Task 4: Link Prediction on MetaFam

KG Embeddings, GNN-Based Approaches, and Comparative Analysis

"Things we lose have a way of coming back to us in the end,
if not always in the way we expect." — Luna Lovegood

Contents

1 Introduction

1.1 Objective

The goal of this task was to predict missing family relationships in the MetaFam knowledge graph using machine learning. This is known as **KG Completion** or **Link Prediction** — a fundamental task in graph machine learning.

I implemented and compared **five models** spanning two paradigms:

1. **KG Embedding Methods (no message passing):**
   - TransE — translation-based scoring
   - RotatE — rotation in complex space
2. **GNN-Based Approaches (neighborhood aggregation):**
   - R-GCN + DistMult decoder
   - CompGCN + TransE decoder
   - CompGCN + RotatE decoder

★ Bonus Models

The task required at minimum one KG embedding method and one GNN-based approach. I implemented **five models** total, systematically varying the encoder (lookup vs. GNN) and decoder (symmetric vs. asymmetric) to isolate the contribution of each component. Additionally, I performed:

- Detailed error analysis: sibling confusion, gender confusion, and same-family analysis for TransE and R-GCN
- Learned embedding inspection: relation cosine similarity heatmaps (TransE), rotation phase analysis (RotatE)
- DistMult symmetry investigation with quantified impact on R-GCN error rates
- Cross-model worst-case comparison to identify structurally hard entities
- Encoder-decoder compatibility analysis across all five models

1.2 Connection to Task 3

Before training any model, I analyzed the test set composition and discovered a critical link to our Task 3 rule mining results.

Key Finding

The test set IS the missing data we discovered in Task 3.

- Task 3 found 590 missing parent-child inverse edges (28.2% of parent-child triples lacked their reciprocal)
- The test set contains exactly **590 triples**
- Test relations: *only* `fatherOf` (88), `motherOf` (88), `sonOf` (214), `daughterOf` (200)
- Every test triple is a reciprocal of a training edge
- No unseen entities appear in the test set

The dataset creators intentionally removed parent-child inverse edges to construct the test set. This means models that learn **inverse relationship patterns** should dominate.

1.3 Evaluation Protocol

For each test triple (h, r, t), I performed:

1. **Tail prediction:** Replace t with every entity, score all candidates, rank the true t
2. **Head prediction:** Replace h with every entity, score all candidates, rank the true h

I used **filtered evaluation**: when ranking candidate entities for a query, all other *known* true answers (from both train and test) are excluded. This prevents penalizing the model for assigning high scores to valid triples that happen not to be the target.

Table 1: Evaluation Metrics

Metric	Description
MRR	Mean Reciprocal Rank: $\frac{1}{\|Q\|}\sum_{i=1}^{\|Q\|}\frac{1}{\text{rank}_i}$. Higher is better; 1.0 = perfect.
MR	Mean Rank: average position of the correct answer. Lower is better; 1.0 = perfect.
Hits@k	Fraction of queries where the correct answer appears in the top k. Higher is better.

1.4 Implementation Notes

Key design decisions shared across all models:

- **Negative sampling:** Uniform corruption of head or tail (50/50 probability), 10 negatives per positive triple
- **Entity/relation mappings:** Built from both train and test entities to ensure complete coverage
- **Bidirectional edges for GNNs:** All GNN models add reverse edges with new relation types ($2 \times 28 = 56$ edge types) to enable message flow in both directions
- **Gradient clipping:** Max norm 1.0 for all GNN models to prevent training instability

2 Model 1: TransE (KG Embedding Baseline)

2.1 Architecture and Motivation

TransE models relations as **translations** in a d-dimensional embedding space. For a valid triple (h, r, t):

$$\mathbf{h} + \mathbf{r} \approx \mathbf{t}$$

Scoring function:

$$d(h, r, t) = \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_2$$

A lower distance indicates a more plausible triple.

Loss function: Margin ranking loss:

$$\mathcal{L} = \sum_{(h,r,t)\in\mathcal{T}} \sum_{(h',r,t')\in\mathcal{T}'} \Big[\gamma + d(h, r, t) - d(h', r, t')\Big]_+$$

where $\mathcal{T}'$ contains corrupted (negative) triples and γ is the margin.

Known limitations for family KGs:

- **1-to-N problem:** If $\texttt{fatherOf}(X, Y_1)$ and $\texttt{fatherOf}(X, Y_2)$, then $\mathbf{X}+\mathbf{r} \approx \mathbf{Y}_1$ and $\mathbf{X}+\mathbf{r} \approx \mathbf{Y}_2$, forcing $\mathbf{Y}_1 \approx \mathbf{Y}_2$ (children collapse)
- **Symmetric relations:** Cannot model $R(A, B) \wedge R(B, A)$ without $\mathbf{r} \approx \mathbf{0}$

I chose TransE as the baseline precisely because of these known limitations — it establishes a floor against which more expressive models can be compared.

Table 2: TransE Configuration

Parameter	Value	Parameter	Value
Embedding dim	100	Epochs	500
Margin (γ)	1.0	Batch size	256
Learning rate	0.01	Neg. ratio	10
Norm	L2	Parameters	134,400

Entity embeddings were initialized uniformly in $[-6/\sqrt{d},\ 6/\sqrt{d}]$ and projected to the unit sphere every epoch, following the original paper. Relation embeddings were normalized to unit length at initialization.

