EmpiriGraph-Psy: A Dataset and LLM Pipeline for Extracting Empirical Relation Graphs from Psychology Abstracts

Anonymous ACL submission

Abstract

Existing scientific relation extraction benchmarks mainly target domains such as computer science and biomedicine, where entities are tasks, methods, datasets, materials, or biological concepts. This leaves a gap for variable-oriented empirical fields such as psychology, where findings are expressed as relations among constructs, measurements, interventions, and outcomes. We introduce variable-centered empirical graph extraction, the task of mapping scientific abstracts to typed graphs whose nodes are normalized variables and whose edges represent empirical and hierarchical relations. To support this task, we construct EmpiriGraph-Psy, a benchmark of 210 psychology abstracts annotated by domain-trained coders with normalized variables, construct–measurement hierarchies, empirical relation types, and validation states. We evaluate frontier and open-weight LLMs using both direct extraction and a staged graph-construction pipeline that separates variable extraction, normalization, hierarchy construction, evidence selection, relation extraction, and edge validation. The staged pipeline substantially outperforms direct extraction, with the best configuration achieving 0.736 macro-F1. Error analysis shows that moderation relations and construct–measurement hierarchies remain the most challenging cases, highlighting the difficulty of extracting higher-order empirical claims and implicit abstraction structure from scientific abstracts. [1]

1 Introduction

Scientific relation extraction aims to identify concepts and relations from unstructured research text and represent them as structured graphs. Existing benchmarks have largely focused on computer science and NLP papers (e.g., Gábor et al. 2018) , where scientific entities are typically tasks, datasets, models, and metrics. While these schemas support model comparison and evaluation, they are less suited to variable-oriented empirical fields such as psychology, social science, behavioral science, and health research. In these disciplines, knowledge is often organized around variables and their empirical relations, including covariation, intervention effects, mechanisms, and contextual conditions.

In this paper, we construct a corpus of psychology abstracts because psychology is a representative variable-oriented empirical field: its findings are commonly expressed as relations among constructs, measurements, interventions, outcomes, and contextual conditions. For example, abstracts may report associations between leadership and personality (e.g., Andersen 2006), effects of psychological interventions on patient outcomes (e.g., Anderson and Ozakinci 2018), or moderation by family environment in the relation between genetic risk and behavior (e.g., Cadoret et al. 1995). Extracting such relations can support the construction of variable-centered knowledge graphs for a broad range of empirical disciplines. It further enables large-scale evidence synthesis and historical analyses of how variables and theories emerge, stabilize, change, or disappear across the literature.

Variable-centered graph extraction poses challenges beyond standard entity and relation extraction. First, variable mentions require semantic normalization: the same construct may be expressed through synonyms, abbreviations, measurement instruments, or theoretically related terms. Second, empirical findings are often stated at multiple levels of abstraction. An abstract may relate broad constructs while also specifying relations among their finer-grained dimensions, as illustrated in Figure 1. Flattening these levels loses theoretical structure, while treating them as unrelated variables fragments the evidence. Third, relation classification requires contextual reasoning, including distinguishing associational, directional, mechanistic,

[1] Experimental code: https://anonymous.4open.science/r/EmpiriGraph-Psy-BAAA/

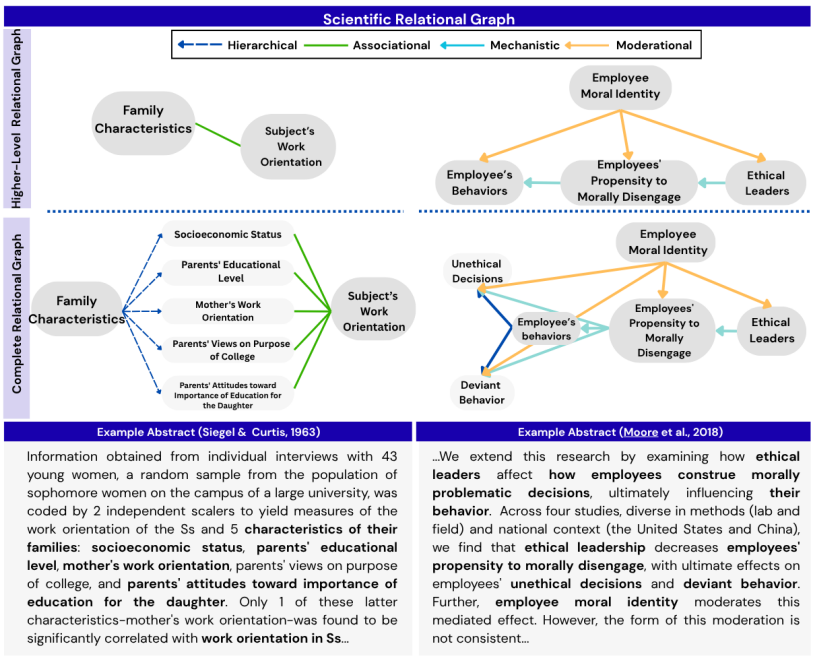

Figure 1: Illustration of the variable-centered relational graph extraction task. The input abstract is transformed into a typed variable graph, where the upper layer represents higher-level constructs and the lower layer represents finer-grained variables or dimensions. Edges capture associational, mechanistic, moderational, and hierarchical relations.

and moderational claims, and identifying whether a relation is validated, hypothesized, or null. These challenges call for an extraction framework that jointly models variable identification, normalization, hierarchy, evidence grounding, and relation classification.

We address these challenges with a multi-stage LLM pipeline that decomposes graph construction into variable extraction, semantic normalization, hierarchy construction, evidence selection, relation extraction, and edge validation. This decomposition follows the structure of the annotation task: variables are first identified and canonicalized, hierarchical edges explicitly encode abstraction structure, and empirical edges are classified as associational, mechanistic, or moderational with their directionality and validation status.

To evaluate predicted graphs, we use a structure-first alignment procedure inspired by maximum common subgraph matching. Rather than relying on surface overlap between variable names, this evaluation aligns predicted and gold graphs under partial node matching and measures whether associational, mechanistic, moderational, and hierarchical edges are recovered with the correct validation status. This allows us to separate structural graph recovery from surface variation in variable naming.

Our contributions are as follows:

- We introduce Empirical Research Knowledge Graph Extraction, a variable-centered task for mapping empirical research abstracts into typed graphs over normalized variables.
- We construct EmpiriGraph-Psy, a benchmark of 210 psychology abstracts annotated with normalized variables, validation states, and associational, mechanistic, moderational, and hierarchical relations.
- We propose a multi-stage LLM pipeline for Empirical Research Knowledge Graph Extraction and show that it substantially outperforms direct prompting across multiple LLMs.
- We introduce a structure-first graph evaluation framework that aligns predicted and gold graphs under partial node matching and measures typed edge recovery.

2 Related Work

2.1 Relation Extraction from Scientific Materials

Recent studies have extensively explored the role of NLP techniques in extracting, organizing, and syn-

thesizing scientific findings from large-scale academic corpora(Wang et al., 2020; Chen et al., 2021; Li et al., 2025; Katz et al., 2024). As noted by Zhao et al. (2024), the number of scientific publications has grown exponentially, making it increasingly infeasible for researchers to manually discover and synthesize important scientific facts embedded in unstructured text. In this context, relation extraction (RE), which aims to automatically identify structured relations among scientific entities, serves as an important infrastructure for the science of science, supporting applications such as scientific knowledge graph construction, text summarization, and scientific leaderboard construction(Luan et al., 2018; Mondal et al., 2021; Dagdelen et al., 2024; Hou et al., 2019; Şahinuç et al., 2024).

Early scientific relation extraction studies can be traced to SEMEVAL-2017(Augenstein et al., 2017; Luan et al., 2017), which introduces a cross-domain scientific keyphrase tagging task, identifying TASK, PROCESS and MATERIAL spans across Computer Science, Materials Science, and Physics to capture domain-specific scientific concepts from text. Later benchmarks expanded scientific relation extraction with richer schemas for scientific entities and relations, including fine-grained relation types in SemEval-2018 Task 7 and joint entity–relation–coreference annotation in SciERC (Gábor et al., 2018; Luan et al., 2018). More recently, SciNLP extends scientific information extraction from abstracts to full-text NLP papers, annotating 60 ACL long papers with fine-grained *task*, *method*, *dataset*, and *metric* entities and 11 relation types, including *UsedFor*, *EvaluatedOn*, *MeasuredBy*, and *CompareWith* (Duan et al., 2025).

Despite the progression of scientific information extraction, existing benchmarks show that entity recognition and type classification are comparatively tractable, while relation discovery remains the main bottleneck. In SemEval-2018 Task 7, systems perform better when candidate relation instances are given, but drop when they must both identify and classify relations from raw abstracts (Gábor et al., 2018). This gap is even more pronounced in full-text settings, where related entities may be separated by longer contexts (Duan et al., 2025). Meanwhile, existing studies on scientific relation extraction mostly focus on NLP or computer science papers, with a smaller line of work considering other domains. For example, biomedical relation extraction (Bio-RE) identifies relations among biomedical entities, such as chemical–disease and gene–disease relations (Shang et al., 2025), while chemistry and materials science studies extract chemical reaction data (Guo et al., 2022) and material synthesis procedures (Yang et al., 2022).

2.2 LLM in Relation Extraction

LLMs exhibit in-context learning (Min et al., 2022) and achieve promising results across many NLP tasks such as text classification, fact retrieval, and natural language inference (Brown et al., 2020; Wei et al., 2022a). Recent studies further show that LLMs exhibit competitive performance on relation extraction tasks under instruction-tuned settings, in some cases approaching or surpassing standard fully supervised methods (Wan et al., 2023; Wadhwa et al., 2023; Tan et al., 2025). LLM has also widely applied in scientific relation extraction across different disciplines including materials science (Dagdelen et al., 2024; Foppiano et al., 2024) and biomedicine, where they facilitate structured knowledge acquisition from scientific literature (Shang et al., 2025; Laskar et al., 2025). Specifically, Chain-of-Thought (CoT) prompting has emerged as an effective mechanism for improving reasoning-intensive NLP tasks by eliciting intermediate reasoning steps from LLMs (Wei et al., 2022b; Wadhwa et al., 2023). It improves relation extraction performance by enhancing relation grounding and aligning dispersed textual evidence with candidate relations, thereby mitigating the aforementioned limitations of conventional supervised approaches in relation discovery (Ma et al., 2023; Wadhwa et al., 2023). These findings suggest that LLMs provide a promising approach for extracting empirical relations which requires contextual understanding and nuanced semantic reasoning. Despite these strengths, no existing LLMs relation extraction system focus on scientific abstraction, and adapting LLMs to this task is non-trivial. We thus apply LLMs to extract empirical research knowledge graphs from scientific abstracts.

