Reconstructing `TensorLog` for Scalable End-to-end Rule Learning

Supplementary Materials

I. Proofs

In this section, we provide detailed proofs for all propositions in this work. Note that we consider the worst-case time complexity for the training phases of end-to-end methods, where the time complexity is derived based on the number of floating-point multiplications and additions.

A. Proof of Proposition 1

Proposition 1. *The time complexity for the training phase of `TensorLog` is $\mathcal{O}(NL(|\mathcal{K}| + |\mathcal{R}||\mathcal{E}|))$.*

Proof. (I) We first prove that the time complexity of a forward computation step for `TensorLog` is $\mathcal{O}(NL(|\mathcal{K}| + |\mathcal{R}||\mathcal{E}|))$. Let $\mathrm{nnz}(M_{r_i})$ be the number of non-zero elements in the sparse matrix M_{r_i}. From Equations (1-2), we know that the complexity for each step in `TensorLog` comes from $\sum_{i=1}^{2n+1} \phi_{r,x}^{(k,l-1)}(w_i^{(r,k,l)} M_{r_i})$. Since the time complexity of $\sum_{i=1}^{2n+1} \phi_{r,x}^{(k,l-1)}(w_i^{(r,k,l)} M_{r_i})$ is $\sum_{i=1}^{2n+1}(\mathrm{nnz}(M_{r_i}) + |\mathcal{E}|)$, where $n = |\mathcal{R}|$. By $\sum_{i=1}^{2n+1} \mathrm{nnz}(M_{r_i}) = |\mathcal{K}|$, we can infer that the time complexity of a forward computation step for `TensorLog` is $\mathcal{O}(NL(|\mathcal{K}| + |\mathcal{R}||\mathcal{E}|))$.

(II) We then prove that the time complexity of a backward propagation step for `TensorLog` is $\mathcal{O}(NL(|\mathcal{K}| + |\mathcal{R}||\mathcal{E}|))$. For a backward propagation step, we know that only $w^{(r,k,l)}$ is trainable. The time complexity for calculating $\frac{\partial \mathcal{L}}{\partial \phi_{r,x}^{(k,l)}} M_{r_i}^\top$ is $\mathrm{nnz}(M_{r_i}) + |\mathcal{E}|$. Therefore, the time complexity of a backward propagation step for `TensorLog` is $\mathcal{O}(NL\sum_{i=1}^{2n+1}(\mathrm{nnz}(M_{r_i})+|\mathcal{E}|)) = \mathcal{O}(NL(|\mathcal{K}|+|\mathcal{R}||\mathcal{E}|))$. □

B. Proof of Proposition 2

Proposition 2. *The time complexity for the training phase of `FastLog` is $\mathcal{O}(NL|\mathcal{K}|)$.*

Proof. (I) From Equations (4-7), the time complexity of a forward computation step for `FastLog` is

$$NL(\underbrace{|\mathcal{K}|}_{\odot} + \underbrace{|\mathcal{K}|}_{\mathcal{F}_{\mathrm{f2e}}}).$$

Therefore, the time complexity of a forward computation step for `FastLog` is $\mathcal{O}(NL|\mathcal{K}|)$.

(II) For a backward propagation step of `FastLog`, we know that only $w^{(r,k,l)}$ is trainable. Let $z = \mathcal{F}_{\mathrm{e2f}}(\phi_{r,x}^{(k,l-1)}) \odot \mathcal{F}_{\mathrm{r2f}}(w^{(r,k,l)})$. The time complexity for calculating $\frac{\partial \mathcal{F}_{\mathrm{f2e}}(z)}{\partial z}$ is $\mathcal{O}(|\mathcal{K}|)$. The time complexity for calculating $\frac{\partial z}{\partial \mathcal{F}_{\mathrm{e2f}}(\phi_{r,x}^{(k,l-1)})}$ is $\mathcal{O}(|\mathcal{K}|)$. The time complexity for calculating $\frac{\partial z}{\partial \mathcal{F}_{\mathrm{r2f}}(w^{(r,k,l)})}$ is $\mathcal{O}(|\mathcal{K}|)$. The time complexity for calculating $\frac{\partial \mathcal{F}_{\mathrm{r2f}}(w^{(r,k,l)})}{\partial w^{(r,k,l)}}$ is $\mathcal{O}(|\mathcal{K}|)$. Therefore, the time complexity of a backward propagation step for `FastLog` is $\mathcal{O}(NL|\mathcal{K}|)$. □

C. Proof of Proposition 3

Proposition 3. *The space complexity for the training phase of `TensorLog` is $\mathcal{O}(NL|\mathcal{R}||\mathcal{E}|)$.*

Proof. (I) We first prove that the space complexity of a forward computation step for `TensorLog` is $\mathcal{O}(NL|\mathcal{R}||\mathcal{E}|)$. For a forward computation step, `TensorLog` requires storing all intermediate estimated truth degrees for all L steps for all N rules to calculate the gradient. Therefore, the space complexity of a forward computation step for `TensorLog` is $\mathcal{O}(NL|\mathcal{R}||\mathcal{E}|)$.

(II) We then prove that the space complexity of a backward propagation step for `TensorLog` is $\mathcal{O}(NL|\mathcal{R}||\mathcal{E}|)$. For a backward propagation step, `TensorLog` requires calculating the gradient of $w^{(r,k,l)}$. The space complexity for calculating $\frac{\partial \mathcal{L}}{\partial \phi_{r,x}^{(k,l)}} M_{r_i}^\top$ is $|\mathcal{E}|$. Therefore, the space complexity of a backward propagation step for `TensorLog` is $\mathcal{O}(NL\sum_{i=1}^{2n+1}|\mathcal{E}|) = \mathcal{O}(NL|\mathcal{R}||\mathcal{E}|)$. □

D. Proof of Proposition 4

Proposition 4. *The space complexity for the training phase of `FastLog` is $\mathcal{O}(NL(|\mathcal{R}| + |\mathcal{E}| + |\mathcal{K}|))$.*

Proof. (I) We first prove that the space complexity of a forward computation step for `FastLog` is $\mathcal{O}(NL(|\mathcal{R}|+|\mathcal{E}|+|\mathcal{K}|))$.For a forward computation step, `FastLog` requires storing the intermediate hidden states for all L steps for all N rules to calculate the gradients, resulting in a space complexity of $\mathcal{O}(NL|\mathcal{K})$. It also requires storing the intermediate estimated truth degrees for all L steps for all N rules to calculate the gradients, resulting in a space complexity of $\mathcal{O}(NL|\mathcal{E})$. Besides, the space complexity of the predicate selection weight is $\mathcal{O}(NL|\mathcal{R}|)$. Therefore, the space complexity of a forward computation step for `TensorLog` is $\mathcal{O}(NL(|\mathcal{R}|+|\mathcal{E}|+|\mathcal{K}|))$.