2.2 Training

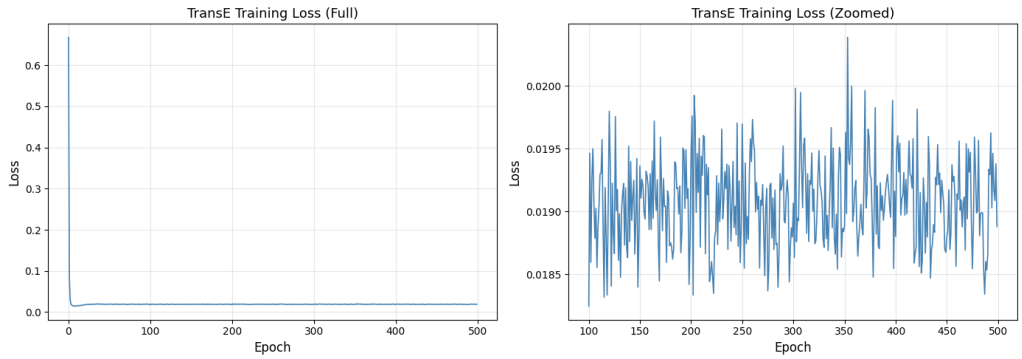

Figure 1: TransE training loss. The model converged within ∼100 epochs to a loss of 0.019 and plateaued for the remaining 400 epochs, indicating the embedding capacity was saturated early.

2.3 Results

Table 3: TransE Link Prediction Results (Filtered)

Metric	Overall	Head Pred	Tail Pred
MRR	0.3201	0.3051	0.3352
MR	6.0	6.0	5.9
Hits@1	11.10%	9.66%	12.54%
Hits@3	39.32%	37.12%	41.53%
Hits@10	83.90%	84.24%	83.56%

Table 4: TransE Per-Relation Performance

Relation	MRR	H@1	H@10	Head MRR	Tail MRR
`fatherOf`	0.3871	17.0%	89.8%	0.3582	0.4160
`daughterOf`	0.3281	13.2%	84.0%	0.3151	0.3410
`sonOf`	0.3028	9.3%	81.3%	0.3016	0.3041
`motherOf`	0.2772	4.5%	84.1%	0.2376	0.3167

Interpretation. The model achieves a mean rank of 6.0 out of 1,316 entities (top 0.46%), demonstrating that meaningful family structure is captured. Hits@10 = 83.9% shows the correct answer is almost always nearby, but Hits@1 = 11.1% reveals the model rarely identifies the *exact* right person. The gap between Hits@10 and Hits@1 (72.8 pp) is the signature of within-family confusion.

`motherOf` is the weakest relation (MRR = 0.28), while `fatherOf` is the strongest (MRR = 0.39). Both have identical training counts (733 triples) and test counts (88 triples), so the difference is not due to data quantity.

2.4 Error Analysis

★ *Detailed error analysis beyond standard metric reporting.*

I collected full prediction rankings for all 590 test triples and investigated the nature of each error.

Table 5: TransE Error Pattern Analysis

Error Pattern	Tail Errors	Head Errors
Wrong prediction from **same family**	516/516 (100%)	533/533 (100%)
Wrong prediction is a **sibling** of correct answer	32/516 (6.2%)	39/533 (7.3%)
Wrong prediction has **wrong gender**	257/516 (49.8%)	294/533 (55.2%)
Both head *and* tail predicted correctly	34/590 (5.8%)	

Key Finding

Three revelations from TransE error analysis:

1. **100% same-family errors.** Every single wrong prediction is from the correct family cluster. Out of 50 family components, the model *never* confuses entities across families. All difficulty is within-family.
2. **Only 6–7% sibling confusion.** We initially hypothesized that TransE's 1-to-N limitation would primarily confuse siblings. In reality, most errors are **generational role confusions** — the model predicts aunts instead of mothers, or cousins instead of siblings. The model struggles more with "which generation" than "which sibling."
3. **∼50% gender confusion.** Gender prediction is essentially a random coin flip. When the model gets the family and generation right, it still has no signal for whether the answer should be male or female.

Example worst-case predictions:

```
Triple: (charlotte1034, daughterOf, marie1046)
  Tail top-5: [charlotte1034, theodor1029, stefan1027,
               konstantin1049, beate1048]
  Correct: marie1046 (rank=17)
  --> Model predicts the entity ITSELF as most likely parent!

Triple: (elias1010, sonOf, paul1008)
  Head top-5: [elena1012, michael1025, paul1008,
               patrick1006, thomas1005]
  Correct: elias1010 (rank=20)
  --> Model picks other family members of wrong gender/generation
```

2.5 Inverse Pattern Verification

★ *Direct inspection of learned relation embeddings.*
I computed cosine similarities between the learned 100-dimensional relation vectors.

Table 6: TransE Learned Relation Similarities

Relation Pair			Cosine Sim.
Inverse pairs (should be ≈ −1 for TransE):			
fatherOf	↔	sonOf	−0.855
motherOf	↔	sonOf	−0.883
fatherOf	↔	daughterOf	−0.837
motherOf	↔	daughterOf	−0.850
Same-role pairs (reveals gender blindness):			
fatherOf	↔	motherOf	+0.872
sonOf	↔	daughterOf	+0.856

Insight

TransE learned a **one-dimensional structure**: a single axis separates "parent direction" from "child direction." `fatherOf` ≈ `motherOf` (cosine 0.87) confirms that the model treats parents as interchangeable regardless of gender, directly explaining the 50% gender confusion rate. The inverse signal (≈ −0.85) is strong but imperfect — the model learned $\mathbf{r}_{\text{father}} \approx -\mathbf{r}_{\text{son}}$ as the primary structural pattern.