3 Background

In this section, we define the task of Empirical Research Knowledge Graph Extraction. Given a scientific text document X containing n tokens, the goal of the task is to create a graph $G = (V, E)$, where V denotes a set of vertices representing scientific variables or constructs and E denotes a set of edges representing the relationships between the

variables. We distinguish between empirical relation edges and conceptual edges. Empirical edges describe substantive relationships among variables, including associational, mechanistic, and conditional relations. Conceptual edges depict abstraction relations between higher-level constructs and lower-level variables, dimensions, or indicators. An edge is denoted as (v_i, r, v_j), where v_i is the head node, v_j is the tail node, and r is the relation between the two nodes. Relation $r \in \mathcal{R}$ is assigned with one of the categories in the following label set:

$$\mathcal{R} = \{\text{Associational}, \text{Mechanistic}, \text{Moderational}, \text{Hierarchical}\}. \quad (1)$$

An associational edge describes that the head node v_i and the tail node v_j covary or are correlated, without specifying a causal or mechanistic relationship between them, such as a correlation, covariation, or a group difference associated with v_i, without claiming that v_i mechanistically affects v_j.

A mechanistic edge is annotated when the text further explains the mechanism underlying the covariance between v_i and v_j, such that variable v_i affects, predicts, influences, enables, or otherwise has a directional effect on variable v_j.

A conditional edge is annotated when a third variable v_k conditions an established relationship between variables, such as $v_i \rightarrow v_j$, by altering its strength, direction, or statistical significance. This edge type primarily captures moderation or interaction effects.A conditional relation in which v_k moderates the relationship $v_i \rightarrow v_j$ is encoded in the knowledge graph as $(v_k, Conditional, v_i)$ and $(v_k, Conditional, v_j)$.

For each empirical edge, we further annotate its validation state, using three labels: *validated*, *null*, and *hypothesized*. An edge is labeled as *validated* when the text reports that the relationship is supported by the study's empirical results. An edge is labeled as *null* when the relationship is tested but not supported, such as when the abstract reports an insignificant effect. An edge is labeled as *hypothesized* when the relationship is proposed as a hypothesis or expected, but the text does not report empirical evidence confirming or rejecting it.

A major annotation ambiguity arises because abstracts often reports variable relationships at different levels of abstraction. For example, as illustrated in Figure 1 (b), the abstract first describes how *ethical leader* indirectly influence *employee's behavior*, and then specifies two concrete forms of *employee's behavior – unethical decision* and *deviant behavior*. In this case, *employees' behavior* is annotated as a higher-level construct, representing the main research objects, while *unethical decision* and *deviant behavior* are annotated as lower-level variables or specific behavioral outcomes.

We introduce hierarchy edges for representing such abstraction relations. A hierarchy edge connects a higher-level construct to a lower-level variable. The former includes constructs, theoretical categories, or research objects, while the latter denotes a more specific dimension, indicator, or measurement of that construct. The lower-level graph retains all specific relationships between different dimensions of a construct or specific measurements. The higher-level graph collapses lower-level variables into their corresponding higher-level constructs, yielding a more abstract representation of the theoretical relationships among constructs. Such multi-level coding allows for different levels of analytic purposes. For downstream projects such as synthesizing research findings into a knowledge graph, higher-level relational graphs provide descriptions of main research objects and relationships. While for analyzing historical changes in scientific findings, a complete relational graph will retain the organization of relationships in the abstract.

4 Dataset and Human Annotation

4.1 Abstract Corpus Construction

We collect a corpus of abstracts from psychology journals. Since the frequency, density, and linguistic realization of different relation types may vary substantially across historical periods, we designed a stratified sample with broad temporal coverage rather than sampling only from recent publications. Our dataset covers abstracts from six psychology journals with long publication histories, relatively high 5-year impact factors, and represents different subfields of psychology. We include only original author-written abstracts and exclude retrospectively added machine-generated abstracts.

4.2 Data Collection and Validation

Three annotators participated in the project. All annotators were psychology students at either the PhD or undergraduate level. Annotations were conducted using a customized annotation platform

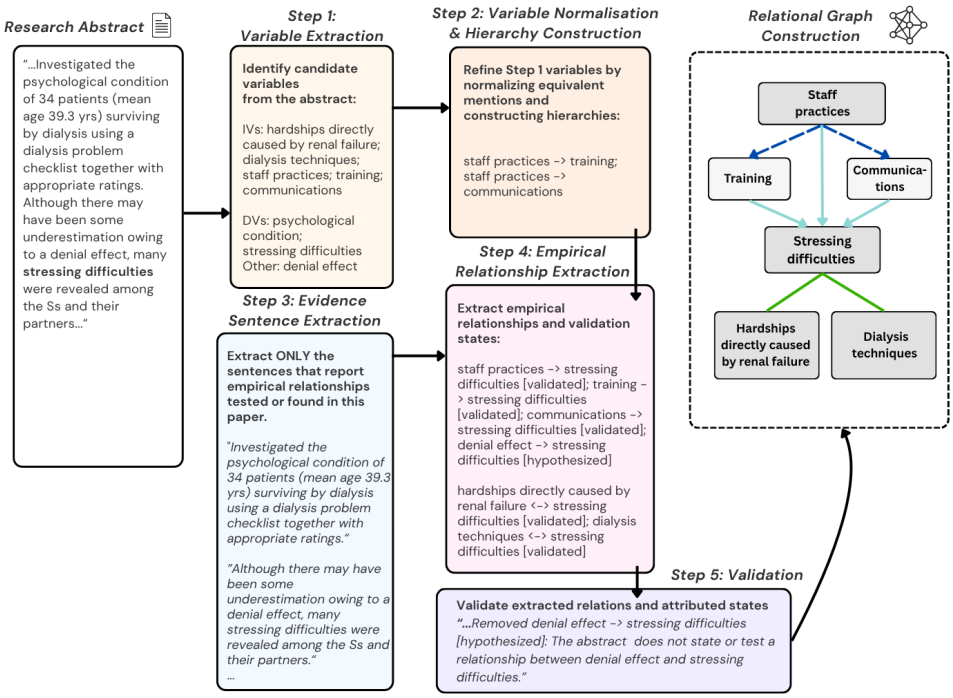

Figure 2: Overview of the proposed variable-centered relational graph extraction pipeline. The system maps a scientific abstract into a structured variable graph through variable extraction, normalization and hierarchy construction, evidence sentence extraction, graph construction, and edge validation.

built on top of Label Studio. The details can be found in Appendix A.1. Prior to the formal annotation phase, all coders received training on the coding scheme and completed a qualification task consisting of 10 abstracts. These materials were used only for training and were excluded from the final coding dataset. After coders passed this initial training stage, they proceeded to the formal annotation task. During annotation, the coding guidelines were iteratively discussed and refined through team discussions.

Gold graphs were constructed through a two-stage process. First, the three annotators jointly covered the full corpus of 210 abstracts, with a 50-abstract subset independently annotated by all three coders for reliability assessment. Second, after agreement analysis, the annotations were reviewed by the annotator team. Disagreements in variable boundaries, variable normalization, hierarchy edges, relation types, and validation-state labels were discussed and resolved. The resulting reviewed annotations were used as the final gold graphs for model evaluation.

5 Methodology

5.1 Pipeline

Figure 2 shows the overall structure of the proposed pipeline. We decompose graph construction into five sequential subtasks: variable extraction, variable normalization and hierarchy construction, evidence sentence extraction, empirical relationship extraction, and validation. Each stage produces a structured intermediate output, which is then passed to the next stage as context. The full prompts used for each stage are provided in Appendix A.7; below we summarize the role of each stage.

Deductive Task

Step 1: Variable Extraction
In Step 1, the model identifies candidate variables from the abstract and proposes normalized variable spans/names. This stage emphasizes variable identification because downstream stages cannot recover variables that are never extracted.

Step 2: Variable Normalization and Hierarchy Construction
In Step 2, the system builds a variable hierarchy and performs canonicalization. Practically, this stage merges near-duplicate mentions, links higher-level and lower-level variables, and enforces a consistent variable vocabulary for later relation extraction.

Step 3: Evidence Sentence Extraction
In Step 3, the model selects evidence-bearing sentences likely to contain relational information with reference of the variable list generated from Step 2. This acts as an information bottleneck that improves precision by reducing distracting context and forcing the model to justify edge decisions from explicit textual evidence.

Step 4: Graph Construction
In Step 4, the model predicts and classifies empirical relations among variables (Associational, Mechanistic, Moderational), producing the initial graph structure.

Step 5: Edge Validation
In Step 5, the system re-validates extracted edges and relation attributes, correcting likely false positives and improving accuracy. This final pass functions as a consistency and quality-control stage before evaluation.

5.2 Graph Evaluation

Exact node-label matching is unsuitable because the same scientific variable may be expressed by different surface forms across annotators or models. We therefore evaluate graphs using a structure-first alignment. The gold graph G refers to the finalized human-annotated graph produced through the procedure described in Section 4.2. Let the gold graph be $G = (V_G, E_G)$ and the predicted graph be $P = (V_P, E_P)$, where each edge is directed and typed:

$$e = (u, v, \tau), \tau \in \mathcal{T}. \tag{2}$$

We search for an injective partial mapping

$$\phi : V_G \to V_P \cup \{\emptyset\}, \tag{3}$$

where $\emptyset$ denotes an unmatched gold node. A gold edge $(u, v, \tau) \in E_G$ is counted as matched if

$$\phi(u) \neq \emptyset, \phi(v) \neq \emptyset, (\phi(u), \phi(v), \tau) \in E_P. \tag{4}$$

The alignment is chosen to maximize typed edge overlap:

$$\phi^\star = \arg\max_{\phi} |\{(u, v, \tau) \in E_G : (\phi(u), \phi(v), \tau) \in E_P\}| . \tag{5}$$

Let $m_\star$ be the number of matched edges under $\phi^\star$. We then compute precision, recall, and F1 score based on $m_\star$, $|E_P|$.

We report three complementary views: M_{typed}, computed on the full directed typed graphs; M_{higher}, computed after projecting both graphs onto higher-level nodes; and M_{agnostic}, computed after collapsing all edge types into a single undirected relation type. Optimization details, preprocessing, per-type scores, and complexity analysis are provided in Appendix A.3.

Because the goal of this paper is graph extraction rather than node-label normalization, we treat node alignment as an auxiliary component of graph-level evaluation. Nevertheless, since the structure-first metric may align nodes with different surface forms, we further validate the aligned node pairs in Appendix A.5. Across all aligned node pairs, the mean embedding-based cosine similarity is 0.735. Manual inspection of 100 stratified randomly sampled pairs found that 87 pairs referred to the same variable or construct. This suggests that most structure-aligned node pairs are semantically valid, although a minority of alignments remain noisy.