(II) We then prove that the space complexity of a backward propagation step for `FastLog` is $\mathcal{O}(NL(|\mathcal{R}|+|\mathcal{E}|+|\mathcal{K}|))$. Let $z = \mathcal{F}_{\mathrm{e2f}}(\phi_{r,x}^{(k,l-1)}) \odot \mathcal{F}_{\mathrm{r2f}}(w^{(r,k,l)})$. The space complexity for calculating $\frac{\partial \mathcal{F}_{\mathrm{f2e}}(z)}{\partial z}$ is $\mathcal{O}(|\mathcal{E}|+|\mathcal{K}|)$. The space complexity for calculating $\frac{\partial z}{\partial \mathcal{F}_{\mathrm{e2f}}(\phi_{r,x}^{(k,l-1)})}$ is $\mathcal{O}(|\mathcal{K}|)$. The space complexity for calculating $\frac{\partial z}{\partial \mathcal{F}_{\mathrm{r2f}}(w^{(r,k,l)})}$ is $\mathcal{O}(|\mathcal{K}|)$. The space complexity for calculating $\frac{\partial \mathcal{F}_{\mathrm{r2f}}(w^{(r,k,l)})}{\partial w^{(r,k,l)}}$ is $\mathcal{O}(|\mathcal{K}|+|\mathcal{R}|)$. Therefore, the space complexity of a backward propagation step for `FastLog` is $\mathcal{O}(NL(|\mathcal{R}| + |\mathcal{E}| + |\mathcal{K}|))$. □

E. Proof of Proposition 5

Proposition 5. *For an arbitrary triple* $(a, r, b) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$*,* $\forall N \geq 1, L \geq 1$: $\mathtt{TensorLog}(\theta_r^{N,L}, a, b) = \mathtt{FastLog}(\theta_r^{N,L}, a, b)$.

Proof. To prove Proposition 5, we first introduce three sparse matrices M_{e2f}, M_{r2f}, and M_{f2e}, where $M_{\text{e2f}} \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{K}|}$ (resp. $M_{\text{r2f}} \in \mathbb{R}^{(2n+1)\times|\mathcal{K}|}$ or $M_{\text{f2e}} \in \mathbb{R}^{|\mathcal{K}|\times|\mathcal{E}|}$) stores the mapping between a head entity (resp. relation or fact) and its corresponding fact (resp. fact or tail entity).

For all $1 \leq k \leq N, 1 \leq l \leq L$, it holds that

$$\begin{aligned}
\phi_{r,a}^{(k,l)} &= \mathcal{F}_{\text{f2e}}(\mathcal{F}_{\text{e2f}}(\phi_{r,a}^{(k,l-1)}) \odot \mathcal{F}_{\text{r2f}}(w^{(r,k,l)})) \\
&= ((\phi_{r,a}^{(k,l-1)} M_{\text{e2f}}) \odot (w^{(r,k,l)} M_{\text{r2f}})) M_{\text{f2e}} \\
&= \phi_{r,a}^{(k,l-1)} ((M_{\text{e2f}} \odot (w^{(r,k,l)} M_{\text{r2f}})) M_{\text{f2e}}) \\
&= \phi_{r,a}^{(k,l-1)} (\sum_{i=1}^{2n+1} w_i^{(r,k,l)} M_{r_i}).
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
\mathtt{FastLog}(\theta_r^{N,L}, a, b) &= (\sum_{k=1}^{N} \phi_{r,a}^{(k,L)}) v_b \\
&= (\sum_{k=1}^{N} ((\cdots (v_a^\top (\sum_{i=1}^{2n+1} w_i^{(r,k,1)} M_{r_i})) \\
&\quad (\sum_{i=1}^{2n+1} w_i^{(r,k,2)} M_{r_i})) \\
&\quad \cdots \\
&\quad (\sum_{i=1}^{2n+1} w_i^{(r,k,L)} M_{r_i}))) v_b \\
&= v_a^\top (\sum_{k=1}^{N} \prod_{l=1}^{L} \sum_{i=1}^{2n+1} w_i^{(r,k,l)} M_{r_i}) v_b \\
&= \mathtt{TensorLog}(\theta_r^{N,L}, a, b).
\end{aligned}$$

□

F. Proof of Proposition 6

Proposition 6. *The time complexity for the training phase of* $\mathtt{FastLog}^{c_1,c_2}$ *is* $\mathcal{O}(NLc_2)$.

Proof. (I) We first prove that the time complexity of a forward computation step for $\mathtt{FastLog}^{c_1,c_2}$ is $\mathcal{O}(NLc_2)$. From Proposition 2, we know that the time complexity of a forward computation step for $\mathtt{FastLog}$ is $\mathcal{O}(NL|\mathcal{K}|)$. From Equations (10-11), we know that only top-c_2 intermediate hidden states involve floating-point multiplications and additions. Therefore, the time complexity of a forward computation step for $\mathtt{FastLog}^{c_1,c_2}$ is $\mathcal{O}(NLc_2)$.

(II) We then prove that the time complexity of a backward propagation step for $\mathtt{FastLog}$ is $\mathcal{O}(NLc_2)$. From Equations (9), we know that only top-c_1 intermediate estimated truth degrees are used to calculate the gradients. From Equations (10), we know that only top-c_2 intermediate hidden states are used to calculate the gradients. Let $z = \hat{\mathcal{F}}_{\text{r2f}}^{c_2}(\hat{\mathcal{F}}_{\text{e2f}}^{c_1}(\phi_{r,x}^{(k,l-1)}), w^{(r,k,l)})$. The time complexity for calculating $\frac{\partial \hat{\mathcal{F}}_{\text{f2e}}(z)}{\partial z}$ is $\mathcal{O}(c_2)$ because z only has c_2 elements. The time complexity for calculating $\frac{\partial z}{\partial w^{(r,k,l)}}$ is $\mathcal{O}(c_2)$ because only the top-c_2 elements in $\hat{\mathcal{F}}_{\text{e2f}}^{c_1}(\phi_{r,x}^{(k,l-1)})$ are used to calculate gradients. The time complexity for calculating $\frac{\partial \hat{\mathcal{F}}_{\text{e2f}}^{c_1}(\phi_{r,x}^{(k,l-1)})}{\partial \phi_{r,x}^{(k,l-1)}}$ is $\mathcal{O}(c_2)$ because only the top-c_2 elements in $\hat{\mathcal{F}}_{\text{e2f}}^{c_1}(\phi_{r,x}^{(k,l-1)})$ are used to calculate gradients. Therefore, the time complexity of a backward propagation step for $\mathtt{FastLog}$ is $\mathcal{O}(NLc_2)$. □

G. Proof of Proposition 7

Proposition 7. *The space complexity for the training phase of* $\mathtt{FastLog}^{c_1,c_2}$ *is* $\mathcal{O}(NL(c_1 + c_2 + |\mathcal{R}|))$.

Proof. (I) We first prove that the space complexity of a forward computation step for $\mathtt{FastLog}^{c_1,c_2}$ is $\mathcal{O}(NL(c_1 + c_2 + |\mathcal{R}|))$. For a forward computation step, $\mathtt{FastLog}^{c_1,c_2}$ requires storing the intermediate estimated truth degrees with the size of c_1 for all L steps for all N rules to calculate the gradients. It also requires storing the intermediate hidden states with the size of c_2 for all L steps for all N rules to calculate the gradients. Besides, the space complexity of the predicate selection weight is $\mathcal{O}(NL|\mathcal{R}|)$. Therefore, the space complexity of a forward computation step for $\mathtt{FastLog}^{c_1,c_2}$ is $\mathcal{O}(NL(c_1 + c_2 + |\mathcal{R}|))$.