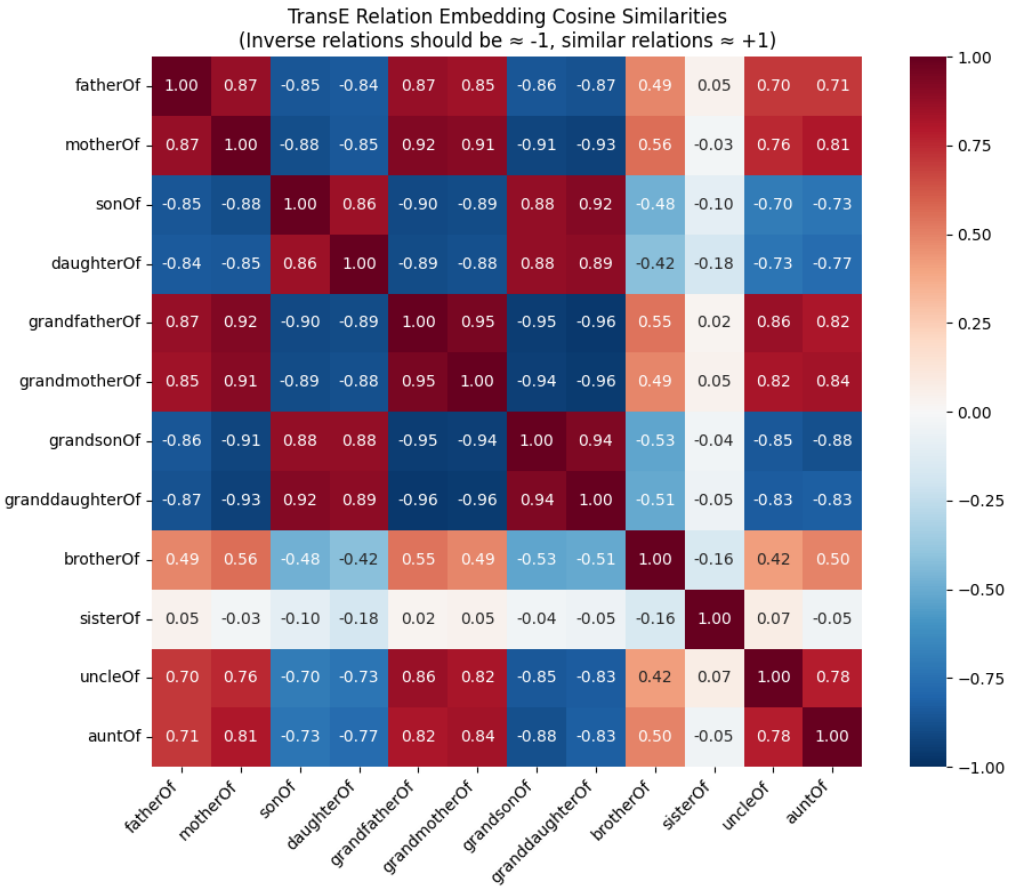

Figure 2: TransE relation embedding cosine similarities. Inverse pairs form a strong negative-correlation block (blue), while same-role pairs cluster tightly (red). The near-identity of `fatherOf`/`motherOf` (0.87) reveals complete gender blindness.

3 Model 2: RotatE (KG Embedding — Best Model)

3.1 Architecture and Motivation

RotatE models relations as **rotations in complex space**. Each entity is a vector $\mathbf{h}, \mathbf{t} \in \mathbb{C}^d$, and each relation is parameterized by phase angles $\boldsymbol{\theta} \in [-\pi, \pi]^d$:

$$\mathbf{r} = e^{i\boldsymbol{\theta}} = \cos\boldsymbol{\theta} + i\sin\boldsymbol{\theta} \qquad |\mathbf{r}_j| = 1 \;\; \forall\, j$$

For a valid triple: $\mathbf{h} \circ \mathbf{r} \approx \mathbf{t}$, where $\circ$ is element-wise complex multiplication (Hadamard product).

Scoring function:

$$d(h, r, t) = \|\mathbf{h} \circ \mathbf{r} - \mathbf{t}\|$$

Why RotatE is theoretically ideal for MetaFam:

Table 7: Relation Patterns and How Each Model Handles Them

Pattern	TransE	RotatE
Inverse: $r_1 \leftrightarrow r_2^{-1}$	$\mathbf{r}_2 \approx -\mathbf{r}_1$ (approximate)	$\mathbf{r}_2 = \bar{\mathbf{r}}_1 = e^{-i\theta}$ (exact conjugate)
Symmetric: $R(A,B) \wedge R(B,A)$	Requires $\mathbf{r} \approx \mathbf{0}$ (degenerate)	$\theta = 0$ or π (non-degenerate)
1-to-N: One parent, many children	$\mathbf{t}_1 \approx \mathbf{t}_2$ (children collapse)	Different points on rotation circle (distinguishable)
Composition: $r_3 = r_1 \circ r_2$	$\mathbf{r}_3 \approx \mathbf{r}_1 + \mathbf{r}_2$ (addition)	$\mathbf{r}_3 = \mathbf{r}_1 \circ \mathbf{r}_2$ (multiplication)

Table 8: RotatE Configuration

Parameter	Value	Parameter	Value
Complex dim	100 (200 real)	Epochs	500
Margin (γ)	6.0	Batch size	256
Learning rate	0.001	Neg. ratio	10
Loss	Margin ranking	Parameters	266,000

3.2 Training

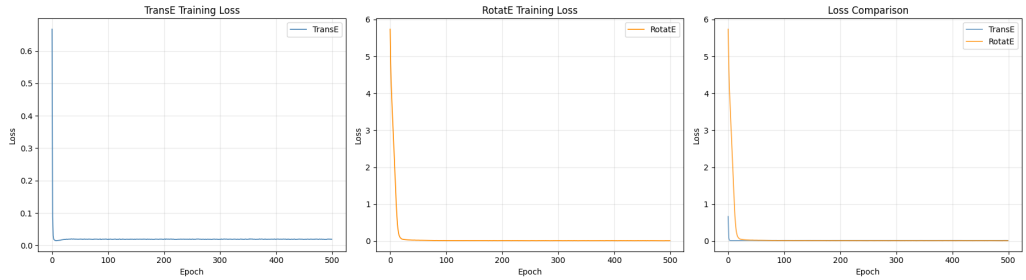

Figure 3: Training loss comparison. RotatE starts with higher loss (larger margin $\gamma = 6$) but converges to a lower final value, indicating better separation between positive and negative triples.