6 Experiment

6.1 Baseline

We first constructed a direct-prompting baseline by providing GPT-5.4 with a general task description, edge-type definitions, and the desired output format, and asking it to generate the complete graph in a single step. To evaluate whether explicit task decomposition improves Empirical Research Knowledge Graph Extraction, we compared three GPT-5.4-based settings: direct one-step prompting, a collapsed pipeline prompt that describes all five stages within a single request, and the full staged pipeline in which the five stages are executed as separate requests.

We then evaluated the full staged pipeline across several high-performing large language models selected with reference to the LLM leaderboard reasoning index ranking[2]: GPT-5.2, GPT-5.4, Claude Sonnet 4.6, Claude Opus 4.7, DeepSeek V4 Pro, and Gemini 3 Flash. We additionally included GPT-4o as a lower-cost comparison model that has been widely-used in annotation tasks. All model outputs were evaluated using the same structural graph evaluation protocol.

6.2 Results

We evaluate the model performance by applying the graph structural evaluation method (Section 5.2) between the gold graph and the model-predicted graphs. The performance of our production model is reported in Table 1.

We produced both micro and macro F1 scores for the best models (GPT-5.4 + GPT-5.2). Micro

[2] https://llm-stats.com/

Type	Micro / Pooled Edges						Macro / Per Abstract		
	Human	LLM	Matched	Precision	Recall	F1	Mean Precision	Mean Recall	Mean F1
Mechanistic	759	757	583	0.770	**0.768**	**0.769**	**0.829**	**0.826**	**0.798**
Associational	282	297	206	0.694	0.730	0.712	0.725	0.742	0.711
Moderational	253	187	151	**0.807**	0.597	0.686	0.728	0.619	0.639
Hierarchical	418	417	272	0.652	0.651	0.651	0.690	0.668	0.662
Higher-Level Graph	768	800	570	0.713	0.742	0.727	0.743	0.789	0.732
Type-Agnostic Graph	1712	1658	1261	0.761	0.737	0.748	0.813	0.815	0.779
Full Graph	1712	1658	1212	0.731	0.708	0.719	0.767	0.771	0.736

Table 1: Structural evaluation of LLM-generated graphs against human-annotated gold graphs. Micro metrics are computed by pooling all edges across abstracts; we aggregate the total number of human-coded edges, LLM-predicted edges, and matched edges, and then compute precision, recall, and F1 from these corpus-level counts. Macro metrics are computed at the abstract level: precision, recall, and F1 are first calculated separately for each abstract and then averaged across abstracts.

F1 is produced by pooling the edges and then computed as a whole, whereas macro F1 is computed by averaging the F1 over each abstract. Our best model (GPT-5.4 for Step 1, 5; GPT-5.2 for the rest) achieved a micro F1 of 0.72 and a macro F1 of 0.74. This is only slightly lower than our mean F1 score reported in inter-annotator agreement by human annotators. Our model performs well on extracting mechanistic and associational relationships. However, it performs less satisfactorily on extracting moderational and hierarchical relationships. Notably, moderational and hierarchical edges are intrinsically harder: the former requires higher-order reasoning beyond binary relation extraction, while the latter requires implicit taxonomic abstraction over variable mentions (Luan et al., 2018; Jia et al., 2019).

We further assessed the robustness of model performance across journals and publish periods, see Appendix 5. This suggests that the model generalizes well across abstracts from different historical periods, despite potential changes in writing style, reporting conventions, and terminology over time. Extraction performance is highly stable across period, with all F1 score above 0.71. The performance varied more across journals, with journal-level F1 ranging from 0.67 to 0.81.

6.3 Model Comparison

Table 2 reports model performance under direct prompting and the proposed pipeline. The direct-prompting baseline obtains balanced but relatively low scores, with an F1 of 0.530. In contrast, all pipeline-based configurations outperform the baseline. The combined GPT-5.2 + GPT-5.4 configuration achieves the best overall performance, with 0.767 precision, 0.771 recall, and 0.736 F1. Among single-model configurations, GPT-5.4 performs best, reaching an F1 of 0.694, followed by GPT-5.2 with an F1 of 0.679.

Different models exhibit distinct error profiles. Gemini 3 Flash achieves the highest recall (0.782) but substantially lower precision, suggesting a tendency to over-generate relations. By contrast, DeepSeek V4 Pro and GPT-4o are more precision-oriented but recover fewer correct relations. We further compare direct prompting, chain-of-thought prompting, and the final staged pipeline using GPT-5.4. This comparison shows a clear improvement from direct prompting to chain-of-thought prompting and then to the staged pipeline, indicating that explicit decomposition improves graph extraction quality beyond model choice alone. Overall, the results show that the proposed pipeline improves the extraction relative to one-step prompting, and that combining GPT-5.2 and GPT-5.4 provides the best precision–recall balance.

6.4 Error Analysis

In addition to Table 1, Figure 3 presents a detailed confusion matrix and breakdown between the gold and predicted edge types. Hierarchy has the highest FN rate (27.4%) — the model misses more than one in four hierarchy edges. This edge type requires inferring that one construct is a component of another, a structural inference rarely made explicit in abstracts. Moderation has the highest type confusion rate (15.3%) — nearly one-sixth of moderation edges are recovered under the wrong type, most often as directional. This reflects the inherent ambiguity of moderation language and the model's tendency to simplify three-way interactions. Di-

Model	Precision	Recall	F1
GPT-5.4 (Direct Prompt)	0.566	0.606	0.528
GPT-5.2 + GPT-5.4	**0.767**	0.771	**0.736**
GPT-5.4	0.754	0.714	0.694
GPT-5.4 (Single Step CoT)	0.691	0.698	0.658
GPT-5.2	0.700	0.726	0.679
GPT-4o	0.738	0.508	0.551
Claude Opus 4.7	0.700	0.651	0.635
Claude Sonnet 4.6	0.660	0.736	0.652
Gemini 3 Flash	0.536	**0.782**	0.598
Deepseek V4 Pro	0.757	0.619	0.641

Note. The model variants are gpt-4o, gpt-5.2, gpt-5.4, claude-sonnet-4.6, claude-opus-4.7, gemini-3-flash, deepseek-v4-pro accordingly. All parameters (i.e., reasoning level, verbosity) are set to 'low' where applicable.

Table 2: Performance (macro-averaged) comparison of different models with respect to the gold graph.

rectional has the lowest type confusion (2.3%) and a moderate FN rate, reflecting its dominant frequency and clearer linguistic markers (e.g., "predicted", "mediated", "caused"). Correlational edges are nearly balanced in FN/FP (10.2% vs. 9.8%), suggesting the model neither systematically over- nor under-produces correlational relationships, but does occasionally impose directionality (8.3% type confusion). FP rates are uniformly lower than FN rates across all types, indicating the model is more likely to miss a relationship than to hallucinate one—a desirable conservative bias for downstream knowledge-graph construction.

7 Conclusion

This study introduced EmpriGraph-Psy, a dataset and LLM pipeline for extracting empirical relation graphs from psychology abstracts. We explored how variable-centered scientific findings can be represented as graphs. We further demonstrated that decomposing graph construction into variable extraction, normalization, evidence selection, relation extraction, and validation substantially improves LLM performance over direct prompting. The best pipeline achieved strong graph-level extraction performance ($F1 = 0.74$), approaching human agreement and showing stable results across publication periods. EmpiriGraph-Psy provides a benchmark and extraction framework for constructing empirical knowledge graphs in psychology, and offers a reference framework for information extraction in broader variable-oriented empirical fields.

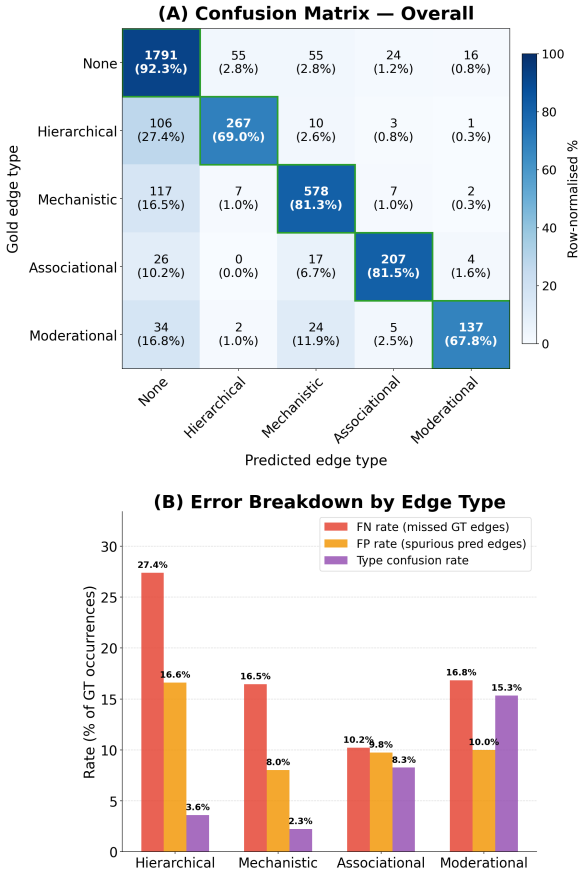

Figure 3: Confusion matrix of LLM-predicted edge types against gold graph edge types and error breakdown by edge types.

Limitations

The current dataset is limited to psychology abstracts. Although the dataset covers multiple psychology subfields and a broad historical period, it remains unclear whether the proposed workflow generalizes to other disciplines such as health science or biology, where abstracts may follow different writing conventions and reporting styles. Future work could extend the workflow of Empirical Research Knowledge Graph Extraction to additional domains to assess the cross-disciplinary robustness of the proposed pipeline.

Our work focuses on extracting empirical and conceptual relationships and evaluating whether LLM-based pipelines can recover such graph structures from abstracts. The current annotation scheme does not include other important scientific components such as samples, methods, statistical procedures, or tasks. Future work may therefore integrate our LLM workflow with prior NLP-based scientific information extraction methods to construct more complete scientific knowledge graphs.

References

Jon Aarum Andersen. 2006. Leadership, personality and effectiveness. *The Journal of Socio-Economics*, 35(6):1078–1091.

Niall Anderson and Gozde Ozakinci. 2018. Effectiveness of psychological interventions to improve quality of life in people with long-term conditions: rapid systematic review of randomised controlled trials. *BMC Psychology*, 6(1):11.

Isabelle Augenstein, Mrinal Das, Sebastian Riedel, Lakshmi Vikraman, and Andrew McCallum. 2017. Semeval-2017 task 10: Scienceie – extracting keyphrases and relations from scientific publications. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 546–555, Vancouver, Canada. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, and 1 others. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901.

Remi J. Cadoret, William R. Yates, E. Troughton, G. Woodworth, and Mark A. Stewart. 1995. Genetic-environmental interaction in the genesis of aggressivity and conduct disorders. *Archives of General Psychiatry*, 52(11):916–924.

