

ArtifactLinker: Linking Scientific Artifacts for Automatic State-of-the-Art Discovery

Haofei Yu^{1,2*} Jiaxuan You¹ Peter Clark²

Bodhisattwa Prasad Majumder² Kyle Richardson²

¹University of Illinois Urbana-Champaign ²Allen Institute for AI
haofei2@illinois.edu, kyler@allenai.org

 Code: <https://github.com/allenai/artifact-linker>

 Data: <https://huggingface.co/datasets/lwaekfjlk/artifact-bench>

Abstract

Scientific artifacts such as models and datasets are foundations for research. With the rapid growth of platforms like HuggingFace, researchers now have access to a large number of artifacts. Yet, a key challenge remains: how can we automatically discover the state-of-the-art (SOTA) model for a given dataset by fully leveraging existing artifacts? We formalize this task as **automatic SOTA discovery** by modeling HuggingFace as an artifact graph, where nodes are models/datasets and edges represent evaluations. We propose ARTIFACTLINKER, a two-stage framework: (1) ranking promising unobserved model–dataset links using Graph Neural Networks (GNNs) or graph-augmented Large Language Models (LLMs), and (2) verifying top-ranked links via coding experiments with LLM-based agents. We further introduce a benchmark named ARTIFACTBENCH with 14,053 artifacts and 51,337 relations to evaluate the performance of both stages. Results show that (1) graph structures between existing artifacts are effective for missing link prediction; (2) end-to-end ranking and verification with ARTIFACTLINKER help discover potential SOTA results and research insights.

1 Introduction

Scientific artifacts are fundamental building blocks of modern research (Heumüller et al., 2020; Cooper et al., 2022; Johnson et al., 2019). Models and datasets on the HuggingFace Hub are representative examples: researchers routinely share, reuse, and extend these artifacts to support reproducibility and drive progress (Marić et al., 2023; Lissa et al., 2020). As the machine learning community continues to produce an increasing number of artifacts across sub-domains (Castaño et al., 2024; Ait et al., 2023; Laufer et al., 2025), a natural question arises: *Can we leverage existing artifacts to support automatic scientific discovery?* We study the HuggingFace ecosystem as a concrete setting for this question because it provides a large, active, and executable collection of models, datasets, and libraries.

Thinking of HuggingFace as an artifact graph. We can conceptualize the HuggingFace ecosystem as a structured graph (Chen et al., 2025b; Laufer et al., 2025). As illustrated in Figure 1, models, datasets, papers, and codebases form nodes, while finetuning, reference, and evaluation relations form edges; evaluation metrics such as F1 scores serve as edge attributes. This graph view is enabled by three properties of HuggingFace: (1) a large and continuously growing artifact collection, (2) unified APIs that support programmatic access, and (3) rich metadata linking artifacts and evaluation results. Unlike academic papers, which often report results without directly linking them to executable resources, HuggingFace connects evaluation outcomes to concrete models and datasets, including many artifacts without formal publications. While prior work has mainly treated HuggingFace as a

*Work done during the internship at the Allen Institute for AI.

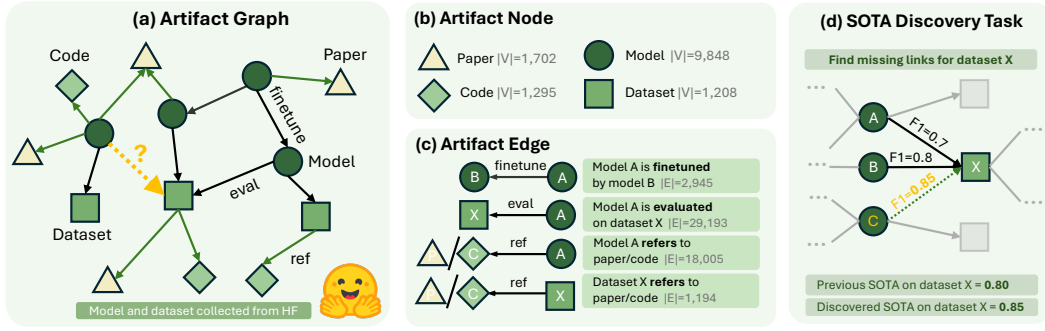


Figure 1: **Artifact graph structure and SOTA discovery task formulation.** (a) **Example graph.** Visualization of the artifact graph, highlighting its sparsity and missing links across artifact types. (b) **Node statistics.** Distribution of nodes across artifact categories. (c) **Edge statistics.** Distribution of edges across relation types. (d) **Task definition.** SOTA discovery formulated as link prediction over the artifact graph.

retrieval source (Silva et al., 2025) or API hub (Shen et al., 2023), we study whether these structured artifact relationships can be converted to an engine for automatic discovery.

Challenges for automatic discovery. Turning this artifact graph into an automatic discovery system presents two key challenges. (1) **Task ambiguity:** “automatic discovery” lacks a precise operational definition, making systematic evaluation difficult (Beel et al., 2025). (2) **Scalability:** the combinatorial number of possible model–dataset pairs makes exhaustive code-based verification prohibitively expensive (Urbanowicz et al., 2022).

Linking artifacts as SOTA discovery. We address **task ambiguity** by instantiating automatic discovery as *SOTA discovery*: identifying model–dataset pairs that achieve previously unseen evaluation scores. In the artifact graph, this becomes a concrete link prediction and ranking problem—identifying missing model–dataset links with high predicted edge attributes. This formulation provides a measurable objective for evaluating automatic discovery systems.

Scalable framework for SOTA discovery. We address **scalability** with a two-stage *rank-then-verify* framework. The ranking stage prioritizes promising model–dataset pairs using graph-based signals, substantially reducing the number of candidates that require expensive execution. The verification stage then synthesizes and executes evaluation code only for highly ranked pairs. This design enables efficient search while ensuring that discovered results are grounded in reproducible execution.

Main contributions. Our work makes three contributions: (1) We construct ARTIFACTBENCH, a benchmark for SOTA discovery in the HuggingFace ecosystem that includes prediction, ranking, and verification tasks; (2) We propose ARTIFACTLINKER, a two-stage rank-then-verify framework and establish baseline results on ARTIFACTBENCH; and (3) We demonstrate ARTIFACTLINKER in an end-to-end Natural Language Inference (NLI) case study, showing its ability to discover new artifact relationships and generate research insights. Together, ARTIFACTBENCH and ARTIFACTLINKER provide a concrete benchmark and initial framework for studying automatic discovery over scientific artifacts.

2 Related Works

HuggingFace platform utilization. HuggingFace has increasingly become a natural platform for studying automatic discovery. Prior work has largely relied on static analyses of its artifacts and relationships to characterize trends in machine learning development (Chen et al., 2025b; Laufer et al., 2025). Beyond serving as a repository, HuggingFace has been conceptualized in multiple ways: as a knowledge graph (Silva et al., 2025), an API hub (Shen et al., 2023), a model card aggregator (Yang et al., 2024), and even an evolutionary tree (Gao & Gao, 2023). Other studies have examined its community dynamics (Rahman et al., 2025a; Castaño et al., 2023). In contrast, our work moves beyond static description and trend analysis and focuses on performance prediction and execution-based verification.

Resource	Model	Data	Paper	Code	Eval. Score	Model Lineage	Execution
Linked PwC	✓	✓	✓	✓	✓	✗	✗
HuggingKG	✓	✓	✓	✗	✗	✓	✗
HuggingGraph	✓	✓	✗	✗	✗	✓	✗
ArtifactBench	✓	✓	✓	✓	✓	✓	✓

Table 1: **Comparison with existing artifact-based benchmarks.** Model, Data, Paper, and Code denote the included artifact types; Eval. Score denotes model–dataset evaluation results with associated metrics and scores; Model Lineage denotes model derivation relations such as fine-tuning; and Execution denotes whether these benchmarks report reproduced evaluation results after execution.

Large-scale prediction for accelerating discoveries. Accelerating scientific discovery has been a major focus in domains such as drug discovery (Stokes et al., 2020; Serrano et al., 2024; Vişan & Neşuţ, 2024; You et al., 2022), materials science (Xie & Grossman, 2018; Butler et al., 2018), and molecular design (Segler et al., 2018), among others (Yu et al., 2025; Cheng et al., 2025). In these settings, experimental verification is prohibitively costly and time-consuming. In contrast, our work focuses on a more tractable class of automatic discovery tasks by leveraging the intrinsic linking structure of HuggingFace artifacts.

LLM-based coding agents for reproducible experimentation. Prior work has explored free-form discovery with generating executable code from research ideas (Lu et al., 2024; Jansen et al., 2024; 2025), though evaluation remains challenging given the open-ended nature of such tasks. Other efforts have focused on reproducing experiments within specific codebases (Bogin et al., 2024; Starace et al., 2025; Kim et al., 2025; Seo et al., 2025; Siegel et al., 2024; Xiang et al., 2025; Bragg et al., 2026), which is challenging due to the complexity of such codebases. In contrast, our tasks rely on reproducing a more concrete/grounded set of research artifacts.

3 Constructing an Artifact Graph from the HuggingFace Hub

We construct an artifact graph from the HuggingFace ecosystem to support artifact discovery and execution-based verification. Existing resources capture complementary aspects of the artifact ecosystem: Linked PwC (Färber & Lamprecht, 2023) organizes models, datasets, papers, and benchmark results, while HuggingKG (Chen et al., 2025a) and HuggingGraph (Rahman et al., 2025b) capture HuggingFace artifacts and their provenance relations. However, as summarized in Table 1, none jointly provides model-dataset evaluation scores, model lineage, and executable artifacts required for prediction followed by verification. This motivates the artifact graph we construct below.

Definition of artifact graphs. We model the artifact ecosystem as a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathcal{V}_m \cup \mathcal{V}_d \cup \mathcal{V}_p \cup \mathcal{V}_c$ contains four node types: models, datasets, papers, and codebases. Each node is associated with semantic attributes derived from its documentation, such as model cards, paper abstracts, and repository descriptions. The primary edge set, $\mathcal{E}_{\text{eval}} \subset \mathcal{V}_m \times \mathcal{V}_d$, represents evaluation relations: an edge (m, d) indicates that model m has been evaluated on dataset d with observed score $f^*(m, d)$. These edges serve as both training supervision and prediction targets. We further include artifact–paper edges $\mathcal{E}_{\text{paper}}$, artifact–codebase edges $\mathcal{E}_{\text{code}}$, and model–model fine-tuning edges $\mathcal{E}_{\text{finetune}} \subset \mathcal{V}_m \times \mathcal{V}_m$. These auxiliary relations enrich the graph topology and provide additional structural context for message passing.

