Jira Link Prediction

Fateme Faraji Daneshgar
Research Associate

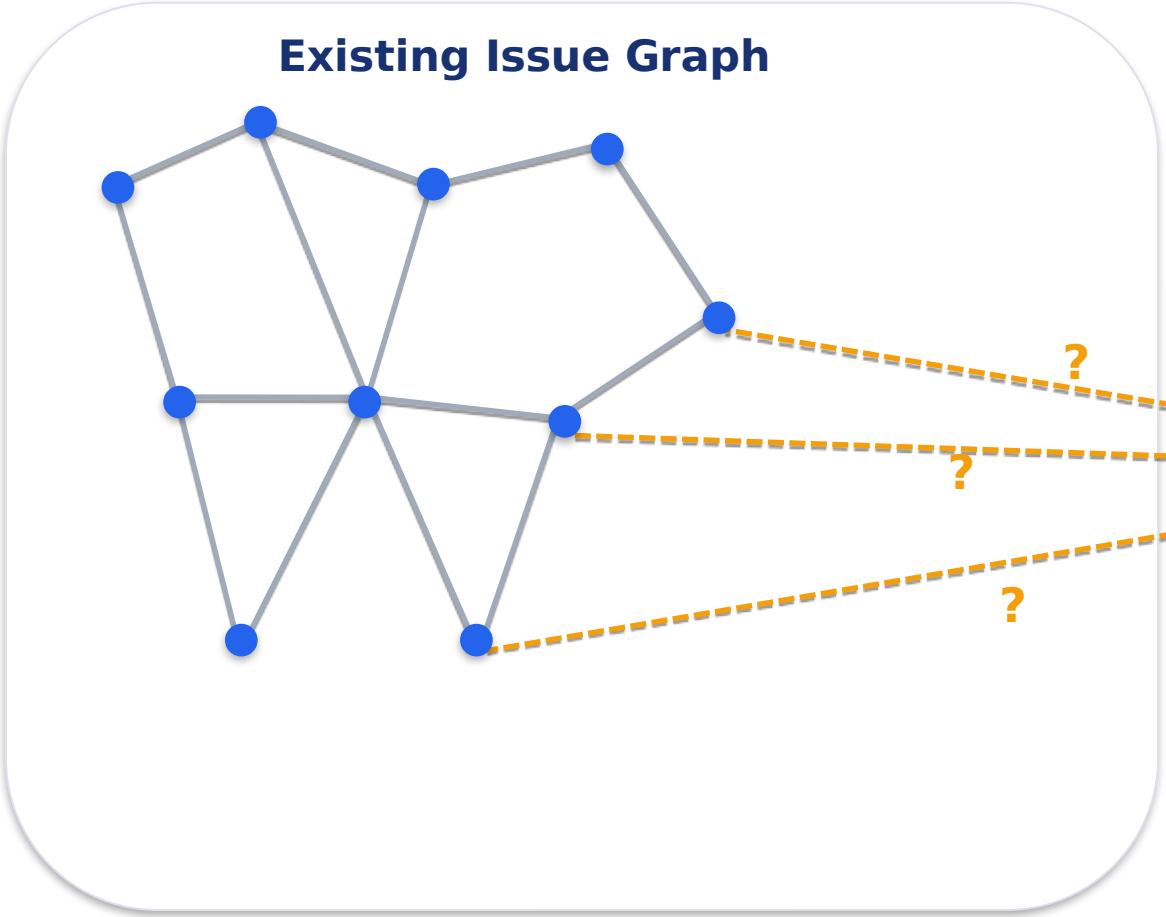
Polytechnique Montréal

DORSAL Laboratory

Agenda

- Problem Modeling
- Attribute-Based Link Prediction
- Pairwise Feature Engineering
- Contrastive Learning
- Classification Model
- Empirical Observations
- Structural Feature Engineering
- Multi-Expert Link Prediction
- LLM-based Link Predictor
- Results
- Conclusion

Problem Modeling



New JIRA Issue

- Summary
- Description
- Metadata

14,124
Issues

5,149
Known Links

Predict Links
Goal

Objective: Given a newly created JIRA issue, predict its relationships with existing issues in the graph.