Xiuying Chen, Hind Alamro, Mingzhe Li, Shen Gao, Xiangliang Zhang, Dongyan Zhao, and Rui Yan. 2021. Capturing relations between scientific papers: An abstractive model for related work section generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6068–6077, Online. Association for Computational Linguistics.

John Dagdelen, Alexander Dunn, Sanghoon Lee, Nicholas Walker, Andrew S. Rosen, Gerbrand Ceder, Kristin A. Persson, and Anubhav Jain. 2024. Structured information extraction from scientific text with large language models. *Nature Communications*, 15(1):1418.

Decheng Duan, Yingyi Zhang, Jitong Peng, and Chengzhi Zhang. 2025. Scinlp: A domain-specific benchmark for full-text scientific entity and relation extraction in nlp. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 14473–14486, Suzhou, China. Association for Computational Linguistics.

Luca Foppiano, Guillaume Lambard, Toshiyuki Amagasa, and Masashi Ishii. 2024. Mining experimental data from materials science literature with large language models: an evaluation study. *Science and Technology of Advanced Materials: Methods*, 4:2356506.

Kata Gábor, Davide Buscaldi, Anne-Kathrin Schumann, Behrang QasemiZadeh, Haïfa Zargayouna, and Thierry Charnois. 2018. Semeval-2018 task 7: Semantic relation extraction and classification in scientific papers. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 679–688, New Orleans, Louisiana. Association for Computational Linguistics.

Jiang Guo, A. Santiago Ibanez-Lopez, Hanyu Gao, Victor Quach, Connor W. Coley, Klavs F. Jensen, and Regina Barzilay. 2022. Automated chemical reaction extraction from scientific literature. *Journal of Chemical Information and Modeling*, 62(9):2035–2045.

Yufang Hou, Charles Jochim, Martin Gleize, Francesca Bonin, and Debasis Ganguly. 2019. Identification of tasks, datasets, evaluation metrics, and numeric scores for scientific leaderboards construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5203–5213, Florence, Italy. Association for Computational Linguistics.

Robin Jia, Cliff Wong, and Hoifung Poon. 2019. Document-level n-ary relation extraction with multiscale representation learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3693–3704, Minneapolis, Minnesota. Association for Computational Linguistics.

Uri Katz, Mosh Levy, and Yoav Goldberg. 2024. Knowledge navigator: Llm-guided browsing framework for exploratory search in scientific literature. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 8838–8855, Miami, Florida, USA. Association for Computational Linguistics.

Md Tahmid Rahman Laskar, Israt Jahan, Elham Dolatabadi, Chun Peng, Enamul Hoque, and Jimmy Xiangji Huang. 2025. Improving automatic evaluation of large language models (LLMs) in biomedical relation extraction via LLMs-as-the-judge. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25483–25497, Vienna, Austria. Association for Computational Linguistics.

Pengjiang Li, Zaitian Wang, Xinhao Zhang, Ran Zhang, Lu Jiang, Pengfei Wang, and Yuanchun Zhou. 2025. Scitopic: Enhancing topic discovery in scientific literature through advanced llm. *arXiv preprint arXiv:2508.20514*.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2017. Scientific information extraction with semi-supervised neural tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2641–2651, Copenhagen, Denmark. Association for Computational Linguistics.

Yi Luan, David Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multitask identification of entities, relations, and coreference for scientific knowledge graph construction.

In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3219–3232, Brussels, Belgium. Association for Computational Linguistics.

Xilai Ma, Jing Li, and Min Zhang. 2023. Chain of thought with explicit evidence reasoning for few-shot relation extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2334–2352, Singapore. Association for Computational Linguistics.

Ciaran McCreesh, Patrick Prosser, and James Trimble. 2017. A partitioning algorithm for maximum common subgraph problems. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 712–719. International Joint Conferences on Artificial Intelligence Organization.

Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Ishani Mondal, Yufang Hou, and Charles Jochim. 2021. End-to-end construction of NLP knowledge graph. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1885–1895, Online. Association for Computational Linguistics.

Samba Ndojh Ndiaye and Christine Solnon. 2011. Cp models for maximum common subgraph problems. In *Principles and Practice of Constraint Programming – CP 2011*, pages 637–644. Springer.

Furkan Şahinuç, Thy Thy Tran, Yulia Grishina, Yufang Hou, Bei Chen, and Iryna Gurevych. 2024. Efficient performance tracking: Leveraging large language models for automated construction of scientific leaderboards. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7963–7977, Miami, Florida, USA. Association for Computational Linguistics.

Yufei Shang, Yanrong Guo, Shijie Hao, and Richang Hong. 2025. Biomedical relation extraction via adaptive document-relation cross-mapping and concept unique identifier. *Preprint*, arXiv:2501.05155.

Xingwei Tan, Yuxiang Zhou, Gabriele Pergola, and Yulan He. 2025. Cascading large language models for salient event graph generation. *arXiv preprint arXiv:2406.18449*.

Somin Wadhwa, Silvio Amir, and Byron C. Wallace. 2023. Revisiting relation extraction in the era of large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15566–15589, Toronto, Canada. Association for Computational Linguistics.

Zhen Wan, Fei Cheng, Zhuoyuan Mao, Qianying Liu, Haiyue Song, Jiwei Li, and Sadao Kurohashi. 2023. GPT-RE: In-context learning for relation extraction using large language models. *arXiv preprint arXiv:2305.02105*.

Kuansan Wang, Zhihong Shen, Chiyuan Huang, Chieh-Hsi Wu, Darrin Eide, and Yuxiao Dong. 2020. Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1(1):396–413.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, and 1 others. 2022a. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022b. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837.

Xianjun Yang, Ya Zhuo, Julia Zuo, Xinlu Zhang, Stephen Wilson, and Linda Petzold. 2022. PcMSP: A dataset for scientific action graphs extraction from polycrystalline materials synthesis procedure text. In *Findings of the Association for Computational Linguistics: EMNLP 2022*. Association for Computational Linguistics.

Xiaoyan Zhao, Yang Deng, Min Yang, Lingzhi Wang, Rui Zhang, Hong Cheng, Wai Lam, Ying Shen, and Ruifeng Xu. 2024. A comprehensive survey on relation extraction: Recent advances and new frontiers. *ACM Computing Surveys*.

A Appendix

A.1 Dataset and Human Annotations

Dataset The final annotated dataset included abstracts from the following six journals: *Journal of Applied Psychology*, *Journal of Consulting and Clinical Psychology*, *Journal of Counseling Psychology*, *Journal of Educational Psychology*, *Journal of Experimental Psychology: General*, and *Behaviour Research and Therapy*. Our inclusion criteria were: (1) peer-reviewed journal articles and (2) empirical studies. Therefore, literature reviews, letters, meta-analyses, and other non-empirical studies were excluded from the dataset. For each journal, we sampled five abstracts from every 10-year period between the 1960s and 2025. The final dataset therefore consisted of 210 abstracts in total, including 30 abstracts per decade and 35 abstracts from each journal.

Human Annotation Three annotators participated in the project. One coder (A) was an undergraduate student in Psychology, and the other two coders (B, C) were PhD students in Psychology. Annotators participated under different project arrangements, including compensated research assistance and unpaid project participation. The annotating interface can be found in Figure 4. The annotation interface extended Label Studio with custom modules for relational graph annotation. Annotators could highlight variable spans, modify normalized variable names in the region panel, assign relation types in the relations panel, and visually inspect the resulting relationship graph rendered in the sidebar after submission.

Annotation Guideline Annotators followed a detailed annotation guideline. The guideline instructed human annotators to proceed through the following steps: (1) locate sentences in the abstract that contain information about key concepts, i.e., variables, and empirical relationships tested or reported in the article; (2) identify variables that were empirically examined in the study; (3) code hierarchical relationships between variables when one variable was presented as a subtype, component, dimension, indicator, or more specific measurement of a broader construct; (4) annotate the four relation types; (5) assign a validation-state label to each empirical relation, using *validated*, *null*, or *hypothesized*; and (6) canonicalize variable names by merging synonyms and coreferential mentions. The full annotation guideline is provided in the anonymous repository linked in the abstract.

Inter-annotator Agreement As reported in Table 3, pairwise agreement was highest between Coder A and Coder C, both of whom were Psychology PhD students, with an F1 score of .82 and Cohen's κ of .60. Comparisons involving Coder B, the Psychology undergraduate student, showed slightly lower but broadly comparable agreement.

Coder comparison	F1	Kappa
Coder A vs. Coder B	0.717	Cohen's $\kappa = 0.566$
Coder A vs. Coder C	0.777	Cohen's $\kappa = 0.590$
Coder B vs. Coder C	0.830	Cohen's $\kappa = 0.559$
All three coders	—	Fleiss' $\kappa = 0.632$

Table 3: Inter-annotator agreement among the three human coders on the 50-article overlap set. Pairwise F1 treats the second coder in each comparison as the reference annotation.

To remain compatible with the access conditions of the original source materials, the released dataset does not redistribute copyrighted abstract text. Instead, we release metadata identifiers and the derived annotation layer, including normalized variables, empirical relation edges, hierarchy edges, validation states, and dataset splits. The annotation layer is released under a research-compatible open license, such as CC BY 4.0, while accompanying code is released separately under an open-source software license. The intended use of the dataset is research on scientific information extraction, empirical relation extraction, knowledge graph construction, and model evaluation. The dataset is not intended for clinical, legal, policy, or individual-level decision-making, nor for making evaluative claims about specific authors, participants, journals, or institutions.

A.2 Model Performance Across Journals and Periods

We assessed the robustness of model performance across journals and publish periods. As shown in Tables 4, extraction performance is highly stable across period, with all F1 score above 0.71. This suggests that the model generalizes well across abstracts from different historical periods, despite potential changes in writing style, reporting conventions, and terminology over time. The performance varied more across journals, see Tables 5, with journal-level F1 ranging from 0.67 to 0.81.