Graph construction. We construct the artifact graph from popular models and datasets on the HuggingFace Hub, with monthly downloads exceeding 1000, and enrich them with referenced arXiv papers and GitHub repositories. The core of the graph consists of structured evaluation records linking models to datasets with reported metrics and scores, while referenced papers, codebases, and model lineage provide additional artifact relations. After filtering isolated entries without any edges, the resulting graph contains 14,053 nodes and 51,337 edges, with detailed statistics shown in Figure 1. Artifact metadata

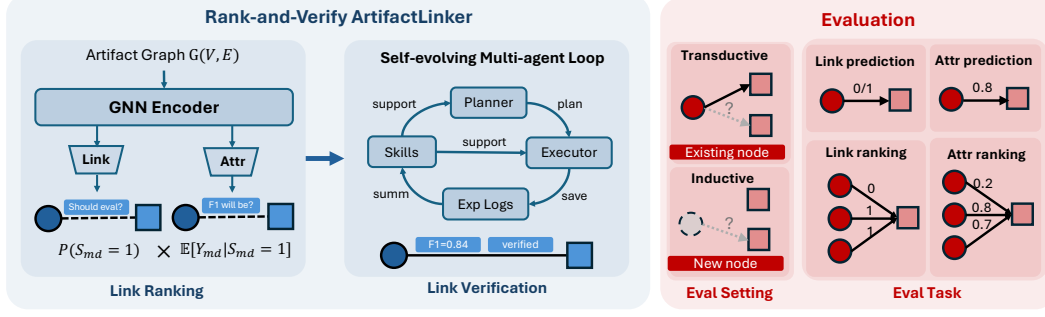


Figure 2: **Overview of ARTIFACTLINKER and its evaluation framework.** (Left) The two-stage rank-and-verify pipeline. A GNN-based ranking model first estimates the ranking score for unobserved model–dataset pairs. The top-ranked candidates are then selected for execution in the verification stage. (Right) Ranking evaluation tasks. We evaluate the system under both transductive (nodes observed during training) and inductive (nodes unseen during training) settings. The evaluation spans four distinct tasks covering both link and attribute prediction/ranking.

is additionally processed into leakage-free textual node representations for downstream models. Full construction details and leakage analysis are provided in Appendix §F.

4 Linking Scientific Artifacts for Automatic SOTA Discovery

We first formalize automatic SOTA discovery using the artifact-graph formulation, and then introduce ARTIFACTLINKER, a scalable two-stage *rank-and-verify* framework.

4.1 Definition of Automatic SOTA Discovery

Building on the artifact graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ defined in Section §3, we treat each observed edge as a realized evaluation record. A ground-truth oracle $f^* : \mathcal{V}_m \times \mathcal{V}_d \rightarrow \mathbb{R}$ returns the benchmark score of a model–dataset pair. Automatic SOTA discovery aims to identify a missing link $(m, d) \notin \mathcal{E}$ that, upon verification, establishes a new state of the art by exceeding the current best score on dataset d :

$$f^*(m, d) > f_{\max}^*(d) = \max_{m' | (m', d) \in \mathcal{E}} f^*(m', d). \quad (1)$$

In other words, the goal is to discover unobserved model–dataset pairs whose verified performance is SOTA. The input to ARTIFACTLINKER is the artifact graph $\mathcal{G}$, and the output is a set of candidate (m, d) pairs.

Rank-and-verify framework. Evaluating all missing links through the code-execution oracle f^* is computationally prohibitive. We therefore adopt a two-stage pipeline: a graph-based ranker first estimates $\hat{f}(m, d)$ for unobserved pairs (§4.2), and a verifier then executes only the top-ranked candidates to identify new SOTA results (§4.3).

4.2 Link Ranking via Graph Modeling

The selection bias problem. The ranking stage estimates the expected performance $\hat{f}(m, d)$ of unobserved model–dataset pairs without executing them. Let $S_{md} \in \{0, 1\}$ indicate whether a pair (m, d) can be successfully evaluated, yielding benchmark score Y_{md} . Training only on observed pairs estimates the *conditional* expectation $\mathbb{E}[Y_{md} | S_{md} = 1]$, implicitly assuming that the pair is viable. However, ranking unobserved pairs requires the *unconditional* expectation $\mathbb{E}[Y_{md}]$, which must also account for model–dataset incompatibility. Ignoring this distinction introduces selection bias and can assign high predicted scores to pairs that ultimately fail to execute.

Ranking formulation. We assume incompatible pairs yield zero utility, $\mathbb{E}[Y_{md} | S_{md} = 0] = 0$. By the law of total expectation,

$$\hat{f}(m, d) = \mathbb{E}[Y_{md}] = P(S_{md} = 1) \cdot \mathbb{E}[Y_{md} | S_{md} = 1]. \quad (2)$$

This factorization decomposes ranking into two subproblems: a *link predictor* estimating the compatibility probability $P(S_{md} = 1)$, and an *attribute predictor* estimating the conditional performance $\mathbb{E}[Y_{md} | S_{md} = 1]$.

Ranking modeling. To instantiate this formulation, we first encode the artifact graph using an L -layer graph encoder:

$$\mathbf{h}_v^{(k+1)} = \text{AGG}^{(k)} \left(\mathbf{h}_v^{(k)}, \left\{ \mathbf{h}_u^{(k)}, e_{uv} : u \in \mathcal{N}(v) \right\} \right), \quad (3)$$

yielding final node embeddings $\mathbf{z}_v = \mathbf{h}_v^{(L)}$. We then parameterize the two subproblems with two specialized heads over the shared representations:

$$P(S_{md} = 1) = \phi_l(\mathbf{z}_m, \mathbf{z}_d), \quad \mathbb{E}[Y_{md} | S_{md} = 1] = \phi_a(\mathbf{z}_m, \mathbf{z}_d), \quad (4)$$

where ϕ_l is the link predictor, ϕ_a is the attribute predictor, and ϵ captures observation noise. Both predictors jointly form to model the ranking function.

Joint training. Let $\hat{S}_{md} = \phi_l(\mathbf{z}_m, \mathbf{z}_d)$ denote predicted compatibility and $\hat{Y}_{md} = \phi_a(\mathbf{z}_m, \mathbf{z}_d)$ the predicted conditional score. We train the link predictor using binary cross-entropy (BCE) over observed positive pairs and randomly sampled negative pairs, while the attribute predictor is trained using mean squared error (MSE) over observed positive pairs:

$$\mathcal{L}_{\text{link}} = \sum_{(m,d) \in \mathcal{E}^+ \cup \mathcal{E}^-} \text{BCE}(\hat{S}_{md}, S_{md}), \quad \mathcal{L}_{\text{attr}} = \sum_{(m,d) \in \mathcal{E}^+} \text{MSE}(\hat{Y}_{md}, Y_{md}), \quad (5)$$

where $\mathcal{E}^-$ contains randomly sampled negative pairs. Beyond realizing the factorization in Eq. (2), the shared encoder provides multi-task regularization. Although ϕ_a receives no direct supervision on incompatible pairs, $\mathcal{L}_{\text{link}}$ shapes the shared representations $\mathbf{z}$ with compatibility information, allowing the attribute predictor to capture the viable-incompatible boundary beyond what can be learned from $\mathcal{E}^+$ alone.

Joint inference. At inference time, we rank each unobserved pair using

$$\hat{f}(m, d) = \phi_l(\mathbf{z}_m, \mathbf{z}_d) \cdot \phi_a(\mathbf{z}_m, \mathbf{z}_d). \quad (6)$$

This multiplicative form naturally suppresses incompatible candidates: as $\phi_l \rightarrow 0$, the final score approaches zero regardless of the regressor output ϕ_a . Among plausible pairs, ϕ_a then differentiates candidates by their expected performance.

4.3 Link Verification with Self-Evolving Multi-Agent Framework

The combination problem. Given the top-ranked candidate pairs $(m, d) \in \mathcal{C}$ collected from the ranking model $\hat{f}(m, d)$, the verification stage must automatically synthesize and execute code to verify the true benchmark score Y_{md} . However, standard coding agents treat each evaluation pair independently, causing repeated failures on shared artifact-specific constraints, such as atypical column schemas or required loading flags (e.g., `trust_remote_code=True`). We term this redundant rediscovery the *combination problem*.

Multi-agent self-evolution. We address this problem with a cross-instance memory loop built into a multi-agent framework. After each execution, an LLM reviewer distills runtime failures and successful workarounds into model-, dataset-, and task-specific memories. A *planner agent* uses these accumulated memories to produce an error-aware evaluation plan, which an *executor agent* implements and debugs through the CodeAct workflow (Wang et al., 2024), with access to HuggingFace metadata APIs. This enables successful fixes to transfer across evaluation pairs that share previously encountered artifacts.

5 Experimental Settings

We show ranking and verification task settings and baselines in this section. Full details about evaluation and baselines are available in Appendix §G.

Ranking task settings. Our evaluation focuses on link and attribute ranking (Figure 2), while prediction metrics are also reported as a byproduct of the point-wise scoring function.

Method	Transductive						Inductive					
	Link Prediction		Link Ranking				Link Prediction		Link Ranking			
	AP $\uparrow$	MCC $\uparrow$	MRR $\uparrow$	Hit ₅ $\uparrow$	R ₅ $\uparrow$	N ₅ $\uparrow$	AP $\uparrow$	MCC $\uparrow$	MRR $\uparrow$	Hit ₅ $\uparrow$	R ₅ $\uparrow$	N ₅ $\uparrow$
Heuristic												
Adamic-Adar	0.001	0.000	0.005	0.005	0.000	0.001	0.001	0.000	0.008	0.006	0.002	0.003
Matrix-Factorization	0.015	0.000	0.052	0.066	0.017	0.028	0.002	0.000	0.001	0.000	0.000	0.000
Katz	0.221	0.339	0.198	0.266	0.173	0.168	0.001	-0.001	0.004	0.002	0.002	0.002
Ranker-based												
Jina-v2-reranker	-	-	0.141	0.194	0.149	0.124	-	-	0.144	0.190	0.137	0.121
Jina-v2+graph	-	-	0.246	0.301	0.211	0.201	-	-	0.138	0.197	0.138	0.115
GNN-based w/ joint training												
BUDDY-style	0.233	0.196	0.281	0.378	0.260	0.237	0.039	0.131	0.163	0.212	0.138	0.126
NeoGNN-style	0.163	0.182	0.278	0.352	0.231	0.227	0.046	0.117	0.194	0.231	0.137	0.138
NCN	0.314	0.207	0.297	0.389	0.256	0.248	0.057	0.147	0.165	0.218	0.140	0.132
NCNC	0.352	0.241	0.309	0.398	0.266	0.259	0.063	0.162	0.173	0.229	0.159	0.145
GATv2Conv	0.202	0.213	0.299	0.379	0.263	0.249	0.043	0.128	0.181	0.235	0.158	0.148

Table 2: **Link prediction and ranking results.** AP denotes PR-AUC. MCC is computed using a threshold selected on the validation set. Hit₅, R₅, and N₅ denote Hits@5, Recall@5, and NDCG@5, respectively. Jina-v2-reranker is built for ranking, so link prediction entries are left empty. All GNN-based methods utilize joint training of links and attributes. More details are provided in Section §5 and Appendix §G.

