GraphAgent: Agentic Graph Language Assistant

Yuhao Yang[1], **Jiabin Tang**[1], **Lianghao Xia**[1], **Xingchen Zou**[2],
Yuxuan Liang[2], **Chao Huang**[†1]

[1]The University of Hong Kong, [2]HKUST (GZ)

Abstract

Real-world data combines structured (*e.g.*, graph connections) and unstructured (*e.g.*, text, visuals) formats, capturing explicit relationships (*e.g.*, social links) and implicit semantic interdependencies (*e.g.*, knowledge graphs). We propose GraphAgent, an automated agent pipeline addressing both explicit and implicit graph-enhanced semantic dependencies for predictive (*e.g.*, node classification) and generative (*e.g.*, text generation) tasks. GraphAgent integrates three components: (i) a Graph Generator Agent creating knowledge graphs for semantic dependencies; (ii) a Task Planning Agent interpreting user queries and formulating tasks via self-planning; and (iii) a Task Execution Agent automating task execution with tool matching. These agents combine language and graph language models to reveal complex relational and semantic patterns. Extensive experiments on diverse datasets validate GraphAgent's effectiveness in graph-related predictive and text generative tasks. GraphAgent is open-sourced at: `https://github.com/HKUDS/GraphAgent`.

1 Introduction

Real-world data exists in a complex ecosystem of interconnected types. Structured data, such as graph-based connections, captures explicit relationships like social networks and user interaction patterns (Fey et al., 2023). Unstructured data, including text and visual content, reveals implicit semantic relationships among entities (Zhong and Mottin, 2023). Integrating these diverse data formats is critical for modern applications, enabling comprehensive and nuanced analysis of complex real-world scenarios (Lu et al., 2024).

Graphs effectively represent relational information across domains. In academic networks, citation graphs connect papers (nodes) through citations (edges) (Chen et al., 2023; Wang et al., 2022), with paper content providing unstructured data for analyzing themes, methodologies, and findings. Combining structured citation data with text enables trend identification and supports *Graph-enhanced Text Generative Tasks*, such as knowledge summaries and scientific question-answering. In e-commerce, user behavior graphs, paired with textual data like product reviews and descriptions, improve recommendation accuracy (Shuai et al., 2022; Li et al., 2023), framing *Graph-related Predictive Tasks* for user-item interaction forecasting.

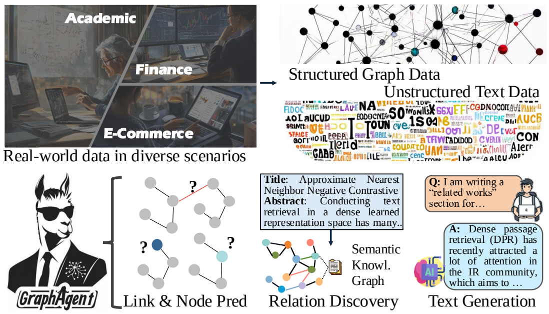
Academic
Finance
E-Commerce
Real-world data in diverse scenarios
Structured Graph Data
Unstructured Text Data
GraphAgent
?
?
?
Link & Node Pred
Title: Approximate Nearest Neighbor Negative Contrastive
Abstract: Conducting text retrieval in a dense learned representation space has many..
Semantic Knowl. Graph
Relation Discovery
Q: I am writing a "related works" section for…
A: Dense passage retrieval (DPR) has recently attracted a lot of attention in the IR community, which aims to …
AI
Text Generation

Figure 1: GraphAgent processes structured and unstructured data, adapting to diverse downstream tasks.

Graph learning methods, particularly Graph Neural Networks (GNNs), excel at mapping nodes and edges into latent representations through message-passing (Hamilton, 2020; Dai et al., 2022; Liu et al., 2022). However, GNNs primarily focus on explicit graph connections, often neglecting complex semantic dependencies in linked textual data, and show limited generalization in zero-shot real-world tasks due to task-specific training needs (Xia and Huang, 2024; Mao et al., 2024). Recent LLM-based approaches, like GraphGPT (Tang et al., 2024a) and LLaGA (Chen et al., 2024a), convert graph data into LLM-compatible tokens but focus on conventional tasks like node classification and link prediction, limiting their ability to handle diverse structured and unstructured data flexibly. This raises a key question: *How can individuals without graph theory or machine learning expertise analyze graph data using natural language to obtain predictions and insights?*

---

† Chao Huang is the corresponding author.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 26349–26368
November 4-9, 2025 ©2025 Association for Computational Linguistics

In this paper, we propose GraphAgent, a fully automated framework to analyze structured and unstructured data, addressing both graph-related predictive and generative tasks. Built on an agentic architecture, GraphAgent enables natural language interaction, empowering non-experts to derive tailored insights from graph-structured data. We tackle three challenges: i) **Constructing Semantic Relationships**: Deriving latent semantic connections from complex data; ii) **Automating Query Understanding**: Interpreting user queries and formulating them into predictive or generative tasks; iii) **Efficient Task Execution**: Accurately implementing tasks to deliver correct results. GraphAgent addresses these with three components: a **Graph Generator Agent** that constructs Semantic Knowledge Graphs (SKGs) from text, a **Task Planning Agent** that interprets queries and plans tasks, and a **Graph Action Agent** that automates task execution.

In summary, our contributions are:

- **Complex Practical Data Integration**. Our framework handles real-world scenarios by merging structured and unstructured data with graph-based relationships, supporting predictive and text generation tasks.
- **Multi-Agent Workflow**. We introduce GraphAgent, an automated graph language assistant, integrates structured and unstructured data analysis. It constructs semantic knowledge graphs (SKGs) from text, formulates predictive and generative tasks, and executes them, uncovering relational and semantic dependencies for diverse tasks.
- **Experimental Evaluation**. GraphAgent excels on diverse data in graph predictive and text generative tasks, with ablation studies confirming component effectiveness. Using small open-source LLMs (*e.g.*, LLaMA-3-8B), it outperforms closed-source models (*e.g.*, GPT-4, Gemini) in generation tasks.

2 Methodology

2.1 Preliminaries

Graph-empowered Agents. Our GraphAgent introduces an automated agentic pipeline for graph predictive and text generation tasks, formulated as $\mathcal{Y} = f(\mathcal{O}; \text{LLM})$. Here, the agentic function $f(\cdot)$ processes an **Observation** $\mathcal{O}$, comprising structured (*e.g.*, explicit graph connections) or unstructured data (*e.g.*, text), to produce an **Action** $\mathcal{Y}$, such as node classifications or text summarization with implicit entity interdependencies.

Graph-Structured Data. In GraphAgent, structured and unstructured data are represented as heterogeneous graphs, $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{N}, \mathcal{R})$, where $\mathcal{V}$ denotes entities, $\mathcal{E}$ represents edges, and $\mathcal{N}$ and $\mathcal{R}$ indicate node and edge types, respectively. Each edge has a meta-type (n_h, r_i, n_t) for the head node n_h, relation r_i, and tail node n_t.

2.2 Graph Generation Agent

To uncover the rich contextual information within unstructured data, GraphAgent designs a Graph Generation Agent that automatically constructs meaningful Semantic Knowledge Graphs (SKGs) from any textual input. For example, for a paper abstract that includes the sentence, “Contrastively trained text-image models have the remarkable ability to perform zero-shot classification”, the model can extract relevant entity nodes such as “text-image models” and “zero-shot classification”.

Iterative Two-Phase Graph Generation Workflow. To capture complex implicit entity-wise dependencies, our graph generation agent operates through an automated two-phase workflow: (1) **Scaffold Knowledge Entity Extraction** and (2) **Knowledge Description Augmentation**. The first phase is dedicated to identifying key knowledge entities or concepts as scaffold knowledge nodes from the provided text, regardless of its format. Specifically, this phase can be formulated as:

$$\mathcal{V}_{\text{scaffold}}^{k=0} = \text{LLM}(\mathbf{x}_{\text{sys\_sk}}, \mathbf{g}_s), \tag{1}$$

where $\mathbf{g}_s$ represents the input unstructured text data, while $\mathbf{x}_{\text{sys\_sk}}$ denotes the system prompt for extracting scaffold knowledge nodes. We adopt an iterative approach to capture both high-level and fine-grained semantic dependencies among multi-grained entities. For example, in an academic paper, high-level entities might include "Machine Learning," while fine-grained entities could be "Self-Supervised Learning" and "Graph Neural Network". Specifically, $\mathcal{V}_{\text{scaffold}}^{k=0}$ refers to the generated vertices during the initial iteration ($k = 0$).

The second phase of knowledge augmentation centers on enhancing and enriching the textual descriptions of the generated entity nodes to ensure accurate, comprehensive, and contextually appropriate language modeling. This critical step ensures that each entity is represented with sufficient detail and semantic clarity. Formally, we define this

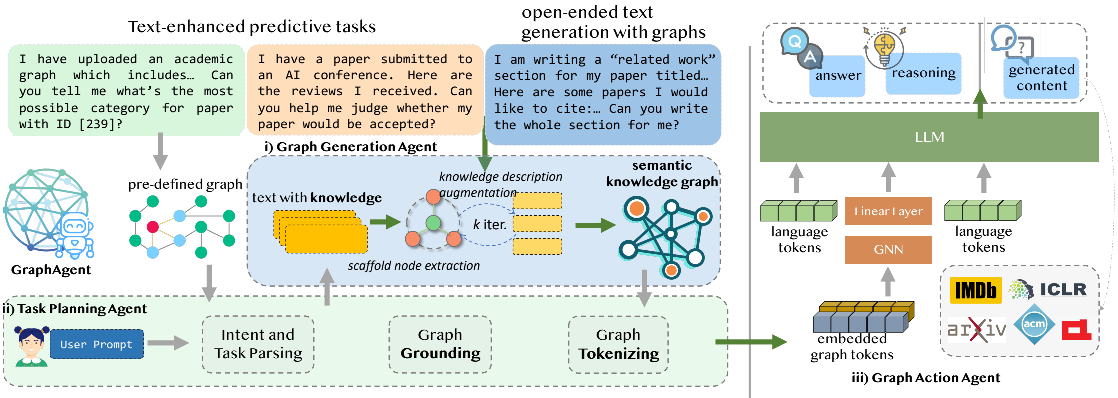
Text-enhanced predictive tasks
open-ended text generation with graphs
I have uploaded an academic graph which includes… Can you tell me what's the most possible category for paper with ID [239]?
I have a paper submitted to an AI conference. Here are the reviews I received. Can you help me judge whether my paper would be accepted?
I am writing a "related work" section for my paper titled… Here are some papers I would like to cite:… Can you write the whole section for me?
i) Graph Generation Agent
GraphAgent
pre-defined graph
text with knowledge
knowledge description augmentation
k iter.
scaffold node extraction
semantic knowledge graph
ii) Task Planning Agent
User Prompt
Intent and Task Parsing
Graph Grounding
Graph Tokenizing
answer
reasoning
generated content
LLM
language tokens
Linear Layer
GNN
language tokens
embedded graph tokens
IMDb
ICLR
arXiv
acm
iii) Graph Action Agent