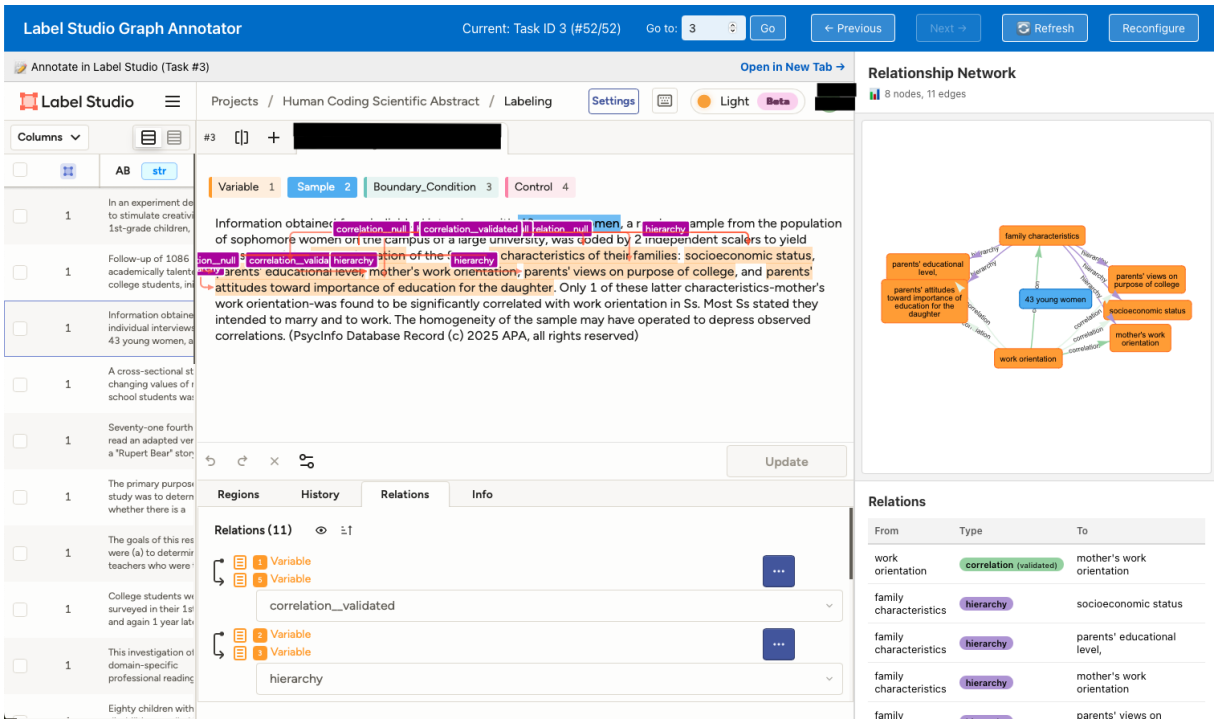

Figure 4: The user interface of annotation task

The strongest performance was observed for *Journal of Consulting and Clinical Psychology* (JCCP; $F1 = 0.807$), which also showed the highest precision (0.863) and high recall (0.806). In contrast, performance was lowest for *Behaviour Research and Therapy* (BRT; $F1 = 0.669$) and *Journal of Experimental Psychology: General* (JEP:G; $F1 = 0.694$). We additionally examined the average number of edges and the distribution of edge types across journals. Lower-performing journals did not systematically contain more edges, nor did they contain a higher proportion of difficult edge types, such as moderational or hierarchical edges. This suggests that differences in journal-level performance are unlikely to arise from differences in graph complexity or relation-type composition. Instead, the variation is more likely attributable to journal-specific reporting styles, such as differences in how explicitly relations are stated or how consistently variables are described.

Overall, the results indicate that the proposed extraction pipeline is temporally robust, with consistently strong performance across publication periods, while journal-level variation remains a more important source of performance heterogeneity.

A.3 Graph Evaluation Details

We evaluate each predicted graph against a gold graph in two stages: node alignment followed by edge scoring under multiple evaluation views.

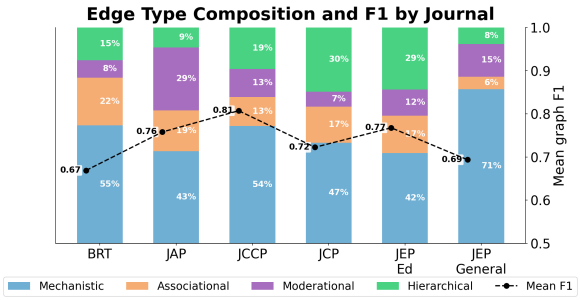

Figure 5: Edge Type Composition by Journal

Preprocessing. After the final complete graphs are formed in Step 5, we apply a set of preprocessing rules before evaluation. These rules programmatically propagate relationships from lower-level to higher-level nodes and remove duplicate relationships. This improves consistency across annotators and models, and reduces errors caused by the complexity of manual annotation.

Graph representation. Let the gold graph be $G = (V_G, E_G)$ and the predicted graph be $P = (V_P, E_P)$. Edges are directed and typed:

$$e = (u, v, \tau), \qquad \tau \in \mathcal{T}, \tag{6}$$

Year Range	Abstracts	Precision	Recall	F1
1960–1970	30	0.818	0.782	0.789
1970–1980	30	0.717	0.804	0.718
1980–1990	30	0.763	0.751	0.721
1990–2000	30	0.769	0.719	0.710
2000–2010	30	0.744	0.809	0.745
2010–2020	30	0.777	0.782	0.743
2020–2025	30	0.779	0.749	0.729

Table 4: Average edge-level precision, recall, and F1 scores across year ranges.

Journal	Abstracts	Precision	Recall	F1
BRT	35	0.716	0.695	0.669
JAP	35	0.798	0.749	0.758
JCCP	35	0.863	0.806	0.807
JCP	35	0.753	0.765	0.723
JEP	35	0.760	0.818	0.768
JEP:G	35	0.711	0.791	0.694

Note. BRT = Behaviour Research and Therapy; JAP = Journal of Applied Psychology; JCCP = Journal of Consulting and Clinical Psychology; JCP = Journal of Counseling Psychology; JEP = Journal of Educational Psychology; JEP:G = Journal of Experimental Psychology: General.

Table 5: Average edge-level precision, recall, and F1 scores across journals.

where

$$\mathcal{T} = \{\texttt{hierarchy}, \texttt{directional}, \texttt{correlational}, \texttt{moderation}\}. \tag{7}$$

Node alignment objective. We search for a partial mapping

$$\phi : V_G \to V_P \cup \{\emptyset\}, \tag{8}$$

where $\emptyset$ indicates an unmatched gold node. The mapping is injective over non-empty assignments:

$$\phi(u) = \phi(v) \neq \emptyset \Rightarrow u = v. \tag{9}$$

Thus, no two gold nodes can be matched to the same predicted node, while multiple gold nodes may remain unmatched.

A gold edge $(u, v, \tau) \in E_G$ is matched if and only if

$$\phi(u) \neq \emptyset, \qquad \phi(v) \neq \emptyset, \qquad (\phi(u), \phi(v), \tau) \in E_P. \tag{10}$$

Define

$$M(\phi) = |\{(u, v, \tau) \in E_G : (\phi(u), \phi(v), \tau) \in E_P\}| . \tag{11}$$

We optimize

$$\phi^\star = \arg\max_\phi M(\phi). \tag{12}$$

This alignment is structure-first. Since the same variable may be realized in an abstract by multiple surface forms, annotators may not produce mutually consistent node labels unless the variable inventory is specified in advance. Under this mapping, nodes with different surface forms may nevertheless be aligned when their typed relational roles are compatible under ϕ.

Search algorithm. Exact optimization of the alignment objective is combinatorial, so we use a branch-and-bound solver with a greedy warm start. First, we compute a greedy initial mapping ϕ_0, yielding an incumbent score $B = M(\phi_0)$. We then run depth-first search over partial mappings ϕ_d.

At each search state, we compute the number of already matched edges $C(\phi_d)$ and an admissible upper bound $U(\phi_d)$ on the number of additional gold edges that could still be matched by any completion of ϕ_d. We use only safe pruning: a branch is pruned when

$$C(\phi_d) + U(\phi_d) \leq B, \tag{13}$$

because even the best possible completion of that branch cannot improve the incumbent. This pruning cannot remove any alignment that could achieve a higher score, provided that $U(\phi_d)$ is an admissible upper bound.

In our implementation, $U(\phi_d)$ is computed typewise. For each relation type $\tau \in \mathcal{T}$, let $R_G^\tau(\phi_d)$ be the number of currently unmatched gold edges of type τ whose endpoints are not ruled out by the partial mapping, and let $R_P^\tau(\phi_d)$ be the number of available predicted edges of type τ that could still be used by unmapped or consistently mapped endpoints. The number of additional matches of type τ cannot exceed

$$\min(R_G^\tau(\phi_d), R_P^\tau(\phi_d)) . \tag{14}$$

Thus, we use

$$U(\phi_d) = \sum_{\tau \in \mathcal{T}} \min(R_G^\tau(\phi_d), R_P^\tau(\phi_d)) . \tag{15}$$

This bound is conservative: it may overestimate the number of additional matches because it ignores some joint compatibility constraints, but it never underestimates the maximum number of matches achievable by a completion. It is therefore safe for pruning.

Whenever a complete mapping with a higher score is found, the incumbent B and the best

mapping are updated. Because search efficiency depends on node order, we run several ordering strategies, including degree-descending, degree-ascending, label order, and seeded random orders, and retain the best incumbent found:

$$M^\star = \max_{s \in S} M_s, \tag{16}$$

where M_s is the best score found under strategy s.

A timeout is enforced per strategy. If any strategy terminates exhaustively, its result is optimal under the specified objective, since the ordering affects only the search order and not the search space. Under timeout, the solver is anytime: it returns the best incumbent found so far, and the resulting alignment score is a lower bound on the optimal alignment score.

Edge scoring. Let $m_G = |E_G|$, $m_P = |E_P|$, and $m_\star = M(\phi^\star)$. We compute:

$$\mathrm{P} = \frac{m_\star}{m_P}, \qquad \mathrm{R} = \frac{m_\star}{m_G}, \qquad \mathrm{F1} = \frac{2\mathrm{PR}}{\mathrm{P}+\mathrm{R}}. \tag{17}$$

These are the complete-graph structural scores under the active evaluation view.

For typed evaluation, we also compute per-type scores. For each $\tau \in \mathcal{T}$, define:

$$E_G^\tau = \{(u,v) : (u,v,\tau) \in E_G\}, \tag{18}$$

$$E_P^\tau = \{(x,y) : (x,y,\tau) \in E_P\}, \tag{19}$$

and

$$M_\tau = \left|\{(u,v) \in E_G^\tau : (\phi^\star(u), \phi^\star(v)) \in E_P^\tau\}\right|. \tag{20}$$

Then

$$\mathrm{P}_\tau = \frac{M_\tau}{|E_P^\tau|}, \qquad \mathrm{R}_\tau = \frac{M_\tau}{|E_G^\tau|}, \qquad \mathrm{F1}_\tau = \frac{2\mathrm{P}_\tau\mathrm{R}_\tau}{\mathrm{P}_\tau + \mathrm{R}_\tau}. \tag{21}$$

Evaluation views. We report three complementary alignment scores: M_{typed}, computed on the full graphs with directed, typed edges; M_{higher}, computed after projecting both graphs onto higher-level nodes by removing lower-level hierarchy children and hierarchy edges; and M_{agnostic}, computed on the full graphs after collapsing all edge types into a single undirected relation type.