Method	Transductive						Inductive					
	Attr Prediction		Attr Ranking				Attr Prediction		Attr Ranking			
	MAE $\downarrow$	RMSE $\downarrow$	τ $\uparrow$	ρ $\uparrow$	Hit ₁ $\uparrow$	N ₁ $\uparrow$	MAE $\downarrow$	RMSE $\downarrow$	τ $\uparrow$	ρ $\uparrow$	Hit ₁ $\uparrow$	N ₁ $\uparrow$
Heuristic												
Global mean	0.187	0.225	0.000	0.000	0.029	0.803	0.187	0.225	0.000	0.000	0.041	0.768
Model mean	0.178	0.220	0.206	0.233	0.399	0.886	0.205	0.240	0.000	0.000	0.041	0.768
Dataset mean	0.128	0.166	0.000	0.000	0.029	0.803	0.128	0.166	0.000	0.000	0.041	0.768
LLM-based												
GPT-5.2	0.137	0.187	0.367	0.413	0.519	0.923	0.138	0.187	0.383	0.431	0.495	0.911
GPT-5.2+graph	0.093	0.136	0.464	0.518	0.591	0.945	0.098	0.144	0.469	0.527	0.573	0.928
Qwen3-8B	0.201	0.263	0.174	0.193	0.404	0.873	0.202	0.262	0.186	0.209	0.413	0.883
Qwen3-8B+graph	0.131	0.181	0.207	0.246	0.452	0.898	0.121	0.173	0.205	0.232	0.395	0.876
GNN-based w/ joint training												
BUDDY-style	0.064	0.101	0.470	0.531	0.573	0.957	0.098	0.141	0.427	0.485	0.453	0.908
NeoGNN-style	0.064	0.100	0.483	0.549	0.542	0.946	0.095	0.135	0.362	0.423	0.418	0.911
NCN	0.064	0.101	0.447	0.508	0.520	0.950	0.097	0.138	0.448	0.515	0.498	0.919
NCNC	0.064	0.100	0.433	0.496	0.520	0.938	0.097	0.138	0.402	0.466	0.449	0.917
GATv2Conv	0.063	0.099	0.508	0.572	0.560	0.955	0.094	0.136	0.387	0.450	0.458	0.915

Table 3: **Attribute prediction and ranking results.** τ , ρ , Hit₁, and N₁ denote Kendall’s Tau, Spearman’s Rho, Hit@1, and NDCG@1, respectively. All GNN-based methods utilize joint training of links and attributes. More details are provided in Section §5 and Appendix §G.

We split the graph into train (80%) and test (20%) sets under two settings: (1) *Transductive*, where edges are split while all nodes remain visible during training; and (2) *Inductive*, where a subset of nodes is held out entirely to evaluate generalization to unseen artifacts. For attribute ranking, candidates are grouped by dataset and metric type.

Ranking baselines. We compare against three groups of baselines: (1) **Heuristics**: Adamic-Adar, Katz, and Matrix Factorization; (2) **LLMs/Rankers**: Jina-v2-reranker¹, GPT-5.2 (OpenAI, 2025), and Qwen3-8B (Yang et al., 2025); and (3) **GNNs**: GATv2Conv (Veličković et al., 2017), BUDDY-style (Chamberlain et al., 2022), NeoGNN-style (Yun et al., 2021), NCN (Wang et al., 2023), and NCNC (Wang et al., 2023). For LLMs and rerankers, we additionally evaluate *+graph* variants that include the target nodes’ 1-hop neighborhoods in the prompt. For a fair GNN comparison, all GNN methods use the same GATv2Conv encoder and differ only in their link-prediction decoders.

¹<https://huggingface.co/jinaai/jina-reranker-v2-base-multilingual>

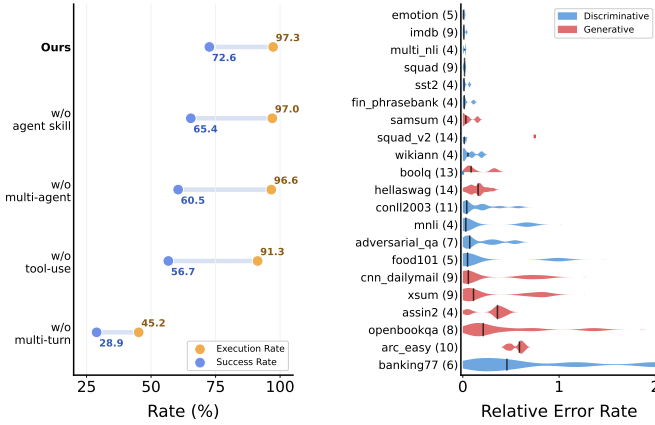


Figure 3: **Agentic verification performance under component ablations.** We ablate each component in our method. Execution rate measures whether the generated code runs successfully, and success rate measures whether the result is no worse than 80% of the ground-truth performance.

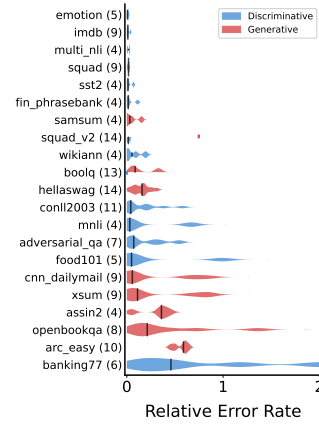


Figure 4: **Error distribution of reproduced verification results.** We show the error distribution across datasets in our reproduced evaluation. The number after each dataset name denotes the number of evaluated models, and discriminative and generative models are shown separately.

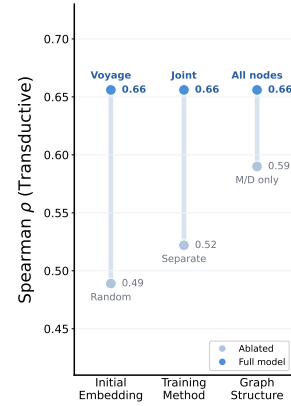


Figure 5: **Ablation study of the GNN-based ranker.** We study three factors: (1) encoder embedding initialization (Voyage vs. random), (2) training strategy (dual-head joint training vs. independent training), and (3) graph structure (model-dataset nodes only vs. the full graph).

Verification task settings. We construct a subset of the benchmark of 263 model–dataset pairs from the test set for automated evaluation reproduction. We filter pairs using three criteria: (1) prioritize popular models and datasets by download count; (2) exclude artifacts that exceed practical execution constraints; and (3) restrict evaluation to bounded metrics—accuracy, F1, BLEU, chrF, and rouge—to ensure consistent evaluation. For each pair, the agent must reproduce the officially reported metric through fully automated code execution.

Verification baselines. We evaluate the contribution of each agent component using four baselines: (1) **Agent-free**: generation without iterative execution feedback; (2) **ReAct agent**: iterative reasoning without HuggingFace API tools; (3) **Tool-use agent**: a single agent responsible for both planning and coding; and (4) **Tool-use multi-agent**: a planner–executor framework without self-evolving memory.

6 Experimental Results

GNN-based ranking models match or outperform LLM-based methods. Tables 2 and 3 show that specialized GNNs are highly competitive when graph structure is available. For transductive link ranking, NCNC achieves the best MRR of 0.309, outperforming the strongest reranker baseline, Jina-v2+graph (0.246). GNNs are also particularly strong for attribute prediction: GATv2Conv achieves an MAE of 0.063, compared with 0.093 for GPT-5.2+graph, while also obtaining the best Kendall’s τ and Spearman’s ρ . Overall, these results demonstrate the value of structural information in the artifact graph.

GNNs retain strong performance under inductive evaluation. Although performance generally decreases from transductive to inductive settings, GNNs remain competitive with LLM-based methods. For inductive attribute prediction, GATv2Conv achieves the best MAE of 0.094, outperforming GPT-5.2+graph at 0.098, while NeoGNN achieves the best RMSE of 0.135. For attribute ranking, NCN reaches a Kendall’s τ of 0.448 and Spearman’s ρ of 0.515, close to GPT-5.2+graph at 0.469 and 0.527. These results suggest that the learned structural representations generalize to unseen artifact combinations rather than memorizing.

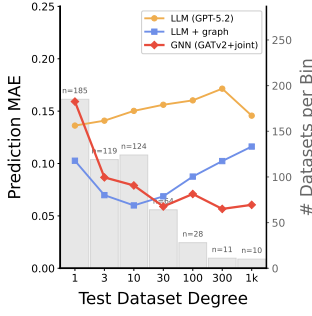


Figure 6: **Degree analysis of attribution prediction results.** We ablate on LLMs, LLMs with 1-hop neighborhood context, and GNN-based methods. We split the test set based on the node degrees of the datasets. Gray bars indicate the degree distribution of dataset nodes.

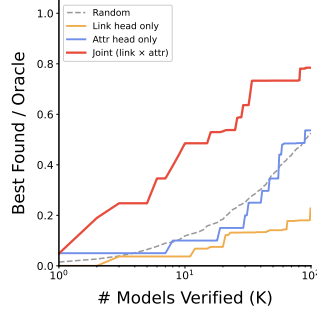


Figure 7: **Verification cost reduction via joint ranking.** We rank candidate models by different scoring functions and verify them in rank order. The y-axis shows the best performance found (normalized by the oracle) after verifying the top-K models (x-axis).

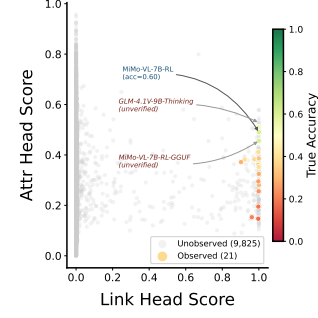


Figure 8: **Case study of model ranking.** We show an example ranking of all models on MathVision. Each dot represents a model, positioned by its link head score (x-axis) and attribute head score (y-axis). Gray dots denote models without existing reported evaluations.