(II) We then prove that the space complexity of a backward propagation step for $\mathtt{FastLog}^{c_1,c_2}$ is $\mathcal{O}(NL(c_1 + c_2 + |\mathcal{R}|))$. Let $z = \hat{\mathcal{F}}_{\text{r2f}}^{c_2}(\hat{\mathcal{F}}_{\text{e2f}}^{c_1}(\hat{\phi}_{r,x}^{(k,l-1)}), w^{(r,k,l)})$. The space complexity for calculating $\frac{\partial \hat{\mathcal{F}}_{\text{f2e}}(z)}{\partial z}$ is $\mathcal{O}(c_1 + c_2)$. The space complexity for calculating $\frac{\partial z}{\partial \hat{\mathcal{F}}_{\text{e2f}}(\hat{\phi}_{r,x}^{(k,l-1)})}$ is $\mathcal{O}(c_2)$. The space complexity for calculating $\frac{\partial z}{\partial w^{(r,k,l)}}$ is $\mathcal{O}(c_2 + |\mathcal{R}|)$. Therefore, the space complexity of a backward propagation step for $\mathtt{FastLog}^{c_1,c_2}$ is $\mathcal{O}(NL(c_1 + c_2 + |\mathcal{R}|))$. □

H. Proof of Proposition 8

Proposition 8. *Given a knowledge graph* $\mathcal{G}$*, for an arbitrary triple* $(a, r, b) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$*,* $\forall N \geq 1, L \geq 1$: $\mathtt{FastLog}(\theta_r^{N,L}, a, b) = \mathtt{FastLog}^{|\mathcal{E}|,|\mathcal{K}|}(\theta_r^{N,L}, a, b)$.

Proof. From Equations (4-6) and (9-11), we know that $\hat{\mathcal{F}}_{\text{e2f}}^{|\mathcal{E}|}$ (resp. $\hat{\mathcal{F}}_{\text{r2f}}^{|\mathcal{K}|}$ or $\hat{\mathcal{F}}_{\text{f2e}}$) is equivalent to $\mathcal{F}_{\text{e2f}}$ (resp. $\mathcal{F}_{\text{r2f}}$ or $\mathcal{F}_{\text{f2e}}$) because both $\mathcal{T}^{|\mathcal{E}|}(\mathbb{T})$ and $\mathcal{T}^{|\mathcal{K}|}(\mathbb{T})$ return the original set $\mathbb{T}$ of tuples. Therefore, Equation (7) can be derived by:

$$
\begin{aligned}
\texttt{FastLog}(\theta_r^{N,L}, a, b) &= \sum_{k=1}^{N} \phi_{r,a}^{(k,L)} v_b \\
&= \sum_{k=1}^{N} \mathcal{F}_{\mathrm{f2e}}(\mathcal{F}_{\mathrm{r2f}}(w^{(r,k,L)}) \odot \mathcal{F}_{\mathrm{e2f}}(\\
&\quad \cdots \\
&\quad \mathcal{F}_{\mathrm{f2e}}(\mathcal{F}_{\mathrm{r2f}}(w^{(r,k,2)}) \odot \mathcal{F}_{\mathrm{e2f}}(\\
&\quad \mathcal{F}_{\mathrm{f2e}}(\mathcal{F}_{\mathrm{r2f}}(w^{(r,k,1)}) \odot \mathcal{F}_{\mathrm{e2f}}(v_x^\top)))) \\
&\quad \cdots))v_b \\
&= \sum_{k=1}^{N} \hat{\mathcal{F}}_{\mathrm{f2e}}(\hat{\mathcal{F}}_{\mathrm{r2f}}^{|\mathcal{K}|}(\hat{\mathcal{F}}_{\mathrm{e2f}}^{|\mathcal{E}|}(\\
&\quad \cdots \\
&\quad \hat{\mathcal{F}}_{\mathrm{f2e}}(\hat{\mathcal{F}}_{\mathrm{r2f}}^{|\mathcal{K}|}(\hat{\mathcal{F}}_{\mathrm{e2f}}^{|\mathcal{E}|}(v_x^\top), w^{(r,k,1)})), \cdots), \\
&\quad w^{(r,k,L)})) \\
&= \texttt{FastLog}^{|\mathcal{E}|,|\mathcal{K}|}(\theta_r^{N,L}, a, b)
\end{aligned}
$$

□

II. Formalization of Existing Methods

In the following, we introduce four SOTA end-to-end rule learning methods that employ TensorLog to learn CRs, including NeuralLP, DRUM, smDRUM, mmDRUM.

A. *The NeuralLP Method*

NeuralLP [1] is the first work that exploits TensorLog operators to learn CRs. Specifically, NeuralLP introduces a set of additional learnable parameters to pay attention to previous steps, thereby learning CRs with dynamic length without using the identity relation. Besides, NeuralLP leverages LSTM [2] networks to estimate both the predicate selection weights and the attention weights. Note that NeuralLP only simulates the inference of one CR, i.e., it holds that $N = 1$. Formally, given a query $(x, r, ?)$ and the maximum length of each rule L, NeuralLP first encodes r as a trainable vector $v_r \in \mathbb{R}^d$, where d denotes the dimensional size. Then an input sequence $(q_1, q_2, \cdots, q_{L+1})$ is created by setting $q_l = v_r$ for all $1 \le l \le L$ and $q_{L+1} = v^{\mathrm{end}}$, where v^{end} is a special trainable vector to capture the boundary of the input sequence. For all $1 \le l \le L+1$, the predicate selection weights $w^{(r,1,l)} \in [0,1]^{2n}$ are estimated by

$$
\begin{aligned}
h_l &= \mathrm{LSTM}(h_{l-1}, q_l), \\
w^{(r,1,l)} &= \mathrm{Softmax}(W h_l + b),
\end{aligned} \tag{14}
$$

where h_0 is a zero-padding d-dimensional vector. $W \in \mathbb{R}^{2n \times d}$ and $b \in \mathbb{R}^{2n}$ are trainable weights and bias, respectively. The attention weights $\alpha^{(r,1,l)} \in [0,1]^l$ are estimated by

$$
\alpha^{(r,1,l)} = \mathrm{Softmax}([h_0^\top h_l; h_1^\top h_l; \cdots; h_{l-1}^\top h_l]), \tag{15}
$$

where $[;]$ is the concatenation operation. The intermediate truth degrees $\phi_{r,x}^{(1,l)} \in \mathbb{R}^{|\mathcal{E}|}$ for NeuralLP are estimated by

$$
\phi_{r,x}^{(1,l)} = \begin{cases} \displaystyle\sum_{i=1}^{2n} (\sum_{j=0}^{l-1} \alpha_j^{(r,1,l)} \phi_{r,x}^{(1,j)})(w_i^{(r,1,l)} M_{r_i}), & 1 \le l \le L, \\ \displaystyle\sum_{j=0}^{L} \alpha_j^{(r,1,L+1)} \phi_{r,x}^{(1,j)}, & l = L+1, \end{cases} \tag{16}
$$

where $\phi_{r,x}^{(1,0)} = v_x^\top$. For an arbitrary triple $(x, r, y) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, the truth degree of (x, r, y) is estimated by

$$
\texttt{NeuralLP}(\theta_r^{1,L}, x, y) = \phi_{r,x}^{(1,L+1)} v_y, \tag{17}
$$

where $\theta_r^{1,L} = \{v_r, W, b\} \cup \theta_{\mathrm{LSTM}}$ is a set of trainable parameters for the head relation r, and θ_{LSTM} is the set of parameters used in the LSTM network.