Figure 2: The overall framework of the proposed GraphAgent.

phase as follows:

$$\mathcal{C}^{k=0}_{\text{scaffold}} = \text{LLM}(\mathbf{x}_{\text{sys\_ka}}, \mathbf{g}_s, \mathcal{V}^{k=0}_{\text{scaffold}}). \qquad (2)$$

where $\mathcal{C}^{k=0}_{\text{scaffold}}$ denotes the node-specific descriptions, while $\mathbf{x}_{\text{sys\_ka}}$ denotes the system prompt for knowledge augmentation. To iteratively execute this two-phase workflow, GraphAgent uses the textual augmentation output from the previous round as the implicit graph input for the next round:

$$\mathcal{V}^{k=j}_{\text{scaffold}} = \text{LLM}(\mathbf{x}_{\text{sys\_sk}}, \mathcal{C}^{k=j-1}_{\text{scaffold}}) \qquad (3)$$

$$\mathcal{C}^{k=j}_{\text{scaffold}} = \text{LLM}(\mathbf{x}_{\text{sys\_ka}}, \mathcal{C}^{k=j-1}_{\text{scaffold}}, \mathcal{V}^{k=j-1}_{\text{scaffold}}). \qquad (4)$$

We then merge the nodes and descriptions generated across different iterations to form the final node set: $\mathcal{V}_{\text{skg}} = \bigcup \mathcal{V}^{k}_{\text{scaffold}}$ and $\mathcal{C}_{\text{skg}} = \bigcup \mathcal{C}^{k}_{\text{scaffold}}$. The relationships among these nodes, denoted as $\mathcal{E}_{\text{skg}}$, are established based on their derivation: if a new node is generated from the textual description of a node in the previous iteration, we connect these two nodes in the semantic knowledge graph. The system prompts used for graph generation are detailed in Table 8 in the Appendix.

2.3 Task Planning Agent

With both structured and unstructured data represented as graphs, GraphAgent employs a task planning agent to automatically interpret user queries and transform the graph data into a unified embedding structure. This facilitates easier utilization by the subsequent predictive and generative modules. Input-output examples of the task planning agent is provided in Table 14 in the Appendix.

2.3.1 Intent Identification and Task Formulation

The task planning agent is initially tasked with formulating meaningful predictive or generative tasks based on the user query prompt. Given a user query prompt $\mathbf{x}_{\text{usr\_p}}$ and a predefined system prompt for task parsing $\mathbf{x}_{\text{sys\_tp}}$, the task planning agent formulates the intended task as follows:

$$\mathbf{g}_s, \mathbf{x}_{\text{usr\_ann}}, \mathbf{t}_{\text{usr}} = \text{LLM}(\mathbf{x}_{\text{sys\_tp}}, \mathbf{x}_{\text{usr\_p}}), \qquad (5)$$

This intent identification and task formulation procedure generates three fundamental types of task attributes within our agent architecture, which is specifically defined as follows:

- **Source graph** $\mathbf{g}_s$ represented by formatted files, textual graph descriptions, or plain documents.
- **Task type** $\mathbf{t}_{\text{usr}}$ is inferred from the query prompt and can be one of `"predictive_predefined"`, `"predictive_wild"`, or `"open_generation"`. This task type symbol is used to automatically select system prompt templates during training or inference for different tasks.
- **User annotation** $\mathbf{x}_{\text{usr\_ann}}$ includes additional task information, such as task descriptions, label candidates for predictive tasks, and generation requirements for generative tasks.

To construct grounded graph tokens that can be understood by the subsequent action agent, the task planning agent follows two stages: i) Graph-Token Grounding, converting graphs with nodes and edges into grounded Python objects; ii) Graph Tokenization, generating tokens from the input that preserve complex interdependencies among graph-structured entities.

2.3.2 Graph-Token Grounding

Our framework reads graph nodes and edges and converts them into grounded Python objects using a graph-building and wrapping tool. Notably, our model can handle diverse graph inputs, regardless of whether an explicit graph with predefined nodes and edges is present. For simplicity, we will demonstrate a scenario where the user uploads a predefined graph. For example, the query prompt might be: "...I want to know which category is correct for the node with ID [305]..." with uploaded graph files such as ["node_list.txt", "edge_list.txt"]. To build a grounded graph object in Python, we utilize the graph-building and wrapping tool (GBW_Tool(·)) with PyG (Fey and Lenssen, 2019) to add nodes and construct edges. Since user-uploaded graphs can have arbitrary node and edge types, we standardize the graphs as heterogeneous graphs, where s_i and r_i represent the types of each node and edge, respectively. Formally, a heterogeneous graph is constructed as:

$$\mathcal{G}^{\text{exp}} = \text{GBW\_Tool}(\mathcal{V}, \mathcal{E}, \mathcal{N}, \mathcal{R}) \tag{6}$$

$$\mathcal{G}^{\text{skg}} = \text{GBW\_Tool}(\mathcal{V}_{\text{skg}}, \mathcal{E}_{\text{skg}}, \mathcal{N}_{\text{skg}}, \mathcal{R}_{\text{skg}}) \tag{7}$$

where $\mathcal{V}, \mathcal{E}, \mathcal{N}, \mathcal{R}$ represent the nodes, edges, node types, and edge types of the explicit graph, respectively. They are obtained by parsing the graph input $\mathbf{g}_s$. Similarly, $\mathcal{V}_{\text{skg}}, \mathcal{E}_{\text{skg}}, \mathcal{N}_{\text{skg}}, \mathcal{R}_{\text{skg}}$ denote the corresponding graph components generated by the aforementioned Graph Generation Agent. This graph grounding module enables our model to convert graph data from various representations and forms into unified `Python` objects, facilitating their subsequent utilization.

2.3.3 Graph Tokenization

The Task Planning Agent converts discrete nodes and edges into embedded representations suitable for action agents based on graph LLMs. This tokenization process consists of two stages: first, encoding the graph into embeddings, and second, retrieving the nodes and their neighbors to create input graph tokens. For the embedding process, we employ a pre-trained text encoder $f_{\text{text\_enc}}$ and a pre-trained GNN f_{gnn}. Graph tokens are generated by initially encoding the textual features $\mathbf{c}$ of the graph nodes and their meta types using the text encoder, followed by modeling geometric features.

$$\mathbf{e}_i^{\text{text}} = f_{\text{text\_encoder}}(\mathbf{c}_i);\ \mathbf{e}_{s_i|r_i}^{\text{text}} = f_{\text{text\_encoder}}(\mathbf{c}_{s_i|r_i}) \tag{8}$$

$$\mathbf{e}_i^{\text{gnn}} = f_{\text{gnn}}(\mathbf{e}_i^{\text{text}}, \mathbf{e}_{s_i}^{\text{text}}, \mathbf{e}_{r_i}^{\text{text}}, \mathcal{V}, \mathcal{E}). \tag{9}$$

For each central node i in our heterogeneous graph, we systematically apply a graph sampling tool to create the subgraph input for the subsequent action agent, which can be formulated as follows:

$$[\mathbf{e}_{N_i}^{\text{gnn}}] = \text{Sampling\_Tool}(\mathcal{G}, \mathbf{E}^{\text{gnn}}, i) \tag{10}$$

2.4 Graph Action Agent

To enhance the capabilities of graph encoding and prediction/generation, we incorporate a trainable Graph Action Agent into our GraphAgent framework, based on the Graph LLM architecture (Tang et al., 2024b; Chen et al., 2024a). This Graph Action Agent is specifically trained to optimize performance for both predictive and text generation tasks involving graph data.

2.4.1 Cross-Task Graph Agent

The graph action agent is capable of handling two categories of diverse tasks, as shown below. The details on the system prompt builder and examples of system prompts are shown in Table 8.

- **Predictive Graph-Language Tasks**. These tasks focus on generating predictions based on user prompts, utilizing both structured and unstructured data. Examples include node classification and link prediction for explicit graph data, as well as document classification based on extracted implicit semantic knowledge graphs (SKGs), such as categorizing news articles. When using implicit SKGs to complement explicit graphs, the graph generator agent uses the observed explicit nodes as initial scaffold nodes to build the SKG. Specifically, for these tasks, our model constructs a system prompt that effectively guides the LLM toward task-specific objectives:

  $$\mathbf{x}_{\text{sys\_pred\_i}} = f_{\text{sys}}(\mathbf{t}_{\text{usr}}, \mathbf{x}_{\text{usr\_ann}}, \mathbf{g}_s), \tag{11}$$

  where the prompt builder function f_{sys} creates an appropriate system prompt based on the task type and user annotations, incorporating $\mathbf{g}_s$ for node or graph textual information. The predictive graph-language tasks are then defined as follows:

  $$\mathbf{y}_{\text{pred}}, \mathbf{y}_{\text{reasoning}} = \text{LLM}(\mathbf{x}_{\text{sys\_pred\_i}}, \{\mathcal{G}^{\text{exp}}|\mathcal{G}^{\text{skg}}\}), \tag{12}$$

  where $\{\mathcal{G}^{\text{exp}}|\mathcal{G}^{\text{skg}}\}$ indicates that the agent can utilize either $\mathcal{G}^{\text{exp}}, \mathcal{G}^{\text{skg}}$ or both. In this context, the LLM generates accurate predictions and reasoning in response to the user's query prompt.

- **Generative Graph-Language Tasks**. The discovered SKGs can serve as robust and comprehensive references for generative language tasks, such as text generation and summarization. These open-ended tasks are typically prompted in

Table 1: Dataset details for training and evaluation. "NC" is short for node classification.