ArtifactCoder reliably reproduces evaluations for widely used artifacts. Beyond prediction on the artifact graph, Figure 3 shows that ArtifactCoder achieves a 72.6% success rate across 263 evaluations. Multi-turn interaction is particularly important: removing it reduces success to 28.9%, while removing tool use lowers performance to 56.7%. Together, self-evolving memory, multi-agent coordination, and tool use substantially improve the reliability of automatic evaluation reproduction.

Link verification in the wild provides a challenging agent benchmark. Figure 4 reveals substantial variation across artifacts. ArtifactCoder achieves near-zero error on common datasets such as SQuAD and IMDb, but struggles with more complex or less frequently used artifacts. ARC-Easy exhibits consistently high relative error across models, while Banking77 frequently exceeds 0.5 and can reach 2.0. These results highlight in-the-wild artifact verification as a challenging setting for evaluating agent reasoning, debugging, and tool-use capabilities, particularly on complex and niche artifacts. As new artifacts continue to emerge, the benchmark can also naturally expand over time.

7 Discussions

Q1: Which factors support effective ranking in the artifact graph? The effectiveness of our GNN framework comes from both informative graph representations and joint training. As shown in Figure 5, stronger initial semantic embeddings provide the largest gain, increasing correlation from 0.49 to 0.66. Adding richer graph structure, including paper and codebase nodes, further improves performance from 0.59 to 0.66. To avoid data leakage, all metadata features are derived from LLM-summarized model cards with ground-truth labels removed. More detailed leakage analysis is in Appendix §F. Finally, shared-encoder joint training improves correlation from 0.52 to 0.66. Together, these results show that effective ranking depends on combining semantic node features, graph topology, and joint optimization.

Q2: Why can GNN-based methods outperform LLM-based methods in attribute prediction? Across all evaluations, the joint GATv2 model achieves an MAE of 0.062, outperforming GPT-5.2 alone (0.137) and GPT-5.2 with 1-hop graph context (0.093). Figure 6 shows that this advantage grows with node degree: GNN MAE decreases monotonically from 0.087 at degree 3–10 to 0.057 at degree 300–1,000 as richer neighborhoods become available. In contrast, LLM+graph degrades to 0.102–0.116 at high degree, where serializing large neighborhoods requires truncation due to context-window limits. At cold start (degree 1–3, (~34%) of datasets), however, both LLM variants outperform GNNs (MAE

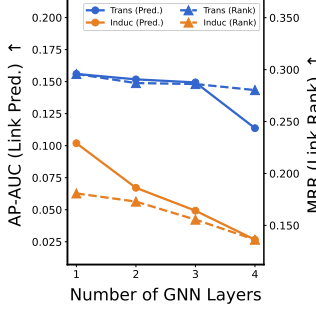


Figure 9: **Ablation study on GNN layer numbers (link ranking and prediction).** GATv2 as the backbone model. Both AP-AUC and MRR metrics are the higher the better.

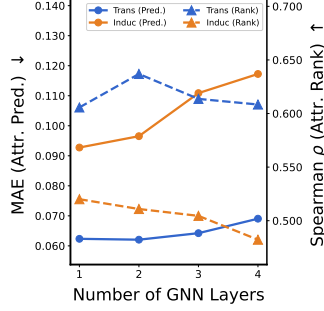


Figure 10: **Ablation study on GNN layer numbers (attribute ranking and prediction).** GATv2 as the backbone model. MAE is the lower the better while Spearman is the higher the better.

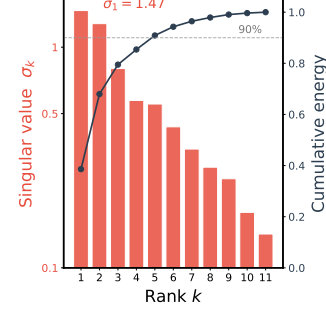


Figure 11: **Rank analysis of NLI accuracy matrix.** SVD applied to the double-centered 12×45 dataset $\times$ model accuracy matrix. Rank-5 recovers 90%+ of the residual variance.

0.103–0.136 vs. 0.159), because textual priors remain useful when graph evidence is sparse. This suggests a natural hybrid strategy: use LLM+graph for cold-start datasets and GNNs for well-connected artifacts.

Q3: How well does the ranking model reduce verification costs? The ranking model $\hat{f}(m, d)$ is designed to prioritize promising SOTA candidates and reduce verification cost. As shown in Figure 7, ranking by $\hat{f}(m, d)$ recovers 50% of existing SOTA results with only 10 verifications on average, compared with more than 60 for the biased attribution baseline. Figure 8 further shows a case study on MathVision, where the model ranks highly both variants of known strong models (e.g., MiMo-VL-7B-RL-GGUF²) and distinct model families (e.g., GLM-4.1V³), demonstrating its ability to generalize beyond model variants.

8 Ablation Studies

GNN decoders. Besides the bilinear decoder used in GATv2, we evaluate specialized link-prediction decoders, including NCN, NCNC, NeoGNN, and BUDDY, while keeping the GATv2 encoder fixed. These decoders exploit additional neighborhood structure and generally improve link prediction and ranking, with NCNC achieving the strongest transductive link-ranking performance. Their benefits are less consistent on attribute prediction and ranking, where the same node representations are shared across variants. This suggests that explicit structural priors are particularly effective for modeling link compatibility, while fine-grained attribute estimation depends more on the learned node representations.

GNN layer number. In Figures 9 and 10, we vary the number of GNN layers from 1 to 4. Both link- and attribute-level performance peaks at one or two layers and degrades thereafter, particularly on the inductive split: AP drops by half from (L=1) to (L=4), while attribute MAE increases by over 25%. This is consistent with over-smoothing around high-degree dataset hubs. Nevertheless, graph aggregation remains essential: removing it entirely (matrix factorization with text embeddings) reduces transductive link MRR from 0.30 to 0.01 and attribute ρ from 0.61 to 0.41. Thus, a shallow non-linear GNN is sufficient to capture the local collaborative-filtering signal without excessive over-smoothing.

9 End-to-end Case Study

We demonstrate ARTIFACTLINKER on Natural Language Inference (NLI), evaluating 45 candidate models across 12 representative NLI datasets (Williams et al., 2018; Bowman et al.,

²<https://huggingface.co/unsloth/MiMo-VL-7B-RL-GGUF>

³<https://huggingface.co/zai-org/GLM-4.1V-9B-Base>

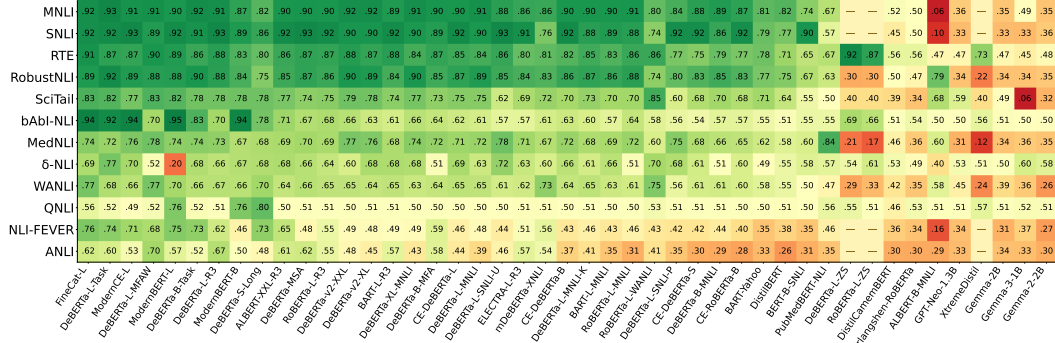


Figure 12: **Verified accuracy matrix for NLI tasks.** We show the accuracy verification results conducted by the ARTIFACTLINKER with 45 models and 12 NLI datasets. We use ST_SE split for RobustNLI. Cells filled with "-" are because these models are two-way pretrained models, while the evaluated datasets are 3-way NLI tasks. Therefore, these models are skipped for typical datasets. Models and datasets details are in Appendix SH.

2015; Nie et al., 2020; Conneau et al., 2018; Bentivogli et al., 2009; Wang et al., 2018) (Figure 12). We choose NLI because its rich collection of models, datasets, and published results provides a well-grounded setting for end-to-end evaluation. The case study illustrates the types of discoveries that ARTIFACTLINKER can extract from an artifact graph.

Q1: Can ARTIFACTLINKER discover unobserved SOTA results in the real world? We identify sileod/deberta-v3-large-tasksource-nli,⁴ a multi-task model whose card reports results on WNLI and MNLI but not SNLI. Our verification obtains an SNLI test accuracy of 0.9212—to our knowledge, the first reported score for this model and within 1pp of the leaderboard SOTA (0.931).⁵ On the less canonical pietrolesci/robust_nli (ST_SE),⁶ the same model reaches 0.920, improving over the previously reported best by 23pp. These results show that ARTIFACTLINKER can uncover strong, previously unreported evaluations.

Q2: Does the model–dataset evaluation matrix exhibit low-rank hidden structure? Applying SVD to the double-centered 12×45 accuracy matrix shows that three components explain 80% of the residual variance and five explain 91% (Figure 11), indicating an effective interaction rank of only 3–5. This helps explain why a shallow GNN is sufficient: much of the model–dataset interaction can be captured in a low-dimensional space. Consistent with this structure, ARTIFACTLINKER’s GNN-based factorization achieves 0.062 MAE over 5,839 transductive edges. The completed evaluation matrix also reveals non-monotonic model capabilities: newer models, including the Gemma family, often underperform older DeBERTa-based models on NLI tasks that the latter solve near-perfectly. Although similar phenomena have been studied in prior NLI work (Naik et al., 2018; Talman & Chatzikiriakidis, 2019; Bhargava et al., 2021; Delbari & Pilehvar, 2025), ARTIFACTLINKER recovers these patterns directly from large-scale evaluation traces, illustrating its ability to support automated empirical analysis.

10 Conclusion

In this paper, we introduce ARTIFACTBENCH, a challenging benchmark for SOTA discovery grounded in the HuggingFace ecosystem. We also present ARTIFACTLINKER, a two-stage rank-and-verify framework, together with an end-to-end Natural Language Inference case study demonstrating its practical potential in realistic discovery settings. Our results establish ARTIFACTBENCH as a meaningful testbed for automated scientific discovery and highlight the promise of rank-and-verify approaches for efficiently exploring large discovery spaces, while providing a foundation for future tasks and research in this area.