The following Proposition 9 shows the time complexity of NeuralLP .

Proposition 9. *Let $\mathcal{K} = \mathcal{G} \cup \mathcal{G}^-$. The time complexity for the training phase of NeuralLP is $\mathcal{O}(L(|\mathcal{K}| + |\mathcal{R}||\mathcal{E}|) + L^2|\mathcal{E}|)$.*

Proof. (I) We first prove that the time complexity of a forward computation step for NeuralLP is $\mathcal{O}(L(|\mathcal{K}| + |\mathcal{R}||\mathcal{E}|) + L^2|\mathcal{E}|)$. From Equations (14-17), we know that the time complexity of a forward computation step for NeuralLP is

$$
\underbrace{L(|\mathcal{K}| + |\mathcal{R}||\mathcal{E}|)}_{\texttt{TensorLog}} + \underbrace{\frac{L(L-1)}{2}|\mathcal{E}|}_{\text{Aggregation}} + \underbrace{(L+1)(8d^2)}_{\text{LSTM network}} + \underbrace{(d^2+d)}_{\text{MLP}},
$$

where d denotes the hidden size. In general, it holds that $L(|\mathcal{K}| + |\mathcal{R}||\mathcal{E}|) + \frac{L(L-1)}{2}|\mathcal{E}| \gg (L+1)(8d^2) + (d^2+d)$. Therefore, the time complexity of a forward computation step for NeuralLP is $\mathcal{O}(L(|\mathcal{K}| + |\mathcal{R}||\mathcal{E}|) + L^2|\mathcal{E}|)$.

(II) We then prove that the time complexity of a forward computation step for NeuralLP is $\mathcal{O}(L(|\mathcal{K}| + |\mathcal{R}||\mathcal{E}|) + L^2|\mathcal{E}|)$. From Equations (14-17), we know that the time complexity of a backward propagation step for NeuralLP is

$$
\underbrace{2L(|\mathcal{K}| + |\mathcal{R}||\mathcal{E}|)}_{\texttt{TensorLog}} + \underbrace{L(L-1)|\mathcal{E}|}_{\text{Aggregation}} + \underbrace{2(L+1)(8d^2)}_{\text{LSTM network}} + \underbrace{2(d^2+d)}_{\text{MLP}}.
$$

In general, it holds that $L(|\mathcal{K}| + |\mathcal{R}||\mathcal{E}|) + L(L-1)|\mathcal{E}| \gg +2(L+1)(8d^2) + 2(d^2+d)$. Therefore, the time complexity of a forward computation step for NeuralLP is $\mathcal{O}(L(|\mathcal{K}| + |\mathcal{R}||\mathcal{E}|) + L^2|\mathcal{E}|)$. □

B. *The DRUM Method*

Different from NeuralLP, DRUM [3] introduces more trainable parameters to learn more CRs, and uses the identity relation to learn rules with dynamic length. Specifically, DRUM leverages N BiLSTM networks to estimate the predicate selection weights. Formally, given a query $(x, r, ?)$, the maximum number of rules to be learnt N, the maximum length of each rule L, DRUM first encodes r as a trainable vector $v_r \in \mathbb{R}^d$, where d denotes the dimensional size. For all $1 \leq k \leq N, 1 \leq l \leq L$, the predicate selection weights $w^{(r,k,l)} \in [0,1]^{2n+1}$ is estimated by

$$\begin{aligned}\overrightarrow{h}_l^{(k)} &= \overrightarrow{\text{BiLSTM}}^{(k)}(\overrightarrow{h}_{l-1}^{(k)}, v_r),\\ \overleftarrow{h}_{L-l+1}^{(k)} &= \overleftarrow{\text{BiLSTM}}^{(k)}(\overleftarrow{h}_{L-l}^{(k)}, v_r),\\ w^{(r,k,l)} &= \text{Softmax}(W[\overrightarrow{h}_l^{(k)}; \overleftarrow{h}_{L-l+1}^{(k)}] + b),\end{aligned} \tag{18}$$

where both $\overrightarrow{h}_0^{(k)}$ and $\overleftarrow{h}_{L+1}^{(k)}$ are set as zero-padding d-dimensional vectors. $W \in \mathbb{R}^{(2n+1)\times 2d}$ and $b \in \mathbb{R}^{2n+1}$ are trainable weights and bias, respectively. For an arbitrary triple $(x,r,y) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, the intermediate truth degrees $\phi_{r,x}^{(k,l)} \in \mathbb{R}^{|\mathcal{E}|}$ for DRUM are estimated by

$$\phi_{r,x}^{(k,l)} = \sum_{i=1}^{2n+1} \phi_{r,x}^{(k,l-1)}(w_i^{(r,k,l)} M_{r_i}), \tag{19}$$

where $\phi_{r,x}^{(k,0)} = v_x^\top$. The truth degree of (x,r,y) is estimated by

$$\texttt{DRUM}(\theta_r^{N,L}, x, y) = \sum_{k=1}^{N} \phi_{r,x}^{(k,L)} v_y, \tag{20}$$

where $\theta_r^{N,L} = \{v_r, W, b\} \cup \bigcup_{1\leq k\leq N} \theta_{\text{BiLSTM}}^{(k)}$ is a set of trainable parameters for the head relation r, and $\theta_{\text{BiLSTM}}^{(k)}$ is the set of parameters used in the k-th BiLSTM network.

C. The smDRUM *Method*

smDRUM [4] is proposed to enhance the faithfulness between DRUM and CRs, by introducing new tensorized operations. Note that smDRUM uses the same way as DRUM to estimate the predicate selection weights. For an arbitrary triple $(x,r,y) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, the intermediate truth degrees $\phi_{r,x}^{(k,l)} \in \mathbb{R}^{|\mathcal{E}|}$ for smDRUM are estimated by

$$\phi_{r,x}^{(k,l)} = \sum_{i=1}^{2n+1} \phi_{r,x}^{(k,l-1)} \otimes (w_i^{(r,k,l)} M_{r_i}), \tag{21}$$

where $\phi_{r,x}^{(k,0)} = v_x^\top$, $\otimes$ is the *max-production* operator, i.e., given two matrices $U \in \mathbb{R}^{a\times m}$ and $V \in \mathbb{R}^{m\times b}$, $(U \otimes V)_{i,j} = \max_{k=1}^{m} U_{i,k} \cdot V_{k,j}$ for all $1 \leq i \leq a$ and $1 \leq j \leq b$. The truth degree of (x,r,y) is estimated by

$$\texttt{smDRUM}(\theta_r^{N,L}, x, y) = \sum_{k=1}^{N} \phi_{r,x}^{(k,L)} v_y, \tag{22}$$

where $\theta_r^{N,L} = \{v_r, W, b\} \cup \bigcup_{1\leq k\leq N} \theta_{\text{BiLSTM}}^{(k)}$ is a set of trainable parameters for the head relation r.