Positive-edge scoring. For validated-only analysis, define

$$E_G^+ = \{e \in E_G : \mathrm{val}(e) = \texttt{validated}\}, \tag{22}$$

$$E_P^+ = \{e \in E_P : \mathrm{val}(e) = \texttt{validated}\}. \tag{23}$$

We then apply the same alignment and scoring procedure to (V_G, E_G^+) and (V_P, E_P^+).

Algorithm 1 Safe branch-and-bound alignment for typed directed graphs

Require: Gold graph $G = (V_G, E_G)$, predicted graph $P = (V_P, E_P)$, edge types $\mathcal{T}$, node order strategy s
Ensure: Best partial mapping $\phi^\star$ and matched-edge count $M^\star$

```
φ0 ← GreedyAlign(G, P, s)
B ← M(φ0)
φ* ← φ0
O ← OrderNodes(V_G, s)
procedure Search(d, φ_d)
    C ← MatchedEdges(φ_d, E_G, E_P)
    U ← UpperBound(φ_d, E_G, E_P, T)
    if C + U ≤ B then
        return                          ▷ safe pruning
    end if
    if d = |O| then
        if C > B then
            B ← C
            φ* ← φ_d
        end if
        return
    end if
    u ← O[d]
    for all x ∈ V_P not already used by φ_d do
        φ' ← φ_d ∪ {u ↦ x}
        Search(d + 1, φ')
    end for
    φ' ← φ_d ∪ {u ↦ ∅}
    Search(d + 1, φ')
end procedure
Search(0, ∅)
return φ*, B
```

Complexity. Let $n = |V_G|$, $m = |V_P|$, $e_G = |E_G|$, $e_P = |E_P|$, and $K = |S|$. Preprocessing operations, including canonicalization, deduplication, type filtering, and hierarchy transforms, are near-linear in edge count per graph in practice.

For alignment, the worst-case complexity is combinatorial:

$$O\left(\sum_{k=0}^{\min(n,m)} \binom{n}{k} \frac{m!}{(m-k)!}\right). \tag{24}$$

Branch-and-bound often reduces the number of explored states substantially, but it does not change the worst-case complexity class. Across K ordering strategies, the complexity is $O(K \cdot T_{\text{search}})$, with T_{search} capped by the timeout per strategy.

Space per graph pair is $O(n + m + e_G + e_P)$ for graph structures, adjacency, and hashed edge sets, plus $O(n + m)$ search-state overhead. At corpus scale, with A independent graph pairs and W workers, wall-clock runtime is approximately

$$T_{\text{wall}} \approx \frac{\sum_{i=1}^{A} T_i}{W} + \text{parallel overhead}. \tag{25}$$

Implementation note. The method is related to maximum common subgraph and maximum common edge-subgraph formulations, which search for structurally consistent correspondences between graphs under injective node mappings (McCreesh et al., 2017; Ndiaye and Solnon, 2011). However, our evaluator optimizes a task-specific objective: maximum typed-directed edge overlap under an injective partial node mapping. It is not a classical induced-MCS objective, because non-edges are not required to be preserved and extra predicted edges are penalized through precision rather than through the alignment constraint. The branch-and-bound solver is exact when search terminates exhaustively; under timeout, it behaves as an anytime solver and returns the best incumbent found.

A.4 Algorithm for Inter-coder Agreement

Pairwise Cohen's kappa. For two raters A, B, collect labels $\{y_A(i), y_B(i)\}_{i=1}^{N}$ across all aligned node-pair items. Let n_{pq} be confusion counts over classes $p, q \in \mathcal{C}$, and $N = \sum_{p,q} n_{pq}$. Observed agreement:

$$P_o = \frac{1}{N} \sum_{c \in \mathcal{C}} n_{cc}. \tag{26}$$

Chance agreement from marginals:

$$P_e = \sum_{c \in \mathcal{C}} \left(\frac{n_{c\cdot}}{N}\right) \left(\frac{n_{\cdot c}}{N}\right). \tag{27}$$

Cohen's kappa:

$$\kappa = \frac{P_o - P_e}{1 - P_e}. \tag{28}$$

We report pairwise κ for all rater pairs (e.g., A-B, A-Gold, B-Gold).

Three-rater Fleiss' kappa. For three-rater agreement, we use the gold graph as pivot: compute $\phi_{G \to A}$ and $\phi_{G \to B}$, keep gold nodes mapped to both raters, then form undirected node pairs on those shared gold nodes. Each item has three labels (y_G, y_A, y_B). Let $m = 3$ raters, $K = |\mathcal{C}| = 5$, N items, and n_{ij} the number of raters assigning item i to class j. Per-item agreement:

$$P_i = \frac{1}{m(m-1)} \sum_{j=1}^{K} n_{ij}(n_{ij} - 1). \tag{29}$$

Mean observed agreement:

$$\bar{P} = \frac{1}{N} \sum_{i=1}^{N} P_i. \tag{30}$$

Category prevalence:

$$p_j = \frac{1}{Nm} \sum_{i=1}^{N} n_{ij}. \tag{31}$$

Chance agreement:

$$\bar{P}_e = \sum_{j=1}^{K} p_j^2. \tag{32}$$

Fleiss' kappa:

$$\kappa_F = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e}. \tag{33}$$

A.4.1 Intercoder Agreement via MCS-Aligned Node-Pair Labels

We compute intercoder agreement on graph annotations using a two-stage protocol: structural node alignment followed by multiclass kappa on aligned node pairs.

Why alignment is required. Coders may use different surface names or granularity for conceptually similar variables. Direct string-based matching would therefore underestimate agreement. To make coder labels comparable, we first align nodes structurally using the same MCS-style logic used in our graph structural evaluation pipeline.

Graphs and labels. For each article, each human coder provides a preprocessed directed typed graph $G = (V, E)$, with edge types in

$$\mathcal{T} = \{\texttt{hierarchy}, \texttt{directional}, \texttt{correlational}, \texttt{moderation}\}. \tag{34}$$

Preprocessing is identical to the main graph evaluation workflow (correlation canonicalization, hierarchy handling, deduplication, and relation-priority collapse).

MCS-style node alignment. Given two rater graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, we estimate an injective mapping

$$\phi : V_1 \to V_2 \tag{35}$$

that maximizes typed directed edge consistency. For a candidate ϕ,

$$M(\phi) = |\{(u, v, \tau) \in E_1 : (\phi(u), \phi(v), \tau) \in E_2\}| \,. \tag{36}$$

We use a greedy multi-order approximation: run greedy mapping under several node orders (degree-descending, degree-ascending, label order, seeded random orders), then select the mapping with the largest $M(\phi)$:

$$\phi^\star = \arg \max_{\phi \in \Phi_{\text{greedy}}} M(\phi). \tag{37}$$

This reuses the same MCS-inspired structural matching principle as the main evaluator.

Agreement unit construction. After alignment, agreement is computed over *ordered node pairs* among mapped nodes:

$$\mathcal{U} = \{(u, v) : u, v \in \mathrm{dom}(\phi^\star),\ u \neq v\}. \tag{38}$$

Each ordered pair $(u, v) \in \mathcal{U}$ is one agreement item. If $m = |\mathrm{dom}(\phi^\star)|$, the article contributes

$$m(m - 1) \tag{39}$$

items. Using ordered pairs preserves directional orientation: the item (u, v) is distinct from (v, u).

5-class directed pair label. Each ordered pair receives one class from

$$\mathcal{C} = \{\texttt{none}, \texttt{hierarchy}, \texttt{directional}, \texttt{correlational}, \texttt{moderation}\}. \tag{40}$$

For rater 1, the label for (u, v) is determined by the directed edge from u to v in G_1. For rater 2, the corresponding label is determined by the directed edge from $\phi^\star(u)$ to $\phi^\star(v)$ in G_2. If no edge is present in the relevant direction, the label is none. If multiple relation types are present for the same ordered pair, we assign a single label by the following priority:

$$\texttt{directional} \succ \texttt{correlational} \succ \texttt{moderation} \succ \texttt{hierarchy} \succ \texttt{none}. \tag{41}$$

Thus, each rater contributes one nominal label per aligned ordered node pair, and disagreements in edge direction are reflected as label disagreements.

Interpretation. Because kappa is computed after MCS-aligned node correspondence, it measures agreement on relation coding decisions under structural correspondence rather than agreement on exact variable strings. Since agreement units are ordered node pairs, the statistic is sensitive to both relation type and directional orientation.

Complexity. Let $n_1 = |V_1|$, $n_2 = |V_2|$, $e_1 = |E_1|$, $e_2 = |E_2|$, and $|S|$ node-order strategies. Per article, alignment cost is approximately

$$O(|S| \cdot n_1 n_2 \cdot d) \,, \tag{42}$$

where d is local edge-consistency check cost with hashed edge lookups. If $m = |\mathrm{dom}(\phi^\star)|$, directed pair-label construction is

$$O(m(m - 1)) = O(m^2). \tag{43}$$

Kappa computation over N total directed-pair items and fixed $K = 5$ classes is linear in N. Thus corpus-level time is additive over articles:

$$\sum_{a=1}^{A} \left(T_{\text{align}}^{(a)} + O(m_a^2) \right) , \tag{44}$$

with memory dominated by graph structures, mappings, and label vectors:

$$O(|V| + |E| + N). \tag{45}$$

A.5 Node Validation

Because the graph-structural evaluation aligns nodes by maximizing matched typed directed edges rather than by directly comparing node labels, we conduct an additional validation of the node correspondences induced by the common graph. This analysis asks whether structurally aligned node pairs also tend to be semantically similar at the label level. If structurally aligned nodes were frequently semantically unrelated, the reported edge-level F1 could be inflated by graph-theoretic coincidences rather than reflecting meaningful relation extraction.

Table 6 and Figure 6 show the cosine similarity distribution for gold–predicted node pairs appearing in the common graph. Across all aligned node pairs, the mean embedding-based cosine similarity is 0.735. The majority of aligned pairs fall in the moderate-to-exact similarity range, suggesting that the MCS-derived alignment is usually pairing variables with related meanings rather than merely matching nodes with compatible graph positions.

Cosine similarity tier	Count	Share
≥ 0.9 exact	444	37.9%
0.7–0.9 high	256	21.9%
0.5–0.7 moderate	228	19.5%
0.3–0.5 weak	140	12.0%
< 0.3 low	103	8.8%

Table 6: Distribution of embedding-based cosine similarities between structurally aligned gold and predicted node labels.