Objective: Train a model that predicts the probability of a link between two JIRA issues.

Two complementary sources of information

Attribute-based

Node Features

- Summary
- Description
- Metadata
- Embeddings

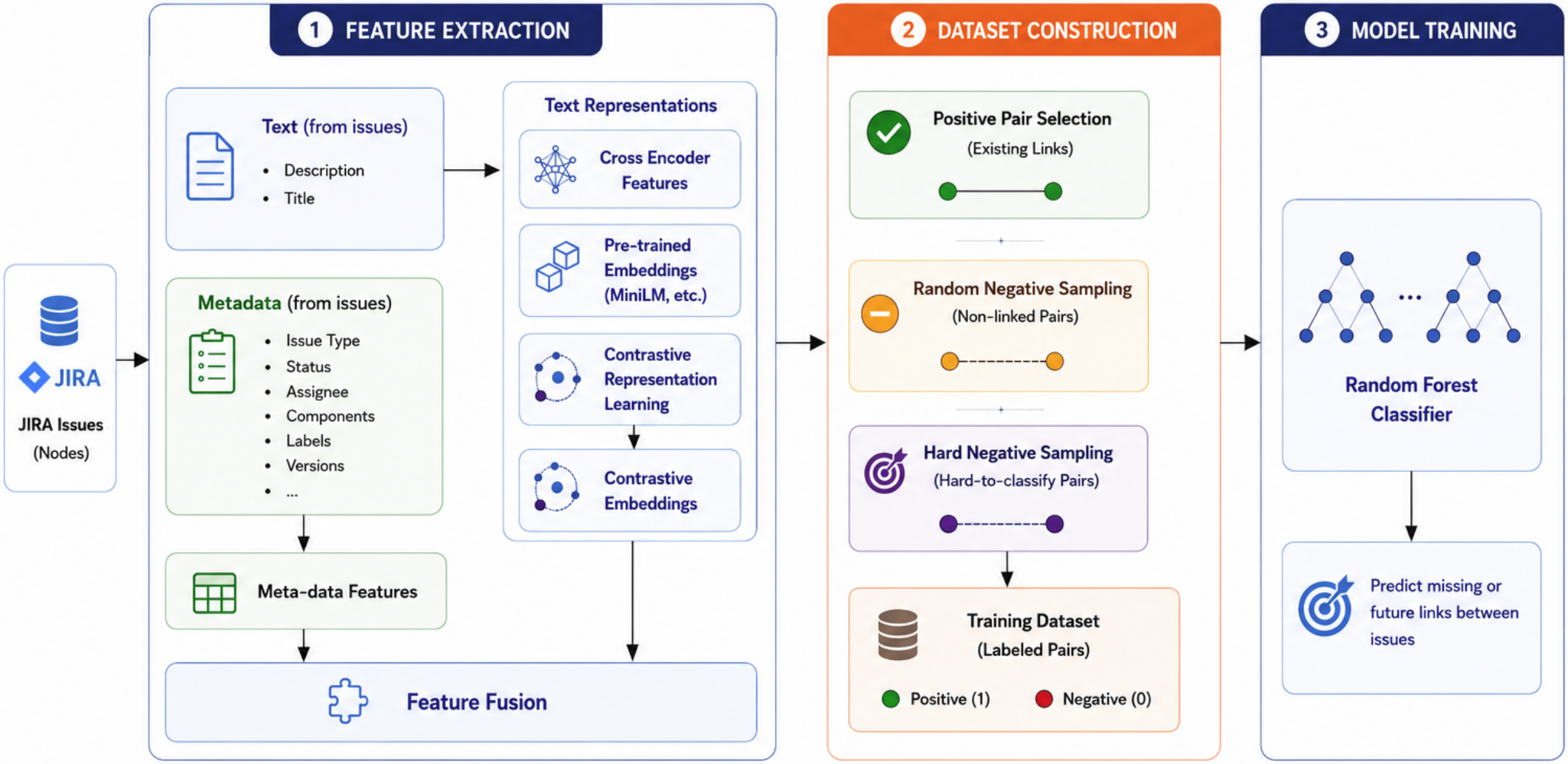
Structure-based

Graph Topology

- Neighborhood
- Connectivity
- Paths
- Graph Context

Link Prediction

Attribute-base Model Training Pipeline



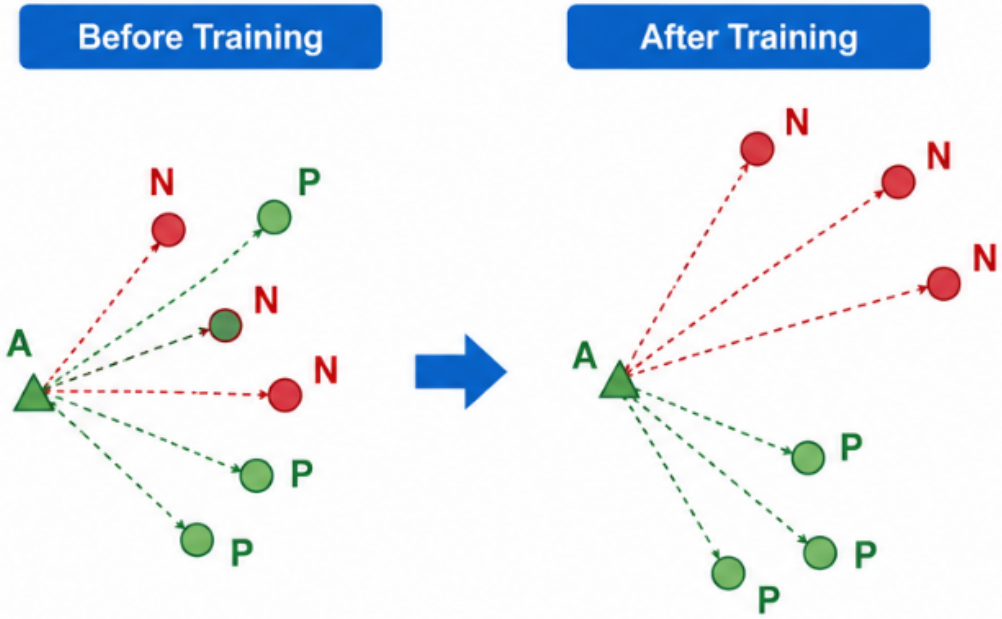
Contrastive Learning

- **Training strategy:**
  - Triplets: (Anchor, Positive, Negative)
  - Positive: linked issue (real relation)
  - **Negative:** hard negative issue
  - Objective: maximize separation between confusing pairs
  - Result: **task-specific embedding space**

Triplet Loss

$$L = \max(0,\, d(a,p) - d(a,n) + \text{margin})$$

$$d(a,n) \geq d(a,p) + \text{margin}$$



Triplet loss pulls linked issues closer and pushes hard negatives farther apart, producing task-specific embeddings.

A Anchor (issue) P Positive (linked issue) N Negative (hard negative)

Pairwise Feature Engineering

Metadata-based Features

Presence features:
both_present_assignee,
both_present_components, ...
Equality features: same_project,
same_assignee, same_reporter, ...
Jaccard similarity for set-valued fields
- components
- labels

Additional metadata similarity indicators

Embedding-based Features

Computed from issue embeddings
distance_cosine
distance_l2
absdiff_mean
prod_mean
Represent semantic similarity between issue pairs

Total extracted pairwise features at this stage: 42

Model Details

Sentence Encoding Model

- all-MiniLM-L6-v2
- 384- dimensional embeddings
- Produces sentence embeddings
- Very fast retrieval