D. The mmDRUM *Method*

mmDRUM [4] is another method proposed to enhance the faithfulness between DRUM and CRs. mmDRUM employs the same way in DRUM to estimate the predicate selection weights. Compared to smDRUM, mmDRUM introduces *max-pooling* to aggregate N rules. For an arbitrary triple $(x,r,y) \in \mathcal{E}\times\mathcal{R}\times\mathcal{E}$, the intermediate truth degrees $\phi_{r,x}^{(k,l)} \in \mathbb{R}^{|\mathcal{E}|}$ for mmDRUM are estimated by

$$\phi_{r,x}^{(k,l)} = \sum_{i=1}^{2n+1} \phi_{r,x}^{(k,l-1)} \otimes (w_i^{(r,k,l)} M_{r_i}), \tag{23}$$

where $\phi_{r,x}^{(k,0)} = v_x^\top$. The truth degree of (x,r,y) is estimated by

$$\texttt{mmDRUM}(\theta_r^{N,L}, x, y) = \max_{k=1}^{N} \phi_{r,x}^{(k,L)} v_y, \tag{24}$$

where $\theta_r^{N,L} = \{v_r, W, b\} \cup \bigcup_{1\leq k\leq N} \theta_{\text{BiLSTM}}^{(k)}$ is a set of trainable parameters for the head relation r.

The following Proposition 10 shows the time complexity of DRUM, smDRUM, and mmDRUM.

Proposition 10. *Let $\mathcal{K} = \mathcal{G} \cup \mathcal{G}^- \cup \{I(e,e) \mid e \in \mathcal{E}\}$. The time complexity for the training phase of* DRUM *,* smDRUM*, and* mmDRUM *is $\mathcal{O}(NL(|\mathcal{K}| + |\mathcal{R}||\mathcal{E}|))$.*

Proof. From Equations (19-24), we know that DRUM, smDRUM and mmDRUM has the same training time complexity.

(I) We first prove that the time complexity of a forward computation step for DRUM, smDRUM, and mmDRUM is $\mathcal{O}(NL(|\mathcal{K}|+|\mathcal{R}||\mathcal{E}|))$. From Equations (18-20), we know that the time complexity of a forward computation step for DRUM is

$$\underbrace{NL(|\mathcal{K}| + |\mathcal{R}||\mathcal{E}|)}_{\texttt{TensorLog}} + \underbrace{2N(L+1)(8d^2)}_{\text{BiLSTM networks}} + \underbrace{N((2d)^2 + 2d)}_{\text{MLPs}},$$

where d denotes the hidden size. In general, it holds that $NL(|\mathcal{K}| + |\mathcal{R}||\mathcal{E}|) \gg N(L+1)(8d^2) + N((2d)^2 + 2d)$. Therefore, the time complexity of a forward computation step for DRUM, smDRUM, and mmDRUM is $\mathcal{O}(NL(|\mathcal{K}| + |\mathcal{R}||\mathcal{E}|))$.

(II) We then prove that the time complexity of a forward computation step for DRUM, smDRUM, and mmDRUM is $\mathcal{O}(NL(|\mathcal{K}|+|\mathcal{R}||\mathcal{E}|))$. From Equations (18-20), we know that the time complexity of a backward propagation step for DRUM, smDRUM, and mmDRUM is

$$\underbrace{2NL(|\mathcal{K}| + |\mathcal{R}||\mathcal{E}|)}_{\texttt{TensorLog}} + \underbrace{4N(L+1)(8d^2)}_{\text{BiLSTM networks}} + \underbrace{2N((2d)^2 + 2d)}_{\text{MLPs}}.$$

In general, it holds that $2NL(|\mathcal{K}| + |\mathcal{R}||\mathcal{E}|) \gg 4N(L+1)(8d^2)+2N((2d)^2+2d)$. Therefore, the time complexity of a forward computation step for DRUM, smDRUM, and mmDRUM is $\mathcal{O}(NL(|\mathcal{K}| + |\mathcal{R}||\mathcal{E}|))$. □

III. FastLog-enhanced Methods

SOTA end-to-end rule learning methods can be enhanced by replacing TensorLog operators with FastLog operators. By X-FL we denote the FastLog-enhanced methods, where X can be NeuralLP, DRUM, smDRUM, and mmDRUM.

A. The NeuralLP*-FL Method*

In the following, we elaborate on enhancing the NeuralLP method with FastLog. It is worth noting that the use of FastLog does not affect the estimation of selection weights. Therefore, we can still use Equation (14-15) for selection weight estimation. Let $\mathcal{K} = \mathcal{G}\cup\mathcal{G}^-$. For an arbitrary triple $(x, r, y) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, the intermediate truth degrees $\phi_{r,x}^{(1,l)} \in \mathbb{R}^{|\mathcal{E}|}$ are estimated by

$$\phi_{r,x}^{(1,l)} = \begin{cases} \mathcal{F}_{\mathrm{f2e}}(\mathcal{F}_{\mathrm{e2f}}(\sum_{j=0}^{l-1} \alpha_j^{(r,1,l)} \phi_{r,x}^{(1,j)}) \odot \mathcal{F}_{\mathrm{r2f}}(w^{(r,1,l)})), & 1 \le l \le L, \\ \sum_{j=0}^{L} \alpha_j^{(r,1,L+1)} \phi_{r,x}^{(1,j)}, & l = L+1, \end{cases} \tag{25}$$

where $\phi_{r,x}^{(1,0)} = v_x$. The truth degree of (x, r, y) is estimated by

$$\texttt{NeuralLP}\text{−FL}(\theta_r^{1,L}, x, y) = \phi_{r,x}^{(1,L+1)} v_y, \tag{26}$$

The following Proposition 11 shows the time complexity of `NeuralLP`-FL.

Proposition 11. *The time complexity of for the training phase of `NeuralLP`-FL is $\mathcal{O}(L|\mathcal{K}| + L^2|\mathcal{E}|)$.*

Proof. (I) We first prove that the time complexity of a forward computation step for `NeuralLP`-FL is $\mathcal{O}(L|\mathcal{K}| + L^2|\mathcal{E}|)$. From Proposition 2, we know that the time complexity of a forward computation step for `FastLog` is $\mathcal{O}(NL|\mathcal{K}|)$. From Equations (25-26), we know that the time complexity of a forward computation step for `NeuralLP`-FL is

$$\underbrace{L|\mathcal{K}|}_{\texttt{FastLog}} + \underbrace{\frac{L(L-1)}{2}|\mathcal{E}|}_{\text{Aggregation}} + \underbrace{(L+1)(8d^2)}_{\text{LSTM network}} + \underbrace{(d^2+d)}_{\text{MLP}}$$

where d denotes the hidden size. In general, it holds that $L|\mathcal{K}| + \frac{L(L-1)}{2}|\mathcal{E}| \gg (L+1)(8d^2) + (d^2 + d)$. Therefore, the time complexity of a forward computation step for `NeuralLP`-FL is $\mathcal{O}(L|\mathcal{K}| + L^2|\mathcal{E}|)$.