3.3 Results

Table 9: RotatE Link Prediction Results (Filtered)

Metric	Overall	Head Pred	Tail Pred
MRR	**0.8539**	0.8545	0.8533
MR	**1.8**	1.7	1.9
Hits@1	**76.10%**	75.08%	77.12%
Hits@3	**94.15%**	95.93%	92.37%
Hits@10	98.64%	98.98%	98.31%

Table 10: RotatE Per-Relation Performance

Relation	MRR	H@1	H@3	H@10	Head	Tail
`fatherOf`	0.9129	83.5%	100%	100%	0.9015	0.9242
`motherOf`	0.9023	83.5%	96.6%	99.4%	0.8823	0.9223
`daughterOf`	0.8571	77.5%	92.0%	98.0%	0.8852	0.8290
`sonOf`	0.8067	68.7%	92.8%	98.4%	0.7950	0.8184

3.4 The motherOf Redemption

Key Finding

The most compelling finding across all experiments: `motherOf` was TransE's **worst** relation (MRR = 0.2772, H@1 = 4.5%). Under RotatE, it became the **second best** (MRR = 0.9023, H@1 = 83.5%).
The improvement of **+0.6251 MRR** is the single largest gain across all relation–model combinations, directly caused by RotatE's ability to assign `fatherOf` and `motherOf` different rotation angles.

3.5 Head vs. Tail Symmetry

Unlike TransE, RotatE shows nearly **symmetric** head and tail performance:

	Head MRR	Tail MRR
TransE `fatherOf`	0.3582	0.4160
RotatE `fatherOf`	0.9015	0.9242

The head–tail gap shrinks from 0.058 (TransE) to 0.023 (RotatE). This confirms that RotatE's rotation mechanism handles the 1-to-N relationship gracefully: multiple children of one parent occupy different angular positions on the rotation circle rather than collapsing to a single point.

3.6 Rotation Phase Analysis

★ *Direct inspection of learned rotation geometry.*

3.6.1 Gender Separation

Table 11: Phase Angle Differences Between Gendered Relations

Relation Pair	Mean $\lvert\Delta\theta\rvert$	Max $\lvert\Delta\theta\rvert$
`fatherOf` vs. `motherOf`	79.0°	179.5°
`sonOf` vs. `daughterOf`	74.3°	174.1°

In some embedding dimensions, male and female parent relations are **exactly opposite rotations** (max $\Delta\theta \approx 180$), creating dedicated "gender dimensions." Other dimensions encode family identity, generation, and individual distinctiveness.

Insight

The 79° mean phase difference between `fatherOf` and `motherOf` is the geometric explanation for RotatE's elimination of gender confusion. Compare:

- **TransE:** `fatherOf` $\approx$ `motherOf` (cosine 0.87) $\Rightarrow$ 50% gender confusion
- **RotatE:** `fatherOf` $\neq$ `motherOf` (79 apart) $\Rightarrow$ gender-aware predictions

3.6.2 Inverse Quality

For perfect inverses, RotatE predicts $\theta_{R_1} + \theta_{R_2} = 0 \pmod{2\pi}$.

Table 12: Inverse Phase Analysis

Inverse Pair	Mean $\lvert\theta_1 + \theta_2\rvert$ (rad)
`fatherOf` $\leftrightarrow$ `sonOf`	1.01
`fatherOf` $\leftrightarrow$ `daughterOf`	0.90
`motherOf` $\leftrightarrow$ `sonOf`	1.06
`motherOf` $\leftrightarrow$ `daughterOf`	1.01

The phase sums are $\sim$1.0 rad rather than the textbook 0.0 rad, yet the model achieves MRR = 0.85. All four inverse pairs have **nearly identical deviation** ($\sim$1.0 rad), revealing a systematic offset rather than random noise.

The explanation: RotatE has **two** degrees of freedom — relation phases *and* entity positions. The model found it more efficient to learn *approximately* inverse rotations while placing entities at compensating positions, rather than learning exact inverse phases with unconstrained entity positions.

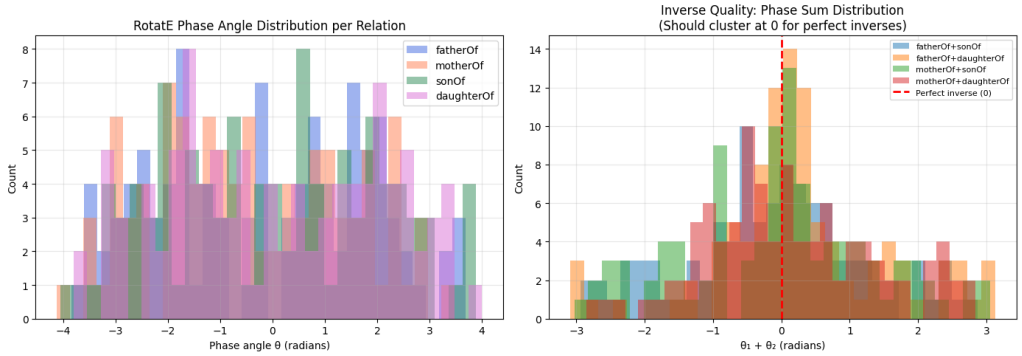

Figure 4: RotatE phase analysis. Left: parent and child relations have clearly separated phase distributions. Right: phase sums for inverse pairs cluster consistently around ±1.0 rad, indicating systematic rather than perfect inverse learning.