	IMDB	**ACM**	**Arxiv-Papers**	**ICLR-Peer Reviews**	**Related Work Generation**	**GovReport Summarization**
Task Type	Predictive	Predictive	Predictive	Predictive	Generative	Generative
Sub-Task	NC	NC	Paper Classification	Paper Judgement Prediction	Text Generation	Text Summarization
Pre-defined Graph?	✓	✓	×	×	×	×
#Train Samples	2,400	-	5,175	3,141	4,155	-
#Eval Samples	-	1000	500	500	500	304
#Tokens	10M	0.8M	30M	45M	93M	2M
#Pre-defined Graph Nodes	11,616	10,942	-	-	-	-
SKG Source	People Entities	Paper	Paper	Paper, Reviews	Multiple Papers	Documents
#SKG Nodes	57,120	20,388	153,555	161,592	875,921	15,621

a direct text format that implicitly contains knowledge, without the need for predefined graphs. For example, to summarize a news article, an SKG $\mathcal{G}^{\text{skg}}$ is automatically constructed from the article's content, which includes rich entities and connections that aid in the summarization task. Additionally, a system prompt is automatically generated to enhance the content generation quality using the graph-structured information:

$$\mathbf{x}_{\text{sys\_gen\_i}} = f_{\text{sys}}(\mathbf{t}_{\text{usr}}, \mathbf{x}_{\text{usr\_ann}}, \mathbf{g}_s) \tag{13}$$

$$\mathbf{y}_{\text{gen}} = \text{LLM}(\mathbf{x}_{\text{sys\_gen\_i}}, \mathcal{G}^{\text{skg}}), \tag{14}$$

where $\mathbf{y}_{\text{gen}}$ is the generated textual output, with input parameters consistent with those used in predictive tasks. In this context, the LLM focuses on producing coherent and contextually accurate content based on both text and graph inputs.

2.4.2 Graph-Instruction Alignment

To teach our agent in comprehending graph-structured data, we implement graph-instruction alignment in the initial fine-tuning stage. Inspired by the work of (Tang et al., 2024b), we utilize the efficient, effective, and easily scalable task of graph-instruction matching as our alignment target. Specifically, we present a set of graph token-instruction pairs:

$$\mathcal{D}^g = [(\mathbf{e}_0, s_0), (\mathbf{e}_1, s_1), ...], \tag{15}$$

$$\mathcal{D}^c = [(\mathbf{c}_0, \mathbf{c}_{s_0}), (\mathbf{c}_1, \mathbf{c}_{s_1}), ...] \tag{16}$$

where $(\mathbf{e}_i, s_i)$ denotes the i-th graph token with meta type s_i, and $(\mathbf{c}_i, \mathbf{c}_{s_i})$ denotes the text description of the i-th graph token and its meta type, correspondingly. We devise two general tasks to achieve fine-grained and comprehensive alignment between the graph tokens and the textual instructions:

- **Intra-type alignment**. This alignment task aims to strengthen the capability of LLMs to interpret graph embedding tokens of certain meta-types through promoting their alignment with the relevant texts. This is conducted by training LLMs to output correct sequence of the texts given a sequence of graph tokens. Specifically, we construct a dataset $\mathcal{D}^{\text{intra}}$ with each entry consists of two sequences of graph tokens and texts, separately: $d_i^{\text{intra}} = ([(\mathbf{e}_j, s_i), ...], [(\mathbf{c}_k, \mathbf{c}_{s_i}), ...])$. Then, we train the alignment with a next-token-prediction Cross-Entropy objective as follows:

$$\text{argmin}_{\Theta}\text{CE\_Loss}(d_i^{\text{intra}}[0] | \text{LLM}(d_i^{\text{intra}}[1])), \tag{17}$$

  where Θ denotes the learnable parameters of the large language model $\text{LLM}(\cdot)$. And indices $[0]$ and $[1]$ indicate the text sequence and the graph token sequence, respectively.
- **Inter-type alignment**. As introducing multiple meta-types in the alignment task can further empower the LLM's comprehension of complex heterogeneous relations, we devise another alignment training objective using inter-type graph tokens. Technically, the dataset $\mathcal{D}^{\text{inter}}$ is constructed by sampling entries that consist graph tokens of different meta-types in the first sequence:

$$\begin{aligned} d_i^{\text{inter}} = ([(\mathbf{e}_m, s_m), (\mathbf{e}_n, s_n), \ldots], \\ [(\mathbf{c}_n, \mathbf{c}_{s_n}), (\mathbf{c}_q, \mathbf{c}_{s_q}), \ldots]) \end{aligned} \tag{18}$$

  Then, the LLM is trained to predict the text sequence and the meta-type sequence of the provided graph tokens:

$$\text{argmin}_{\Theta}\text{CE\_Loss}(d_i^{\text{inter}}[0] | d_i^{\text{inter}}[1])). \tag{19}$$

2.4.3 Agent Task Finetuning

To enhance GraphAgent's performance on different agent tasks, we propose to finetune the action agent with diverse graph-language instructions covering different agent tasks. Recall that with the task planning agent we have the user requested task $\mathbf{t} \in \mathcal{T}$ from the query prompt. For each $\mathbf{t}$ in the instruction dataset, we pair it with a special systematic prompt to distinguish between various tasks during training. The systematic prompt contains brief description of the task being handled. Formally, the agent task finetuning dataset is constructed as:

$$\begin{aligned} \mathcal{D}^{\text{multi}} = \Big\{ (\{(\mathbf{x}_{\text{pred}}, \mathbf{x}_{\text{reasoning}}) \mid \mathbf{x}_{\text{gen}}\}, \\ \{\mathcal{G}^{\text{exp}} \mid \mathcal{G}^{\text{skg}}\}, \mathbf{t}_i, \mathbf{a}_i) \Big\}. \end{aligned} \tag{20}$$

For each instruction-output pair, the graph provided can be an explicit graph, an automatically discovered SKG, or both. For predictive tasks, the output includes a prediction and reasoning, while for generative tasks, the output is the objective.

Further, to facilitate a smooth learning curve for multi-tasking the graph language model, we take inspiration from curriculum learning techniques (Xu et al., 2020; Bengio et al., 2009) and sort our training tasks into different difficulty levels. We start training with easier tasks to build the model's foundational graph-language understanding. As training progresses, we gradually introduce more complex tasks to refine the model's capabilities. The details are demonstrated in Table 6.

3 Evaluation

We evaluate the effectiveness of our GraphAgent framework through a focused evaluation addressing key Research Questions (**RQs**):

- **RQ1**: How well does GraphAgent capture graph relational information and textual semantic interdependencies for graph-related predictive tasks?
- **RQ2**: How effectively does GraphAgent perform predictive tasks by leveraging implicit textual semantic interdependencies?
- **RQ3**: How does GraphAgent perform in graph-enhanced text generation with implicit dependency understanding compared to SOTA LLMs?
- **RQ4**: How do GraphAgent's key components impact its performance in ablation studies?

3.1 Experimental Settings

3.1.1 Implementation Details

In GraphAgent, task planning and graph generation agents use `GPT3.5-Turbo` with few-shot prompts to enhance query handling and semantic knowledge graph (SKG) discovery. `PyG` converts structural data into graph objects for grounding. Text-attributed graph embeddings are created with Sentence-BERT (`all-mpnet-base-v2`). The graph action agent employs Llama3-8b (Llama Team, 2024). A learnable linear layer connects textual and graph representations (Liu et al., 2024; Tang et al., 2024a). A pre-trained heterogeneous graph model (Tang et al., 2024b) encodes text-graph node pairs, projected through the adaptation layer and processed with the LLM for integrated reasoning.

3.1.2 Datasets

A summary of the diverse datasets used is provided in Table 1. We place detailed discussion on the datasets in Appendix A.1.

3.1.3 Baseline Methods

We compare GraphAgent against a diverse set of baseline models across graph-related predictive tasks and text generation, including homogeneous GNNs, heterogeneous graph models, graph LLMs, and state-of-the-art LLMs with retrieval-augmented generation (RAG) systems.

- **Graph-Related Predictive Tasks**. We evaluate: i) **Homogeneous GNNs**, including SAGE (Hamilton et al., 2017) and GAT (Velickovic et al., 2018); ii) **Heterogeneous Graph Models**, such as HAN (Wang et al., 2019b), HGT (Hu et al., 2020), and HetGNN (Zhang et al., 2019); and iii) **Graph LLMs**, using HiGPT (Tang et al., 2024b), a state-of-the-art model for complex heterogeneous graphs.
- **Graph-Enhanced Text Generation**. We compare with: i) **Open-Source LLMs**, including Llama 3 series (Llama Team, 2024), Mistral NeMo[1], and Qwen2-72b (Yang et al., 2024); ii) **Closed-Source LLMs**, such as Deepseek-Chat-V2, GPT4o-mini, and Gemini-1.5-Flash via their APIs; iii) **LLM-empowered RAG Systems**, specifically GraphRAG[2], which enhances LLMs with graph-based RAG.

3.2 Evaluation Protocols

We implement comprehensive and consistent training strategies across all models. We apply full fine-tuning for our model and all baseline models requiring supervised fine-tuning. For model selection, we utilize validation sets with early-stopping for predictive tasks, while monitoring training loss decreasing rate for alignment training and generative tasks. To ensure fair comparison, we maintain consistent feature encoder (all-mpnet-base-v2) across all models including GNNs and Graph LLMs. We use identical prompt templates across all LLM-based models, with GraphLLMs receiving additional graph tokens for embedding injection and basic meta type descriptions (detailed in Table 8). The iterative steps are set to 2 for discovering two-hop knowledge graphs per query prompt.

For evaluation, we adopt different metrics based on task types. In graph-related predictive tasks with

---

[1] https://mistral.ai/news/mistral-nemo/

[2] `https://github.com/microsoft/graphrag`

Table 2: Zero-shot learning performance evaluation: We assess our model's transfer capabilities by training on IMDB dataset with few-shot learning, then evaluating node classification performance on ACM dataset under zero-shot conditions, utilizing both graph structural and textual information.

Metric	Trained on	SAGE	GAT	HAN	HGT	HetGNN	HiGPT	**GraphAgent**	*Imprv.*
Micro-F1	IMDB-1	32.93±4.18	35.67±0.53	34.07±1.11	32.40±0.14	37.43±4.34	45.40±0.89	**51.21±1.32**	*12.8%*
(%)	IMDB-40	31.73±0.05	23.93±1.44	26.97±1.94	35.60±0.99	31.80±0.16	50.50±0.77	**74.98±1.24**	*48.5%*
Macro-F1	IMDB-1	26.47±2.69	29.08±1.31	22.50±4.16	16.31±0.05	31.39±4.68	41.77±1.24	**46.82±1.43**	*12.1%*
(%)	IMDB-40	31.17±0.17	21.41±0.71	23.13±1.32	27.49±1.22	31.44±0.17	45.85±0.89	**74.98±1.12**	*63.5%*
AUC	IMDB-1	49.34±2.47	52.48±0.38	51.28±0.86	50.00±0.00	53.18±2.95	59.69±0.82	**64.10±1.25**	*7.4%*
(%)	IMDB-40	48.67±0.13	43.20±1.08	45.45±1.46	51.48±0.43	48.72±0.06	63.60±0.51	**80.90±1.01**	*27.2%*

ground truth, we use Micro-F1 (Mi-F1), Macro-F1 (Ma-F1), and AUC metrics. For graph-enhanced generative tasks that are open-ended, we primarily rely on the PPL score using state-of-the-art models (Llama3-70b, Qwen2-72b) to measure fluency, rather than reference-based similarity metrics which can be misleading due to their limitations in text generation evaluation. Additionally, we incorporate the LLM-as-judge approach for better approximation of human judgment. This comprehensive evaluation framework ensures robust and meaningful comparison across different model architectures while addressing the limitations of conventional evaluation metrics for generative tasks.