Classification Model

- Random Forest
- Captures nonlinear decision boundaries
- Feature importance
- Probability output

Cross Encoder Model

- cross-encoder/stsb-roberta-base
- RoBERTa-base encoder (12-layer Transformer encoder)
- Joint encoding of issue pairs
- Models cross-text token interactions
- Fine-tuned on Semantic Textual Similarity (STS)
- Outputs a semantic similarity score

Evaluation

- precision: 0.35
- recall: 0.46
- f1: 0.397

Empirical Observations

After evaluating our model and investigating false positives, we observed:

1 Identical Text (Depends on Project)

Summary: *identical*
Description: *identical*

Same Project — Sometimes linked, sometimes not

- Issue A ⋯ Issue B — ✓ Link
- Issue C ⋯ Issue D — ✗ No Link

Different Projects — Should be linked

- Issue E (Proj X) ⋯ Issue F (Proj Y) — ✓ Link

Observation:
Identical issues are not always linked within the same project, but should be linked across different projects.

2 Highly Similar Text (Partially Predictable)

Summary: *very similar*
Description: *very similar*

Issue A:

- Issue B — ✓ Link
- Issue C — ✗ No Link
- Issue D — ✗ No Link
- Issue E — ✓ Link

Observation:
Only a small fraction of highly similar issues are actually linked.
≈20% of linked issue pairs belong to this category.

3 Text Similarity Is Not Sufficient (Real Challenge)

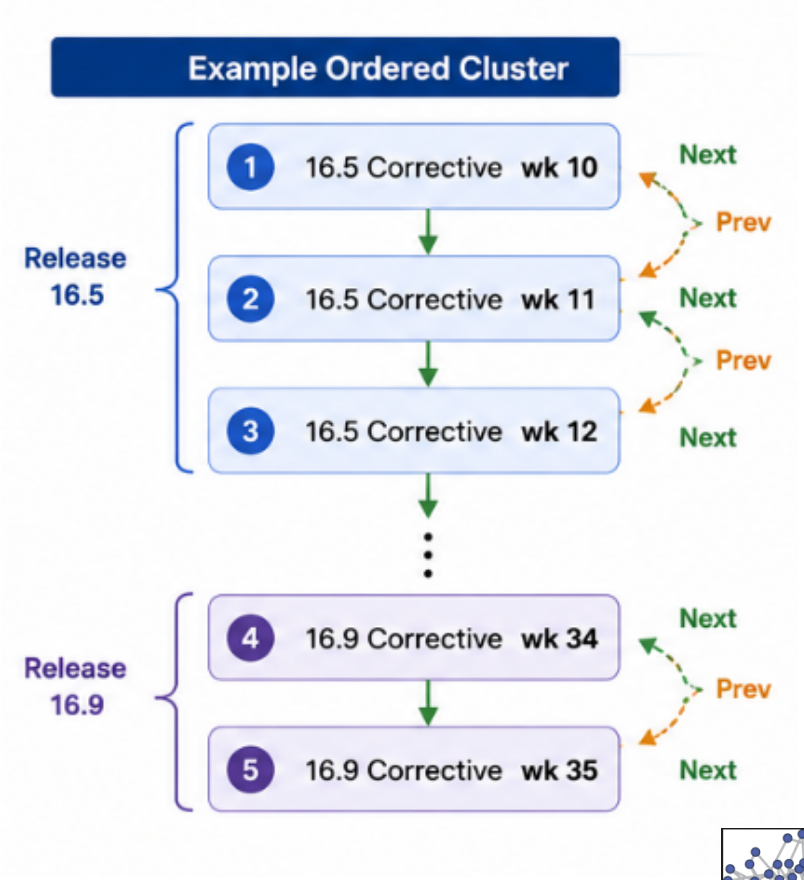
Low similarity (but linked): Issue A ⋯ Issue B ✓

High similarity (but not linked): Issue C ⋯ Issue D ✗

Observation:
Links exist between issues with low textual similarity, while many highly similar issue pairs are not linked.