(II) We then prove that the time complexity of a backward propagation step for `NeuralLP`-FL is $\mathcal{O}(L|\mathcal{K}| + L^2|\mathcal{E}|)$. For a backward propagation step for `NeuralLP`-FL, we know that both $w^{(r,1,l)}$ and $\alpha^{(r,1,l)}$ are trainable. Let $z = \mathcal{F}_{\mathrm{e2f}}(\sum_{j=0}^{l-1} \alpha_j^{(r,1,l)} \phi_{r,x}^{(1,j)}) \odot \mathcal{F}_{\mathrm{r2f}}(w^{(r,1,l)})$. The time complexity for calculating $\frac{\partial \mathcal{F}_{\mathrm{f2e}}(z)}{\partial z}$ is $\mathcal{O}(|\mathcal{K}|)$. The time complexity for calculating $\frac{\partial z}{\partial \mathcal{F}_{\mathrm{e2f}}(\sum_{j=0}^{l-1} \alpha_j^{(r,1,l)} \phi_{r,x}^{(1,j)})}$ is $\mathcal{O}(|\mathcal{K}|)$. The time complexity for calculating $\frac{\partial \mathcal{F}_{\mathrm{e2f}}(\sum_{j=0}^{l-1} \alpha_j^{(r,1,l)} \phi_{r,x}^{(1,j)})}{\partial \alpha^{(r,1,l)}}$ is $\mathcal{O}(L^2|\mathcal{E}|)$. The time complexity for calculating $\frac{\partial z}{\partial \mathcal{F}_{\mathrm{r2f}}(w^{(r,1,l)})}$ is $\mathcal{O}(|\mathcal{K}|)$. The time complexity for calculating $\frac{\partial \mathcal{F}_{\mathrm{r2f}}(w^{(r,1,l)})}{\partial w^{(r,1,l)}}$ is $\mathcal{O}(|\mathcal{K}|)$. Therefore, the time complexity of a backward propagation step for `NeuralLP`-FL is $\mathcal{O}(L|\mathcal{K}| + L^2|\mathcal{E}|)$. □

By being enhanced by `FastLog`, the time complexity of a forward computation step for `NeuralLP` is reduced from $\mathcal{O}(L(|\mathcal{K}| + |\mathcal{R}||\mathcal{E}|) + L^2|\mathcal{E}|)$ to $\mathcal{O}(L|\mathcal{K}| + L^2|\mathcal{E}|)$. The time complexity of a backward propagation step for `NeuralLP` is reduced from $\mathcal{O}(L(|\mathcal{K}|+|\mathcal{R}||\mathcal{E}|)+L^2|\mathcal{E}|)$ to $\mathcal{O}(L|\mathcal{K}|+L^2|\mathcal{E}|)$. The following Proposition 12 demonstrates the correctness of `NeuralLP`-FL.

Proposition 12. *For an arbitrary triple $(a, r, b) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, $\forall L \ge 1$: `NeuralLP`$-\mathrm{FL}(\theta_r^{1,L}, a, b) =$ `NeuralLP`$(\theta_r^{1,L}, a, b)$.*

Proof. To prove Proposition 12, we first introduce three sparse matrices M_{e2f}, M_{r2f}, and M_{f2e}, where $M_{\mathrm{e2f}} \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{K}|}$ (resp. $M_{\mathrm{r2f}} \in \mathbb{R}^{2n\times|\mathcal{K}|}$ or $M_{\mathrm{f2e}} \in \mathbb{R}^{|\mathcal{K}|\times|\mathcal{E}|}$) stores the mapping between a head entity (resp. relation or fact) and its corresponding fact (resp. fact or tail entity).

For all $1 \le l \le L$, it holds that

$$\begin{aligned} \phi_{r,a}^{(l)} &= \mathcal{F}_{\mathrm{f2e}}(\mathcal{F}_{\mathrm{e2f}}(\sum_{j=0}^{l-1} \alpha_j^{(r,l)} \phi_{r,a}^{(j)}) \odot \mathcal{F}_{\mathrm{r2f}}(w^{(r,l)})) \\ &= ((\sum_{j=0}^{l-1} \alpha_j^{(r,l)} \phi_{r,a}^{(j)}) M_{\mathrm{e2f}} \odot (w^{(r,l)} M_{\mathrm{r2f}})) M_{\mathrm{f2e}} \\ &= (\sum_{j=0}^{l-1} \alpha_j^{(r,l)} \phi_{r,a}^{(j)})((M_{\mathrm{e2f}} \odot (w^{(r,l)} M_{\mathrm{r2f}})) M_{\mathrm{f2e}}) \\ &= (\sum_{j=0}^{l-1} \alpha_j^{(r,l)} \phi_{r,a}^{(j)})(\sum_{i=1}^{2n} w_i^{(r,l)} M_{r_i}) \\ &= \sum_{i=1}^{2n}(\sum_{j=0}^{l-1} \alpha_j^{(r,l)} \phi_{r,a}^{(j)})(w_i^{(r,l)} M_{r_i}) \end{aligned}$$

Therefore, we have

$$\begin{aligned} \texttt{NeuralLP}\text{−FL}(\theta_r^{L}, a, b) &= \phi_{r,a}^{(L+1)} v_b \\ &= \sum_{j=0}^{L} \alpha_j^{(r,L+1)} \phi_{r,a}^{(j)} \\ &= \sum_{j=0}^{L} \alpha_j^{(r,L+1)} (\\ &\quad \sum_{i=1}^{2n}(\sum_{k=0}^{j-1} \alpha_k^{(r,j)} \phi_{r,a}^{(k)})(w_i^{(r,j)} M_{r_i})) \\ &= \texttt{NeuralLP}(\theta_r^{L}, a, b) \end{aligned}$$

□

This proposition reveals that the efficacy of `NeuralLP` will not be impaired by applying `FastLog`.

B. The `DRUM`-FL Method

In the following, we elaborate on enhancing the `DRUM` method with `FastLog`. Let $\mathcal{K} = \mathcal{G} \cup \mathcal{G}^- \cup \{I(e, e) \mid e \in \mathcal{E}\}$. Similarly with `DRUM`, `DRUM`-FL also uses Equation (18) for selection weight estimation. For all $1 \le k \le N, 1 \le l \le L$, the intermediate truth degrees $\phi_{r,x}^{(k,l)} \in \mathbb{R}^{|\mathcal{E}|}$ are estimated by

$$\phi_{r,x}^{(k,l)} = \mathcal{F}_{\mathrm{f2e}}(\mathcal{F}_{\mathrm{e2f}}(\phi_{r,x}^{(k,l-1)}) \odot \mathcal{F}_{\mathrm{r2f}}(w^{(r,k,l)})), \tag{27}$$

where $\phi_{r,x}^{(k,0)} = v_x^\top$. The truth degree of (x, r, y) is estimated by

$$\texttt{DRUM}\text{−FL}(\theta_r^{N,L}, x, y) = (\sum_{k=1}^{N} \phi_{r,x}^{(k,L)}) v_y, \tag{28}$$

The following Proposition 13 shows the time complexity of `DRUM`-FL.

Proposition 13. *The time complexity for the training phase of* `DRUM`*-FL is* $\mathcal{O}(NL|\mathcal{K}|)$.

Proof. (I) We first prove that the time complexity of a forward computation step for `DRUM`-FL is $\mathcal{O}(NL|\mathcal{K}|)$. From Proposition 2, we know that the time complexity of a forward computation step for `FastLog` is $\mathcal{O}(NL|\mathcal{K}|)$. From Equations (27-28), we know that the time complexity of a forward computation step for `DRUM`-FL is

$$\underbrace{NL|\mathcal{K}|}_{\texttt{FastLog}} + \underbrace{2N(L+1)(8d^2)}_{\text{BiLSTM networks}} + \underbrace{N((2d)^2+2d)}_{\text{MLPs}}$$

where d denotes the hidden size. In general, it holds that $NL|\mathcal{K}| \gg +2N(L+1)(8d^2) + ((2d)^2+2d)$. Therefore, the time complexity of a forward computation step for `DRUM`-FL is $\mathcal{O}(NL|\mathcal{K}|)$.