3.3 Graph Prediction Task with Explicit and Implicit Graph Contexts (RQ1)

We evaluate GraphAgent's performance on node classification with explicit graph structures, enhancing methods by integrating semantic knowledge graphs from node text, using both semantic KG and explicit connections as dual graph token inputs. Following prior work (Tang et al., 2024a,b; Chen et al., 2024a), we adopt a zero-shot evaluation framework for real-world applicability. Models are trained on IMDB under few-shot settings (1 and 40 shots) and tested on 1,000 unseen ACM dataset nodes. We apply Chain-of-Thought (Wei et al., 2022) for inference in GraphAgent and other LLM-enhanced methods.

Table 2 shows GraphAgent outperforms the state-of-the-art graph language model, HiGPT, by over 28% across metrics. This improvement results from integrating a graph generation agent, an automated task planning agent, and dual fine-tuning (graph-text alignment and agent task fine-tuning). These components enable GraphAgent to construct rich semantic knowledge graphs, capture interdependencies, and understand complex relationships in structured and unstructured graph contexts, leading to superior downstream task performance.

3.4 Graph Prediction with Implicit Semantic Interdependencies (RQ2)

We assess GraphAgent's performance on predictive tasks requiring complex semantic interdependency understanding, comparing against state-of-the-art LLMs. GraphAgent constructs semantic knowledge graphs (SKGs) using a dual-agent system for task planning and graph generation, with SKG nodes serving as semantic anchors for enriched input representation. Evaluations on Arxiv-Papers and ICLR-Peer Reviews datasets (Table 4) test GraphAgent across task-specific (GraphAgent-Task Expert), comprehensive (GraphAgent-General), and zero-shot (GraphAgent-Zero-Shot) settings. Unlike GNNs and GraphLLMs needing explicit graphs, GraphAgent competes with fine-tuned and GraphRAG-augmented LLMs:

- **High Performance with Smaller Size.** With 8B parameters, GraphAgent outperforms larger LLMs like Llama3-70b and Qwen2-72b by 31.9% across metrics on both datasets. By capturing interdependencies via SKGs and maintaining contextual awareness, GraphAgent integrates local and global patterns, enabling robust reasoning for complex tasks.
- **Strong Generalization in Multi-task and Zero-shot Settings.** GraphAgent-General surpasses task-specific models on Arxiv-Papers, leveraging SKGs for enhanced text-graph reasoning, with competitive results on ICLR-Peer Reviews. In zero-shot settings, GraphAgent matches Deepseek-Chat-V2 and Gemini-1.5-Flash, showcasing robust generalization through SKG integration and specialized tuning.
- **Advantages over SFT and RAG.** GraphAgent outperforms vanilla SFT LLMs and GraphRAG by leveraging SKG integration for better knowledge utilization. By using graph embedding tokens for efficient, consolidated representation, it reduces token overhead and mitigates hallucination, ensuring reliable performance.

3.5 Graph-Enhanced Text Generation (RQ3)

We evaluate GraphAgent's graph-enhanced text generation using perplexity (PPL) and LLM-based assessment. Results are shown in Table 3, Figure 4, and zero-shot GovReport data in Table 5.

Table 3: Performances on ACL-EMNLP related works content generation. Light grey denotes that the score is computed with the same-family model.

Method	Model Size	PPL-Llama3-70b		PPL-Qwen2-72b	
		Mean	Max	Mean	Max
Open-sourced LLMs					
Llama3-8b	8B	7.016	13.061	7.491	12.787
Mistral-Nemo	12B	7.367	15.967	6.872	12.065
Llama3-70b	70B	6.168	14.436	5.877	12.897
Qwen2-72b	72B	6.043	11.675	5.325	11.302
API-based Commercial LLMs					
Deepseek-Chat-V2	236B→21B	5.632	13.483	5.144	10.337
GPT4o-mini	-	7.277	15.480	6.818	13.267
Gemini-1.5-Flash	-	5.188	10.399	5.377	10.779
Finetuned LLMs					
Llama3-8b Finetuned	8B	7.682	19.452	7.629	18.757
GraphRAG Implementations					
Llama3-8b + GraphRAG	8B	7.098	18.092	6.539	14.722
Llama3-70b + GraphRAG	70B	6.590	14.827	6.135	14.163
GraphAgent-Task Expert	8B	3.805	10.316	4.069	11.685
GraphAgent-General	8B	**3.618***	**8.000***	**3.867***	**8.775***

Table 4: Performance comparison with state-of-the-art LLMs on complex graph prediction tasks involving implicit semantic relationships. Results marked with * indicate statistical significance ($p<0.01$) compared to the second-best performer.

Method	Model Size	Arxiv-Papers			ICLR-Peer Reviews		
		Mi-F1	Ma-F1	AUC	Mi-F1	Ma-F1	AUC
Open-sourced LLMs							
Llama3-8b	8B	0.514	0.289	0.527	0.402	0.394	0.502
Mistral-Nemo	12B	0.510	0.292	0.615	0.272	0.246	0.380
Llama3-70b	70B	0.630	0.330	0.635	0.434	0.421	0.551
Qwen2-72b	72B	0.632	0.472	0.700	0.344	0.277	0.509
API-based Commercial LLMs							
Deepseek-Chat-V2	236B→21B	0.746	0.580	0.757	0.362	0.312	0.516
GPT4o-mini	-	0.592	0.343	0.634	**0.692***	0.592	0.591
Gemini-1.5-Flash	-	0.748	0.504	0.714	0.684	0.487	0.533
Finetuned LLMs							
Llama3-8b Finetuned	8B	0.794	0.593	0.736	0.620	0.554	0.553
GraphRAG Implementations							
Llama3-8b + GraphRAG	8B	0.516	0.288	0.601	0.430	0.427	0.517
Llama3-70b + GraphRAG	70B	0.603	0.324	0.623	0.308	0.296	0.401
GraphAgent-Task Expert	8B	0.820	0.620	0.768	0.686	**0.620***	**0.615***
GraphAgent-General	8B	**0.840***	**0.621***	**0.769***	0.667	0.604	0.607
GraphAgent-Zero-Shot	8B	0.739	0.512	0.701	0.538	0.531	0.563

- **Improved Generation via Lower Perplexity**. Table 3 shows GraphAgent's lower PPL scores compared to baselines, validated by Llama3-70b and Qwen2-72b, with enhanced fluency and clarity. SFT and GraphRAG variants underperform, failing to capture complex reasoning. Our semantic knowledge graphs improve GraphAgent's reasoning and comprehension.

- **Enhanced Quality via LLM-based Evaluation.** Using LLM-as-judge (Zheng et al., 2024), which correlates strongly with human judgment compared to BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004), we evaluated GraphAgent with GPT-4 (prompts in Table 13) against Llama3-8b, its fine-tuned version, Mistral Nemo, and Llama3-70b. On 200 test samples (Figure 4), GraphAgent achieves 114% quality improvement over Llama3-8b and 45% over Llama3-70b, outperforming same-sized models in 67% of cases and leading open-source models in 58%, despite 8B parameters and low input overhead.

- **Cross-domain Document Summarization.** GraphAgent excels in GovReport summarization (Table 9 in Appendix), producing well-organized summaries (green) without task-specific optimization. Table 5 shows GraphAgent achieves lower PPL scores than Llama3-8b and Mistral-Nemo, matching the fluency of leading LLMs in zero-shot tasks. Our approach, leveraging semantic knowledge graphs and multi-task graph-based training, ensures robust generalization.

3.6 Ablation Study

To evaluate each component in GraphAgent, we conducted an ablation study with the following variants: • **(-) SKG** removes the graph generation agent and excludes semantic knowledge graph tokens from LLM input. • **(-) Alignment** omits the graph-instruction alignment tuning described in Section 2.4.2. • **(-) Cur. Strategy** eliminates the curriculum learning strategy for agent task training (Section 2.4.3), instead training all tasks simultaneously across all epochs. Figure 3 presents the comparative results between GraphAgent and its variants on both predictive and generative tasks. Our analysis reveals two key findings:

- For predictive tasks, semantic knowledge graphs generated by the graph generation agent show the strongest impact, as their supplementary information substantially enhances model performance. In contrast, for generative tasks, the alignment stage proves crucial for maintaining high performance, likely because these tasks demand sophisticated reasoning capabilities, making alignment essential for learning deeper graph-instruction understanding.

- The curriculum training strategy shows consistent improvements across both task types. By enabling gradual progression from simpler predictive tasks to more complex generative ones, this approach allows the model to effectively assimilate knowledge from various graph-instruction pairs, resulting in stronger performance.

4 Related Work

Graph Representation Learning models complex relationships using graph embedding techniques (Chen et al., 2020; Wu et al., 2020). Graph

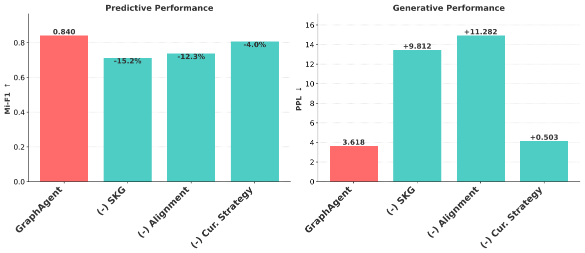

Figure 3: Ablation study comparing GraphAgent with its variants on both graph-related prediction and graph-enhanced text generation tasks.

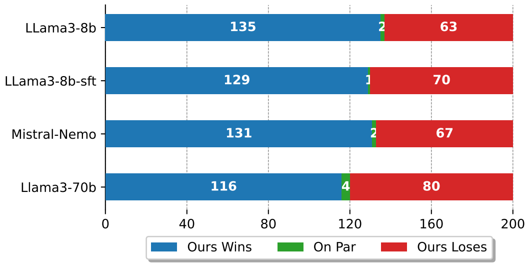

Figure 4: Comparative evaluation results: GPT-4o as judge assessing our proposed GraphAgent framework against state-of-the-art open-source LLMs.

