Knowledge Graph Agent

Hristo Efimov
Vela Partners, Internship

March 15, 2024

Abstract

This paper introduces an AI Agent that efficiently extracts insights from unstructured data by building knowledge graphs around a specific person or company entity. Given an initial query, the agent leverages web search capabilities to collect relevant text-based data. It then applies advanced natural language processing (NLP) techniques to extract key relationships from the gathered text. We explore and evaluate various NLP-based data extraction approaches for optimal performance, focusing on accuracy and processing speed as key metrics. Finally, the agent constructs a comprehensive knowledge graph, providing a clear and interconnected representation of the queried entity's context. This approach offers a powerful tool for transforming unstructured data into structured knowledge graphs, revealing valuable insights.

1 Introduction

The abundance of unstructured data presents a valuable resource for businesses, researchers, and individuals. However, deriving meaningful insights from vast repositories of information remains a complex and time-consuming process. Knowledge graphs offer an elegant solution to represent complex data effectively [1]. By depicting information as interconnected entities and relationships, they offer a more intuitive way to navigate the complexities of information and extract insights from it. Despite their potential, creating knowledge graphs from unstructured datasets remains challenging, largely due to the need for sophisticated data retrieval and processing techniques. Traditional approaches frequently involve extensive model fine-tuning and vast training datasets for machine learning models.

Large Language Model (LLM) agents are advanced AI systems based on large language models. They blend natural language understanding and task execution, often augmented with additional tools and systems to enhance their performance.[1] This paper introduces an LLM Agent that leverages the Generative Pre-trained Transformer (GPT) models developed by OpenAI to extract and filter essential data from text, format it, and construct knowledge graphs around the relevant information. It combines web search capabilities with advanced NLP techniques to extract nuanced relationships directly from unstructured text, requiring no upfront fine-tuning of the underlying language model and minimal resource expenditure.

In order to achieve this, the agent utilizes a web search tool and a knowledge graph builder tool. The process begins with an initial query representing a person or company (our input entity). The web search tool leverages existing search APIs, such as Bing, to gather text-based data related to the query. Before analyzing the collected information with an LLM model, it is combined with data about the input entity from the model's base knowledge. After that, we explore various prompt engineering techniques, such as Chain-of-Thought and Retrieval Augmented Generation, to identify and extract key relationships from the text and analyze the performance of different NLP-based data extraction approaches. Finally, the graph builder tool synthesizes the extracted information into a comprehensive knowledge graph around that entity. This structured representation enables users to visualize the entity's context and quickly identify valuable insights.

The paper focuses on extracting insights about people or companies. The analyzed data ranges from personal information and professional affiliations about individuals to product offerings and investments about corporate entities. However, the agent is not constrained to these queries and could be modified or used directly to extract relevant information about other entities.

2 Methodology

This section introduces the methods we apply to engineer prompts and the large language models we use to process data.

Chain-of-Thought (CoT) Prompting

CoT prompting is a technique designed to enhance the reasoning capabilities of LLMs by guiding them through a series of intermediate reasoning steps. This approach provides the LLM with step-by-step reasoning examples to demonstrate how to break down complex problems into smaller, more manageable tasks. It encourages the model to demonstrate its reasoning process, improving accuracy and transparency [2].

[1]More on LLM Agents: `https://www.promptingguide.ai/research/llm-agents`

Zero-shot CoT Prompting

Zero-shot CoT prompting is a variant of CoT where a model generates a response without any additional context or prior examples specific to the task at hand. This technique is particularly useful for our agent's queries where pre-existing examples are scarce. It relies on the model's intrinsic knowledge and ability to understand and follow instructions in the prompt. By appending the instruction "Let's think step by step" to our prompts, we encourage the model to unfold its reasoning in a structured manner[3].[2]

Prompt Chaining

Prompt Chaining is a technique used to handle complex tasks by breaking them down into more manageable subtasks. It involves taking the output of one prompt as the input for the next, creating a sequence of queries that leads to the final result.[3] This method allows us to break down the agent's process into several steps: knowledge generation, entity identification, relationship extraction, and JSON conversion.

GPT models

At the core of our knowledge graph agent lie the GPT models developed by OpenAI. Specifically, we use the advanced `gpt-4-0125-preview` model for complex data extraction and analysis tasks. For more lightweight tasks, such as converting data to JSON objects, we utilize the `gpt-3.5-turbo-0125` model to prioritize speed and efficiency.