(II) We then prove that the time complexity of a backward propagation step for `DRUM`-FL is $\mathcal{O}(NL|\mathcal{K}|)$. For a backward propagation step of `DRUM`-FL, we know that only $w^{(r,k,l)}$ is trainable. Let $z = \mathcal{F}_{\text{e2f}}(\phi_{r,x}^{(k,l-1)}) \odot \mathcal{F}_{\text{r2f}}(w^{(r,k,l)})$. The time complexity for calculating $\frac{\partial \mathcal{F}_{\text{f2e}}(z)}{\partial z}$ is $\mathcal{O}(|\mathcal{K}|)$. The time complexity for calculating $\frac{\partial z}{\partial \mathcal{F}_{\text{e2f}}(\phi_{r,x}^{(k,l-1)})}$ is $\mathcal{O}(|\mathcal{K}|)$. The time complexity for calculating $\frac{\partial z}{\partial \mathcal{F}_{\text{r2f}}(w^{(r,k,l)})}$ is $\mathcal{O}(|\mathcal{K}|)$. The time complexity for calculating $\frac{\partial \mathcal{F}_{\text{r2f}}(w^{(r,k,l)})}{\partial w^{(r,l,l)}}$ is $\mathcal{O}(|\mathcal{K}|)$. Therefore, the time complexity of a backward propagation step for `DRUM`-FL is $\mathcal{O}(NL|\mathcal{K}|)$. □

By being enhanced by `FastLog`, the time complexity of a forward computation step for `DRUM` is reduced from $\mathcal{O}(NL(|\mathcal{K}| + |\mathcal{R}||\mathcal{E}|))$ to $\mathcal{O}(NL|\mathcal{K}|)$. The time complexity of a backward propagation step for `DRUM` is reduced from $\mathcal{O}(NL(|\mathcal{K}| + |\mathcal{R}||\mathcal{E}|))$ to $\mathcal{O}(NL|\mathcal{K}|)$. The following Proposition 14 demonstrates the correctness of `DRUM`-FL.

Proposition 14. *For an arbitrary triple* $(a, r, b) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, $\forall N \geq 1, L \geq 1 :$ `DRUM`$-\text{FL}(\theta_r^{N,L}, a, b) =$ `DRUM`$(\theta_r^{N,L}, a, b)$.

Proof. To prove Proposition 14, we first introduce three sparse matrices M_{e2f}, M_{r2f}, and M_{f2e}, where $M_{\text{e2f}} \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{K}|}$ (resp. $M_{\text{r2f}} \in \mathbb{R}^{(2n+1)\times|\mathcal{K}|}$ or $M_{\text{f2e}} \in \mathbb{R}^{|\mathcal{K}|\times|\mathcal{E}|}$) stores the mapping between a head entity (resp. relation or fact) and its corresponding fact (resp. fact or tail entity).

For all $1 \leq k \leq N, 1 \leq l \leq L$, it holds that

$$\begin{aligned}\phi_{r,a}^{(k,l)} &= \mathcal{F}_{\text{f2e}}(\mathcal{F}_{\text{e2f}}(\phi_{r,a}^{(k,l-1)}) \odot \mathcal{F}_{\text{r2f}}(w^{(r,k,l)})) \\ &= ((\phi_{r,a}^{(k,l-1)} M_{\text{e2f}}) \odot (w^{(r,k,l)} M_{\text{r2f}}))M_{\text{f2e}} \\ &= \phi_{r,a}^{(k,l-1)}((M_{\text{e2f}} \odot (w^{(r,k,l)} M_{\text{r2f}}))M_{\text{f2e}}) \\ &= \phi_{r,a}^{(k,l-1)}(\sum_{i=1}^{2n+1} w_i^{(r,k,l)} M_{r_i})\end{aligned}$$

Therefore, we have

$$\begin{aligned}\texttt{DRUM}-\text{FL}(\theta_r^{N,L}, a, b) &= (\sum_{k=1}^{N} \phi_{r,a}^{(k,L)})v_b \\ &= (\sum_{k=1}^{N}((\cdots(v_a^\top(\sum_{i=1}^{2n+1} w_i^{(r,k,1)} M_{r_i})) \\ &\quad (\sum_{i=1}^{2n+1} w_i^{(r,k,2)} M_{r_i})) \\ &\quad \cdots \\ &\quad (\sum_{i=1}^{2n+1} w_i^{(r,k,L)} M_{r_i})))v_b \\ &= v_a^\top(\sum_{k=1}^{N}\prod_{l=1}^{L}\sum_{i=1}^{2n+1} w_i^{(r,k,l)} M_{r_i})v_b \\ &= \texttt{DRUM}(\theta_r^{N,L}, a, b)\end{aligned}$$

□

This proposition reveals that the efficacy of `DRUM` will not be impaired by applying `FastLog`.

C. The `smDRUM`*-FL Method*

Similar to `DRUM`-FL, for all $1 \leq k \leq N, 1 \leq l \leq L$, the formalization of `smDRUM`-FL is defined as

$$\phi_{r,x}^{(k,l)} = \mathcal{F}_{\text{f2e}}^{\max}(\mathcal{F}_{\text{e2f}}(\phi_{r,x}^{(k,l-1)}) \odot \mathcal{F}_{\text{r2f}}(w^{(r,k,l)})), \tag{29}$$

where $\phi_{r,x}^{(k,0)} = v_x^\top$. $\mathcal{F}_{\text{f2e}}^{\max} : \mathbb{R}^{|\mathcal{K}|} \to \mathbb{R}^{|\mathcal{E}|}$ is a function such that the i-th elements of $\mathcal{F}_{\text{f2e}}^{\max}(v)$ is

$$[\mathcal{F}_{\text{f2e}}^{\max}(v)]_i = \max_{j:\text{tail}(\tau_j)=i} v_j. \tag{30}$$

The truth degree of (x, r, y) is estimated by

$$\texttt{smDRUM}-\text{FL}(\theta_r^{N,L}, x, y) = (\sum_{k=1}^{N} \phi_{r,x}^{(k,L)})v_y. \tag{31}$$

Note that `smDRUM`-FL has the same time complexity as `DRUM`-FL. The following Proposition 15 demonstrates the correctness of `smDRUM`-FL.

Proposition 15. *For an arbitrary triple* $(a, r, b) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, $\forall N \geq 1, L \geq 1 :$ `smDRUM`$-\text{FL}(\theta_r^{N,L}, a, b) =$ `smDRUM`$(\theta_r^{N,L}, a, b)$.

Proof. To prove Proposition 15, we first introduce three sparse matrices M_{e2f}, M_{r2f}, and M_{f2e}, where $M_{\text{e2f}} \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{K}|}$ (resp. $M_{\text{r2f}} \in \mathbb{R}^{(2n+1)\times|\mathcal{K}|}$ or $M_{\text{f2e}} \in \mathbb{R}^{|\mathcal{K}|\times|\mathcal{E}|}$) stores the mapping between a head entity (resp. relation or fact) and its corresponding fact (resp. fact or tail entity).