Neural Networks (GNNs) capture node dependencies via message-passing (Dwivedi et al., 2023; Huang et al., 2024a). Architectures like Graph Convolutional Networks (GCNs) (Kipf and Welling, 2017; Jin et al., 2020; Wu et al., 2024) aggregate neighbors through localized convolutions, while Graph Attention Networks (GAT) (Veličković et al., 2018; Zhang et al., 2022; Hao et al., 2023) use attention to weigh neighbor importance. In our GraphAgent, GNNs serve as graph tokenizers for LLM integration.

Graph Language Models. Recent works enhance graph model generalization by combining Large Language Models (LLMs) with GNNs (Tang et al., 2024b). GraphGPT (Tang et al., 2024a) integrates graph encoders with LLMs via an alignment projector. LLaGA (Chen et al., 2024b) reorganizes nodes into structure-aware sequences for LLMs. ZeroG (Li et al., 2024) enables zero-shot transfer learning for cross-dataset generalization. Unlike these models, which focus on explicit graph topology, our framework addresses both explicit and implicit graph-enhanced semantic dependencies, supporting diverse predictive and generative tasks.

LLM-empowered Agents improve user interaction by integrating diverse data with intuitive communication (Shinn et al., 2023; Xie et al., 2023). Language-based assistants enhance reasoning and decision-making (Yao et al., 2023; Jimenez et al., 2024), while vision-based assistants provide contextual insights from visual data (Koh et al., 2024; Hong et al., 2024). Embodied agents leverage LLMs for navigation and interaction in robotics and smart systems (Brehmer et al., 2024; Huang et al., 2024b). However, few agents handle relational data alongside textual information. Our framework bridges this gap, integrating for both predictive and generative tasks.

5 Conclusion

We present GraphAgent, a multi-agent framework that integrates graph-based reasoning with advanced language modeling to address complex scenarios involving relational and textual data. GraphAgent's automated pipeline, comprising a graph generator agent for semantic interdependencies, a task planning agent for query interpretation, and a task execution agent, enhances large language models' adaptability across diverse datasets. It excels in graph prediction and open-ended text generation tasks. Future work will extend GraphAgent to incorporate visual information from multi-modal data, improving its ability to handle relational, textual, and visual content.

6 Limitations

While GraphAgent demonstrates strong performance in integrating structured and unstructured data for predictive and generative tasks, it has some limitations. First, the framework relies on specific LLMs (*e.g.*, LLaMA-3-8B, GPT3.5-Turbo) for its agents, which may limit adaptability to other models without further fine-tuning. Second, the current implementation focuses on text and graph data, lacking integration of visual information, which we plan to address in future work.

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML*, pages 41–48.

Johann Brehmer, Joey Bose, Pim De Haan, and Taco S Cohen. 2024. Edgi: Equivariant diffusion for planning with embodied agents. *NeurIPS*, 36.

Bo Chen, Jing Zhang, Fanjin Zhang, Tianyi Han, Yuqing Cheng, Xiaoyan Li, Yuxiao Dong, and Jie Tang. 2023. Web-scale academic name disambiguation: the whoiswho benchmark, leaderboard, and toolkit. In *KDD*, pages 3817–3828.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and deep graph convolutional networks. In *ICML*, pages 1725–1735. PMLR.

Runjin Chen, Tong Zhao, Ajay Jaiswal, Neil Shah, and Zhangyang Wang. 2024a. Llaga: Large language and graph assistant. *ICML*.

Runjin Chen, Tong Zhao, AJAY KUMAR JAISWAL, Neil Shah, and Zhangyang Wang. 2024b. Llaga: Large language and graph assistant. In *ICML*.

Enyan Dai, Wei Jin, Hui Liu, and Suhang Wang. 2022. Towards robust graph neural networks for noisy graphs with sparse labels. In *WSDM*, pages 181–191.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2023. Benchmarking graph neural networks. *JMLR*, 24(43):1–48.

Matthias Fey, Weihua Hu, Kexin Huang, Jan Eric Lenssen, Rishabh Ranjan, Joshua Robinson, Rex Ying, Jiaxuan You, and Jure Leskovec. 2023. Relational deep learning: Graph representation learning on relational databases. *arXiv preprint arXiv:2312.04615*.

Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*.

Xinyu Fu, Jiani Zhang, Ziqiao Meng, and Irwin King. 2020. Magnn: Metapath aggregated graph neural network for heterogeneous graph embedding. In *Proceedings of the web conference 2020*, pages 2331–2341.

William L Hamilton. 2020. *Graph representation learning*. Morgan & Claypool Publishers.

William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*, pages 1024–1034.

Qianyue Hao, Wenzhen Huang, Tao Feng, Jian Yuan, and Yong Li. 2023. Gat-mf: Graph attention mean field for very large scale multi-agent reinforcement learning. In *KDD*, pages 685–697.

Xiaoxin He, Xavier Bresson, Thomas Laurent, Adam Perold, Yann LeCun, and Bryan Hooi. 2023. Harnessing explanations: Llm-to-lm interpreter for enhanced text-attributed graph representation learning. *arXiv preprint arXiv:2305.19523*.

Wenyi Hong and 1 others. 2024. Cogagent: A visual language model for gui agents. In *CVPR*, pages 14281–14290.

Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *WWW*, pages 2704–2710. ACM / IW3C2.

Kexin Huang, Ying Jin, Emmanuel Candes, and Jure Leskovec. 2024a. Uncertainty quantification over graph with conformalized graph neural networks. *NeurIPS*, 36.

Wenlong Huang and 1 others. 2024b. Grounded decoding: Guiding text generation with grounded models for embodied agents. *NeurIPS*, 36.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. 2024. Swe-bench: Can language models resolve real-world github issues? In *ICLR*.

Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph structure learning for robust graph neural networks. In *KDD*, pages 66–74.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. 2024. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. *ACL*.

Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian McAuley. 2023. Text is all you need: Learning language representations for sequential recommendation. In *KDD*, pages 1258–1267.

Yuhan Li, Peisong Wang, Zhixun Li, Jeffrey Xu Yu, and Jia Li. 2024. Zerog: Investigating cross-dataset zero-shot transferability in graphs. In *KDD*, pages 1725–1735.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024. Visual instruction tuning. *NeurIPS*, 36.

Nian Liu, Xiao Wang, Deyu Bo, Chuan Shi, and Jian Pei. 2022. Revisiting graph contrastive learning from the perspective of graph spectrum. *NeurIPS*, 35:2972–2983.

AI @ Meta Llama Team. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Yujie Lu, Xianjun Yang, Xiujun Li, Xin Eric Wang, and William Yang Wang. 2024. Llmscore: Unveiling the power of large language models in text-to-image synthesis evaluation. *NeurIPS*, 36.

Haitao Mao, Zhikai Chen, Wenzhuo Tang, Jianan Zhao, Yao Ma, Tong Zhao, Neil Shah, Michael Galkin, and Jiliang Tang. 2024. Graph foundation models. *ICML*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *NeurIPS*, 36.

Jie Shuai, Kun Zhang, Le Wu, Peijie Sun, Richang Hong, Meng Wang, and Yong Li. 2022. A review-aware graph contrastive learning framework for recommendation. In *SIGIR*, pages 1283–1293.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. 2024a. Graphgpt: Graph instruction tuning for large language models. In *SIGIR*, pages 491–500.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Long Xia, Dawei Yin, and Chao Huang. 2024b. Higpt: Heterogeneous graph language model. In *KDD*.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, and 1 others. 2018. Graph attention networks. In *ICLR (Poster)*. OpenReview.net.

Xiao Wang, Deyu Bo, Chuan Shi, Shaohua Fan, Yanfang Ye, and S Yu Philip. 2022. A survey on heterogeneous graph embedding: methods, techniques, applications and sources. *TBD*, 9(2):415–436.

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. 2019a. Heterogeneous graph attention network. In *WWW*, pages 2022–2032.

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, and 1 others. 2019b. Heterogeneous graph attention network. In *WWW*, pages 2022–2032. ACM.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS*, 35:24824–24837.

Zhihao Wu, Zhao Zhang, and Jicong Fan. 2024. Graph convolutional kernel machine versus graph convolutional networks. *NeurIPS*, 36.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *TPAMI*, 32(1):4–24.

Lianghao Xia and Chao Huang. 2024. Anygraph: Graph foundation model in the wild. *arXiv preprint arXiv:2408.10700*.

Tianbao Xie, Fan Zhou, Zhoujun Cheng, Peng Shi, Luoxuan Weng, Yitao Liu, Toh Jing Hua, Junning Zhao, Qian Liu, Che Liu, and 1 others. 2023. Openagents: An open platform for language agents in the wild. In *COLM*.

Benfeng Xu, Licheng Zhang, Zhendong Mao, Quan Wang, Hongtao Xie, and Yongdong Zhang. 2020. Curriculum learning for natural language understanding. In *ACL*, pages 6095–6104.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, and 1 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *ICLR*.

Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and Nitesh V. Chawla. 2019. Heterogeneous graph neural network. In *KDD*, pages 793–803. ACM.

Wentao Zhang, Ziqi Yin, Zeang Sheng, Yang Li, Wen Ouyang, Xiaosen Li, Yangyu Tao, Zhi Yang, and Bin Cui. 2022. Graph attention multi-layer perceptron. In *KDD*, pages 4560–4570.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *NeurIPS*, 36.

Zhiqiang Zhong and Davide Mottin. 2023. Knowledge-augmented graph machine learning for drug discovery: From precision to interpretability. In *KDD*, pages 5841–5842.

A Appendix

Table 5: GovReport summarization performance. Evaluation scores are presented with same-family model comparisons highlighted in light grey.

Method	Model Size	PPL-Llama3-70b		PPL-Qwen2-72b	
		Mean	Max	Mean	Max
Llama3-8b	8B	9.476	25.355	7.564	17.443
Mistral-Nemo	12B	9.333	28.537	7.194	19.347
Llama3-70b	70B	6.473	14.724	5.629	11.813
Qwen2-72b	72B	7.134	16.075	5.494	11.294
Deepseek-Chat-V2	236B→21B	8.246	21.176	7.311	18.092
GPT4o-mini	-	10.332	23.300	6.576	10.213
Gemini-1.5-Flash	-	7.374	18.408	6.133	9.237
GraphAgent-General	8B	6.736	20.362	5.936	27.196

A.1 Dataset Details.

- **Graph-Related Predictive Tasks**. For tasks that involve explicit graph relational information, we utilize two benchmark datasets: IMDB (Fu et al., 2020) and ACM (Wang et al., 2019a). In contrast, for predictive tasks that do not depend on explicit graph structures, we have curated two additional datasets: Arxiv-Papers (He et al., 2023) and ICLR-Peer Reviews[3]. The Arxiv-Papers dataset comprises published papers from Arxiv in 2023, from which we randomly sampled a subset. This dataset is created by analyzing the titles and abstracts of these papers to classify whether they are likely to be accepted. The ICLR-Peer Reviews dataset features pairs of papers and their corresponding reviews from ICLR 2024, specifically focusing on borderline cases that pose challenges in determining acceptance. This dataset is used for both training and testing purposes.