3 Data Retrieval

The foundation of our AI agent's functionality lies in its ability to retrieve and process relevant data efficiently. We utilize a combination of external web data and information generated from the LLM's rich pre-existing knowledge base[4]. This approach allows us to create a dynamic and comprehensive understanding of the queried subject.

3.1 Web Data

To gather information from the web, we utilize the Bing Web Search API for general web searches and the Bing Entity Search API for tailored entity data. The Web Search API returns the most relevant search results for the initial entity query. For our purposes, we will use the snippets that Bing provides for each search result, which contain an overview of the content found on the corresponding web page. For well-known entities, e.g., prominent figures and large organizations, the Bing Entity Search API is particularly effective in returning structured summaries (usually key information from Wikipedia).

3.2 Knowledge Generation

Inspired by [4], we also extract relevant information from the LLM's pre-existing knowledge using prompts focused on entity identification and relationship extraction. This process is crucial for filling in the gaps not covered by external data sources. Additionally, we instruct the model to return an empty response if no relevant information is available about an entity. In this case we utilize the `gpt-4-0125-preview` model due to its superior factual knowledge capabilities and more up-to-date training data[5].

[2]Examples on CoT and Zero-shot CoT: `https://www.promptingguide.ai/techniques/cot`

[3]More on Prompt Chaining: `https://www.promptingguide.ai/techniques/prompt_chaining`

[4]Also known as parametric knowledge, this refers to the intrinsic information that the model learned during training

[5]At the time of writing, `gpt-4-0125-preview` is trained on data up to December 2023: `https://platform.openai.com/docs/models/gpt-4-and-gpt-4-turbo`

4 Extracting Entity Relationships through LLM Prompting

In this section, we aim to extract relevant relationships between entities within a given body of text. To achieve this, we compare two different prompt design approaches. The first approach breaks down the extraction task into two separate prompts, whereas the second method combines all the steps into a single prompt. We integrate the information collected in the data retrieval process as context into our prompts. This approach allows the agent to identify relevant entities and relationships without upfront fine-tuning of the underlying language model. We also look into standardizing the model's output by parsing it into a JSON object.

4.1 Method 1

Here, we chain two prompts by splitting the task of extracting relationships from the context into two subtasks. The first step is responsible for identifying entities related to the query in the given context data. We use the following prompt, where `{query}` is the input entity and `{data}` contains all information acquired in the data retrieval process:

Prompt
Identify all unique specific entities related to {query} that are mentioned in the
text below. These entities could include people, companies, locations, universities,
professional affiliations, etc. Only include entities mentioned in the text.

Text:
”””
{data}
”””

The prompt returns a list of relevant entities from the context information. The second prompt then uses the text to extract relevant relationships between the input entity and the entities identified above:

Prompt
Extract the relevant relationships between {query} and the entities you identified above.
The relationships should be in the format ”{query} - relationship - identified entity”.
The relationships should be maximum 2 words and can be a verb or a noun.
The relationships should not be a form of the verb ’be’ or ’have’.
Focus on relationships where the entity is a specific person, company,
location, university, professional affiliation, etc., not an attribute to {query}.

The focus of the prompt is to get balanced information on the person or company in the query. In order to get more relevant data, it introduces a format and rules for the relationships the model should follow. The following is an example of the resulting text-based output containing a list of relationships around the input query `"Yigit Ihlamur"`:

Output
1. Yigit Ihlamur - co-founder - Vela Partners LLC
2. Yigit Ihlamur - worked at - Google's Cloud division
3. Yigit Ihlamur - studied at - University of Oxford
4. Yigit Ihlamur - advisor - Incubate Fund US
5. Yigit Ihlamur - board observer - Cartken
6. Yigit Ihlamur - board observer - DagKnows
7. Yigit Ihlamur - invested in - Meditopia
8. Yigit Ihlamur - resides in - San Francisco Bay Area

4.2 Method 2

This approach combines entity identification and relationship extraction into a single, more concise prompt (`{query}` and `{data}` defined as before):