For all $1 \leq k \leq N, 1 \leq l \leq L$, it holds that

$$\begin{aligned}\phi_{r,a}^{(k,l)} &= \mathcal{F}_{\text{f2e}}^{\max}(\mathcal{F}_{\text{e2f}}(\phi_{r,a}^{(k,l-1)}) \odot \mathcal{F}_{\text{r2f}}(w^{(r,k,l)})) \\ &= ((\phi_{r,a}^{(k,l-1)} M_{\text{e2f}}) \odot (w^{(r,k,l)} M_{\text{r2f}})) \otimes M_{\text{f2e}} \\ &= \phi_{r,a}^{(k,l-1)}((M_{\text{e2f}} \odot (w^{(r,k,l)} M_{\text{r2f}})) \otimes M_{\text{f2e}}) \\ &= \phi_{r,a}^{(k,l-1)} \otimes ((M_{\text{e2f}} \odot (w^{(r,k,l)} M_{\text{r2f}}))M_{\text{f2e}}) \\ &= \phi_{r,a}^{(k,l-1)} \otimes (\sum_{i=1}^{2n+1} w_i^{(r,k,l)} M_{r_i})\end{aligned}$$

TABLE I: Hyper-parameter settings for DRUM-FL+ on different datasets.

Hyper-parameter	Dataset			
	UMLS	FB15k-237	Wikidata5M	Freebase
Maximum number of rules N	100	3	10	3
Maximum size of body atoms L	3	3	3	3
Training time	7.3m	1.5h	24h	22.2h
Batch size	64	64	16	2
Learning rate	1e-3	1e-3	3e-4	1e-3
Hidden size d for estimating predicate selection weights	128	128	2048	128

Therefore, we have

$$\begin{aligned}
\texttt{smDRUM}-\mathrm{FL}(\theta_r^{N,L}, a, b) &= (\sum_{k=1}^{N} \phi_{r,a}^{(k,L)})v_b \\
&= (\sum_{k=1}^{N}((\cdots(v_a^\top \otimes (\sum_{i=1}^{2n+1} w_i^{(r,k,1)} M_{r_i})) \\
&\otimes (\sum_{i=1}^{2n+1} w_i^{(r,k,2)} M_{r_i})) \\
&\cdots \\
&\otimes (\sum_{i=1}^{2n+1} w_i^{(r,k,L)} M_{r_i})))v_b \\
&= v_a^\top (\sum_{k=1}^{N} \bigotimes_{l=1}^{L} \sum_{i=1}^{2n+1} w_i^{(r,k,l)} M_{r_i})v_b \\
&= \texttt{smDRUM}(\theta_r^{N,L}, a, b)
\end{aligned}$$

□

This proposition reveals that the efficacy of smDRUM will not be impaired by applying FastLog.

D. *The mmDRUM-FL Method*

Similar to DRUM-FL and smDRUM-FL, the formalization of mmDRUM-FL is defined as

$$\phi_{r,x}^{(k,l)} = \mathcal{F}_{\mathrm{f2e}}^{\max}(\mathcal{F}_{\mathrm{e2f}}(\phi_{r,x}^{(k,l-1)}) \odot \mathcal{F}_{\mathrm{r2f}}(w^{(r,k,l)})), \qquad (32)$$

where $\phi_{r,x}^{(k,0)} = v_x^\top$. The truth degree of (x, r, y) is estimated by

$$\texttt{mmDRUM}-\mathrm{FL}(\theta_r^{N,L}, x, y) = \max_{k=1}^{N} \phi_{r,x}^{(k,L)} v_y. \qquad (33)$$

Note that mmDRUM-FL has the same time complexity as DRUM-FL and smDRUM-FL. The following Proposition 16 shows the correctness of mmDRUM-FL.

Proposition 16. *For an arbitrary triple $(a, r, b) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, $\forall N \geq 1, L \geq 1$: mmDRUM$-\mathrm{FL}(\theta_r^{N,L}, a, b)$ = mmDRUM$(\theta_r^{N,L}, a, b)$.*

Proof. To prove Proposition 16, we first introduce three sparse matrices M_{e2f}, M_{r2f}, and M_{f2e}, where $M_{\mathrm{e2f}} \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{K}|}$ (resp. $M_{\mathrm{r2f}} \in \mathbb{R}^{(2n+1)\times|\mathcal{K}|}$ or $M_{\mathrm{f2e}} \in \mathbb{R}^{|\mathcal{K}|\times|\mathcal{E}|}$) stores the mapping between a head entity (resp. relation or fact) and its corresponding fact (resp. fact or tail entity).

For all $1 \leq k \leq N, 1 \leq l \leq L$, it holds that

$$\begin{aligned}
\phi_{r,a}^{(k,l)} &= \mathcal{F}_{\mathrm{f2e}}^{\max}(\mathcal{F}_{\mathrm{e2f}}(\phi_{r,a}^{(k,l-1)}) \odot \mathcal{F}_{\mathrm{r2f}}(w^{(r,k,l)})) \\
&= ((\phi_{r,a}^{(k,l-1)} M_{\mathrm{e2f}}) \odot (w^{(r,k,l)} M_{\mathrm{r2f}})) \otimes M_{\mathrm{f2e}} \\
&= \phi_{r,a}^{(k,l-1)}((M_{\mathrm{e2f}} \odot (w^{(r,k,l)} M_{\mathrm{r2f}})) \otimes M_{\mathrm{f2e}}) \\
&= \phi_{r,a}^{(k,l-1)} \otimes ((M_{\mathrm{e2f}} \odot (w^{(r,k,l)} M_{\mathrm{r2f}}))M_{\mathrm{f2e}}) \\
&= \phi_{r,a}^{(k,l-1)} \otimes (\sum_{i=1}^{2n+1} w_i^{(r,k,l)} M_{r_i})
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
\texttt{mmDRUM}-\mathrm{FL}(\theta_r^{N,L}, a, b) &= (\max_{k=1}^{N} \phi_{r,a}^{(k,L)})v_b \\
&= (\max_{k=1}^{N}((\cdots(v_a^\top \otimes (\sum_{i=1}^{2n+1} w_i^{(r,k,1)} M_{r_i})) \\
&\otimes (\sum_{i=1}^{2n+1} w_i^{(r,k,2)} M_{r_i})) \\
&\cdots \\
&\otimes (\sum_{i=1}^{2n+1} w_i^{(r,k,L)} M_{r_i})))v_b \\
&= v_a^\top (\max_{k=1}^{N} \bigotimes_{l=1}^{L} \sum_{i=1}^{2n+1} w_i^{(r,k,l)} M_{r_i})v_b \\
&= \texttt{mmDRUM}(\theta_r^{N,L}, a, b)
\end{aligned}$$

□

This proposition reveals that the efficacy of mmDRUM will not be impaired by applying FastLog.

IV. Hyper-parameters for DRUM-FL+

To help reproduce our results for DRUM-FL+, we provide the detailed hyper-parameter settings used in our experiments. Table I reports the detailed hyper-parameter settings for DRUM-FL+ in regard to different datasets.

References

[1] F. Yang, Z. Yang, and W. W. Cohen, "Differentiable learning of logical rules for knowledge base reasoning," in *NIPS*, 2017, pp. 2319–2328.

[2] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[3] A. Sadeghian, M. Armandpour, P. Ding, and D. Z. Wang, "DRUM: end-to-end differentiable rule mining on knowledge graphs," in *NeurIPS*, 2019, pp. 15 321–15 331.

[4] X. Wang, D. J. T. Cucala, B. C. Grau, and I. Horrocks, "Faithful rule extraction for differentiable rule learning models," in *ICLR*, 2024.