- **Graph-Enhanced Text Generation**. To demonstrate the text generation capabilities of model, we evaluate its performance in generating related work for research papers and summarizing lengthy documents using graph-enhanced semantic dependencies. **First**, we collected datasets from the ACL and EMNLP conferences, covering the years 2020 to 2023, including both the "main" and "findings" tracks. We extracted the related work sections from these papers and organized them into approximately 5,000 topic-content pairs. For generating related work, GraphAgent takes a list of paper titles and their corresponding abstracts-input that can be provided by users. Using this information, scaffold knowledge graphs are created and subsequently processed by the Graph Action Agent, which comprehends the data to produce comprehensive related work for the specified papers. **Second**, we utilize the GovReport dataset [4] to evaluate GraphAgent as a language assistant for document summarization. This dataset comprises detailed reports from government research agencies, including the Congressional Research Service and the U.S. Government Accountability Office. It necessitates the summarization of longer documents, maintaining richer context and semantic interdependencies, unlike other summarization datasets.

A.2 Detailed Implementation of GraphAgent

To ensure reproducibility of our experimental results, we provide comprehensive implementation details and technical specifications of our GraphAgent framework in this section.

A.2.1 System prompts of GraphAgent

Tables 8 and 13 present the comprehensive system prompts used in our framework. Specifically, Table 8 details the system prompts for the three core components: the *Task Planning Agent*, the *Graph Generation Agent*, and the task-specific prompt builders for the *Multi-Task Agent*. Additionally, Table 13 outlines the system prompts employed in our LLM-as-judge evaluation protocol.

Table 6: Curriculum training strategy in terms of epochs and different ratios of data mixing.

	Alignment Data	Predictive Data	Generative Data
Epoch 1	10%	70%	20%
Epoch 2	5%	60%	35%
Epoch 3	0%	50%	50%
Afterwards	0%	40%	60%

A.2.2 Curriculum training strategy

We employ a curriculum learning strategy to effectively train our graph language model for multi-task scenarios. As shown in Table 6, the training process begins with fundamental tasks to establish basic graph-language understanding, then progressively introduces more challenging components - advancing from predictive tasks to generative tasks. This graduated approach ensures robust model development and optimal performance across diverse task requirements.

---

[3] `https://github.com/ranpox/iclr2024-openreview-submissions`

[4] `https://huggingface.co/datasets/ccdv/govreport-summarization`

Table 7: Examples for the system prompts and system prompt builder we used in GraphAgent

System prompt for intent and task parsing [$\mathbf{x}_{\text{system\_tp}}$] You are very powerful assistant for graph-related tasks for diverse user inputs. You can do great in parsing the following important properties from the user input: 1. "graph source". This is either the uploaded file paths if the user uploads pre-defined graph for the task, or the user input contents as texts or documents that contain knowledge. 2. "graph task". the graph task type to handle, must be one of "predictive_predefined", "predictive_wild", "open_generation". You should infer the graph task to handle from the user input. 3. "user annotations". Any additional information the user provided in the query prompt. Could be task description, label candidates or specific requirements. You are provided with two realistic examples to help you excel in the task: <few shot examples>.
System prompt for scaffold knowledge node extraction at the 0-th step of the Graph Generation Agent [$\mathbf{x}_{\text{system\_sk\_0}}$] You are very powerful assistant for graph-related tasks for diverse user inputs. You can do great in detecting and extracting the important scaffold nodes from the user input. A list of scaffold nodes reflect the top-level concepts or entities in the content, that are useful to form a knowledge graph for the content. You should carefully examine the input content to decide your extraction strategy. 1. For a general long document of a certain scenario, consider several most high-level aspects that are useful to grasp the key concepts in the document. Do not propose too specific concepts as scaffold nodes. It is very vital to be general and be abstract in your proposed scaffold nodes. 2. For inputs that are more formatted and contain specific entities, relationships, or concepts, you can directly adopt the key entities or concepts listed in the input as scaffold nodes. In this case, it is essential to accurately concentrate on the high-level formatted concepts or entities. For your output, use auto-increment ids to number the scaffold nodes, and infer the general type for each. You are provided with several examples to help you excel in the task: <few shot examples>.
System prompt for scaffold knowledge node extraction after the 0-th step of the Graph Generation Agent [$\mathbf{x}_{\text{system\_sk\_1}}$] You are very powerful assistant for graph-related tasks for diverse user inputs. You can do great in detecting and extracting the important scaffold nodes from the user input. A list of scaffold nodes should be informative and representative of the key points in the text, that are useful to form a knowledge graph for the content. You should carefully examine the input content to decide your extraction strategy. You also need to provide a description of the extracted keywords for each scaffold node. The description should be detailed and informative, and can contain two parts: 1) a brief description of the keywords based on the contexts in the text, and 2) a detailed description of the keywords based on your own knowledge. You are provided with several examples to help you excel in the task: <few shot examples>.
System prompt for knowledge description augmentation of the Graph Generation Agent [$\mathbf{x}_{\text{system\_ka}}$] You are a powerful assistant in generating information textual descriptions for a list of scaffold nodes. Each scaffold node represents a high-level key point or topic in the text, and your goal is to provide comprehensive and detailed texts related to each scaffold node. The texts can be from your own knowledge base with references to the original input content. Texts should be detailed and you should never miss any important information. You can never miss any node in the input. You should parse corresponding texts for each scaffold node in the input. You should always return the same number of scaffold nodes as the input. You are provided with several examples to help you excel in the task: <few shot examples>.
System prompt builder template for graph multi task agent [$\mathbf{x}_{\text{system\_ka}}$] You are a powerful assistant in accomplishing diverse user required tasks with the help of structured knowledge as graphs. The current user requested task is of type: <$\mathbf{t}_{\text{user}}$>. The detailed request or provided information is: <$\mathbf{x}_{\text{user\_ann}}, \mathbf{g}_s$>. [**If predictive in the wild or open generation:**] For the required task, a heterogeneous knowledge graph is built to assist you as useful and informative knowledge references. There are <num. of meta types> types of nodes and edges in the graph, separately: <meta types>. The graph tokens for each type are: [<meta type>: <graph>]. [**If predictive with pre-defined graphs:**] For the required task, a pre-defined heterogeneous graph is provided as information reference. There are <num. of meta types> types of nodes and edges in the graph, separately: <meta types>. The graph tokens for each type are: [<meta type>: <graph>]. Additionally, a heterogeneous knowledge graph is also constructed to augment your knowledge for the task. There are <num. of meta types> types of nodes and edges in the graph, separately: <meta types>. The graph tokens for each type are: [<meta type>: <graph>]. Please generate response that satisfies the user's request.<$\mathbf{x}_{\text{user\_ann}}$>. Provide concise reasoning if the task involves certain prediction.

Table 8: Examples for the system prompts and system prompt builder we used in GraphAgent

System prompt for intent and task parsing [$\mathbf{x}_{\text{system\_tp}}$] You are very powerful assistant for graph-related tasks for diverse user inputs. You can do great in parsing the following important properties from the user input: 1. "graph source". This is either the uploaded file paths if the user uploads pre-defined graph for the task, or the user input contents as texts or documents that contain knowledge. 2. "graph task". the graph task type to handle, must be one of "predictive_predefined", "predictive_wild", "open_generation". You should infer the graph task to handle from the user input. 3. "user annotations". Any additional information the user provided in the query prompt. Could be task description, label candidates or specific requirements. You are provided with two realistic examples to help you excel in the task: <few shot examples>.
System prompt for scaffold knowledge node extraction at the 0-th step of the Graph Generation Agent [$\mathbf{x}_{\text{system\_sk\_0}}$] You are very powerful assistant for graph-related tasks for diverse user inputs. You can do great in detecting and extracting the important scaffold nodes from the user input. A list of scaffold nodes reflect the top-level concepts or entities in the content, that are useful to form a knowledge graph for the content. You should carefully examine the input content to decide your extraction strategy. 1. For a general long document of a certain scenario, consider several most high-level aspects that are useful to grasp the key concepts in the document. Do not propose too specific concepts as scaffold nodes. It is very vital to be general and be abstract in your proposed scaffold nodes. 2. For inputs that are more formatted and contain specific entities, relationships, or concepts, you can directly adopt the key entities or concepts listed in the input as scaffold nodes. In this case, it is essential to accurately concentrate on the high-level formatted concepts or entities. For your output, use auto-increment ids to number the scaffold nodes, and infer the general type for each. You are provided with several examples to help you excel in the task: <few shot examples>.
System prompt for scaffold knowledge node extraction after the 0-th step of the Graph Generation Agent [$\mathbf{x}_{\text{system\_sk\_1}}$] You are very powerful assistant for graph-related tasks for diverse user inputs. You can do great in detecting and extracting the important scaffold nodes from the user input. A list of scaffold nodes should be informative and representative of the key points in the text, that are useful to form a knowledge graph for the content. You should carefully examine the input content to decide your extraction strategy. You also need to provide a description of the extracted keywords for each scaffold node. The description should be detailed and informative, and can contain two parts: 1) a brief description of the keywords based on the contexts in the text, and 2) a detailed description of the keywords based on your own knowledge. You are provided with several examples to help you excel in the task: <few shot examples>.
System prompt for knowledge description augmentation of the Graph Generation Agent [$\mathbf{x}_{\text{system\_ka}}$] You are a powerful assistant in generating information textual descriptions for a list of scaffold nodes. Each scaffold node represents a high-level key point or topic in the text, and your goal is to provide comprehensive and detailed texts related to each scaffold node. The texts can be from your own knowledge base with references to the original input content. Texts should be detailed and you should never miss any important information. You can never miss any node in the input. You should parse corresponding texts for each scaffold node in the input. You should always return the same number of scaffold nodes as the input. You are provided with several examples to help you excel in the task: <few shot examples>.
System prompt builder template for graph multi task agent [$\mathbf{x}_{\text{system\_ka}}$] You are a powerful assistant in accomplishing diverse user required tasks with the help of structured knowledge as graphs. The current user requested task is of type: <$\mathbf{t}_{\text{user}}$>. The detailed request or provided information is: <$\mathbf{x}_{\text{user\_ann}}, \mathbf{g}_s$>. [**If predictive in the wild or open generation:**] For the required task, a heterogeneous knowledge graph is built to assist you as useful and informative knowledge references. There are <num. of meta types> types of nodes and edges in the graph, separately: <meta types>. The graph tokens for each type are: [<meta type>: <graph>]. [**If predictive with pre-defined graphs:**] For the required task, a pre-defined heterogeneous graph is provided as information reference. There are <num. of meta types> types of nodes and edges in the graph, separately: <meta types>. The graph tokens for each type are: [<meta type>: <graph>]. Additionally, a heterogeneous knowledge graph is also constructed to augment your knowledge for the task. There are <num. of meta types> types of nodes and edges in the graph, separately: <meta types>. The graph tokens for each type are: [<meta type>: <graph>]. Please generate response that satisfies the user's request.<$\mathbf{x}_{\text{user\_ann}}$>. Provide concise reasoning if the task involves certain prediction.