> ***Prompt***
> Step 1: Identify all unique specific entities related to {query} that are mentioned in the
> text below. These entities could include people, companies, locations, universities,
> professional affiliations, etc. Only include entities mentioned in the text.
>
> Step 2: Extract the relevant relationships between {query} and the entities you have identified.
> The relationships should be in the format "{query} - relationship - identified entity".
> The relationship should be maximum 2 words and can be a verb or a noun.
> The relationship should not be a form of the verb 'be' or 'have'.
> Focus on relationships where the entity is a specific person, company,
> location, university, professional affiliation, etc., not an attribute to {query}.
>
> Step 3: Ensure that all relationships follow the rules:
> 1. The relationships must be in the format "{query} - relationship - identified entity".
> 2. The relationships must not be a form of the verb 'be', e.g., 'is', 'are', 'was', 'were'.
> 3. The relationships must not be a form of the verb 'have', e.g., 'has', 'had'.
> Remove the relationships that don't follow the rules.
>
> Think step-by-step. Only output the list of relationships.
>
> Text:
> """
> {data}
> """

Steps 1 and 2 are identical to the two prompts above. In addition, the prompt includes a revision step (Step 3), which ensures that the extracted relationships are as relevant as possible and follow the specified format. We also use Zero-shot CoT prompting by simply including the instruction "Think step-by-step.". This step-by-step reasoning often leads to improved accuracy and a clearer understanding of how the model arrives at its conclusions [3].[6] In both prompt designs, the context data is delimited by triple quotation marks, which helps separate the instructions from the data in the input.[7] The following is an example of the returned text containing a list of relationships corresponding to the input query `"Yigit Ihlamur"`:

> ***Output***
> 1. Yigit Ihlamur - co-founder - Vela Partners
> 2. Yigit Ihlamur - General Partner - Vela Partners
> 3. Yigit Ihlamur - Advisor - Incubate Fund US
> 4. Yigit Ihlamur - Board Observer - Cartken
> 5. Yigit Ihlamur - Board Observer - DagKnows
> 6. Yigit Ihlamur - attended - University of Oxford
> 7. Yigit Ihlamur - worked at - Google
> 8. Yigit Ihlamur - invested in - Meditopia
> 9. Yigit Ihlamur - located in - San Francisco Bay Area

[6] [5] shows that the APE-engineered prompt "Let's work this out in a step by step way to be sure we have the right answer." might perform better in zero-shot scenarios.

[7] `https://platform.openai.com/docs/guides/prompt-engineering/strategy-write-clear-instructions`

4.3 Comparing the two approaches

Since the first approach utilizes two sequential prompts, generating the final result often takes at least twice as long as the second method. In Table 1, we show the execution times of both methods on the same queries. The execution times of the first method are consistently longer than the times of the second one, generally by a factor of two.

Additionally, a qualitative analysis revealed that Method 2 tends to extract a more focused set of relationships compared to Method 1. For example, given the query `"Yigit Ihlamur,"` depending on the retrieved data, the first approach often includes potentially irrelevant relationships like `"Yigit Ihlamur - has connections on - LinkedIn"` or `"Yigit Ihlamur - owns - www.ihlamur .org"`. This is likely due to the final revision step in the second method. We could introduce an additional revision prompt in the first approach, but it would come at the expense of an even higher execution time.

Overall, the second method consistently identifies and extracts more relevant connections while requiring approximately less than half the time to return a result.

Query	Approach 1 (s)	Approach 2 (s)
Yigit Ihlamur	21.9	5.7
Elon Musk	26.7	8.0
Donald Trump	15.4	9.0
Revolut	17.1	8.7
SpaceX	14.2	6.7
Hugging Face	15.7	8.1
Vela Partners	10.2	6.0
Berbatov	9.8	4.7
Anthropic	19.1	8.0
Sam Altman	17.8	3.9
Average	**16.8**	**6.9**

Table 1: Execution times of the two prompting approaches over 10 different queries, with the average time over all queries in the last row. The times shown are in seconds, rounded to one decimal place. **Note:** the times of individual runs can fluctuate between runs.

