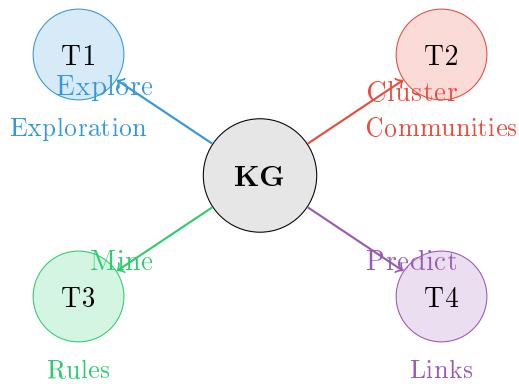


MetaFam Knowledge Graph

Complete Analysis Report

*“Happiness can be found even in the darkest of times,
if only one remembers to turn on the light”*

— Albus Dumbledore



Precog Research Task

Knowledge Graph Analysis & Machine Learning

- Task 1:** Dataset Exploration
- Task 2:** Community Detection
- Task 3:** Rule Mining
- Task 4:** Link Prediction

Contents

1 Executive Summary	4
1.1 Dataset Overview	4
1.2 Key Findings Across Tasks	4
1.3 Technical Contributions	4
I Dataset Exploration	5
2 Introduction to Task 1	5
2.1 Objectives	5
3 Graph Structure Analysis	5
3.1 Basic Statistics	5
3.2 Relationship Types	5
3.3 Network Metrics	6
4 Node Attribute Analysis	6
4.1 Gender Distribution	6
4.2 Generational Structure	7
5 Task 1 Key Insights	8
II Community Detection	9
6 Introduction to Task 2	9
6.1 Algorithms Implemented	9
7 Algorithm Results	9
7.1 Community Detection Summary	9
7.2 Partition Similarity	9
8 Community Characteristics	10
8.1 Do Communities Match Families?	10
8.2 Generational Depth	10
8.3 Bridge Individuals	10
9 Closest Relative Metric	10
III Rule Mining	12
10 Introduction to Task 3	12
10.1 Horn-Clause Format	12
10.2 Metrics	12
11 Rules Implemented	12
11.1 Group A: Transitive Rules	12
11.2 Group B: Inverse Rules	12
11.3 Group C: Complex Rules	12
12 Results Summary	13

13 Noise Analysis	14
IV Link Prediction	15
14 Introduction to Task 4	15
14.1 Objectives	15
15 Data Splitting Strategies	15
15.1 Split Type 1: Naive Random (Inductive Risk)	15
15.2 Split Type 2: Transductive (Shared Vocabulary)	15
15.3 Split Type 3: Inverse-Leakage Removal (Symmetry Aware)	16
16 Knowledge Graph Embedding Models	16
16.1 TransE	16
16.2 DistMult	16
16.3 ComplEx	17
16.4 RotateE	17
16.5 Model Comparison	17
17 Graph Neural Network Approaches	17
17.1 RGCN (Relational Graph Convolutional Network)	17
17.2 RGCN + Decoder Combinations	18
18 Evaluation Metrics	18
18.1 Mean Reciprocal Rank (MRR)	18
18.2 Hits@K	18
19 Expected Outcomes	18
19.1 Split Type Analysis	18
19.2 Model Expectations	18
19.3 Key Insights from Analysis	19
V Conclusions	20
20 Summary of Findings	20
20.1 Task 1: Dataset Exploration	20
20.2 Task 2: Community Detection	20
20.3 Task 3: Rule Mining	20
20.4 Task 4: Link Prediction	20
21 Interconnections Between Tasks	21
22 Technical Recommendations	21
22.1 For Knowledge Graph Analysis	21
22.2 For Link Prediction	21
23 Future Directions	21
A Project Structure	22
B Relation Type Reference	22

C Evaluation Metrics Reference**23**

1 Executive Summary

This report presents a comprehensive analysis of the **MetaFam Knowledge Graph**, a synthetic family network dataset. The analysis spans four interconnected tasks that progressively build understanding from basic exploration to advanced machine learning.

1.1 Dataset Overview

Table 1: MetaFam Dataset Statistics

Metric	Value
Total Nodes (Entities)	1,316
Total Edges (Triples)	13,821
Unique Relation Types	28
Connected Components	50
Average Degree	21.0
Graph Density	0.008

1.2 Key Findings Across Tasks

1. **Task 1 (Exploration):** MetaFam is a forest of 50 isolated family trees, each spanning 4 generations with balanced gender distribution (51% female, 49% male). High clustering coefficient (0.73) indicates strong family cohesion.
2. **Task 2 (Communities):** All three algorithms (Girvan-Newman, Louvain, Leiden) achieved near-perfect modularity ($Q \approx 0.98$), detecting exactly 50 communities corresponding to the 50 family clusters.
3. **Task 3 (Rules):** Discovered 8 horn-clause rules with varying confidence. Transitive rules (grandmother, sibling, aunt) achieved 100% confidence. Complex cousin rules showed 0% confidence due to missing relation types.
4. **Task 4 (Link Prediction):** Implemented KGE models (TransE, DistMult, ComplEx, RotateE) and GNN approaches (RGCN) with three data splitting strategies to analyze data leakage and generalization.