Conclusion:
Link prediction depends on more than semantic similarity.
Additional contextual and structural information is required.

Structural Feature Augmentation



1) Cluster Nodes

Identify the cluster to which each node belongs based on release/version and other numerical identifiers.

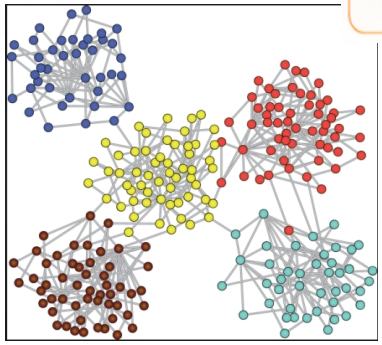
Each node is annotated with its cluster ID.

2) Next Node (Next-Node)

For each node, identify its immediate successor within the same cluster according to the defined order.

3) Previous Node (Prev-Node)

For each node, identify its immediate predecessor within the same cluster according to the defined order.



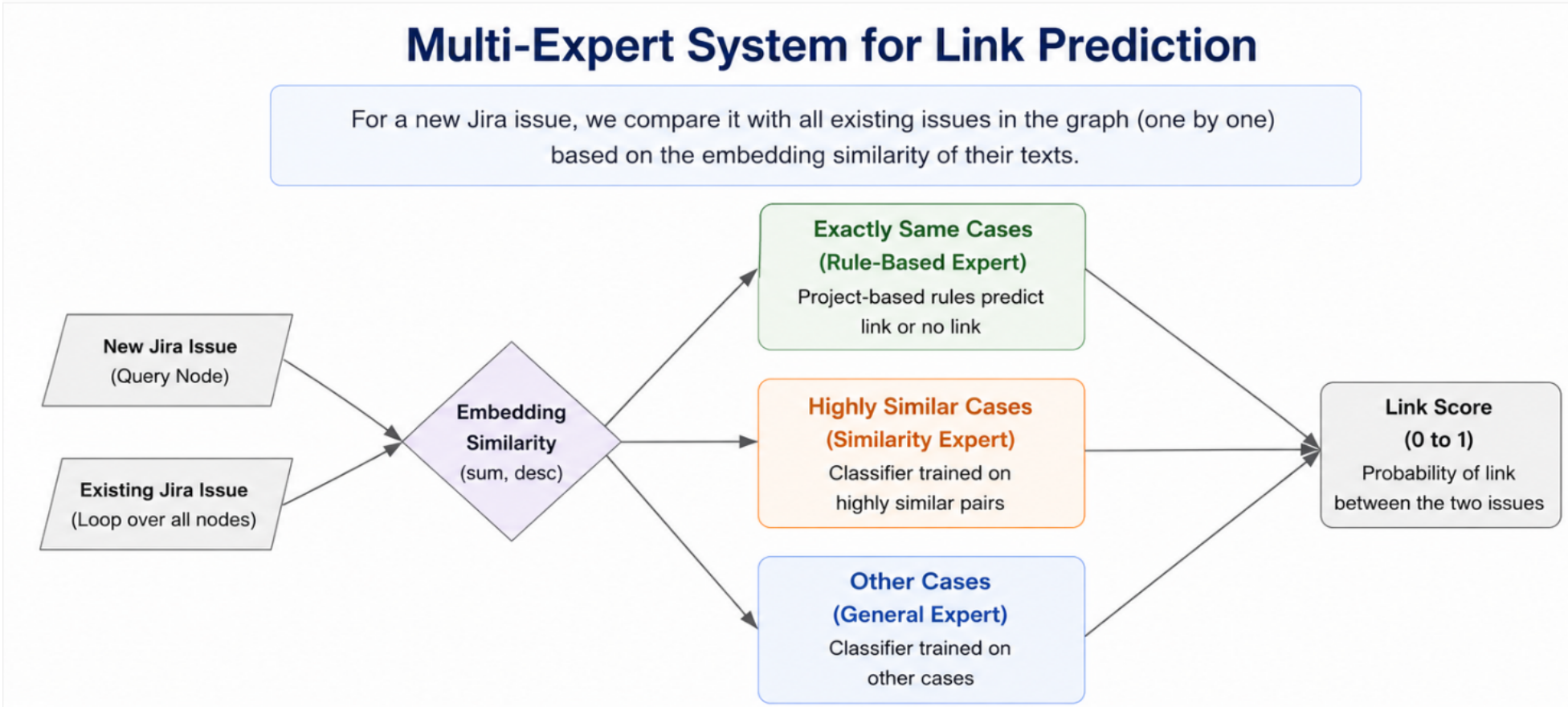
Heuristic Ordering Rules

Nodes in an ordered cluster are sorted according to release, version, and other numbers.
