4132 Course Work Algorithm Summary

Mingyuan LIU

November 26,2025

Algorithm 1: Preprocessing Pipeline for LOTR Texts

Input: Raw LOTR books $\{B_1, B_2, B_3\}$
Output: Tokenized paragraphs P, vocabulary V
foreach *book* B_i **do**
    Attempt to read B_i with multiple encodings;
    Remove chapter titles and metadata;
    Normalize whitespace and remove duplicate blank lines;
    Split text into paragraphs by blank lines;
Initialize empty list P;
foreach *paragraph* p **do**
    Tokenize p using regex into words and punctuation;
    if *p is not the first paragraph* **then**
        prepend token $\langle PARA \rangle$
    Prepend $\langle BOS \rangle$ and append $\langle EOS \rangle$;
    Append processed paragraph to P;
Build vocabulary V using: special tokens + all unique tokens sorted alphabetically;
Convert all tokens in P into token IDs using V;
Save P as `paragraphs.json` and V as `vocab.json`;

Algorithm 2: Dataset Construction

Input: Paragraph token ID sequences P, vocabulary V
Output: Training samples $(style, x, y, mask)$
Let $N = |P|$, $K = 3$;
Compute $n_{\text{book}} = N/3$;
for $i = 1$ **to** N **do**
  if $i < n_{book}$ **then**
    style_id = 0
  else
    if $i < 2n_{book}$ **then**
      style_id = 1
    else
      style_id = 2
foreach *sequence* s **do**
  $x = s[0{:}|s| - 1]$;
  $y = s[1{:}|s|]$;
Pad sequences to same length and create mask with 1 for real tokens and 0 for padding;

Algorithm 3: Retro Bahdanau Additive Attention

Input: Current state $h_t \in R^H$, history $H_{1:t-1} \in R^{(t-1)\times H}$
Output: Context c_t, attention weights α
if $t = 1$ **then**
  Return zero context and zero attention vector;
Compute score: $e_i = v^\top \tanh(W_h H_i + W_s h_t)$;
Normalize weights: $\alpha_i = \exp(e_i)/\sum_j \exp(e_j)$;
Compute context: $c_t = \sum_i \alpha_i H_i$;
return c_t, α;

Algorithm 4: Style-Conditioned Retro-Attention LSTM Language Model

Input: Token IDs $x_{1:T}$, style ID s
Output: Logits $o_{1:T}$
Embed tokens: $E_t = \text{TokenEmb}(x_t)$;
Embed style: $S = \text{StyleEmb}(s)$ and repeat across all T steps;
Construct LSTM input: $Z_t = [E_t; S]$;
Run LSTM to obtain $H_{1:T}$;
for $t = 1$ **to** T **do**
    Current state: $h_t = H_t$;
    Past states: $H_{1:t-1}$;
    $(c_t, \alpha_t) = \text{RetroAttention}(h_t, H_{1:t-1})$;
    Feature vector: $u_t = [h_t;\ c_t;\ S]$;
    Logits: $o_t = W_o u_t + b_o$;
return $o_{1:T}$;

Algorithm 5: Training Loop for Style-Conditioned Retro-Attention LM

Input: DataLoader, model parameters θ
Output: Best model $\theta^\star$
for *epoch* $= 1$ **to** E **do**
    foreach *batch* $(style, x, y, mask)$ **do**
        Compute logits $o = f_\theta(x, style)$;
        Compute masked loss: $L = (\text{CE}(o, y) \cdot mask) / \sum mask$;
        Backpropagate $\nabla_\theta L$;
        Clip gradients to threshold τ;
        Update parameters;
    Save parameters if validation loss improves;