4.4 Converting to JSON format

The standardization of output is crucial for downstream applications and integration with other systems. However, the structure and format of the LLM output can often vary, so creating a text-to-JSON parser from scratch could be challenging. Thus, we convert the text-based list of relationships by using another prompt. OpenAI provides a JSON mode for both its `gpt-4-0125-preview` and `gpt-3.5-turbo-0125` models, which constrains the models only to generate strings that parse into valid JSON objects[8]. Utilizing `gpt-3.5-turbo-0125` for this conversion task proves sufficient. The following is an example of the returned JSON object containing a list of relationships corresponding to the input query ”Yigit Ihlamur”:

[8] `https://platform.openai.com/docs/guides/text-generation/json-mode`

JSON output

```
{
    ”relationships”: [
        {”src”: ”Yigit Ihlamur”, ”relationship”: ”co-founder”, ”tgt”: ”Vela Partners”},
        {”src”: ”Yigit Ihlamur”, ”relationship”: ”General Partner”, ”tgt”: ”Vela Partners”},
        {”src”: ”Yigit Ihlamur”, ”relationship”: ”Advisor”, ”tgt”: ”Incubate Fund US”},
        {”src”: ”Yigit Ihlamur”, ”relationship”: ”Board Observer”, ”tgt”: ”Cartken”},
        {”src”: ”Yigit Ihlamur”, ”relationship”: ”Board Observer”, ”tgt”: ”DagKnows”},
        {”src”: ”Yigit Ihlamur”, ”relationship”: ”attended”, ”tgt”: ”University of Oxford”},
        {”src”: ”Yigit Ihlamur”, ”relationship”: ”worked at”, ”tgt”: ”Google”},
        {”src”: ”Yigit Ihlamur”, ”relationship”: ”invested in”, ”tgt”: ”Meditopia”},
        {”src”: ”Yigit Ihlamur”, ”relationship”: ”located in”, ”tgt”: ”San Francisco Bay Area”}
    ]
}
```

5 Building and Visualizing the Graph

We utilize the NetworkX Python library[9] to construct a graph representation of the identified entities and the relationships between them. After creating the graph structure with NetworkX, we plot the knowledge graph with the Matplotlib library. See Figure 1 and Figure 2a.

An important edge case to consider is when two entities have multiple relationships corresponding to multiple edges between two nodes in the graph. Since NetworkX cannot automatically draw multiple edges between nodes, we will aggregate them into a single edge in the visualization. This edge will have a composite label listing all the relationships between the two entities. For example, the relationships `"Yigit Ihlamur - co-founder - Vela Partners"` and `"Yigit Ihlamur - General Partner - Vela Partners"` are represented in Figure 1 with one edge with a combined label `"co-founder, General Partner"`. In addition, to avoid cluttering the graph with too many entities, users can limit the visualization to a subset of the nodes, as demonstrated in Figure 2b.

6 Conclusion

This paper introduces a Knowledge Graph Agent that seamlessly extracts insights from text-based data and constructs knowledge graphs around entities of interest. We have streamlined the process by leveraging state-of-the-art LLMs alongside advanced prompting techniques, such as Prompt Chainging and CoT. This approach demonstrates LLMs' potential to transform unstructured information into actionable knowledge, facilitating subsequent data analysis.

6.1 Limitations and Future Work

While promising, this system has many limitations that provide opportunities for future development.

- **Data Quality:** The current approach relies heavily on Bing's search capabilities and the model's pre-trained knowledge. Diversifying information sources by exploring alternative search APIs or knowledge bases, such as Google Search API and Crunchbase API, could mitigate biases and improve consistency. Furthermore, developing a more comprehensive data retrieval process, incorporating web scraping and LLM embeddings search, would enable the system to handle larger and more diverse datasets.
- **Scalability:** The data retrieval and processing pipeline would benefit from optimizations to enhance its scalability for handling large-scale knowledge graph generation.

[9]`https://networkx.org/documentation/stable/`

- **Knowledge Graph Builder:** The current graph builder does not produce complex and exhaustive graphs. A more comprehensive version could include relationships of entities other than the input query, e.g., by further extracting data about each identified entity in the graph.
- **Visualizations:** Implementing dynamic and interactive knowledge graph visualizations, e.g., with the D3.js library, could significantly enhance the user experience. We could also integrate additional features, such as exploring specific entities and relationships in more detail or expanding the graph based on user queries.

References

[1] C. Peng, F. Xia, M. Naseriparsa, and F. Osborne, "Knowledge graphs: Opportunities and challenges," *Artif. Intell. Rev.*, vol. 56, no. 11, pp. 13 071–13 102, 2023. DOI: `10.1007/S10462-023-10465-9`.

[2] J. Wei *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *CoRR*, vol. abs/2201.11903, 2022. DOI: `10.48550/arXiv.2201.11903`.

[3] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," *CoRR*, vol. abs/2201.11903, 2023. DOI: `10.48550/arXiv.2205.11916`.