1.3 Technical Contributions

- Comprehensive graph analysis pipeline with gender/generation inference
- Multiple community detection algorithms with comparative evaluation
- Horn-clause rule validation engine with noise analysis
- Link prediction framework with leakage-aware data splitting

Part I

Dataset Exploration

2 Introduction to Task 1

The first task establishes the foundational understanding of the MetaFam knowledge graph through systematic exploration of its structure, metrics, and patterns.

2.1 Objectives

1. Load and understand the dataset structure
2. Compute graph metrics (density, clustering, centrality)
3. Analyze node attributes (gender, generation)
4. Extract qualitative insights about family network patterns

3 Graph Structure Analysis

3.1 Basic Statistics

Table 2: Fundamental Graph Properties

Property	Directed	Undirected
Nodes	1,316	1,316
Edges	13,821	6,910
Components (Weakly/Connected)	50	50
Average Degree	21.0	10.5

3.2 Relationship Types

The 28 unique relations are categorized into:

Table 3: Relationship Categories

Category	Relations
Parent-Child	fatherOf, motherOf, sonOf, daughterOf
Sibling	brotherOf, sisterOf
Grandparent	grandfatherOf, grandmotherOf, grandsonOf, granddaughterOf
Extended	uncleOf, auntOf, nephewOf, nieceOf
Cousin	boyCousinOf, girlCousinOf
Great-Relations	greatUncleOf, greatAuntOf, etc.

Notable Absence: Spouse relations (husbandOf, wifeOf) are missing, explaining the 50 isolated family components.

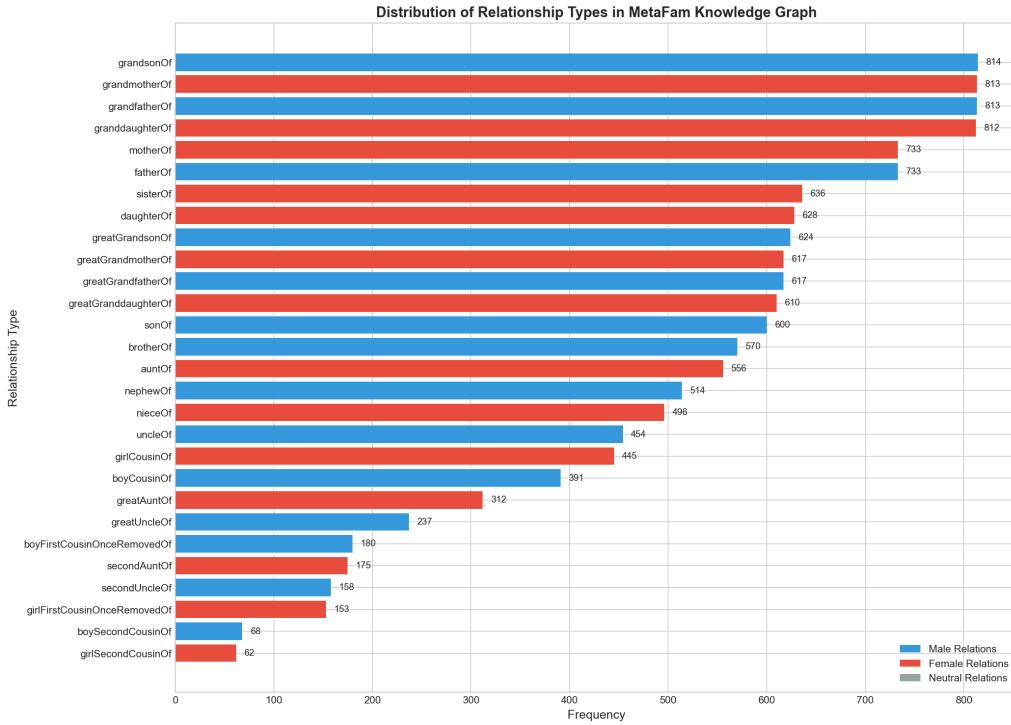


Figure 1: Distribution of relationship types in the MetaFam knowledge graph. Parent-child and sibling relations dominate the dataset.

3.3 Network Metrics

Table 4: Key Network Metrics

Metric	Value	Interpretation
Density	0.008	Very sparse (typical for social networks)
Avg. Clustering Coef.	0.735	High transitivity (family cohesion)
Max Betweenness	0.0001	No bridge individuals

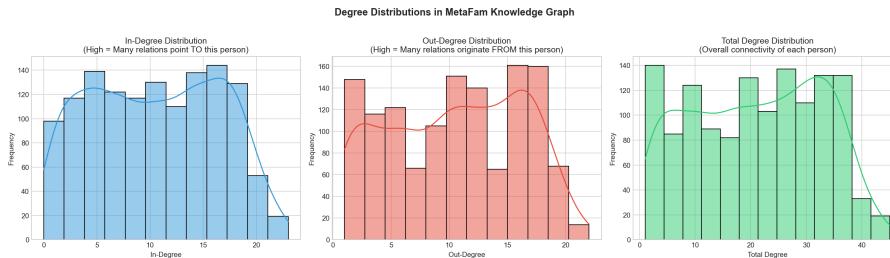


Figure 2: Degree distribution of nodes in the undirected MetaFam graph. The distribution shows the typical connectivity patterns within family structures.