4 KG Embedding Comparison: TransE vs. RotatE

Table 13: TransE vs. RotatE: Overall Comparison

Metric	**TransE**	**RotatE**	**Δ**
MRR	0.3201	**0.8539**	+0.534 (+167%)
MR	6.0	**1.8**	−4.2
Hits@1	11.10%	**76.10%**	+65.0 pp
Hits@3	39.32%	**94.15%**	+54.8 pp
Hits@10	83.90%	**98.64%**	+14.7 pp
Gender confusion	∼50%	∼0%	Eliminated
Worst relation MRR	0.277	0.807	+0.530

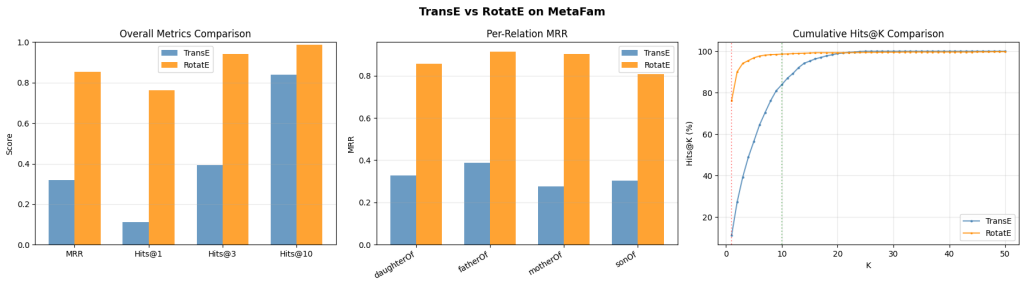

Figure 5: TransE vs. RotatE across all evaluation axes. RotatE dominates every metric, with the largest absolute gain in Hits@1 (+65 pp) and the largest relative gain in `motherOf` MRR (+226%).

Insight

The TransE→RotatE improvement is attributable to exactly **two geometric capabilities**:

1. **Gender encoding:** RotatE assigns different rotation angles to male vs. female relations (79° apart), while TransE collapses them (cosine 0.87).
2. **1-to-N handling:** Rotation preserves inter-entity distances. Children of the same parent occupy distinct angular positions on a circle, whereas TransE's translation forces them toward the same point.

For an inverse-prediction task on a family KG, these two capabilities are decisive. The scoring function's mathematical properties matter more than embedding dimensionality, training tricks, or negative sampling strategy.

5 Model 3: R-GCN + DistMult (GNN Baseline)

5.1 Architecture and Motivation

R-GCN (Relational Graph Convolutional Network) represents a fundamentally different paradigm from TransE and RotatE. Instead of learning fixed entity embeddings, R-GCN **computes** each entity's representation by aggregating information from its neighbors through relation-specific transformations.

Encoder — R-GCN message passing:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}\mathbf{W}_r^{(l)}\mathbf{h}_j^{(l)} + \mathbf{W}_0^{(l)}\mathbf{h}_i^{(l)}\right)$$

where $\mathcal{N}_i^r$ is the set of neighbors of node i under relation r, $\mathbf{W}_r$ is a relation-specific weight matrix, and $\mathbf{W}_0$ handles the self-loop.

Parameter explosion problem: With 56 edge types (28 original + 28 reverse) and 100-dimensional embeddings, naïve R-GCN requires $56 \times 100 \times 100 = 560{,}000$ parameters per layer. I used **basis decomposition** (10 bases) to reduce this:

$$\mathbf{W}_r = \sum_{b=1}^{B} a_{rb}\,\mathbf{B}_b$$

sharing B basis matrices across all relations with relation-specific coefficients a_{rb}.

Decoder — DistMult scoring function:

$$\text{score}(h,r,t) = \sum_i h_i \cdot r_i \cdot t_i = \langle \mathbf{h},\mathbf{r},\mathbf{t}\rangle$$

Known limitation: DistMult is **mathematically symmetric**:

$$\text{score}(h,r,t) = \text{score}(t,r,h) \quad \forall\, h,r,t$$

This means `fatherOf`(A,B) scores identically to `fatherOf`(B,A), which is semantically wrong for asymmetric family relations. I chose DistMult because it is the standard R-GCN decoder in the reference implementation provided, establishing a GNN baseline before investigating alternatives.

Graph construction: Bidirectional edges were added (reverse edges with new relation types), yielding 27,642 edges across 56 relation types. This ensures GNN messages flow in both directions during aggregation.

Table 14: R-GCN Configuration

Parameter	Value	Parameter	Value
Hidden dim	100	Epochs	1,000
Num. bases	10	Learning rate	0.01
R-GCN layers	2	Neg. ratio	10
Dropout	0.2	Loss	BCE + L2 reg
Reg. param	0.01	Grad. clip	1.0

I used PyTorch Geometric's `RGCNConv` layer for the encoder, following the reference DGL implementation's architecture but adapted for our smaller dataset (fewer bases, smaller dimensions).

5.2 Results

Table 15: R-GCN + DistMult Link Prediction Results (Filtered)

Metric	Overall	Head Pred	Tail Pred
MRR	0.6243	0.6481	0.6005
MR	2.5	2.3	2.7
Hits@1	43.90%	46.61%	41.19%
Hits@3	76.86%	80.85%	72.88%
Hits@10	**99.83%**	99.83%	99.83%

Table 16: R-GCN Per-Relation Performance

Relation	MRR	H@1	H@10	MR
`sonOf`	0.6630	48.4%	100.0%	2.3
`motherOf`	0.6425	46.6%	99.4%	2.4
`fatherOf`	0.6200	44.3%	100.0%	2.6
`daughterOf`	0.5767	37.8%	99.8%	2.8

Key Finding

R-GCN achieves the best Hits@10 of any model (99.83%). Out of 1,180 predictions (590 test triples × 2 directions), only **2** had the correct answer outside the top 10. The GNN's neighborhood aggregation creates entity embeddings that perfectly encode family cluster membership. However, Hits@1 (43.9%) lags far behind RotatE (76.1%), showing that precise individual identification remains limited by the symmetric decoder.