The ordering determines the previous-node and next-node relationships.

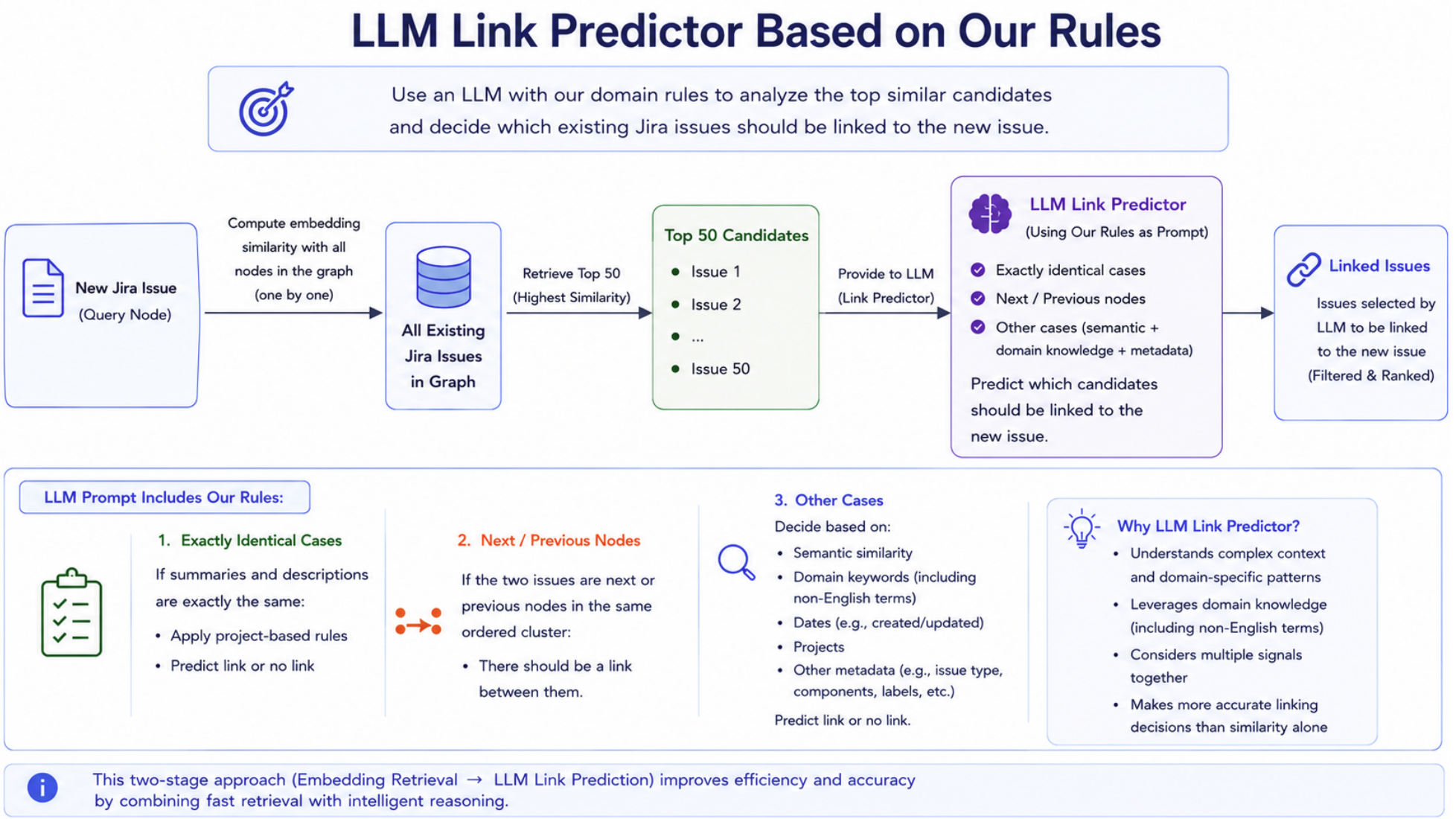
Rule	Condition	Ordering Strategy
1 **Same release and different versions**	Release is the same, versions are different	Order based on **version**
2 **Same release/version and different numbers**	Release and version are the same, other numbers are different	Order based on **numbers**
3 **Just release or version**	Only release or only version is available	Order based on **release/version**
4 **Just numbers**	Only numbers are available (no release or version)	Order based on **numbers**
5 **Tie-breaking**	If multiple nodes have identical keys according to the above rules	• Order by **creation date** (ascending) • If still tied, order by **issue key** (lexicographical)