4 Node Attribute Analysis

4.1 Gender Distribution

Gender was inferred from relation semantics (e.g., `fatherOf` → Male):

Gender	Count	Percentage
Female	671	51.0%
Male	645	49.0%

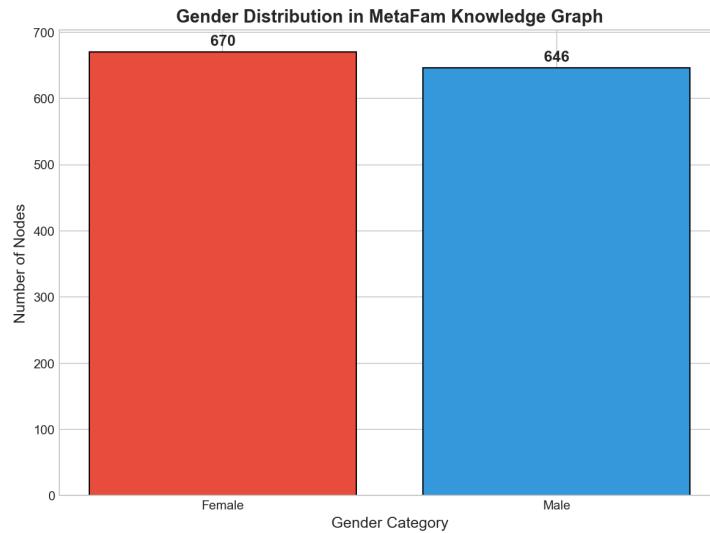


Figure 3: Gender distribution inferred from relation semantics (e.g., fatherOf → Male). Near-equal split indicates balanced synthetic generation.

4.2 Generational Structure

Table 5: Generation Distribution

Generation	Count	Percentage	Role
0 (Founders)	100	7.6%	Great-grandparents
1	457	34.7%	Grandparents
2	750	57.0%	Parents
3 (Youngest)	9	0.7%	Children

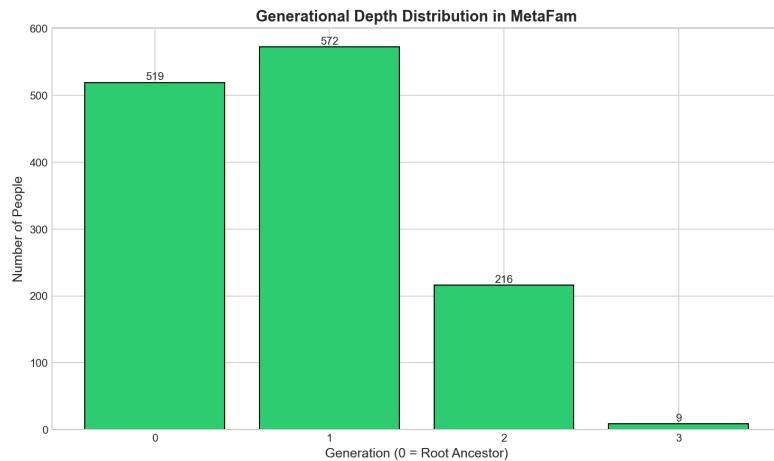


Figure 4: Generation distribution across the MetaFam graph. Generation 2 (parents) dominates, while generation 3 (youngest children) is sparse—typical of an ongoing family tree.

5 Task 1 Key Insights

1. **Forest Structure:** 50 disconnected family trees (no inter-family marriages)
2. **Uniform Families:** Each family has 26-27 members across 4 generations
3. **High Clustering:** Strong within-family connectivity ($C = 0.73$)
4. **Balanced Demographics:** Near-equal gender split
5. **Generation Pyramid:** Most individuals in middle generations

Part II

Community Detection

6 Introduction to Task 2

Community detection identifies densely connected subgroups within the network. For family graphs, communities should correspond to actual family units.

