

Relation-Conditioned Continuous Time for TKBC (Design & Spec)

Mục tiêu: Thay thế hoàn toàn cơ chế time hiện tại (Fourier + Linear + gate trộn với 1) bằng **Relation-conditioned continuous time + residual gate**, nhằm buộc time thực sự ảnh hưởng tới scoring và tránh *static shortcut*.

Tài liệu này là **đặc tả kỹ thuật chính xác** (công thức + hàm + chú thích) để giao cho AI (Copilot/Cursor) code lại mô hình theo đúng ý đồ.

1. Mục tiêu thiết kế

Mô hình mới cần thoả mãn:

1. **Time phụ thuộc relation:** mỗi relation có động học thời gian riêng
2. **Không có đường thoát “tắt time”** (không gate về vector 1)
3. **Giữ PairRE core** để so sánh công bằng với baseline
4. **Hỗ trợ continuous time & extrapolation**

2. Ký hiệu

- h, t : head/tail entity embedding, $\in \mathbb{R}^d$
- r^H, r^T : relation embeddings kiểu PairRE, $\in \mathbb{R}^d$
- r : relation id
- τ : thời gian liên tục (scalar), đã chuẩn hoá từ `time_id`
- d : dimension embedding (vd 128)
- K : số tần số Fourier (vd 16 hoặc 32)
- ω_k : tần số (global, shared)

3. Chuẩn hoá thời gian (BẮT BUỘC)

Cho `time_id = l \in \{0, \dots, T-1\}`.

Khuyến nghị dùng miền **[0,1]** để ổn định pha:

$$\tau(l) = \frac{l}{T-1}$$

Hoặc scale nhẹ nếu cần:

$$\tau(l) = s \cdot \frac{l}{T-1}, \quad s \in \{1, 10\}$$

Không dùng $[0, 100]$ ở phiên bản đầu tiên vì dễ aliasing sin/cos.

4. Relation-conditioned Continuous Time Encoder

4.1 Ý tưởng cốt lõi

Thay vì time embedding dùng chung $m(\tau)$, ta học:

$$m_r(\tau) = f_r(\tau)$$

Mỗi relation có: - **Trend riêng** (tốc độ thay đổi): a_r - **Amplitude riêng**: $A_{r,k}$ - **Phase riêng**: $P_{r,k}$

Tần số ω_k được **chia sẻ toàn cục** để giảm tham số.

4.2 Time features theo relation

Tạo vector $z_r(\tau) \in \mathbb{R}^{1+K}$:

Trend:

$$z_{r,0}(\tau) = a_r \cdot \tau$$

Periodic:

$$z_{r,k}(\tau) = A_{r,k} \cdot \sin(\omega_k \tau + P_{r,k}), \quad k = 1..K$$

Gộp:

$$z_r(\tau) = [z_{r,0}, z_{r,1}, \dots, z_{r,K}]$$

4.3 Chiếu lên không gian embedding

Dùng projection chung + nonlinearity:

$$m_r(\tau) = \tanh(W_{proj} z_r(\tau) + b_{proj}) \in \mathbb{R}^d$$

Giải thích: - tanh giữ ổn định, tránh nổ - Khác với Linear cũ vì input đã *relation-specific* và có cấu trúc thời gian

4.4 Tham số cần học

Per relation: - $a_r \in \mathbb{R}$ (trend) - $A_r \in \mathbb{R}^K$ (amplitude) - $P_r \in \mathbb{R}^K$ (phase)

Global: - $\omega \in \mathbb{R}^K$ (frequency, nên fixed ban đầu) - W_{proj}, b_{proj}

Khởi tạo khuyến nghị: - $a_r \sim \mathcal{N}(0, 0.01)$ - $A_r \sim \mathcal{N}(0, 0.1)$ - $P_r \sim \text{Uniform}(-\pi, \pi)$ - W_{proj} : Xavier init

5. Residual Gate (QUAN TRỌNG NHẤT)

Thay gate cũ có đường thoát về vector 1 bằng **residual gate**:

$$g_r(\tau) = \mathbf{1} + \beta \cdot \tanh(m_r(\tau))$$

- β : hyperparameter (hoặc learnable), khuyến nghị **0.1–0.5**, bắt đầu **0.2**
- Không dùng α_r ở phiên bản đầu tiên để tránh *time off*

Tính chất: - Luôn có tác động multiplicative của time - Không thể collapse về **1** nếu $m_r(\tau)$ thay đổi

6. Scoring Function (PairRE core)

PairRE interaction:

$$x(h, r, t) = h \circ r^H - t \circ r^T$$

Áp dụng gate:

$$\tilde{x}(h, r, t, \tau) = x(h, r, t) \circ g_r(\tau)$$

Score:

$$\phi(h, r, t, \tau) = -\|\tilde{x}(h, r, t, \tau)\|_1$$

(Giữ L1 để so sánh công bằng với baseline PairRE)

7. Các hàm cần implement (API gợi ý)

7.1 Normalize time

```
def normalize_time(time_id: LongTensor, T: int, scale: float = 1.0) -> FloatTensor:
    """Return tau in [0, scale]."""
```

$$\tau = scale \cdot \frac{time_id}{T - 1}$$

7.2 RelationConditionedTimeEncoder

Input: - `rel_id`: shape [B] - `tau`: shape [B]

Output: - `m`: shape [B, d]

Steps: 1. lookup `a_r`, `A_r`, `P_r` 2. compute `z0 = a_r * tau` 3. compute `zk = A_r * sin(omega * tau + P_r)` 4. concat `z = [z0, zk]` 5. project: `m = tanh(z @ W_proj.T + b_proj)`

7.3 Residual gate

```
def residual_gate(m: FloatTensor, beta: float = 0.2) -> FloatTensor:
    return 1.0 + beta * torch.tanh(m)
```

7.4 Forward scoring (single triple)

```
1. tau = normalize_time(time_id)
2. m = time_encoder(rel_id, tau)
3. g = residual_gate(m)
4. x = h * rH - t * rT
5. x_t = x * g
6. score = -L1_norm(x_t)
```

8. Optional regularizers (khuyến nghị)

8.1 Time smoothness

$$\mathcal{L}_{smooth} = \|m_r(\tau_{l+1}) - m_r(\tau_l)\|_2^2$$

8.2 Frequency bound

$$\mathcal{L}_\omega = \sum_k \max(0, |\omega_k| - \omega_{max})^2$$

9. Checklist tránh lặp lại lỗi cũ

Sau khi code xong, chạy diagnostics: - **Test A:** `m_r(t)` có cấu trúc (ít noise hơn) - **Test B:** score delta theo time tăng rõ - **Test D:** gradient time vẫn tồn tại

10. Thông số mặc định để bắt đầu

- `d = 128`
 - `K = 16`
 - `beta = 0.2`
 - `tau scale = 1.0`
 - `omega` : fixed, log-spaced
 - Optimizer / LR giữ như PairRE baseline
-

Lưu ý: Đây là tài liệu *research-grade specification*, không phải README cho end-user. Dùng trực tiếp để giao cho AI code lại module time + scoring.