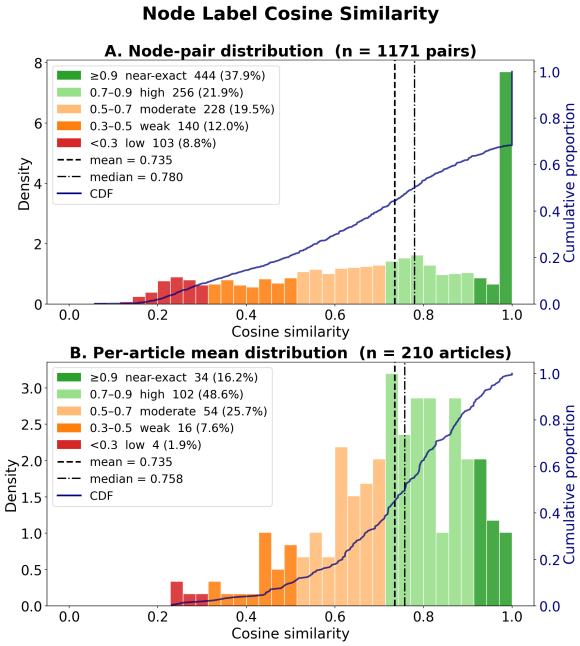

Figure 6: Distribution of cosine similarities between structurally aligned gold and predicted node labels.

We further stratify aligned node pairs by embedding-based cosine similarity and then randomly sample 100 pairs in proportion to the observed tier distribution. Human inspection of this sample assesses whether aligned nodes refer to the same empirical variable or construct despite surface-form differences. In this sample, 87 pairs were judged to refer to the same variable or construct. Almost all cosine similarity ≥ 0.5 pairs refer to the same construct. About one-third of the $0.3 - 0.5$ similarity tier node pairs are correctly paired; an example is 'general working model of attachment' and 'attachment variables,' yielding cosine similarity of 0.35 but refer to the same variable in the abstract. All < 0.3 sample pairs are incorrectly paired in reality. This result supports the validity of the typed-directed edge F1 metric: most matched edges are evaluated over semantically corresponding variable pairs, even though node-label similarity is not used as the primary alignment objective.

Edge Type	Mean Cosine Similarity
Mechanistic	0.732
Associational	0.752
Moderational	0.764
Hierarchical	0.727

Table 7: Edge type cosine similarity derived by averaging node similarity from both sides of the edge.

Table 7 reports edge-level cosine similarity by relation type, computed by averaging the cosine similarities of the two aligned node pairs forming each matched edge. The mean similarities are broadly comparable across relation types, ranging from 0.727 for hierarchical edges to 0.764 for moderational edges. This suggests that the structural alignment procedure produces semantically plausible node correspondences across all relation categories, rather than relying on a single relation type with unusually high label similarity. Hierarchical edges show the lowest mean similarity, which is expected because they often connect broader constructs with finer-grained variables or measurements.

Overall, the node-pair cosine analysis and manual validation provide a validity check for the structural evaluation procedure. Since the primary alignment objective does not use node-label similarity, the observed semantic similarity among common-graph node pairs suggests that the structural matching procedure usually recovers meaningful node correspondences. Therefore, the reported graph F1 can be interpreted as a relation-extraction score over largely semantically aligned variables, rather than as an artifact of arbitrary node matching.

A.6 AI Assistance Statement

The manuscript was written by the authors. ChatGPT (https://chat.openai.com/) was used only for language polishing. Claude Code was used for coding assistance such as debugging during implementation. Code and final manuscript were checked and verified by the authors.

A.7 Prompting Details

Prompt: Step 1 Variable Extraction

```
developer_content = dedent("""\
You are a scientific-text analyst
    specializing in variable detection in
    research abstracts. Extract only true
    study variables explicitly investigated,
```

```
     measured, categorized, manipulated, or
    reported as substantive constructs in
    the abstract. Minimize false positives.
    Do not include generic outcomes,
    contextual phrases, narrative
    descriptions, methodological wording,
    summary language, or overly broad
    fragments unless they are clearly
    treated as variables in the study.

When identifying variables, prefer specific
    constructs or named categories over
    umbrella phrases when both appear. Avoid
     including text spans that are merely
    explanatory, incidental, or derivative
    of another listed variable.

Success criteria: include a text span only
    if the abstract presents it as a study
    variable through direct investigation,
    measurement, categorization,
    manipulation, or substantive reporting.
    If a term is ambiguous, exclude it
    unless the abstract clearly treats it as
     a variable. When both a broad label and
     its explicit named subcategories appear
    , include the subcategories and include
    the broad label only if the abstract
    itself treats that label as a reported
    variable or category set.
""")

user_content = dedent(f"""\
<task>
Your task is to extract the variable
    relationships from the abstract.
    Identify and label which variables are
    the independent variables (IVs),
    dependent variables (DVs), moderators,
    and mediators, if any.
</task>

<context>
ABSTRACT:
{abstract}
</context>

<output_format>
Return a JSON object with this exact
    structure:

{
    "independent_variables": ["IV 1", "IV
    2"],
    "dependent_variables": ["DV 1", "DV 2"],
    "moderators": ["Mod 1"],
    "mediators": ["Med 1"],
    "final_variable_list": ["var 1", "var 2",
     "var 3"]
}
</output_format>

<rules>
Rules for final_variable_list:
- Must be the union of all variables across
    all four role lists
- Each variable appears only once
- If no moderators or mediators exist,
    return empty lists for those fields

Return ONLY the JSON object, with no
    additional text.
</rules>
""")
```

Prompt: Step 2 Hierarchy Detection and Variable Normalization

```
developer_content = dedent("""\
You are an expert scientific-text analyst
    specializing in identifying hierarchical
relationships and normalizing variables in
    research abstracts.

<definitions>
WHAT IS A VALID HIERARCHY?

CASE 1: Construct decomposition

The abstract identifies, distinguishes, or
    analyzes components or subthemes of a
    broader concept, AND identifying those
    components is the main reported finding
    (typical in qualitative / mixed-methods
    studies).

NOT Case 1:
- Components mentioned only for definition
    or background
- Appositional phrases ("X, including A, B,
    C") that merely list what something
    consists of without treating each as a
    reported finding

Examples:
- Big Five personality -> Extraversion
- Theme -> Subtheme
- "we constructed X themes"
- "the analysis revealed X themes/subthemes"
- "we identified X themes"
- "thematic analysis showed..."

CASE 2: Abstract-to-specific relationship

The abstract first mentions a broad
    construct-level variable/construct, AND
    THEN specifies lower-level dimensions/
    indicators that affect that same outcome
    .

In CASE 2, BOTH conditions must be met:

(a) The study measures/analyzes several
    characteristics/aspects of a given
    construct,
and these aspects are then each examined in
    relation to other variables. This may
    include validated effects, null effects,
     proposed relationships, or hypothesised
     relationships. However, do not invent
    an overarching higher-level variable
    yourself; the higher-level variable must
     be explicitly mentioned in the original
     text.
```

```
Therefore, you may not extract a shared head
     noun from multiple variable names and
    treat it as a higher-level construct.
    For example, do not extract "depictive
    gestures" from "dynamic depictive
    gestures" and "nondynamic depictive
    gestures" and code it as the higher-
    level variable.

(b) In the abstract, each lower-level
    variable appears in at least one
    directional, correlational, or
    moderation relationship with other
    variables.

RELAXATION RULES:

The higher-level variable does not
    necessarily need to be a measured
    variable or part of the model. It may
    also be a construct umbrella.

Stick to the abstract and check whether it
    mentions how the higher-level variable
    relates to other variables or how the
    study tests the relationship of the
    higher-level variable.

The abstract might not explicitly claim a
    decomposition from higher-level variable
     to lower-level variables. You are
    allowed to infer implicit decomposition
    between the higher-level variable and
    lower-level variables using your
    knowledge.

Example 1:
If the abstract first states a general
    relationship such as "this study
    examines how personality influences
    leadership," and then discusses several
    personality dimensions such as
    extraversion, agreeableness, and
    conscientiousness, then code:

Personality -> Extraversion
Personality -> Agreeableness
Personality -> Conscientiousness

Example 2:
If the abstract states that the study
    examines how training affects spelling
    memory, and then measures spelling
    memory through several indicators, then
    code:

Spelling memory -> Short-term spelling
    performance
Spelling memory -> Long-term spelling
    performance
Spelling memory -> Stability of spelling
    knowledge
</definitions>

<rules>
Different levels of the same variable:

Do not code a hierarchical relationship when
     a variable is only divided into
    different
levels of the same variable.

A variable may be divided only into
    different levels of the same variable,
    not into different subcomponents. The
    rule for judging whether something is "
    different levels of the same variable"
    is: does this component actually vary,
    get manipulated, or get
measured in this study?

For example, self-esteem (high/low) are
    levels, because "high self-esteem" is
    not itself a component that is
    manipulated or measured for variation.

This SHOULD NOT BE CODED AS A HIERARCHY. A
    hierarchy requires meaningful
    subcomponents that can vary
    independently. For example, extraversion
     can vary independently, and spelling
    memory can be measured by different
    indicators.

Code format for levels:
Variable (level 1, level 2)

Workflow:

1. Check for combined concepts

If a phrase expresses relationships among
    multiple concepts, for example
    interaction, association, effect, "A and
     B", or impact of A on B, split it into
    the individual
concepts.

Be cautious. You should only split combined
    concepts.

2. Identify the hierarchical relationships
    and levels in the abstract.

If you spot a hierarchical relationship,
    check whether both the higher-level
    variable and all lower-level variables
    appear in the variable list. If not, add
     variables from the abstract if they are
     included in a hierarchical relationship
    .

3. Code the hierarchy and levels.

For hierarchy relationships, put the upper-
    level variable in "variable" and lower-
    level variables in "components". Include
     the lower-level variables in "
    final_variable_list" as well.

For levels, do NOT put different levels in
    components, and do NOT put different
    levels in final_variable_list. Only code
     them as:

Variable (level 1, level 2)
```

```
Do NOT change a variable's role position. If
     a variable is an independent variable
    in the input variable list, do not move
    it to moderators or other positions.

4. Final check

- Make sure all hierarchical relationships
    in the abstract are coded, and both the
    higher-level variable and all lower-
    level variables in each hierarchy
    relationship are added into the variable
     list.
- Check again whether each hierarchy
    relationship meets either CASE 1 or CASE
     2. If not, delete that hierarchy
    relationship.
- Check whether variables coded as hierarchy
     sub-levels actually have manipulated or
     observed variation in the text. If
    there is no actual variation, reconsider
     whether it should be treated as a
    different level.

Use exact variable names where possible.

Return only valid JSON.
</rules>
""")

user_content = dedent(f"""\
<task>
You are given an abstract and a variable
    list. Normalize the variable list,
    detect valid hierarchy relationships,
    and distinguish hierarchy from different
     levels of the same variable.
</task>

<output_format>
Return a JSON object with this exact
    structure:

{
  "hierarchy": {
    "hierarchical_relationships": [
      {
        "higher_level_variable": "",
        "lower_level_variable": ""
      }
    ]
  },
  "final_variable_list": ["", ""],
  "independent_variables": [
    {
      "variable": "",
      "components": ["", ""]
    }
  ],
  "dependent_variables": [
    {
      "variable": "",
      "components": ["", ""]
    }
  ],
  "moderators": ["", ""],
  "mediators": ["", ""],
  "other_variables_controls": [
    {
      "name": "",
      "role": ""
    }
  ]
}
</output_format>

<input_variables>
VARIABLE LIST:
{step1_json}
</input_variables>

<context>
ABSTRACT:
{abstract}
</context>
""")
```