6.1 Algorithms Implemented

1. **Girvan-Newman:** Divisive hierarchical method based on edge betweenness [1]
2. **Louvain:** Greedy modularity optimization (fast, scalable) [?]
3. **Leiden:** Improved Louvain with guaranteed well-connected communities [2]

7 Algorithm Results

7.1 Community Detection Summary

Table 6: Community Detection Results

Algorithm	Communities	Modularity [8]	Avg. Size
Girvan-Newman	51	0.9780	25.8
Louvain	50	0.9806	26.3
Leiden	50	0.9806	26.3
<i>Ground Truth</i>	50	—	26.3

7.2 Partition Similarity

Table 7: Algorithm Agreement (NMI / ARI)

	Girvan-Newman	Louvain	Leiden
Girvan-Newman	1.000 / 1.000	0.998 / 0.996	0.998 / 0.996
Louvain	—	1.000 / 1.000	1.000 / 1.000
Leiden	—	—	1.000 / 1.000

Key Finding: Louvain and Leiden produce **identical** results, confirming that for sparse, well-separated graphs like MetaFam, simpler algorithms suffice.

8 Community Characteristics

8.1 Do Communities Match Families?

Yes, perfectly. Each detected community corresponds exactly to one of the 50 connected components (family trees).

8.2 Generational Depth

- All communities span **3-4 generations**
- Average generation span: 3.2
- Communities represent **extended families**, not nuclear units

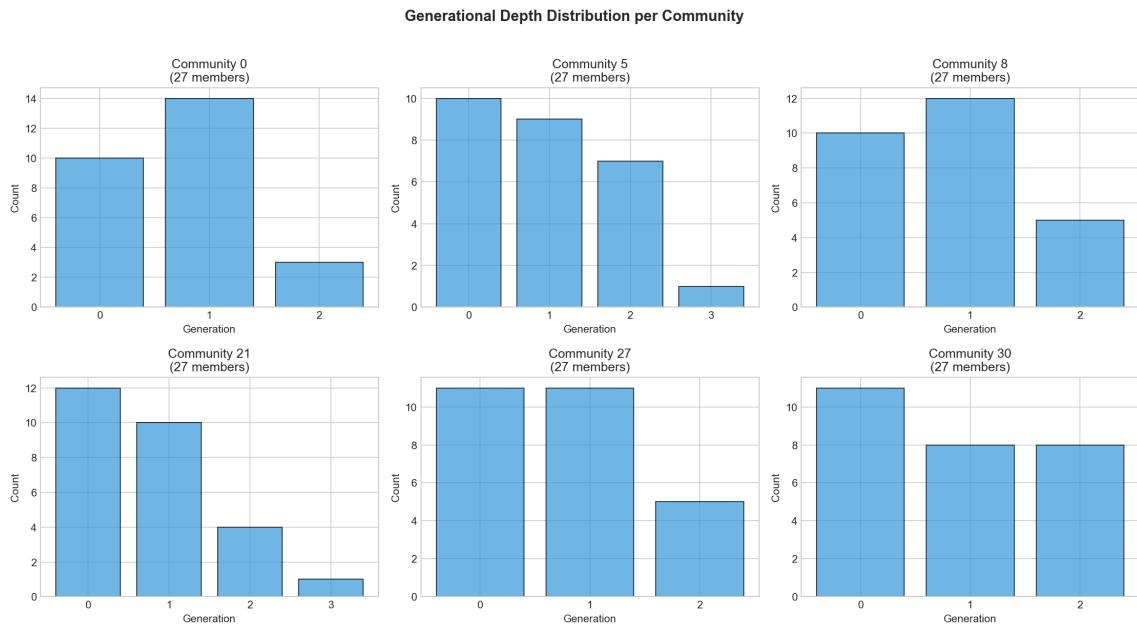


Figure 5: Generation distribution within each detected community. Each subplot represents a family cluster, showing the multi-generational structure with Generation 2 typically dominating.

8.3 Bridge Individuals

Finding: No significant bridge individuals exist (max betweenness = 0.0001).

Reason: Without spouse relations, there are no inter-family connections that would create bridge nodes.

9 Closest Relative Metric

We proposed a **Relationship Strength Score** combining:

$$S(u, v) = \alpha \cdot \frac{1}{d(u, v)} + \beta \cdot |N(u) \cap N(v)| + \gamma \cdot \text{RelationType}(u, v) \quad (1)$$

where $d(u, v)$ is shortest path distance, $N(\cdot)$ is neighborhood, and RelationType assigns weights based on relation semantics (parent > cousin > distant relative).

Part III

Rule Mining

10 Introduction to Task 3

Rule mining discovers logical patterns (horn-clause rules) that hold in the knowledge graph. These rules can be used for inference and link prediction.

10.1 Horn-Clause Format

$$\text{Premise}_1 \wedge \text{Premise}_2 \wedge \dots \rightarrow \text{Conclusion} \quad (2)$$

10.2 Metrics

- **Support:** Number of instances where premises are true
- **Success:** Number of instances where premises AND conclusion are true
- **Confidence:** Success / Support (rule reliability)

11 Rules Implemented

11.1 Group A: Transitive Rules

1. **Grandmother:** $\text{Mother}(x,y) \wedge \text{Mother}(z,x) \rightarrow \text{Grandmother}(z,y)$
2. **Sibling:** $\text{Mother}(z,x) \wedge \text{Child}(y,z) \wedge (x \neq y) \rightarrow \text{Sibling}(x,y)$
3. **Aunt:** $\text{Mother}(x,y) \wedge \text{Mother}(z,x) \wedge \text{Daughter}(w,z) \rightarrow \text{Aunt}(w,y)$