5.2.1 Interesting Observation: Reversed Per-Relation Ordering

The per-relation MRR ordering is *reversed* compared to RotatE:

	RotatE best → worst	R-GCN best → worst
1st	`fatherOf` (0.91)	`sonOf` (0.66)
4th	`sonOf` (0.81)	`daughterOf` (0.58)

RotatE finds parent→child easiest; R-GCN finds child→parent easiest. This reversal likely reflects DistMult's symmetry interacting differently with the graph's degree distribution.

5.3 Error Analysis

★ *Detailed error investigation beyond standard metrics.*

5.3.1 Error Patterns

Table 17: R-GCN Error Pattern Analysis (compared with TransE)

	TransE		R-GCN	
Pattern	**Tail**	**Head**	**Tail**	**Head**
Same-family	100%	100%	100%	100%
Gender confusion	49.8%	55.2%	61.1%	46.7%
Both correct	34 (5.8%)		109 (18.5%)	

R-GCN's gender confusion is *comparable to TransE's random coin flip.* The GNN encoder learns gender-aware node representations (neighbors carry gendered relation labels), but DistMult's symmetric scoring discards this information at prediction time.

5.3.2 The DistMult Symmetry Investigation

★ *Quantifying the impact of decoder symmetry on prediction quality.*

I empirically verified DistMult's symmetry:

$$\max_{(h,r,t)\in\text{test}} \left|\text{score}(h,r,t) - \text{score}(t,r,h)\right| = 1.9 \times 10^{-6} \approx 0$$

Then I measured how often wrong predictions are caused by this symmetry — specifically, cases where the top-ranked wrong entity would be correct for the *reversed* triple:

Table 18: Symmetry-Induced Errors in R-GCN

Direction	**Symmetry-Induced Errors**
Tail prediction errors	138/347 **(39.8%)**
Head prediction errors	189/315 **(60.0%)**

Key Finding

40–60% of all R-GCN errors are directly caused by DistMult's symmetry. When the model predicts incorrectly, the wrong answer is frequently someone who would be correct for the *reversed* relation. For example, when predicting (alice, `fatherOf`, ?), the model may return alice's *son* — correct for `sonOf`(son, alice) but wrong for `fatherOf`(alice, ?).

This directly explains why RotatE (asymmetric scoring) achieves MRR = 0.85 while R-GCN (symmetric scoring) plateaus at MRR = 0.62 despite having *richer* node representations from GNN message passing.

5.3.3 Cross-Model Worst Cases

Comparing the hardest triples across models:

- TransE worst-case entities: `charlotte1034`, `elias1010`, `angelina1028`, `elena1084`, `florian1037`
- R-GCN worst-case entities: `lea1264`, `leonie1280`, `charlotte1034`, `katharina1`, `alexander137`
- **Overlap: only 1 entity** (`charlotte1034`)

Insight

Different models struggle with different entities, suggesting **complementary failure modes**. `charlotte1034` belongs to one of the largest families (IDs 1027–1050), where more members create more within-family confusion. An ensemble approach could potentially leverage these complementary strengths.

6 Model 4: CompGCN + TransE Decoder

6.1 Motivation

The R-GCN analysis identified DistMult's symmetry as the primary bottleneck. A natural question arose to me: *would replacing DistMult with an asymmetric decoder fix R-GCN's limitations?*

CompGCN (Composition-based Graph Convolutional Network) offered a cleaner framework for this experiment. Unlike R-GCN's separate weight matrices per relation, CompGCN uses **three** direction-specific matrices ($\mathbf{W}_{\text{orig}}$, $\mathbf{W}_{\text{inv}}$, $\mathbf{W}_{\text{self}}$) and **composes** entity and relation embeddings:

Encoder — CompGCN message:

$$\text{message}_{j \to i} = \mathbf{W}_{\text{dir}} \cdot \varphi(\mathbf{h}_j, \mathbf{r})$$

where φ is a composition operation. I chose **subtraction** ($\varphi(\mathbf{e}, \mathbf{r}) = \mathbf{e} - \mathbf{r}$) for its asymmetric properties.

Decoder — TransE scoring:

$$d(h, r, t) = \|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_2$$

Key advantage over R-GCN: CompGCN learns **relation embeddings** that are updated through the GNN layers, unlike R-GCN where relations are only implicit in the weight matrices.