⁴<https://huggingface.co/sileod/deberta-v3-large-tasksource-nli>

⁵<https://nlp.stanford.edu/projects/snli/>

⁶https://huggingface.co/datasets/pietrolesci/robust_nli

References

- Adem Ait, Javier Luis Cánovas Izquierdo, and Jordi Cabot. On the suitability of hugging face hub for empirical studies. *ArXiv*, abs/2307.14841, 2023. URL <https://api.semanticscholar.org/CorpusId:260203268>.
- Joeran Beel, Min-Yen Kan, and Moritz Baumgart. Evaluating sakana’s ai scientist: Bold claims, mixed results, and a promising future? In *ACM SIGIR Forum*, volume 59, pp. 1–20. ACM New York, NY, USA, 2025.
- Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. The fifth pascal recognizing textual entailment challenge. *TAC*, 7(8):1, 2009.
- Prajwal Bhargava, Aleksandr Drozd, and Anna Rogers. Generalization in nli: Ways (not) to go beyond simple heuristics. In *Proceedings of the Second Workshop on Insights from Negative Results in NLP*, pp. 125–135, 2021.
- Ben Bogin, Kejuan Yang, Shashank Gupta, Kyle Richardson, Erin Bransom, Peter Clark, Ashish Sabharwal, and Tushar Khot. Super: Evaluating agents on setting up and executing tasks from research repositories. *Proceedings of EMNLP*, 2024.
- Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 632–642, 2015.
- Jonathan Bragg, Mike D’Arcy, Nishant Balepur, Dan Bareket, Bhavana Dalvi, Sergey Feldman, Dany Haddad, Jena D Hwang, Peter Jansen, Varsha Kishore, Kyle Richardson, Amanpreet Singh, Harshit Suarana, Aryen Tiktinsky, Rosni Vasu, Guy Wiener, and Chloe Anastasiades. Astabench: Rigorous benchmarking of ai agents with a scientific research suite. *Proceedings of ICLR*, 2026.
- Keith T Butler, Daniel W Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. Machine learning for molecular and materials science. *Nature*, 559(7715):547–555, 2018.
- Joel Castaño, Silverio Martínez-Fernández, Xavier Franch, and Justus Bogner. Analyzing the evolution and maintenance of ml models on hugging face. *2024 IEEE/ACM 21st International Conference on Mining Software Repositories (MSR)*, pp. 607–618, 2023. URL <https://api.semanticscholar.org/CorpusId:265351447>.
- Joel Castaño, Rafael Cabañas, Antonio Salmer’on, David Lo, and Silverio Mart’inez-Fern’andez. How do machine learning models change? *ArXiv*, abs/2411.09645, 2024. URL <https://api.semanticscholar.org/CorpusId:274023512>.
- Benjamin Paul Chamberlain, Sergey Shirobokov, Emanuele Rossi, Fabrizio Frasca, Thomas Markovich, Nils Hammerla, Michael M Bronstein, and Max Hansmire. Graph neural networks for link prediction with subgraph sketching. *arXiv preprint arXiv:2209.15486*, 2022.
- Qiaosheng Chen, Kaijia Huang, Xiao Zhou, Weiqing Luo, Yuanning Cui, and Gong Cheng. Benchmarking recommendation, classification, and tracing based on hugging face knowledge graph. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 3433–3443, 2025a.
- Qiaosheng Chen, Kaijia Huang, Xiaofang Zhou, Weiqing Luo, Yuanning Cui, and Gong Cheng. Benchmarking recommendation, classification, and tracing based on hugging face knowledge graph. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2025b. URL <https://api.semanticscholar.org/CorpusId:278886655>.
- Junyan Cheng, Peter Clark, and Kyle Richardson. Language modeling by language models. *Proceedings of NeurIPS*, 2025.

- Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2475–2485, 2018.
- Ned Cooper, Tiffanie N. Horne, Gillian R. Hayes, Courtney Heldreth, Michal Lahav, Jess Holbrook, and Lauren Wilcox. A systematic review and thematic analysis of community-collaborative approaches to computing research. *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, 2022. URL <https://api.semanticscholar.org/CorpusId:248419416>.
- Zahra Delbari and Mohammad Taher Pilehvar. Beyond accuracy: Revisiting out-of-distribution generalization in nli models. In *Proceedings of the 29th Conference on Computational Natural Language Learning*, pp. 557–570, 2025.
- Michael Färber and David Lamprecht. Linked papers with code: the latest in machine learning as an rdf knowledge graph. *arXiv preprint arXiv:2310.20475*, 2023.
- Sarah Gao and Andrew Gao. On the origin of llms: An evolutionary tree and graph for 15, 821 large language models. *ArXiv*, abs/2307.09793, 2023. URL <https://api.semanticscholar.org/CorpusId:259983028>.
- R. Heumüller, Sebastian Nielebock, J. Krüger, and F. Ortmeier. Publish or perish, but do not forget your software artifacts. *Empirical Software Engineering*, 25:4585 – 4616, 2020. URL <https://api.semanticscholar.org/CorpusId:220070385>.
- Peter Alexander Jansen, Marc-Alexandre Côté, Tushar Khot, Erin Bransom, Bhavana Dalvi, Bodhisattwa Prasad Majumder, Oyvind Tafjord, and Peter Clark. Discoveryworld: A virtual environment for developing and evaluating automated scientific discovery agents. *ArXiv*, abs/2406.06769, 2024. URL <https://api.semanticscholar.org/CorpusId:270380311>.
- Peter Alexander Jansen, Oyvind Tafjord, Marissa Radensky, Pao Siangliulue, Tom Hope, Bhavana Dalvi, Bodhisattwa Prasad Majumder, D. S. Weld, and Peter Clark. Codescientist: End-to-end semi-automated scientific discovery with code-based experimentation. *ArXiv*, abs/2503.22708, 2025. URL <https://api.semanticscholar.org/CorpusId:277451644>.
- Seth Johnson, F. Samsel, G. Abram, Daniel L. Olson, Andrew J. Solis, Bridger Herman, P. Wolfram, C. Lenglet, and Daniel F. Keefe. Artifact-based rendering: Harnessing natural and traditional visual media for more expressive and engaging 3d visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 26:492–502, 2019. URL <https://api.semanticscholar.org/CorpusId:199001020>.
- Gyeongwon James Kim, Alex Wilf, Louis philippe Morency, and Daniel Fried. From reproduction to replication: Evaluating research agents with progressive code masking. *ArXiv*, abs/2506.19724, 2025. URL <https://api.semanticscholar.org/CorpusId:280000499>.
- Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Benjamin Laufer, Hamidah Oderinwale, and Jon Kleinberg. Anatomy of a machine learning ecosystem: 2 million models on hugging face. *ArXiv*, abs/2508.06811, 2025. URL <https://api.semanticscholar.org/CorpusId:280566415>.
- C. J. Lissa, A. Brandmaier, Loek Brinkman, Anna-Lena Lamprecht, Aaron Peikert, Marijn E. Struiksmā, and Barbara M. I. Vreede. Worcs: A workflow for open reproducible code in science. *Data Sci.*, 4:29–49, 2020. URL <https://api.semanticscholar.org/CorpusId:234357246>.
- Chris Lu, Cong Lu, R. T. Lange, J. Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery. *ArXiv*, abs/2408.06292, 2024. URL <https://api.semanticscholar.org/CorpusId:271854887>.

- T. Marić, Dennis Gläser, Jan-Patrick Lehr, Ioannis Papagiannidis, B. Lambie, Christian H. Bischof, and Dieter Bothe. A pragmatic workflow for research software engineering in computational science. *ArXiv*, abs/2310.00960, 2023. URL <https://api.semanticscholar.org/CorpusId:263605602>.
- Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 2340–2353, 2018.
- Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. Adversarial nli: A new benchmark for natural language understanding. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 4885–4901, 2020.
- OpenAI. Introducing OpenAI gpt-5.2, April 2025. URL <https://openai.com/index/introducing-gpt-5-2/>. Accessed: 2026-01-06.
- Mohammad Shahedur Rahman, Peng Gao, and Yuede Ji. Hugginggraph: Understanding the supply chain of llm ecosystem. *ArXiv*, abs/2507.14240, 2025a. URL <https://api.semanticscholar.org/CorpusId:280270972>.
- Mohammad Shahedur Rahman, Peng Gao, and Yuede Ji. Hugginggraph: Understanding the supply chain of llm ecosystem. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 5997–6005, 2025b.
- Marwin HS Segler, Mike Preuss, and Mark P Waller. Planning chemical syntheses with deep neural networks and symbolic ai. *Nature*, 555(7698):604–610, 2018.
- Minju Seo, Jinheon Baek, Seongyun Lee, and Sung Ju Hwang. Paper2code: Automating code generation from scientific papers in machine learning. *ArXiv*, abs/2504.17192, 2025. URL <https://api.semanticscholar.org/CorpusId:278033490>.
- D. Serrano, F. C. Luciano, B. J. Anaya, Baris Ongoren, Aytug Kara, Gracia Molina, Bianca I Ramirez, Sergio A Sánchez-Guiraes, Jesus A. Simon, Greta Tomietto, Chrysi Rapti, Helga K. Ruiz, Satyavati Rawat, Dinesh Kumar, and A. Lalatsa. Artificial intelligence (ai) applications in drug discovery and drug delivery: Revolutionizing personalized medicine. *Pharmaceutics*, 16, 2024. URL <https://api.semanticscholar.org/CorpusId:273366814>.
- Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Y. Zhuang. Taskbench: Benchmarking large language models for task automation. *ArXiv*, abs/2311.18760, 2023. URL <https://api.semanticscholar.org/CorpusId:265506220>.
- Zachary S. Siegel, Sayash Kapoor, Nitya Nagdir, Benedikt Stroebl, and Arvind Narayanan. Core-bench: Fostering the credibility of published research through a computational reproducibility agent benchmark. *Trans. Mach. Learn. Res.*, 2024, 2024. URL <https://arxiv.org/pdf/2409.11363.pdf>.
- Kanishka Silva, Marcel R. Ackermann, Heike Fliegl, G. Gesese, Fidan Limani, Philipp Mayr, Peter Mutschke, A. Oelen, Muhammad Asif Suryani, Sharmila Upadhyaya, Benjamin Zapilko, Harald Sack, and Stefan Dietze. Research knowledge graphs in nfdr4datascience: Key activities, achievements, and future directions. *ArXiv*, abs/2508.02300, 2025. URL <https://api.semanticscholar.org/CorpusId:280421789>.
- Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, et al. Paperbench: Evaluating ai’s ability to replicate ai research. *arXiv preprint arXiv:2504.01848*, 2025.
- Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackermann, et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702, 2020.
- Aarne Talman and Stergios Chatzikyriakidis. Testing the generalization power of neural network models across nli benchmarks. In *Proceedings of the 2019 ACL workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pp. 85–94, 2019.