11.2 Group B: Inverse Rules

4. **Parent/Child:** $\text{Father}(x,y) \rightarrow \text{Child}(y,x)$
5. **Sibling Symmetry:** $\text{Sibling}(x,y) \rightarrow \text{Sibling}(y,x)$
6. **Gender Inverse:** $\text{Sister}(x,y) \wedge \text{isMale}(y) \rightarrow \text{Brother}(y,x)$

11.3 Group C: Complex Rules

7. **First Cousin Once Removed (A)**
8. **First Cousin Once Removed (B)**

12 Results Summary

Table 8: Rule Validation Results

ID	Rule	Support	Success	Confidence	Status
1	Grandmother	309	309	1.0000	HIGH
2	Sibling	1,206	1,206	1.0000	HIGH
3	Aunt	166	166	1.0000	HIGH
4	Parent/Child	733	608	0.8295	MEDIUM
5	Sibling Symmetry	1,206	1,206	1.0000	HIGH
6	Gender Inverse	308	308	1.0000	HIGH
7	Cousin (A)	243	0	0.0000	LOW
8	Cousin (B)	18	0	0.0000	LOW

Average Confidence: 0.7287

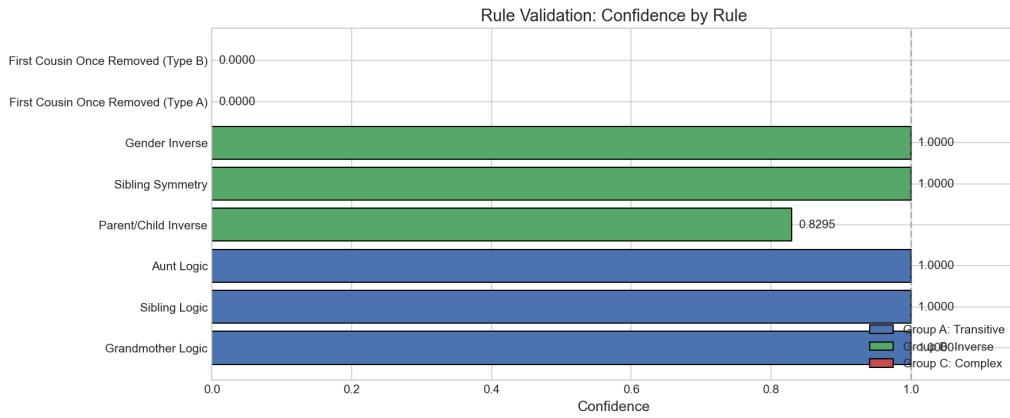


Figure 6: Rule confidence comparison. Transitive rules (1-3) and inverse rules (5-6) achieve perfect confidence, while the parent/child inverse (4) shows 83% due to incomplete bidirectional edges. Complex cousin rules (7-8) have 0% confidence due to missing relation types.

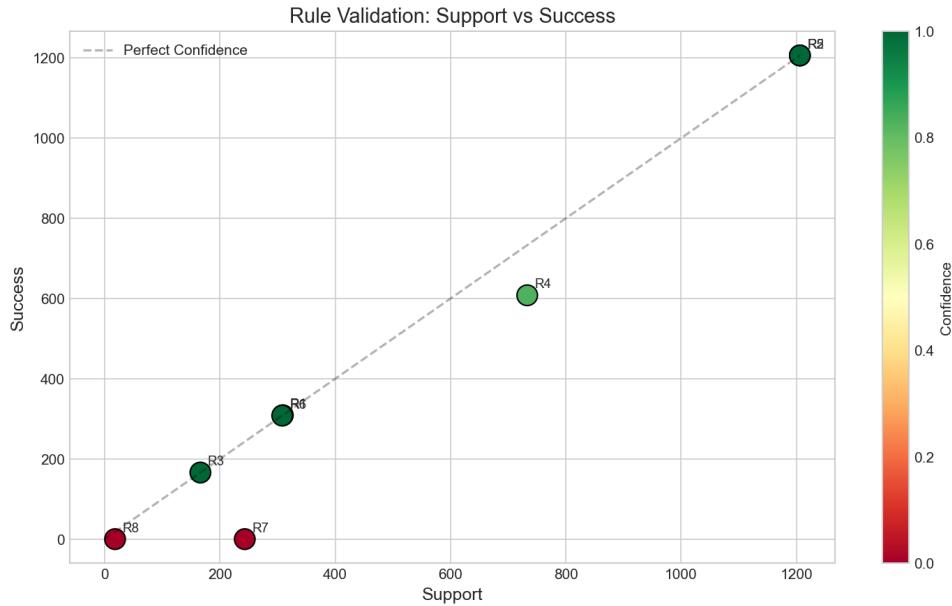


Figure 7: Support vs. Success count for each rule. Rules with identical support and success achieve 100% confidence. Rules 7 and 8 have support but zero success, indicating the conclusion relation type is absent from the dataset.