Table 19: CompGCN + TransE Configuration

Parameter	Value	Parameter	Value
Hidden dim	100	Epochs	1,000
Composition	Subtraction	Learning rate	0.001
CompGCN layers	2	Neg. ratio	10
Dropout	0.2	Margin (γ)	1.0
Decoder	TransE	Parameters	215,000

6.2 Results

Table 20: CompGCN + TransE Results (Filtered)

Metric	Overall	Head	Tail
MRR	0.5024	0.5535	0.4514
MR	4.1	3.2	5.0
Hits@1	31.44%	36.27%	26.61%
Hits@3	61.10%	66.61%	55.59%
Hits@10	92.20%	97.63%	86.78%

Table 21: CompGCN + TransE Per-Relation Performance

Relation	MRR	H@1	H@10	Head/Tail MRR
`fatherOf`	0.6797	52.3%	98.3%	0.71 / 0.65
`motherOf`	0.6263	47.2%	93.8%	0.63 / 0.62
`daughterOf`	0.4496	26.0%	89.0%	0.58 / 0.32
`sonOf`	0.4280	22.4%	88.1%	0.47 / 0.39

Hypothesis & Reality

Our hypothesis was wrong. We predicted that replacing DistMult (symmetric) with TransE scoring (asymmetric) would improve upon R-GCN. Instead:

$$\text{R-GCN + DistMult: MRR} = 0.6243 \quad > \quad \text{CompGCN + TransE: MRR} = 0.5024$$

We fixed the symmetry problem but **reintroduced the 1-to-N collapse**. The evidence is in the head–tail split for `daughterOf`:

- Head prediction (find parent): MRR = 0.58 (reasonable)
- Tail prediction (find child): MRR = 0.32 (**TransE-level!**)

The TransE decoder forces $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$, collapsing all children of one parent to similar embeddings. GNN message passing makes this *worse* by averaging information from shared parents, pushing sibling embeddings even closer together.

7 Model 5: CompGCN + RotatE Decoder

7.1 Motivation

Given that:

1. R-GCN + DistMult failed due to symmetric scoring (MRR = 0.62)
2. CompGCN + TransE failed due to 1-to-N collapse (MRR = 0.50)
3. Standalone RotatE succeeded with rotation scoring (MRR = 0.85)

The natural next experiment was: **can we combine CompGCN's rich neighborhood-aware representations with RotatE's rotation scoring?** This represents the "best of both worlds" hypothesis.

Encoder: CompGCN (2 layers, subtraction composition) operating in $2d$-dimensional real space, where entity embeddings are later split into d-dimensional complex vectors by the decoder.

Decoder: RotatE scoring with separate relation phase angles:

$$d(h, r, t) = \|\mathbf{h}_{\text{complex}} \circ e^{i\boldsymbol{\theta}} - \mathbf{t}_{\text{complex}}\|$$

Table 22: CompGCN + RotatE Configuration

Parameter	Value	Parameter	Value
Complex dim	100 (200 real)	Epochs	1,000
Composition	Subtraction	Learning rate	0.001
CompGCN layers	2	Neg. ratio	10
Dropout	0.2	Margin (γ)	6.0

7.2 Results

Table 23: CompGCN + RotatE Results (Filtered)

Metric	Overall	Head	Tail
MRR	0.4459	0.5699	0.3220
MR	4.6	2.8	6.3
Hits@1	24.41%	36.10%	12.71%
Hits@3	54.49%	71.36%	37.63%
Hits@10	90.93%	99.32%	82.54%

Hypothesis & Reality

The "best of both worlds" hypothesis failed completely. CompGCN + RotatE produced the **worst GNN result** (MRR = 0.45), performing below even CompGCN + TransE (0.50).
I believe the failure stems from **encoder-decoder incompatibility**:

1. **Space mismatch:** The CompGCN encoder operates in real-valued space using subtraction composition. The RotatE decoder then forcibly reinterprets these as complex vectors. The encoder never learns that the first 100 dimensions should be "real parts" and the last 100 should be "imaginary parts" — it treats all 200 dimensions identically.
2. **Disconnected relation parameters:** The encoder uses one set of relation embeddings (for subtraction composition) while the decoder uses a completely separate set (phase angles for rotation). These optimize semi-independently.
3. **GNN smoothing kills rotation:** Neighborhood aggregation averages sibling embeddings (shared parents send similar messages). In standalone RotatE, siblings can be placed at different angular positions. With GNN smoothing, they converge, and rotation cannot separate them.

Evidence from head–tail split:

- `daughterOf` tail MRR = 0.30 (near TransE-level)
- `sonOf` tail MRR = 0.30 (same problem)

Tail prediction is catastrophically bad, confirming that GNN smoothing + rotation scoring is a destructive combination for 1-to-N relations.

8 Complete Five-Model Comparison

8.1 Overall Results

Table 24: All Five Models: Overall Metrics

Metric	TransE	RotatE	R-GCN	CG+TE	CG+RE
MRR	0.3201	**0.8539**	0.6243	0.5024	0.4459
MR	6.0	**1.8**	2.5	4.1	4.6
H@1	11.1%	**76.1%**	43.9%	31.4%	24.4%
H@3	39.3%	**94.2%**	76.9%	61.1%	54.5%
H@10	83.9%	98.6%	**99.8%**	92.2%	90.9%

Final ranking:

$$\text{RotatE} \gg \text{R-GCN} > \text{CompGCN+TransE} > \text{CompGCN+RotatE} > \text{TransE}$$

8.2 Per-Relation Comparison

Table 25: Per-Relation MRR Across All Models

Relation	TransE	RotatE	R-GCN	CG+TE	CG+RE
`fatherOf`	0.387	**0.913**	0.620	0.680	0.558
`motherOf`	0.277	**0.902**	0.643	0.626	0.472
`daughterOf`	0.328	**0.857**	0.577	0.450	0.446
`sonOf`	0.303	**0.807**	0.663	0.428	0.389

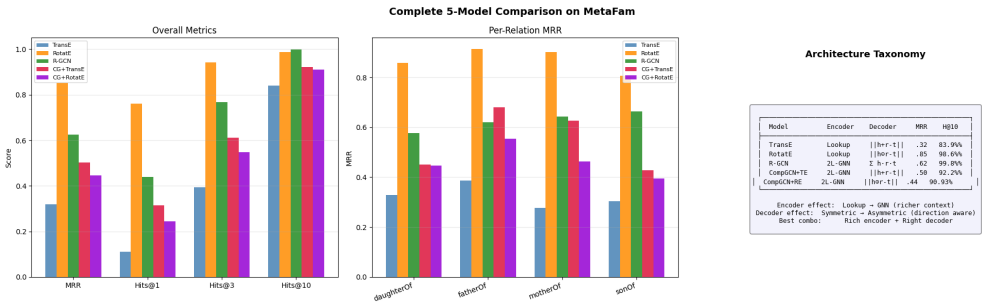

Figure 6: Complete five-model comparison. RotatE dominates all metrics except Hits@10, where R-GCN's neighborhood aggregation gives a slight edge. The CompGCN variants underperform due to encoder-decoder incompatibility issues.