- R. Urbanowicz, Robert F. Zhang, Yuhan Cui, and Pranshu Suri. Streamline: A simple, transparent, end-to-end automated machine learning pipeline facilitating data analysis and algorithm comparison. In *Genetic Programming Theory and Practice*, 2022. URL <https://api.semanticscholar.org/CorpusId:250048789>.
- Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
- A. Vişan and I. Neguţ. Integrating artificial intelligence for drug discovery in the context of revolutionizing drug delivery. *Life*, 14, 2024. URL <https://api.semanticscholar.org/CorpusId:267570328>.
- Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pp. 353–355, 2018.
- Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*, 2024.
- Xiyuan Wang, Haotong Yang, and Muhan Zhang. Neural common neighbor with completion for link prediction. *arXiv preprint arXiv:2302.00890*, 2023.
- Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long papers)*, pp. 1112–1122, 2018.
- Yanzheng Xiang, Hanqi Yan, Shuyin Ouyang, Lin Gui, and Yulan He. Scireplicate-bench: Benchmarking llms in agent-driven algorithmic reproduction from research papers. *ArXiv*, abs/2504.00255, 2025. URL <https://api.semanticscholar.org/CorpusId:277467991>.
- Tian Xie and Jeffrey C Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical review letters*, 120(14):145301, 2018.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.
- Xinyu Yang, Weixin Liang, and James Zou. Navigating dataset documentations in ai: A large-scale analysis of dataset cards on hugging face. *ArXiv*, abs/2401.13822, 2024. URL <https://api.semanticscholar.org/CorpusId:267212153>.
- Yujie You, Xin Lai, Ying Pan, Huiru Zheng, J. Vera, Suran Liu, Senyi Deng, and Le Zhang. Artificial intelligence in cancer target identification and drug discovery. *Signal Transduction and Targeted Therapy*, 7, 2022. URL <https://doi.org/10.1038/s41392-022-00994-0>.
- Haofei Yu, Keyang Xuan, Fenghai Li, Kunlun Zhu, Zijie Lei, Jiaxun Zhang, Ziheng Qi, Kyle Richardson, and Jiaxuan You. Tinscientist: An interactive, extensible, and controllable framework for building research agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 558–590, 2025.
- Seongjun Yun, Seoyoon Kim, Junhyun Lee, Jaewoo Kang, and Hyunwoo J Kim. Neo-gnns: Neighborhood overlap-aware graph neural networks for link prediction. *Advances in Neural Information Processing Systems*, 34:13683–13694, 2021.

A Visualization of Artifact Graph

Figure 13 shows the full-size visualization of all nodes and edges we included in our collected artifact graph.

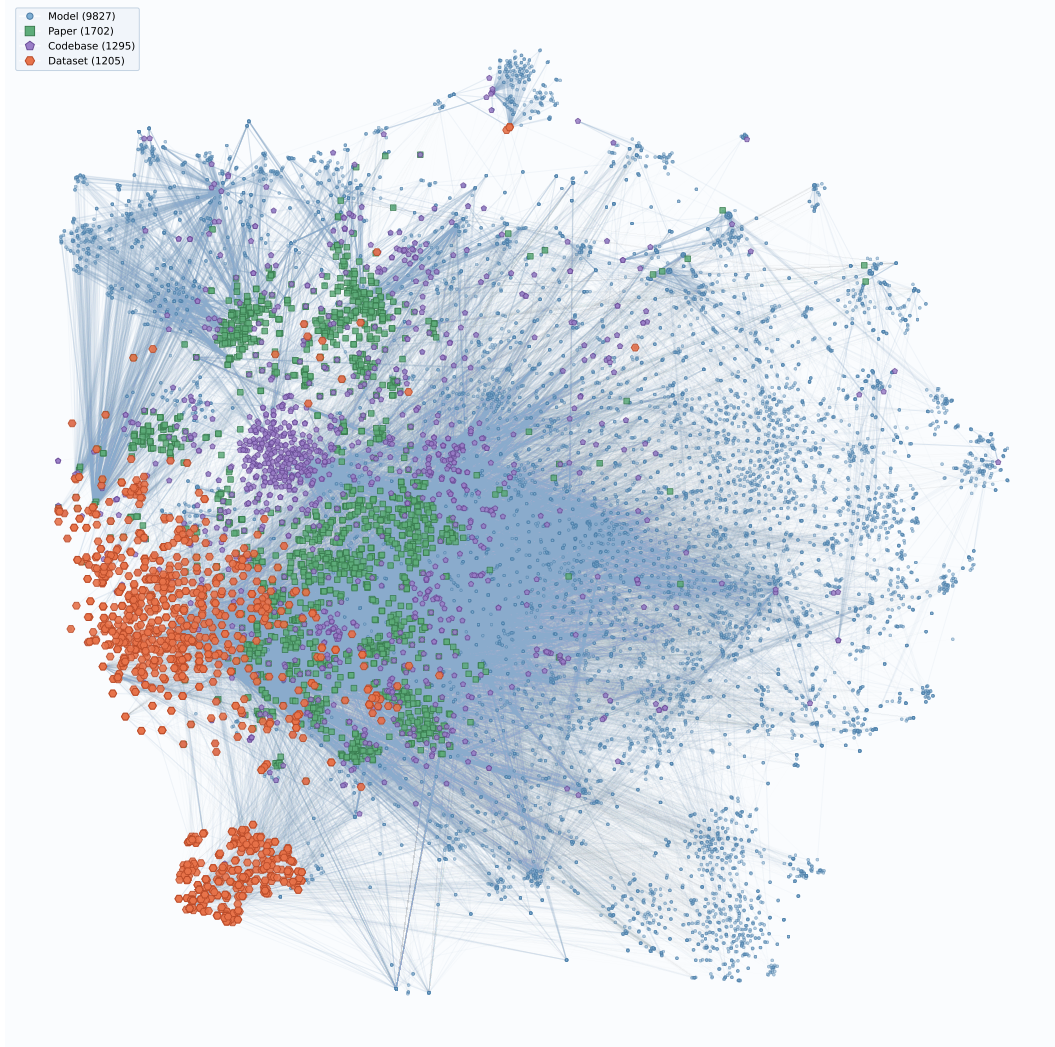


Figure 13: Visualization of collected artifact graph.

B Limitations

Computational considerations in verification While our two-stage framework effectively reduces the search space through candidate filtering in the prediction phase, the verification stage requires actual code execution for validation. For certain large-scale experiments or resource-intensive datasets, full reproduction may require non-trivial computational costs depending on available hardware resources. We note that our current implementation successfully handles the majority of common datasets, though scaling to extremely high-throughput scenarios across diverse hardware environments remains an interesting direction for future optimization.

Evaluation metric coverage Our framework currently emphasizes objective, quantitative metrics such as Accuracy, F1, and Exact Match, which are widely reported in model documentation and amenable to automatic extraction. These metrics cover a substantial portion

of commonly used evaluation schemes in NLP. While our approach could potentially be extended to incorporate tasks involving human evaluation or qualitative assessment, we leave the integration of such subjective metrics as a natural extension for future work, as they would require additional methodological considerations for automated processing.

C Ethics Consideration

Our work focuses on building an automatic discovery framework over open-source artifacts available on HuggingFace. All experiments are conducted exclusively on publicly released models and datasets that are freely accessible to the research community. We do not introduce any new human or sensitive data, nor do we attempt to deanonymize or misuse existing artifacts. The goal of our framework is to advance automated, reproducible, and scalable scientific discovery, and to help researchers more efficiently identify promising model–dataset interactions. Nevertheless, we acknowledge that automated benchmarking may propagate existing biases and limitations present in the underlying models and datasets. To mitigate this, we emphasize transparency in data collection and reproducibility in our verification pipeline. We encourage the community to view our work as a step toward building more reliable and responsible auto-discovery systems, rather than a replacement for human oversight.

D Potential Risks

Metadata quality considerations Our prediction stage uses metadata from the HuggingFace community, including model descriptions, architectural specifications, and training configurations. As with any crowdsourced platform, there may occasionally be instances of incomplete or imprecise documentation that could affect initial ranking predictions.

Performance-oriented discovery scope ARTIFACTLINKER is designed to identify high-performing model-dataset combinations based on standard evaluation metrics. As with any automated performance discovery tool, users should exercise standard research practices by conducting appropriate safety and ethical assessments before deploying discovered models in production environments, particularly for applications involving user-facing content generation. We view our framework as a research tool that augments human decision-making rather than replacing it. Users retain full responsibility for evaluating whether discovered models meet the safety, fairness, and ethical requirements of their specific use cases, and we encourage comprehensive evaluation beyond performance metrics alone.

E Scientific Artifacts

E.1 Data License

All data in ARTIFACTBENCH are derived from publicly available resources on the HuggingFace Hub, including models, datasets, papers, codebases, and their associated metadata. These resources are released under heterogeneous public licenses, such as Apache 2.0, MIT, and CC-BY. We release ARTIFACTBENCH under the Open Database License (ODbL), which permits use, sharing, and modification of the database, while requiring proper attribution and that derivative databases be released under the same license. We will retain license metadata for the original artifacts whenever available and encourage users to comply with the licenses of the underlying resources.

E.2 Model License

Our work uses several foundation models for inference, including chatgpt-4o-latest, o3-2025-04-16, voyage-3, and Qwen3-8B. We access chatgpt-4o-latest and o3-2025-04-16 through the official OpenAI API, and we use voyage-3 through the official VoyageAI inference API. These API-based models are closed-source and governed by their respective

proprietary terms of use. We use them only for academic and non-commercial research purposes, with inputs derived from publicly available data. We do not modify these models and use them as provided through their public APIs. Qwen3-8B is released under the Apache 2.0 license and is used for inference in accordance with its license terms.