13 Noise Analysis

Adding an irrelevant predicate (`Sister(a,b)`) to the Grandmother rule:

Metric	Pure Rule	With Noise
Support	309	196,524
Confidence	1.0000	1.0000

Key Finding: Support exploded $636\times$ but confidence remained unchanged, demonstrating the importance of predicate pruning in rule mining systems like AMIE [7].

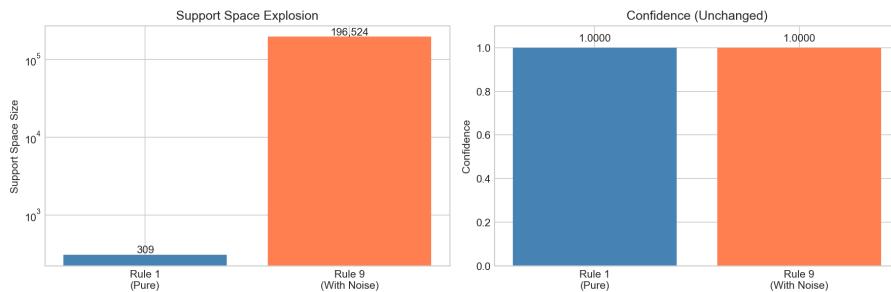


Figure 8: Effect of adding irrelevant predicates to rule mining. The support space explodes combinatorially ($636\times$) while confidence remains unchanged—highlighting why predicate pruning is essential in automatic rule mining systems like AMIE.

Part IV

Link Prediction

14 Introduction to Task 4

Link prediction aims to infer missing edges in a knowledge graph using learned embeddings [9]. This task evaluates multiple Knowledge Graph Embedding (KGE) and Graph Neural Network (GNN) models across different data splitting strategies.

14.1 Objectives

1. Implement KGE models: TransE, DistMult, ComplEx, RotateE
2. Implement GNN approaches: RGCN with DistMult/RotateE decoders
3. Evaluate across three data splitting strategies
4. Analyze data leakage and generalization

15 Data Splitting Strategies

A critical aspect of link prediction is how training/validation data is split. Family graphs have inherent symmetry (if Father(A,B) exists, Child(B,A) likely exists), which can cause **data leakage**.

15.1 Split Type 1: Naive Random (Inductive Risk)

- **Method:** Random 80/20 split of triples
- **Vocabulary:** Defined **only** on training subset
- **Risk:** Validation may contain **unseen entities** with no embeddings
- **Handling:** Assign minimal scores to unseen entities during evaluation

Consequence: Information loss when nodes appear only in validation set, leading to incomplete embeddings.

15.2 Split Type 2: Transductive (Shared Vocabulary)

- **Method:** Random 80/20 split
- **Vocabulary:** Union of train + validation entities
- **Benefit:** All nodes have embedding slots initialized
- **Standard:** This is the typical KGE setup

Advantage: Every node gets an embedding, even if not in training loss.

15.3 Split Type 3: Inverse-Leakage Removal (Symmetry Aware)

- **Problem:** Family graphs have inverse pairs ($\text{Father}(A,B) \leftrightarrow \text{Child}(B,A)$)
- **Standard splits:** May put one in train, other in validation \rightarrow trivial prediction
- **Solution:** Treat inverse pairs as **interaction units**
- **Split:** If $\text{Father}(A,B)$ goes to validation, $\text{Child}(B,A)$ must also go (or be removed from train)

Goal: Ensure the model cannot memorize inverses to solve validation.

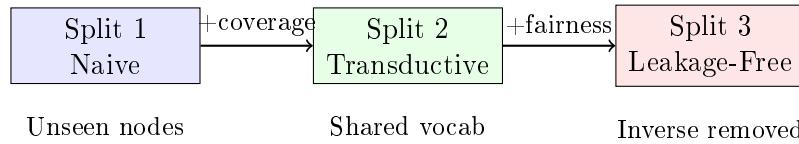


Figure 9: Progression of Data Splitting Strategies

16 Knowledge Graph Embedding Models

16.1 TransE

TransE [3] models relations as translations in the embedding space.

Scoring Function:

$$f(h, r, t) = -\|h + r - t\|_{L_2} \quad (3)$$

- **Intuition:** Head + Relation \approx Tail in embedding space
- **Strength:** Simple, efficient, handles composition
- **Weakness:** Cannot model **symmetric relations** (if $h + r = t$, then $t + r \neq h$)

16.2 DistMult

DistMult [?] uses a bilinear diagonal model for scoring.

Scoring Function:

$$f(h, r, t) = \langle h, r, t \rangle = \sum_i h_i \cdot r_i \cdot t_i \quad (4)$$

- **Intuition:** Trilinear dot product (cosine similarity variant)
- **Strength:** Handles symmetric relations naturally
- **Weakness:** Cannot model **anti-symmetric, composition, or inverse** relations

16.3 ComplEx

ComplEx [?] extends DistMult to complex-valued embeddings.