Prompt 3: Relevant Empirical Sentence Extraction

```
developer_content = dedent("""\
You are a scientific-text analyst.

<rules>
Your job is to extract ONLY the sentences
    that report empirical relationships
    tested or found in this paper.

Keep only the sentences that explicitly
    state an empirical relationship and:
(a) propose it,
(b) test or estimate it, or
(c) report findings about it.

The most critical rule is to keep sentences
    that describe:
- what/how this study does
- what this study tests
- what this study found

Exclude sentences describing:
- previous research has found
- the literature shows
- prior work
- other background information
- ground truth statements not tied to the
    current study's method or findings

INCLUDE these sentence types:
- Hypotheses / research questions:
  phrases such as "this study", "we
    hypothesize", "we predict", "we test",
  "we examine", "we propose", "we argue", "
    this research examine"

- Methods / analysis:
  phrases such as "we argue", "we conducted",
     "we examine", "we test",
  "we analyze", "we estimate", "model", "our
     model introduce", or other
  sentences that describe the method used to
     test the relationship between variables

- Results / findings:
```

```
  phrases such as "we find", "results show",
      "our findings", "we demonstrate",
  "we argue"

EXCLUDE these sentence types:
- Prior work / background:
  sentences mentioning prior research, the
     literature, prior work, or describing
  ground truth / phenomena without linking
     to the current study's method or
     findings

- Implications / practical advice:
  sentences such as "we discuss the
     implications", "our results contribute
     to the
  theory", "our findings have implications
     for"

CONFLICT RULE:
If a sentence contains both included and
    excluded elements, prioritize inclusion.
Include the sentence if it contains any
    included element, even if it also
    contains
excluded elements.

Output only exact sentences from the
    abstract. Do not paraphrase.
Return only valid JSON.
</rules>
""")

user_content = dedent(f"""\
<task>
Extract only the relevant empirical
    sentences from this abstract.
</task>

<input_variables>
VARIABLE LIST:
{vars_json}
</input_variables>

<context>
ABSTRACT:
{abstract}
</context>

<output_format>
Return a JSON object with this exact
    structure:

{"relevant_sentences": ["The exact first
    sentence from the abstract.", "The exact
     second sentence from the abstract."]}
</output_format>

<step_specific_rules>
- "relevant_sentences" must contain EXACT
    text from the abstract
- Do not paraphrase
- If no relevant sentences are found, return
     an empty list for "relevant_sentences"

Return ONLY the JSON object, no additional
    text.
</step_specific_rules>
""")
```

Prompt 4: Empirical Relationship Extraction

```
developer_content = dedent("""\
You are an expert scientific-text analyst
    specializing in extracting empirical
relationships from research abstracts.

<rules>
VARIABLE LIST ENFORCEMENT

- Only extract relationships where ALL
    variables are in the VARIABLE LIST.
- If a variable is semantically equivalent
    to one in the list, normalize to the
    list
  version.
- If no close match exists, discard the
    relationship.

ONE CATEGORY PER PAIR

- A variable pair can appear in only one
    category.
- If both directional and correlational
    apply, keep directional.

USE RELEVANT SENTENCES

- Ground all relationships primarily in the
    provided relevant sentences.

RELATIONSHIP TYPES

DIRECTIONAL (Var1 -> Var2):
One variable affects or influences another
    with clear direction.

CORRELATIONAL (Var1 <-> Var2):
Non-directional association with no causal
    claim.

MODERATION:
A third variable modifies the relationship
    between two others.

IMPORTANT:
- Moderation must also include the
    underlying X-Y relationship.
- Interaction effects produce both
    directional and moderation entries.

VALIDATION STATE

- validated: supported by results
- null: tested but not supported
- hypothesized: proposed but not confirmed

GENERAL

- Use exact variable names from VARIABLE
    LIST.
- Return only valid JSON.
</rules>
""")

user_content = dedent(f"""\
<task>
Extract all directional, correlational, and
    moderation relationships from the
    abstract.
</task>

<procedure>
Step 0: Keep only variables from the
    VARIABLE LIST.

Step 1: Use RELEVANT SENTENCES as primary
    evidence.

Step 2: Detect moderation first.

Step 3: Classify relationships:
- causal -> directional
- non-causal association -> correlational
- conditional relationship -> moderation

Step 4: Assign validation state:
- validated
- null
- hypothesized

Step 5: Ensure no pair appears in multiple
    categories.

Step 6: Remove any relationship using
    variables outside the list.

Step 7: Re-check completeness, especially
    for null effects.
</procedure>

<output_format>
Return a JSON object with this exact
    structure:

{
  "directional": [
    ["source_var", "direction", "target_var",
      "validated|null|hypothesized"]
  ],
  "correlational": [
    ["var1", "correlation", "var2", "
    validated|null|hypothesized"]
  ],
  "moderation": [
    {
      "moderator": "moderator_var",
      "moderated_variables": ["var1", "var2
    "],
      "validation": "validated|null|
    hypothesized"
    }
  ]
}
</output_format>

<input_variables>
VARIABLE LIST:
{variable_list_str}
</input_variables>

<context>
ABSTRACT:
{abstract}
</context>
```

```
<evidence>
RELEVANT SENTENCES:
{sentences_str}
</evidence>
""")
```

Prompt 5: Relationship Validation and Correction

```
developer_content = dedent("""\
Your task is to validate variable names and
    variable relationships, including
    directional, correlational, moderation,
    and hierarchy relationships, using the
following rules.

<rules>
-ONLY allow removal of edges marked as "
    hypothesized" in this step.
-Do NOT remove directional or correlational
    relationships coded as "validated" or
"null" in this step.
-ONLY allow changing categorization between
    directional and correlational
    relationships
in this step.
</rules>

<definitions>
RELATIONSHIP TYPES

DIRECTIONAL (Var1 -> Var2)

Code as directional when the sentence or
    immediate context explicitly states that
     Var1 predicts, affects, influences,
    increases, decreases, causes, explains,
    contributes to, leads to, results in,
    enables, determines, drives, mediates,
    or
otherwise changes Var2.

For example, code A -> B as directional when:

- A impacts, increases, or influences B
- a neural region A enables a function B
- past feature A predicts future outcome B
- manipulated independent variable A results
     in differences in outcome variable B in
     experiments
- B mediates the relationship between A and
    C, coded as A -> B -> C

CORRELATIONAL (Var1 <-> Var2)

Code as correlational when the text only
    states that Var1 and Var2 are associated
    , correlated, related, linked, connected
    , coupled, covarying, or different
    across naturally occurring groups,
    without explicitly assigning directional
     or predictor/outcome roles.

MODERATION

A third variable modifies the relationship
    between other variables.
- M impacts the strength of the relationship
     between X and Y: code as M moderates X-
    Y
- M impacts the mediated relationship
    between X and Y through Z: code as M
    moderates X-Z-Y

IMPORTANT:
- Moderation must also include the
    underlying relationship.
- Interaction effects produce both
    directional and moderation entries.

VALIDATION STATE

- validated: supported by results
- null: tested but not supported
- hypothesized: proposed but not confirmed
</definitions>
""")

user_content = dedent(f"""\
<task>
Validate variable names and variable
    relationships, including directional,
    correlational, moderation, and hierarchy
     relationships, using the following
    rules.
</task>

<procedure>
Step 0: Remove any relationship that
    duplicates a hierarchy edge.

Step 1: Check validation labels.

Remove weak relationships labeled "
    hypothesized" unless the abstract
    explicitly states them as the primary
    hypothesis or they are tested in the
    model.

Step 2: Distinguish direction vs.
    correlation.

Check whether the previous coding
    incorrectly treated a directional
    relationship as a correlation, or a
    correlation as a directional
    relationship. When a directional or
    correlational relationship is miscoded,
    convert it to the correct type instead
    of deleting it.

If the abstract includes equal evidence for
    correlation and direction, code it as
    directional.

Step 3: Validate moderation.

Ensure the moderated relationship is
    correctly specified. It may involve more
     than two variables. For example, if the
```

[2]In the prompt above, the relationship labels were later renamed in our article: `correlational` edges correspond to what we call *associational* edges, and `directional` edges correspond to what we call *mechanistic* edges.

```
      abstract says "M moderates the mediated
      relationship," code M as moderating all
      variables in that mediated path.

When an abstract explicitly claims that the
     interaction or interactive effect of A
     and B significantly impacts C, make sure
      you extract two directional
     relationships:

- A -> C
- B -> C

and two moderation relationships:

- A moderates B-C
- B moderates A-C

Check whether a boundary condition has been
     mistakenly counted as moderation.

For example:
- "The results are validated across
     different conditions" is not moderation.
- A moderation requires a comparison across
     different levels of the moderator.
- "The relationship between A and B does not
      differ under different levels of M" is
     moderation with null validation.

Step 4: Check repetitive variable names.

Ensure each variable name refers to a
     distinct variable in the abstract.
     Identify synonyms and coreference
     relations. Merge them.

If multiple names refer to the same
     underlying variable or relationship,
     merge them as A(B), and update the
     relationships accordingly.
</procedure>

<output_format>
Return a JSON object with this exact
     structure:

{
  "hierarchy": [
    ["upper_var", "hierarchy", "lower_var"]
  ],
  "directional": [
    ["source_var", "direction", "target_var",
      "validated|null|hypothesized"]
  ],
  "correlational": [
    ["var1", "correlation", "var2", "
     validated|null|hypothesized"]
  ],
  "moderation": [
    {
      "moderator": "mod_var",
      "moderated_variables": ["var1", "var2
     "],
      "validation": "validated|null|
     hypothesized"
    }
  ],
  "justifications": [
    {
      "change": "description",
      "reason": "based on abstract"
    }
  ]
}
</output_format>

<context>
ABSTRACT:
{abstract}
</context>

<input_variables>
VARIABLE LIST:
{variable_list}
</input_variables>

<current_state>
CURRENT HIERARCHY:
{step2_hierarchy}

CURRENT DIRECTIONAL:
{step4_directional}

CURRENT CORRELATIONAL:
{step4_correlational}

CURRENT MODERATION:
{step4_moderation}
</current_state>
""")
```