Table 9: Qualitative comparison for a GovReport task between GraphAgent, llama3-8b and Llama3-70b.

[User]
Here is a government report concerning specific topics. Your task is to write a paragraph that highly condenses the content without losing any important information.
Consistent with the discretion afforded by the APA, Regulations.gov and agency-specific comment websites use required and optional fields on comment forms to collect some identity information from commenters. In addition to the text of the comment, agencies may choose to collect identity information by requiring commenters to fill in other fields, such as name, address, and email address before they are able to submit a comment. Regardless of the fields required by the comment form, the selected agencies all accept anonymous comments in practice. Specifically, in the comment forms on Regulations.gov and agency- specific comment websites, a commenter can submit under a fictitious name, such as Änonymous Anonymous,ënter a single letter in each required field, or provide a fabricated address. In each of these scenarios, as long as a character or characters are entered into the required fields, the comment will be accepted. Further, because the APA does not require agencies to authenticate submitted identity information, neither Regulations.gov nor the agency-specific comment websites contain mechanisms to check the validity of identity information that commenters submit through comment forms. Regulations.gov and agency-specific comment websites also collect some information about public users' interaction with their websites through application event logs and proxy server logs, though the APA does not require agencies to collect or verify it as part of the rulemaking process. This information, which can include a public user's Internet Protocol (IP) address, browser type and operating system, and the time and date of webpage visits, is collected separately from the comment submission process as part of routine information technology management for system security and performance, and cannot be reliably connected to specific comments. Seven of the 10 selected agencies have documented some internal guidance associated with the identity of commenters during the three phases of the public comment process: intake, analysis, and response to comments. However, the focus and substance of this guidance varies by agency and phase of the comment process. As shown in Table 1, for selected agencies that have guidance associated with the identity of commenters, it most frequently relates to the comment intake or response to comment phases of the public comment process. The guidance for these phases addresses activities such as managing duplicate comments (those with identical or near-identical comment text but varied identity information) or referring to commenters in a final rule...

[GraphAgent]
The report examines how government agencies manage and disclose identity information associated with public comments on proposed regulations. It finds that while agencies are not required to collect or verify identity information, they do collect some information from commenters and accept anonymous comments. However, the report notes inconsistencies in how agencies post identity information, particularly with duplicate comments. It also highlights a lack of transparency in disclosing these practices to the public, potentially hindering meaningful participation in the rulemaking process. The report recommends that agencies establish clear policies for posting comments and communicate these practices to the public more effectively. Since issuing the report, some agencies have begun implementing these recommendations, but others have not yet done so.

[Llama3-8b]
The Government Accountability Office (GAO) report examines the treatment of identity information by 10 federal agencies during the public comment process. While agencies are not required to collect or verify identity information, they may choose to do so through comment forms. However, the report finds that agencies vary in their treatment of identity information, with some accepting anonymous comments and others requiring commenters to provide name, address, and email address. The report also notes that agencies do not consistently post duplicate comments, which can lead to inconsistent presentation of identity information to the public. Additionally, agencies do not provide clear information about their posting practices, which can limit public users' ability to explore and use the data. The GAO recommends that agencies establish policies for posting comments and provide clear communication to the public about their posting practices, particularly with regard to identity information and duplicate comments. Since the report's issuance, some agencies have taken steps to address these recommendations.

[Llama3-70b]
The government's public comment process, facilitated by Regulations.gov and agency-specific websites, collects identity information from commenters, but allows anonymous comments and does not verify submitted information. While agencies may require certain fields to be filled, commenters can submit fictitious information or minimal data, and comments will still be accepted. The treatment of identity information varies across agencies, with some posting all comments and others posting only a sample of duplicate comments, which can lead to inconsistent presentation of identity information to the public. Furthermore, agencies do not effectively communicate their posting practices and limitations to the public, potentially leading to inaccurate conclusions about the comments submitted. To address this, recommendations have been made to establish policies for posting comments and clearly communicate them to the public, with some agencies having taken action to implement these changes.

Table 10: Qualitative comparison for a NLP-Related Works task between GraphAgent, fine-tuned llama3-8b and Llama3-70b.

[User]
I am writing a "related works" section for my research paper in the NLP domain. The topic to cover is "Dense Passage Retrieval". I would like to discuss the following papers in the section:
Title: Representation Degeneration Problem in Training Natural Language Generation Models. Abstract: ...
Title: Deeper Text Understanding for IR with Contextual Neural Language Modeling. Abstract: ...
Title: Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. Abstract: ...
Title: Unsupervised Corpus Aware Language Model Pre-training for Dense Passage Retrieval. Abstract: ...
Title: PAIR: Leveraging Passage-Centric Similarity Relation for Improving Dense Passage Retrieval. Abstract:...
Further, Use @CITE[id]@ to cite a paper in the paragraph. For example, @CITE[1]@ will be replaced by the citation of the paper with id 1. At the end of the generated content, list all the references in the format of ID: Title.

[GraphAgent]
Dense passage retrieval (DPR) has recently attracted a lot of attention in the IR community, which aims to retrieve relevant passages from a large corpus with the help of a query. DPR is based on the neural network, which can learn query and passage representations from the corpus and query separately. The retrieval model is usually trained by contrastive loss, which is to pull the query representation close to the passage representation when they are relevant, and push them apart when they are irrelevant. In the early stage , DPR is mainly based on the BERT model @CITE[4]@, which is pre-trained by masked language modeling. However, BERT is prone to degenerate into a linear mapping to the word embeddings @CITE[5]@, which leads to a performance drop. To alleviate this problem , several methods are proposed to train DPR model. For example , @CITE[3]@ propose the use of the negative sampling strategy, which is to sample a negative passage from the corpus to calculate the contrastive loss. @CITE[1]@ propose the use of the nearest neighbor negative sampling strategy, which is to sample a negative passage from the corpus with the nearest distance to the query. @CITE[2]@ propose the use of the contrastive loss with the corpus-level information, which is to pull the passage representation close to the corpus-level representation when they are relevant.
References: ID: 1. Title: Representation Degeneration Problem in Training Natural Language Generation Models
ID: 2. Title: Deeper Text Understanding for IR with Contextual Neural Language Modeling
ID: 3. Title: Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval
ID: 4. Title: Unsupervised Corpus Aware Language Model Pre-training for Dense Passage Retrieval
ID: 5. Title: PAIR: Leveraging Passage-Centric Similarity Relation for Improving Dense Passage Retrieval

[Llama3-70b]
Dense passage retrieval has gained significant attention in recent years, leveraging the power of neural networks to learn complex language patterns and query-document relations @CITE[2]@. This approach has many intriguing advantages over sparse retrieval, such as better capturing of language structures and improved performance on queries written in natural languages @CITE[2]@. However, the effectiveness of dense retrieval often requires combination with sparse retrieval, and the training mechanisms used in dense retrieval can be a bottleneck @CITE[3]@. To address this, various techniques have been proposed, including Approximate Nearest Neighbor Negative Contrastive Learning @CITE[3]@, unsupervised corpus-aware language model pre-training @CITE[4]@, and leveraging passage-centric similarity relations @CITE[5]@. These advancements have significantly improved the performance of dense passage retrieval models, making them a promising approach for various natural language processing tasks.

[Llama3-8b Fine-tuned]
Recent advancements in PLMs such as BERT (De- vlin et al., 2018), Roberta (@CITE[1]@, and GPT (@CITE[3]@ have enabled numerous unsupervised techniques to derive dense represen- tations of queries and passages for retrieval. These approaches primarily use a Bi-Encoder structure to embed text in a low-dimensional space and learn similarity relations via contrastive learning, con- trasting traditional sparse retrieval methods like BM25 or DeepCT (@CITE[2]@;. DPR (pi- oneered an unsupervised dense passage retrieval framework, affirming the feasibility of using dense representations for retrieval independently. This efficient and operational approach was further refined by subsequent studies (@CITE[4]@; ; ; @CITE[5]@ that focused on high-quality negative sample min- ing, additional passage relation analysis, and extra training. The essence of these methods is to repre- sent texts in a target space where queries are closer to relevant and distant from irrelevant passages.
References: ID: 1. Title: Representation Degeneration Problem in Training Natural Language Generation Models ID: 2. Title: Deeper Text Understanding for IR with Contextual Neural Language Modeling ID: 3. Title: Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval ID: 4. Title: Unsupervised Corpus Aware Language Model Pre-training for Dense Passage Retrieval ID: 5. Title: PAIR: Leveraging Passage-Centric Similarity Relation for Improving Dense Passage Retrieval

Table 11: Visualization examples for automatically discovered semantic knowledge graphs for a GovReport sample.

[Original Text with Knowledge]

In our June 2019 report, we found that, while abuse deficiencies cited in nursing homes were relatively rare from 2013 through 2017, they became more frequent during that time, with the largest increase in severe cases. Specifically, abuse deficiencies comprised less than 1 percent of the total deficiencies in each of the years we examined, which is likely conservative. Abuse in nursing homes is often underreported by residents, family, staff, and the state survey agency, according to CMS officials and stakeholders we interviewed. However, abuse deficiencies more than doubled–from 430 in 2013 to 875 in 2017–over the 5-year period. (See appendix II.) In addition, abuse deficiencies cited in 2017 were more likely to be categorized at the highest levels of severity– deficiencies causing actual harm to residents or putting residents in immediate jeopardy–than they were in 2013. In light of the increased number and severity of abuse deficiencies, it is imperative that CMS have strong nursing home oversight in place to protect residents from abuse; however, we found oversight gaps that may limit the agency's ability to do so. Specifically, we found that CMS: (1) cannot readily access data on the type of abuse or type of perpetrator, (2) has not provided guidance on what information nursing homes should include in facility-reported incidents, and (3) has numerous gaps in its referral process that can result in delayed and missed referrals to law enforcement. In our June 2019 report, we found that CMS's data do not allow for the type of abuse or perpetrator to be readily identified by the agency. Specifically, CMS does not require the state survey agencies to record abuse and perpetrator type and, when this information is recorded, it cannot be easily analyzed by CMS. Therefore, we reviewed a representative sample of 400 CMS narrative descriptions–written by state surveyors–associated with abuse deficiencies cited in 2016 and 2017 to identify the most common types of abuse and perpetrators. From this review, we found that physical abuse (46 percent) and mental/verbal abuse (44 percent) occurred most often in nursing homes, followed by sexual abuse (18 percent). Furthermore, staff, which includes those working in any part of the nursing home, were more often the perpetrators (58 percent) of abuse in deficiency narratives, followed by resident perpetrators (30 percent) and other types of perpetrators (2 percent). (See appendix III for examples from our abuse deficiency narrative review.)...