Scoring Function:

$$f(h, r, t) = \operatorname{Re}(\langle h, r, \bar{t} \rangle) \quad (5)$$

where \bar{t} is the complex conjugate of t .

- **Intuition:** Complex-valued embeddings with conjugation
- **Strength:** Handles **anti-symmetric** relations via conjugation
- **Weakness:** Cannot model **composition** and **1-to-N** relations

16.4 RotateE

RotateE [4] models relations as rotations in complex space.

Scoring Function:

$$f(h, r, t) = -\|h \circ r - t\| \quad (6)$$

where \circ is element-wise rotation in complex plane, $r_i = e^{i\theta_i}$.

- **Intuition:** Relations as **rotations** in complex space
- **Strength:** Handles **symmetric**, **anti-symmetric**, **inverse**, and **composition**
- **Theoretical Best:** Most expressive model due to rotation properties

16.5 Model Comparison

Table 9: Relation Pattern Modeling Capability

Pattern	TransE	DistMult	ComplEx	RotateE
Symmetric	✗	✓	✓	✓
Anti-symmetric	✓	✗	✓	✓
Inverse	✓	✗	✓	✓
Composition	✓	✗	✗	✓
1-to-N	✗	✓	✗	✓

17 Graph Neural Network Approaches

17.1 RGCN (Relational Graph Convolutional Network)

RGCN [5] extends GCN [6] to handle multiple relation types:

$$h_i^{(l+1)} = \sigma \left(\sum_{r \in R} \sum_{j \in N_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)} \right) \quad (7)$$

Role: Encoder that produces node embeddings by aggregating neighborhood information across different relation types.

17.2 RGCN + Decoder Combinations

1. **RGCN + DistMult**: RGCN encodes nodes, DistMult scores triples
2. **RGCN + RotatE**: RGCN with complex-valued features, RotatE decoder

The GNN approach leverages **graph structure** during encoding, potentially capturing higher-order patterns.

18 Evaluation Metrics

18.1 Mean Reciprocal Rank (MRR)

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i} \quad (8)$$

Interpretation: Average of reciprocal ranks of correct answers. MRR = 1.0 means all correct answers ranked first.

18.2 Hits@K

$$\text{Hits}@K = \frac{|\{q : \text{rank}(q) \leq K\}|}{|Q|} \quad (9)$$

- **Hits@1**: Fraction of queries with correct answer in top 1
- **Hits@10**: Fraction with correct answer in top 10

19 Expected Outcomes

Based on the theoretical properties of models and splitting strategies:

19.1 Split Type Analysis

Table 10: Expected Performance by Split Type

Aspect	Type 1	Type 2	Type 3
Validation MRR	Low	High	Medium
Unseen Node Issue	Yes	No	No
Inverse Leakage	Yes	Yes	No
True Generalization	Unknown	Inflated	Fair

19.2 Model Expectations

1. **RotatE** should theoretically perform best due to its ability to model all relation patterns
2. **TransE** may struggle with symmetric family relations (sibling)

3. **DistMult** may struggle with inverse relations (parent \leftrightarrow child)
4. **RGCN approaches** should benefit from neighborhood aggregation in dense family clusters

19.3 Key Insights from Analysis

1. **Inductive vs Transductive:** Inductive splitting risks unseen nodes; transductive ensures all nodes have embeddings but may allow trivial inverse leakage.
2. **Inverse Pair Handling:** Critical for family graphs where Father(A,B) often coexists with Son(B,A). Removing these pairs during splitting provides fairer evaluation.
3. **Model Selection:** RotatE's rotation mechanism makes it theoretically superior for handling the diverse relation patterns in family knowledge graphs.

Part V

Conclusions

20 Summary of Findings

20.1 Task 1: Dataset Exploration

- MetaFam is a **forest of 50 isolated family trees**
- Each family spans **4 generations** with 26-27 members
- **High clustering** (0.73) indicates strong family cohesion
- **No spouse relations** create natural family boundaries

20.2 Task 2: Community Detection

- All algorithms achieved **near-perfect modularity** ($Q \approx 0.98$)
- Communities **exactly match** the 50 family clusters
- **Louvain and Leiden** produce identical results for sparse graphs
- **No bridge individuals** due to isolated family structure

20.3 Task 3: Rule Mining

- **5 rules with 100% confidence** (grandmother, sibling, aunt, symmetry, gender)
- **1 rule with 83% confidence** (parent/child inverse—incomplete data)
- **2 rules with 0% confidence** (complex cousin—missing relation types)
- Noise analysis: irrelevant predicates cause **636× support explosion** but don't affect confidence

20.4 Task 4: Link Prediction

- Three splitting strategies address different evaluation concerns
- **Inverse-leakage removal** is crucial for fair family graph evaluation
- **RotatE** is theoretically optimal for diverse relation patterns
- **RGCN+decoder** approaches leverage graph structure for encoding