E.3 Data Usage

The data used in this study consist of publicly available scientific artifact information collected from the HuggingFace Hub, including model specifications, dataset descriptions, paper links, codebase metadata, and evaluation results. Our collection focuses exclusively on scientific artifacts and does not intentionally include personally identifiable information. All resources are collected from publicly accessible pages and used for research purposes in accordance with the HuggingFace Hub Terms of Service and the licenses associated with the original artifacts.

F Artifact Graph Data Details

This section provides additional details on the construction of our artifact graph and evaluates whether textual artifact metadata introduces shortcut signals into downstream prediction.

F.1 Graph Construction Pipeline

HuggingFace artifact crawling. We begin from models and datasets on the HuggingFace Hub and collect their associated metadata and documentation, including model cards, dataset cards, repository identifiers, tags, and referenced artifacts. We retain only models and datasets with at least 1,000 monthly downloads, as low-usage artifacts often provide sparse or insufficiently validated evaluation evidence, making reported results more difficult to reliably verify or reproduce. We retain canonical HuggingFace identifiers whenever available so that each artifact can be mapped back to its original Hub resource. This filtering focuses the graph on widely used and sufficiently documented artifacts while reducing noise from incomplete or low-quality entries.

Evaluation record extraction. The primary supervision in our graph comes from evaluation results reported in HuggingFace model and dataset cards. We parse evaluation tables and textual result sections into structured records of the form

$$(m, d, \text{metric}, \text{score}),$$

where m denotes a model, d a dataset or benchmark, and the remaining fields specify the reported metric and score. Most evaluation records are directly extracted from HuggingFace cards. When an entry is incomplete or ambiguous, we use a search-augmented LLM agent to retrieve supporting evidence from associated public sources, such as arXiv papers and GitHub repositories, and resolve the corresponding model, dataset, metric, and score. Multiple metrics for the same model–dataset pair are retained as separate evaluation records. We normalize artifact and metric names where possible and discard records that cannot be reliably resolved.

Contextual enrichment and filtering. We further enrich the graph with related arXiv papers, GitHub repositories, and model–model lineage relations such as fine-tuning. These relations are primarily obtained from references in artifact cards; when explicit links are unavailable, the search-augmented LLM agent is used to identify the corresponding public resources. We remove isolated artifacts (artifact without any edges), entries without usable descriptions, unresolved references, and low-quality metadata, and merge duplicate references that resolve to the same artifact. After filtering, the final graph contains 14,053 nodes and 51,337 edges. A breakdown by node and edge type is shown in Figure 1.

Backbone	AP				MCC			
	L0	L1	L2	L3	L0	L1	L2	L3
BUDDY	0.215	0.223	0.208	0.226	0.234	0.226	0.228	0.234
NeoGNN	0.210	0.227	0.208	0.209	0.208	0.202	0.210	0.195
NCN	0.283	0.292	0.304	0.300	0.214	0.214	0.201	0.216
NCNC	0.260	0.269	0.289	0.275	0.248	0.256	0.248	0.251
GATv2	0.149	0.142	0.149	0.146	0.247	0.254	0.252	0.252

Table 4: **Metadata masking analysis for link prediction.** L0–L3 progressively remove potential textual shortcut signals. Performance remains stable across masking levels. We use 3-layer GATv2 as the backbone.

Backbone	R^2				Hits@1			
	L0	L1	L2	L3	L0	L1	L2	L3
BUDDY	0.799	0.797	0.799	0.800	0.573	0.533	0.596	0.547
NeoGNN	0.794	0.790	0.793	0.801	0.569	0.573	0.556	0.547
NCN	0.788	0.797	0.798	0.798	0.533	0.578	0.560	0.542
NCNC	0.793	0.799	0.794	0.794	0.578	0.560	0.569	0.556
GATv2	0.796	0.797	0.798	0.793	0.578	0.600	0.569	0.596

Table 5: **Metadata masking analysis for attribute prediction.** Both R^2 and Hits@1 remain stable under progressively stronger masking. We use 3-layer GATv2 as the backbone.

Metadata processing. The artifact graph requires more semantic information for link prediction and verification. We use an LLM to summarize extracted model and dataset cards, paper abstracts, and repository descriptions into concise textual node representations. These representations are used to initialize GNN node features and as textual context for LLM-based baselines. All quantitative evaluation information, including metric values and benchmark scores, is removed before constructing these representations to prevent direct target leakage.

Construction quality audit. To assess the reliability of the graph construction pipeline, we randomly sample 100 model–dataset evaluation edges and manually verify each extracted (model, dataset, metric, score) record against publicly available evidence. We additionally check whether the reported evaluation protocol is consistent with the recorded result. Among the sampled edges, 96 are verified, with the recorded score matching the supporting evidence within a 2% relative tolerance, while the remaining 4 contain extraction or protocol inconsistencies.

F.2 Metadata Leakage Analysis

Leakage level settings. Removing numerical evaluation scores alone may not eliminate all potential shortcut signals. Artifact metadata may still reveal dataset names, task names, training data, node identities, or qualitative leaderboard claims correlated with evaluation links. We therefore conduct a masking analysis with four increasingly aggressive metadata levels: **L0**, score-redacted metadata; **L1**, additionally masking dataset names; **L2**, additionally masking task-family information; and **L3**, additionally masking node-identifying entities and leaderboard-style phrases.

Leakage evaluation. Table 4 and 5 reports link-prediction performance. Across all GNN backbones, AP and MCC remain largely stable from L0 to L3, showing no systematic degradation under increasingly aggressive masking. A similar pattern holds for attribute prediction (Table 5). Both R^2 and Hits@1 remain stable across masking levels, suggesting that score prediction is likewise not driven primarily by directly identifiable benchmark

metadata. Text-based LLM baselines are more sensitive to masking: link-prediction AP decreases from 0.820 under L1 to 0.738 under L3, while MCC decreases from 0.615 to 0.419. This confirms that textual metadata can contain shortcut signals. In contrast, the stability of the GNN results indicates that our main graph-based conclusions are not primarily driven by benchmark-name, node-identity, or leaderboard leakage.

G Experimental Details

G.1 GNN Architecture Details

ARTIFACTLINKER uses a common encoder–decoder architecture across all GNN variants. Specifically, every variant shares the same GATv2 graph encoder and attribute decoder, while differing in the link decoder used to score candidate model–dataset pairs. The default GATv2 variant uses a bilinear decoder; NCN and NCNC replace it with their corresponding common-neighbor-based decoders. For NeoGNN and BUDDY, we implement structure-aware decoders inspired by their respective structural modeling strategies, rather than reproducing the full original architectures. The attribute prediction head is shared across all variants and operates on the same node representations produced by the GATv2 encoder.

Shared GATv2 encoder. The encoder stacks three GATv2 message-passing layers, each followed by GraphNorm, a PReLU activation, and feature dropout ($p=0.2$); residual connections are added whenever the input and output dimensions align. Representations from different depths are aggregated using JumpingKnowledge (concatenation followed by a linear projection). The first GATv2 layer projects the 1024-dimensional Voyage text embeddings to 128×8 attention heads, intermediate layers preserve the multi-head representation, and the final layer produces a 128-dimensional node embedding. Unless noted otherwise, all variants therefore use the same single-layer encoder, hidden width 128, 8 attention heads, and 1024-dimensional input features.

GATv2 (GATv2 + bilinear decoder). Our default GATv2 variant uses a bilinear link decoder applied to the two encoded node representations. Together with the shared regression attribute head, this model contains approximately 4.80M parameters.

NCN (GATv2 + NCN decoder). The NCN variant keeps the GATv2 encoder unchanged and replaces the bilinear link decoder with a Neural Common Neighbor (NCN) decoder. For each candidate model–dataset pair, the decoder incorporates the pooled representations of their common neighbors together with the pairwise node representations, providing an explicit first-order structural signal. This variant contains approximately 4.87M parameters.

NCNC (GATv2 + NCNC decoder). NCNC uses the same GATv2 encoder and extends the NCN link decoder with a completion module that estimates virtual common-neighbor information for candidate pairs with sparse observed neighborhoods. The additional completion module increases the model size to approximately 4.93M parameters.

NeoGNN-style (GATv2 + structural-overlap decoder). Rather than reproducing the full NeoGNN architecture (Yun et al., 2021), we adopt its core idea of augmenting learned node representations with explicit neighborhood-overlap signals. Our NeoGNN-style decoder incorporates structural features including common neighbors, Adamic–Adar, and resource allocation statistics, which are fused with the GATv2 node representations before link scoring. This variant contains approximately 4.87M parameters.

BUDDY-style (GATv2 + sketch-based decoder). Similarly, our BUDDY-style variant does not reproduce the complete original BUDDY architecture (Chamberlain et al., 2022). Instead, it adopts its sketch-based structural modeling principle by augmenting the GATv2 representations with compact neighborhood sketch features that approximate local subgraph structure without explicit subgraph extraction for each candidate pair. This model contains approximately 4.87M parameters.

Joint training. All variants share the same attribute regression decoder and are trained jointly using the link and attribute objectives, with $\lambda_{\text{attr}} = 5$. Consequently, the comparison primarily isolates the effect of different link-side structural inductive biases while keeping the underlying graph encoder and attribute predictor fixed. We refer to the latter two variants as NeoGNN-style and BUDDY-style to emphasize that they instantiate the corresponding structural ideas within our shared GATv2 encoder-decoder framework rather than serving as exact reproductions of the original models.

G.2 Training Details

Our primary ARTIFACTLINKER experiments jointly train both decoder heads on a single shared encoder; the link and attribute objectives are optimized together so the encoder learns representations useful for both tasks.

Negative sampling. Positive examples are the observed (model, dataset, metric) triples in the training split. For every positive we sample 2 negative (model, dataset) pairs uniformly from pairs absent in all splits (negative sampling ratio 2). Negatives enter only the link term; the attribute term is computed on positive edges alone.

Training objective. We minimize $\mathcal{L} = \mathcal{L}_{\text{link}} + 5.0 \cdot \mathcal{L}_{\text{attr}}$. The link term $\mathcal{L}_{\text{link}}$ is a class-balanced binary cross-entropy: the mean of a positive-only and a negative-only BCE, so positives and negatives contribute equally regardless of the 1:2 sampling ratio. The attribute term $\mathcal{L}_{\text{attr}}$ is a mean-squared error on positive edges. The attribute head is *link-conditioned*: it receives the predicted link logit together with the pooled pair representation, rather than scoring the pair in isolation.

