 2 **DROPOUT** = 0.5

The model components as follows:

- **Encoder**: Bidirectional LSTM.
- **Attention** Combining the encoder outputs with the decoder's current hidden state.
- **Decoder**
- **Seq2Seq**

I define the loss function as cross-entropy (ignoring the <PAD> token) and use the Adam optimizer with a learning rate of 1×10^{-3}. Training is conducted for 10 epochs, with the training loss reported after each epoch.

4 Evaluation Results

The result are as follows:

- ROUGE-1 F-score > 0.23
- ROUGE-2 F-score > 0.05
- ROUGE-L F-score > 0.22

These results validate our implementation of the seq2seq with attention framework for abstractive summarization, demonstrating that it generates summaries with substantial lexical and structural overlap with the reference summaries.