[Scaffold Node $k = 0$] (top-level aspects or concepts)

Node meta type: Policy Objectives And Goals

Text attribute: The policy objectives and goals of this report are to ensure that CMS has strong nursing home oversight in place to protect residents from abuse. The report aims to identify and address the gaps in CMS's ability to monitor and respond to abuse deficiencies effectively.

Node meta type: Stakeholder Impact and Implications

Text attribute: Stakeholders impacted by this report include nursing home residents, their families, staff, state survey agencies, and CMS officials. The implications are that without proper oversight and reporting mechanisms, abuse in nursing homes may continue to be underreported and inadequately addressed. Node meta type: Methodology and Evidence

Text attribute: The methodology involved reviewing a representative sample of 400 CMS narrative descriptions associated with abuse deficiencies cited in 2016 and 2017. This review aimed to identify the most common types of abuse and perpetrators. Additionally, interviews with CMS officials and stakeholders provided insights into the underreporting and handling of abuse incidents.

Node meta type: Findings and Recommendations

Text attribute: Key findings include the underreporting of abuse in nursing homes, the doubling of abuse deficiencies from 2013 to 2017, and the increased severity of these deficiencies. Recommendations include requiring state survey agencies to report abuse and perpetrator type, providing guidance on facility-reported incidents, and improving the referral process to law enforcement.

Node meta type: Implementation and Evaluation

Text attribute: As of November 2019, the Department of Health and Human Services (HHS) had not implemented the recommendations made in the report. The implementation and evaluation of these recommendations are crucial to improving CMS's oversight and protecting nursing home residents from abuse.

[Scaffold Node $k = 1$] (Fine-grained concepts as keywords. Text attributes are omitted.)

abuse deficiencies in nursing homes ; Centers for Medicare & Medicaid Services (CMS) ; oversight gaps ; recommendations for improvement ; nursing home oversight ; resident protection ; abuse deficiencies ; CMS's ability to monitor ; nursing home residents ; abuse in nursing homes ; state survey agencies ; CMS officials ; CMS narrative descriptions ; abuse deficiencies ; underreporting of abuse incidents ; CMS officials and stakeholders ; underreporting of abuse ; abuse deficiencies ; state survey agencies ; referral process to law enforcement ; Department of Health and Human Services ; Centers for Medicare & Medicaid Services ; nursing home residents ; abuse ;

Table 12: Visualization examples for automatically discovered semantic knowledge graphs for an Arxiv sample, similarly for NLP-Related Works data.

[Original Text with Knowledge]

Title: A Simple Zero-shot Prompt Weighting Technique to Improve Prompt Ensembling in Text-Image Models. Abstract: Contrastively trained text-image models have the remarkable ability to perform zero-shot classification, that is, classifying previously unseen images into categories that the model has never been explicitly trained to identify. However, these zero-shot classifiers need prompt engineering to achieve high accuracy. Prompt engineering typically requires hand-crafting a set of prompts for individual downstream tasks. In this work, we aim to automate this prompt engineering and improve zero-shot accuracy through prompt ensembling. In particular, we ask "Given a large pool of prompts, can we automatically score the prompts and ensemble those that are most suitable for a particular downstream dataset, without needing access to labeled validation data?". We demonstrate that this is possible. In doing so, we identify several pathologies in a naive prompt scoring method where the score can be easily overconfident due to biases in pre-training and test data, and we propose a novel prompt scoring method that corrects for the biases. Using our proposed scoring method to create a weighted average prompt ensemble, our method outperforms equal average ensemble, as well as hand-crafted prompts, on ImageNet, 4 of its variants, and 11 fine-grained classification benchmarks, all while being fully automatic, optimization-free, and not requiring access to labeled validation data.

[Scaffold Node $k = 0$] (top-level aspects or concepts)

Node meta type: Research Background

Text attribute: Contrastively trained text-image models possess the ability to perform zero-shot classification, which involves categorizing unseen images into untrained categories. However, achieving high accuracy in zero-shot classification often requires meticulous prompt engineering, typically involving hand-crafted prompts tailored for specific downstream tasks.

Node meta type: Research Question

Text attribute: The research question addressed in this work is whether it is possible to automatically score and ensemble the most suitable prompts from a large pool for a particular downstream dataset, without relying on labeled validation data. This question stems from the need to automate and improve the accuracy of zero-shot classification through better prompt engineering.

Node meta type: Methodology

Text attribute: The methodology involves identifying and addressing pathologies in a naive prompt scoring method, which can be overly confident due to biases in pre-training and test data. The authors propose a novel prompt scoring method that corrects for these biases, enabling the creation of a weighted average prompt ensemble that is fully automatic and optimization-free.

Node meta type: Key Results

Text attribute: The key results demonstrate that the proposed prompt weighting technique outperforms both equal average ensemble and hand-crafted prompts on ImageNet, four of its variants, and 11 fine-grained classification benchmarks. The method achieves this while remaining fully automatic, not requiring optimization, and without access to labeled validation data.

[Scaffold Node $k = 1$] (Fine-grained concepts as keywords. Text attributes are omitted.)

zero-shot prompt weighting ; automating prompt engineering ; zero-shot classification accuracy ; zero-shot classification ; meticulous prompt engineering ; automatic scoring ; ensemble prompts ; zero-shot classification ; prompt engineering ; naive prompt scoring method ; novel prompt scoring method ; weighted average prompt ensemble ; prompt weighting technique ; fully automatic ;

A.3 Qualitative Analysis of Graph-enhanced Text Generation Tasks

We evaluated GraphAgent against Llama3-8b and Llama3-70b on two distinct graph-enhanced text generation tasks, with results presented in Tables 10 and 9 (Appendix). The experiments demonstrate GraphAgent's significant performance advantages over Llama3-8b while achieving comparable results to the much larger Llama3-70b. Notably, in academic writing tasks (Table 10), GraphAgent effectively leverages knowledge graphs to capture citation relationships and research development paths, producing well-organized summaries (highlighted in green). In contrast, Llama3-8b exhibits notable limitations in both instruction following and citation formatting accuracy (highlighted in red).

This section presents our automatically generated semantic knowledge graphs (SKGs) through two visualized examples in Tables 11 and 12 from GovReport and Arxiv datasets. We visualize each SKG at two levels: $k = 0$ hop showing high-level aspect nodes (highlighted in green) and $k = 1$ hop displaying keyword nodes (highlighted in blue), along with augmented textual attributes for $k = 0$ scaffold nodes. These examples demonstrate our Graph Generation Agent's capability in extracting crucial information, revealing multi-hop relationships, and generating coherent semantic representations. For the related work text generation, while containing paper-based SKGs as multiple sub-graphs, individual papers follow similar SKG patterns to the Arxiv examples.

Table 14 demonstrates the multi-agent collaboration workflow using the IMDB movie classification dataset as an example. The process begins when the *Task Planning Agent* receives a user request, performing request decomposition and task planning, while also processing graph data in scenarios with pre-defined graphs. Next, the *Graph Generation Agent* constructs a task-relevant semantic knowledge graph. Finally, the *Multi-Task Agent* automatically constructs a system prompt based on the task type and executes task inference. This workflow highlights GraphAgent's versatility in automatically adapting to different scenarios, whether handling pre-defined graphs or generating new ones, while maintaining consistent performance across various task types.

Table 13: System Prompts for LLM-based Performance Evaluation.

You are a professional researcher in computer science, AI. You are good at reading and judging written drafts of research papers. Now, your task is to judge between two paragraphs of "related work" section of the same topic. You have to decide which paragraph is written in a better way in terms of the following criteria: 1. It should strictly cover all the references provided. It is not acceptable if only part of the references is cited. 2. It is encouraged that the written content discusses how the related works differ from each other. 3. It is encouraged that detailed technical information is discussed for each reference. 4. It is encouraged that concise, neutral, and objective language is used. Here are the two paragraphs with the same topic <topic>: A:<content a>; B:<content b> Directly give your answers as “‘A is better“‘, “‘B is better“‘ or “‘On par“‘(Use this very sparingly). Then, give a very short reasoning and reflection on why you think so.

Table 14: Illustrative Examples of GraphAgent’s Multi-Agent Pipeline.

[User]
Here I have uploaded a relational graph involving movies, directors and actors. Can you tell me which category does the movie with node <GRAPH_NODE_ID_[7]> belong to? Is it action, comedy or drama?
[Predictive Task with Pre-defined Graph Relations]

[Agent (Task Planning Agent)]
Sure! I will first decompose your request as follows for further actions:
<graph source: [uploaded files]>, <task type: predictive_predefined>, <user annotation>: <"Can you tell me which category does the movie with the target node? Is it action, comedy or drama?">
[Agent (Task Planning Agent)]
Since pre-defined graphs are given, I will try to ground the edge and node list files into readable objects. Then I will encode the graph and index the neighbors for node ID [7] for further actions.
[Function Call: <Graph Grounding> GBW_Tool(·)] → $\mathcal{G}^{\text{exp}}, \mathbf{E}^{\text{gnn}}$
[Agent (Graph Generation Agent)]
Now the graph tokens for the pre-defined graph are ready. I will further try to discover a semantic knowledge graph for the target node and the sampled neighbors.
[Detecting and extracting scaffold nodes for the texts of the nodes...] → $\mathcal{V}^{k=0}_{\text{scaffold}}$
[Augmenting textual descriptions for the scaffold nodes] → $\mathcal{C}^{k=0}_{\text{scaffold}}$
[Iteratively performing until a two-hop knowledge graph is constructed.]
[Function Call: GBW_Tool(·)] → $\mathcal{G}^{\text{skg}}, \mathbf{E}^{\text{gnn\_skg}}$
[Agent (Graph Action Agent)]
Since we already have the pre-defined graph and the built SKG, I will construct an appropriate system prompt according to the user request.
<$\mathbf{x}_{\text{system\_pred}}$>: You are a powerful assistant in accomplishing diverse user required tasks with the help of graphs...
[Agent (Graph Action Agent)]
According to the provided information, the neighbor nodes and the semantic knowledge graph nodes, the most likely category for the movie is ... The reasoning process behind is...