Attribute target. For attribute prediction, we use mean squared error (MSE) as the regression loss. Targets $y \in [0, 1]$ are clamped to $[10^{-7}, 1 - 10^{-7}]$ and mapped to logit space via $\text{logit}(y) = \log \frac{y}{1-y}$. We then minimize the MSE between the predicted raw attribute logits and the logit-transformed targets. At evaluation, the predicted logits are clipped to $[-10, 10]$ before applying a sigmoid to recover metric values in $[0, 1]$.

Optimization. We use Adam (Kingma & Ba, 2014) with learning rate 2×10^{-3} and weight decay 10^{-5} , under a cosine-annealing schedule that decays the learning rate to 10^{-5} over 1,500 epochs.

Model selection. Because the available training data size is limited, we do not further partition the training data to construct a separate validation set. Instead, all models are trained for a fixed budget of 1,500 optimization steps, and we report the checkpoint from the final training step without any form of model selection. The same checkpoint is used across all downstream tasks and evaluation metrics on test-set performance. All methods are evaluated using the same splits and candidate construction, so the GNN, heuristic, reranker, and LLM-based methods are directly comparable within each task.

Link task evaluation data. For link prediction and link ranking, we evaluate model-dataset edges under two settings. The *transductive* setting uses an edge-level split with `test_ratio=0.2` and seed 42. The *inductive* setting uses a disjoint model partition, where test models are unseen during training. In both settings, evaluation negatives are constructed by full enumeration rather than subsampling of models: any model-dataset pair that is not an observed positive in any split is treated as a negative candidate. This yields approximately 5.3M evaluation pairs in the transductive link-prediction setting, with positive prevalence around 0.1%.

Attribute task evaluation data. Attribute prediction and attribute ranking are evaluated only on test positive edges with valid numeric metric metadata. Ground-truth values are obtained from the normalized edge metadata, where all numeric targets are scaled to $[0, 1]$. The scalar target is selected differently for the two attribute tasks. For attribute prediction,

we select one metric per edge: the alphabetically first numeric metric available for that edge. Edges without any numeric metric are excluded. For attribute ranking, we select one metric per dataset: the most frequent numeric metric among that dataset’s test edges. Only edges containing the selected metric are retained, and datasets are excluded if they have fewer than two valid edges or if all target values are identical. Therefore, attribute prediction and attribute ranking are evaluated on related but not identical subsets of test edges.

Link verification data. For link verification, we construct a curated benchmark of 263 model–dataset pairs with sufficiently well-specified evaluation protocols and computationally feasible artifacts. We acknowledge that this selection introduces a bias toward popular and easier-to-execute artifacts; however, the purpose of this benchmark is to enable controlled comparison of verification agents rather than to estimate success over arbitrary real-world artifacts. Importantly, even under this relatively favorable setting, automatic reproduction remains challenging: our full agent reproduces only 72.6% of the pairs, with substantially lower success under ablations. We further evaluate robustness in a less curated setting by randomly sampling 50 model–dataset edges without the feasibility filters. In this in-the-wild stress test, only 5 of 50 pairs execute successfully, and only 1 reproduces the reported score within 20% relative error. Common failures arise from quantized or very large models, nonstandard loading procedures, and incomplete evaluation documentation. Together, these results suggest that the curated 263-pair benchmark already presents a nontrivial verification challenge, while fully automatic verification of arbitrary artifacts remains substantially more difficult.

Link prediction setting. Link prediction is evaluated over the full pool of test positives and fully enumerated negatives. This setting measures whether a method can distinguish observed model–dataset links from all unobserved pairs under extreme class imbalance. MCC is computed using a fixed decision threshold; the joint GNN uses a sigmoid threshold of 0.5, while heuristic baselines use 0.9.

Link ranking setting. Link ranking is evaluated independently for each test dataset. The candidate set contains the dataset’s positive test models together with all other models that are not known positives for that dataset across train and test. This creates approximately 10^4 candidates per dataset, again with full negatives and no sampling. This setting measures whether a method can rank true model–dataset links above all plausible alternatives for the same dataset.

Attribute prediction setting. Attribute prediction is evaluated on the per-edge scalar targets. The model predicts an attribute logit $\hat{\ell}$, which is converted to a bounded score by

$$\hat{y} = \sigma(\text{clip}(\hat{\ell}, -10, 10)).$$

The prediction is compared against the normalized ground-truth value $y \in [0, 1]$. This setting measures whether a method can estimate the expected evaluation score of a known model–dataset edge.

Attribute ranking setting. Attribute ranking is evaluated at the dataset level using the selected metric for each qualifying dataset. This setting measures whether a method can rank models by their relative performance on the same dataset, rather than merely predicting calibrated scalar scores across heterogeneous metrics. The NDCG@1 reported for this task uses a continuous top-1 regret-ratio form,

$$\text{NDCG@1} = \frac{y_{\text{top-1}}}{\max_i y_i},$$

which differs from the binary-relevance NDCG used in link ranking. Because this metric is saturated, simple mean-based baselines can already obtain high scores.

Link verification data

G.3 Compute and Budget

The GNN-based ARTIFACTLINKER checkpoint is ~ 5 M parameters (< 100 MB on disk). GNN training runs in < 1 hour per configuration on a single NVIDIA A100. Node text embeddings are computed once via the Voyage AI voyage-3 endpoint (dimension 1024) and cached; embedding the 14,053-node graph is a one-time API charge under \$20. LLM-based baselines (verification, prediction, ranking) over the full NLI evaluation suite total $< \$1,000$. Closed-source baseline model sizes are not publicly disclosed.

H Details of Case Study

Dataset name We use short names for dataset identifiers in figures and tables. Table 6 lists the correspondence between each dataset short name and its full dataset ID.

Model name To improve readability, we use short names for model identifiers in figures and tables. Table 7 lists the correspondence between each short name and its full HuggingFace model ID.

Table 6: Mapping between NLI dataset short names and HuggingFace dataset identifiers.

Short name	Full dataset ID
SNLI	stanfordnlp/snli
MNLI	nyu-mll/multi_nli
RTE	SetFit/rte
QNLI	SetFit/qnli
ANLI	facebook/anli
SciTail	allenai/scitail
WANLI	alisawuffles/WANLI
MedNLI	araag2/MedNLI
bAbI-NLI	tasksource/babi_nli
δ -NLI	tasksource/defeasible-nli
NLI-FEVER	pietrolesci/nli_fever
RobustNLI-ST-SE	pietrolesci/robust_nli (ST_SE split)

Table 7: Mapping between model short names and full HuggingFace model identifiers (all 45 evaluated models).

Short name	Full model ID
<i>Zero-shot NLI</i>	
DeBERTa-L-ZS	MoritzLaurer/deberta-v3-large-zeroshot-v2.0
RoBERTa-L-ZS	MoritzLaurer/roberta-large-zeroshot-v2.0-c
XtremeDistil	MoritzLaurer/xtremedistil-l6-h256-zeroshot-v1.1-all-33
mDeBERTa-XNLI	MoritzLaurer/mDeBERTa-v3-base-xnli-multilingual-nli-2mil7
DeBERTa-L-MFAW	MoritzLaurer/DeBERTa-v3-large-mnli-fever-anli-ling-wanli
DeBERTa-B-MFA	MoritzLaurer/DeBERTa-v3-base-mnli-fever-anli
<i>Multi-task NLI</i>	
DeBERTa-L-Task	sileod/deberta-v3-large-tasksource-nli
DeBERTa-B-Task	sileod/deberta-v3-base-tasksource-nli
ModernBERT-L	tasksource/ModernBERT-large-nli
ModernBERT-B	tasksource/ModernBERT-base-nli
DeBERTa-S-Long	tasksource/deberta-small-long-nli
FineCat-L	dleemiller/finecat-nli-l
ModernCE-L	dleemiller/ModernCE-large-nli
<i>SNLI+MNLI+FEVER+ANLI</i>	
ALBERT-XXL-R3	ynie/albert-xxlarge-v2-snli_mnli_fever_anli_R1_R2_R3-nli
RoBERTa-L-R3	ynie/roberta-large-snli_mnli_fever_anli_R1_R2_R3-nli
BART-L-R3	ynie/bart-large-snli_mnli_fever_anli_R1_R2_R3-nli
ELECTRA-L-R3	ynie/electra-large-discriminator-snli_mnli_fever_anli_R1_R2_R3-nli
DeBERTa-L-R3	Joelzhang/deberta-v3-large-snli_mnli_fever_anli_R1_R2_R3-nli
DeBERTa-MSA	NDugar/debertav3-mnli-snli-anli
<i>Cross-encoder NLI</i>	
CE-DeBERTa-L	cross-encoder/nli-deberta-v3-large
CE-DeBERTa-B	cross-encoder/nli-deberta-v3-base
CE-DeBERTa-S	cross-encoder/nli-deberta-v3-small
CE-RoBERTa-B	cross-encoder/nli-roberta-base
<i>DeBERTa-MNLI</i>	
DeBERTa-B-MNLI	microsoft/deberta-base-mnli
DeBERTa-L-MNLI	microsoft/deberta-large-mnli
DeBERTa-XL-MNLI	microsoft/deberta-xlarge-mnli
DeBERTa-v2-XL	microsoft/deberta-v2-xlarge-mnli
DeBERTa-v2-XXL	microsoft/deberta-v2-xxlarge-mnli
DeBERTa-L-MNLI-K	khalidalt/DeBERTa-v3-large-mnli
DeBERTa-L-SNLI-P	pepa/deberta-v3-large-snli
DeBERTa-L-SNLI-U	utahnlp/snli_microsoft_deberta-v3-large_seed-1
<i>BART/RoBERTa</i>	
BART-L-MNLI	facebook/bart-large-mnli
BART-Yahoo	joeddav/bart-large-mnli-yahoo-answers
RoBERTa-L-MNLI	roberta-large-mnli
RoBERTa-L-WANLI	alisawuffles/roberta-large-wanli
<i>Domain-specific / smaller</i>	
PubMedBERT-NLI	pritamdeka/PubMedBERT-MNLI-MedNLI
Erlangshen-RoBERTa	IDEA-CCNL/Erlangshen-Roberta-330M-NLI
DistilCamemBERT	cmarkea/distilcamembert-base-nli
DistilBERT	typeform/distilbert-base-uncased-mnli
ALBERT-B-MNLI	prajjwal1/albert-base-v2-mnli
BERT-B-SNLI	textattack/bert-base-uncased-snli
<i>Generative LLM</i>	
GPT-Neo-1.3B	EleutherAI/gpt-neo-1.3B
Gemma-2B	google/gemma-2b
Gemma-2-2B	google/gemma-2-2b
Gemma-3-1B	google/gemma-3-1b-it