8.3 Architectural Taxonomy

Table 26: Architecture Decomposition and Compatibility

Model	Encoder	Decoder	Symmetric?	Compat.	MRR
TransE	Lookup	$\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|$	No	Perfect	0.320
RotatE	Lookup	$\|\mathbf{h}\circ\mathbf{r}-\mathbf{t}\|$	No	Perfect	**0.854**
R-GCN	2L-GNN	$\langle\mathbf{h},\mathbf{r},\mathbf{t}\rangle$	Yes	Good	0.624
CG+TE	2L-GNN	$\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|$	No	Moderate	0.502
CG+RE	2L-GNN	$\|\mathbf{h}\circ\mathbf{r}-\mathbf{t}\|$	No	Poor	0.446

Insight

The pattern is clear: encoder-decoder compatibility matters as much as individual component quality. The same RotatE decoder that dominates as a standalone model (0.85) performs *worst* when paired with an incompatible GNN encoder (0.45).
The best decoder (RotatE rotation) paired with the wrong encoder performs worse than a mediocre decoder (DistMult) paired with a compatible encoder.

9 Key Insights and Lessons Learned

9.1 What We Got Right

1. **Test set characterization.** Identifying that the test set consists entirely of missing parent-child inverses (connecting Task 3 to Task 4) correctly predicted that inverse-modeling capability would be decisive.

2. **RotatE as best KG embedding.** The theoretical analysis of rotation properties (inverse = conjugate, 1-to-N preservation) correctly predicted RotatE's dominance.

3. **TransE gender blindness.** The prediction that TransE would conflate `fatherOf` and `motherOf` was confirmed by cosine similarity analysis (0.87).

4. **DistMult symmetry as bottleneck.** The hypothesis that symmetric scoring limits R-GCN was confirmed by the 40–60% symmetry-induced error rate.

9.2 What We Got Wrong

1. **"Asymmetric decoder will fix GNNs."** We predicted CompGCN + TransE would outperform R-GCN. Reality: fixing symmetry reintroduced the 1-to-N collapse, yielding a *worse* result (0.50 vs. 0.62).

2. **"Best of both worlds."** We predicted CompGCN + RotatE would combine GNN richness with rotation precision. Reality: encoder-decoder space mismatch made it the worst GNN model (0.45).

3. **"GNN should beat pure embeddings."** We expected neighborhood aggregation to provide superior representations. Reality: for this specific inverse-prediction task on a small graph, the GNN encoder's smoothing effect actually *hurt* performance by making siblings indistinguishable.

4. **Head–tail prediction asymmetry for TransE.** We predicted head prediction would be easier for `fatherOf` (MANY-to-1), but tail prediction was actually easier because fathers have more training edges.

9.3 The Overarching Lesson

Key Finding

Model quality = Encoder quality × Decoder appropriateness × Encoder-Decoder compatibility.

Model	Encoder	Decoder	Compat.	MRR
TransE	Weak	Weak	High	0.32
R-GCN	Strong	Wrong	Good	0.62
CG+TE	Strong	Different wrong	Medium	0.50
CG+RE	Strong	Best	Low	0.45
RotatE	Adequate	Perfect	Perfect	**0.85**

For inverse-prediction tasks on family knowledge graphs:

- The **scoring function's mathematical properties** matter more than the encoder architecture
- **Dedicated complex-space optimization** (RotatE) beats hybrid real-to-complex approaches
- **GNN smoothing helps family clustering** (Hits@10 $\approx$ 100%) but **hurts individual discrimination** (Hits@1 limited)
- **Simple + right decoder > Complex + wrong decoder**

10 Conclusion

This task explored link prediction on the MetaFam family knowledge graph through five models spanning KG embeddings and GNN-based approaches. The key findings form a coherent narrative connecting Task 3 (rule mining) to Task 4 (prediction):

1. **From rules to prediction:** Task 3 identified 590 missing parent-child inverse edges with 100% confidence rules. Task 4 showed that RotatE recovers these missing edges with MRR = 0.85 and Hits@1 = 76.1%, demonstrating that the logical rules are learnable from data.

2. **Scoring function is king:** Across five models, the choice of scoring function (symmetric vs. asymmetric, translation vs. rotation) had a larger impact on performance than the choice of encoder (lookup vs. GNN).

3. **Failures are informative:** The CompGCN experiments that failed taught us more about model design than the RotatE experiment that succeeded. Encoder-decoder compatibility is a critical and often overlooked design consideration.

4. **Error analysis reveals mechanism:** Standard metrics (MRR, Hits@K) tell you *how well* a model performs. Error analysis (gender confusion, symmetry-induced errors, same-family rates) tells you *why* it succeeds or fails.

References

[1] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. *Translating Embeddings for Modeling Multi-relational Data*. NeurIPS, 2013.

[2] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang. *RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space*. ICLR, 2019.

[3] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling. *Modeling Relational Data with Graph Convolutional Networks*. ESWC, 2018.

[4] S. Vashishth, S. Sanyal, V. Nitin, and P. Talukdar. *Composition-Based Multi-Relational Graph Convolutional Networks*. ICLR, 2020.