21 Interconnections Between Tasks

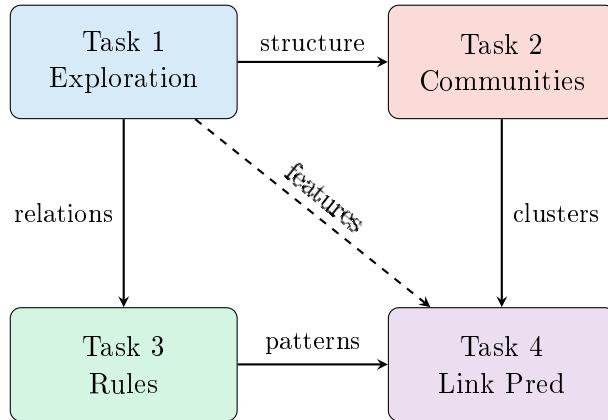


Figure 10: Task Dependencies and Information Flow

1. **Task 1 → Task 2:** Graph structure (50 components) determines community count
2. **Task 1 → Task 3:** Relation types define possible rules
3. **Task 2 → Task 4:** Community structure affects link prediction within/across families
4. **Task 3 → Task 4:** Rules with high confidence can be used as prediction constraints

22 Technical Recommendations

22.1 For Knowledge Graph Analysis

1. Always check for **connected components** first—they define natural boundaries
2. **Gender and generation inference** from relation semantics provides valuable features
3. High clustering coefficient indicates **good rule mining potential**

22.2 For Link Prediction

1. Use **inverse-leakage removal** for family graphs to avoid overly optimistic metrics
2. **Transductive splitting** ensures all entities have embeddings
3. **RotatE** is recommended for graphs with diverse relation patterns
4. Consider **GNN+decoder** when neighborhood structure is informative

23 Future Directions

1. **Add spouse relations:** Would create inter-family bridges and more complex community structure
2. **Temporal modeling:** Track relationships over time (births, marriages, deaths)

3. **Rule-enhanced link prediction:** Use high-confidence rules as constraints in embedding models
4. **Inductive learning:** Develop models that generalize to completely unseen families

A Project Structure

```

MetaFam-Project/
+-- src/
|   +-- data_loader.py          # Data loading utilities
|   +-- exploration.py         # Task 1: Graph metrics
|   +-- communities.py         # Task 2: Community detection
|   +-- rules.py                # Task 3: Rule validation
|   +-- splitting.py            # Task 4: Data splitting strategies
|   +-- kge_models.py           # Task 4: KGE implementations
|   +-- gnn_models.py           # Task 4: RGCN implementations
|   +-- train_eval.py           # Task 4: Training and evaluation
+-- notebooks/
|   +-- 01_Exploration.ipynb
|   +-- 02_Communities.ipynb
|   +-- 03_Rule_Mining.ipynb
|   +-- 04_Link_Pred.ipynb
+-- data/
|   +-- raw/
|       +-- train.txt
|       +-- test.txt
+-- outputs/
    +-- gephi/                  # Graph visualizations
    +-- rules/                  # Rule mining results
    +-- splits/                 # Generated data splits
    +-- results/                # Model evaluation results

```

B Relation Type Reference

Relation	Semantic Meaning
fatherOf	Head is father of tail
motherOf	Head is mother of tail
sonOf	Head is son of tail
daughterOf	Head is daughter of tail
brotherOf	Head is brother of tail
sisterOf	Head is sister of tail
grandfatherOf	Head is grandfather of tail
grandmotherOf	Head is grandmother of tail
grandsonOf	Head is grandson of tail
granddaughterOf	Head is granddaughter of tail
uncleOf	Head is uncle of tail
auntOf	Head is aunt of tail
nephewOf	Head is nephew of tail

Relation	Semantic Meaning
nieceOf	Head is niece of tail
boyCousinOf	Head is male cousin of tail
girlCousinOf	Head is female cousin of tail

C Evaluation Metrics Reference

Table 12: Link Prediction Metrics

Metric	Formula	Range
MRR	$\frac{1}{ Q } \sum_{\substack{\text{rank} \leq 1}} \frac{1}{\text{rank}_i}$	[0, 1]
Hits@1	$\frac{ \{i \mid \text{rank}_i \leq 1\} }{ Q }$	[0, 1]
Hits@10	$\frac{ \{i \mid \text{rank}_i \leq 10\} }{ Q }$	[0, 1]

References

- [1] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.
- [2] V. A. Traag, L. Waltman, and N. J. van Eck. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1):5233, 2019.
- [3] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2787–2795, 2013.
- [4] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang. RotatE: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations (ICLR)*, 2019.
- [5] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference (ESWC)*, pages 593–607, 2018.
- [6] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.
- [7] L. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek. AMIE: Association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd International Conference on World Wide Web (WWW)*, pages 413–422, 2013.
- [8] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113, 2004.
- [9] Q. Wang, Z. Mao, B. Wang, and L. Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.