Purpose: These ordering rules ensure a consistent and meaningful sequence of nodes within each cluster, which is used to identify previous-node and next-node relationships.

Experiments



LLM Link Predictor



Results- Identical cases

========== Nodes With True Neighbors ==========

=======LLM=======	=======RF=======
Average Precision: 64%	Average Precision: 56%
Average Recall: 66%	Average Recall: 68%
Average F1: 62%	Average F1: 57%

========== Nodes With No True Neighbors ==========

Correct Empty Prediction Percentage: 40.32% (LLM) 17.74%(RF)

Average FP per Node: 2.0 (LLM) 5.2 (RF)

Results- highly similar cases

========== Nodes With True Neighbors ==========

=======LLM=======	=======RF=======
Average Precision: 53%	Average Precision: 40%
Average Recall: 63%	Average Recall: 73%
Average F1: 54%	Average F1: 44%

========== Nodes With No True Neighbors ==========

Correct Empty Prediction Percentage: 30.32% (LLM) 12.03%(RF)

Average FP per Node: 1.8 (LLM) 5.3 (RF)

Results- other cases

========== Nodes With True Neighbors ==========

========LLM========	========RF========
Average Precision: 18%	Average Precision: 20%
Average Recall: 18%	Average Recall: 77%
Average F1: 17%	Average F1: 26%

========== Nodes With No True Neighbors ==========

Correct Empty Prediction Percentage: 84.50% (LLM) 9.23%%(RF)

Average FP per Node: 0.2 (LLM) 10.6 (RF)

Estimated Overall Performance

The ratio of nodes in different cases(2% , 11% and 87%)

========== Nodes With True Neighbors ==========

=======LLM=======	=======RF=======
Average Precision: 22.27%	Average Precision: 22.92%
Average Recall: 23.91%	Average Recall: 76.38%
Average F1: 21.97%	Average F1: 28.6%

========== Nodes With No True Neighbors ==========

Correct Empty Prediction Percentage: 77.65% (LLM) 9.7%(RF)

Average FP per Node: 0.41 (LLM) 9.9 (RF)

==========Runtime==========

LLM = 3 min

RF = 1 min

Future Works

- Improving the LLM prompt
- Adding domain keyword to RF model
- Experimenting with other encoders

Thank you for your attention!