[4] J. Liu *et al.*, "Generated knowledge prompting for commonsense reasoning," *CoRR*, vol. abs/2110.08387, 2022. DOI: `10.48550/arXiv.2110.08387`.

[5] Y. Zhou *et al.*, "Large language models are human-level prompt engineers," 2023. DOI: `10.48550/arXiv.2211.01910`.

A Visualizations

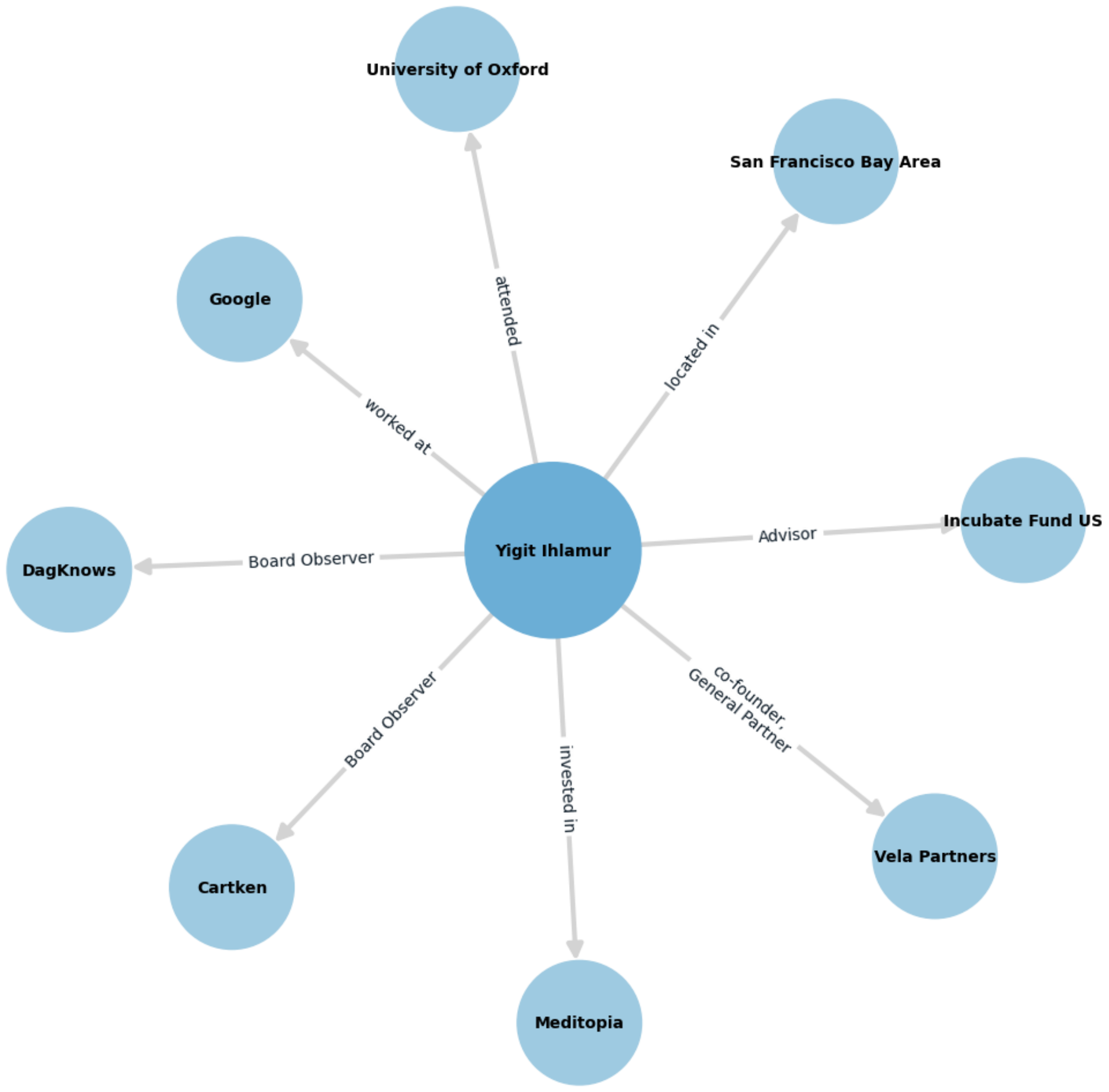
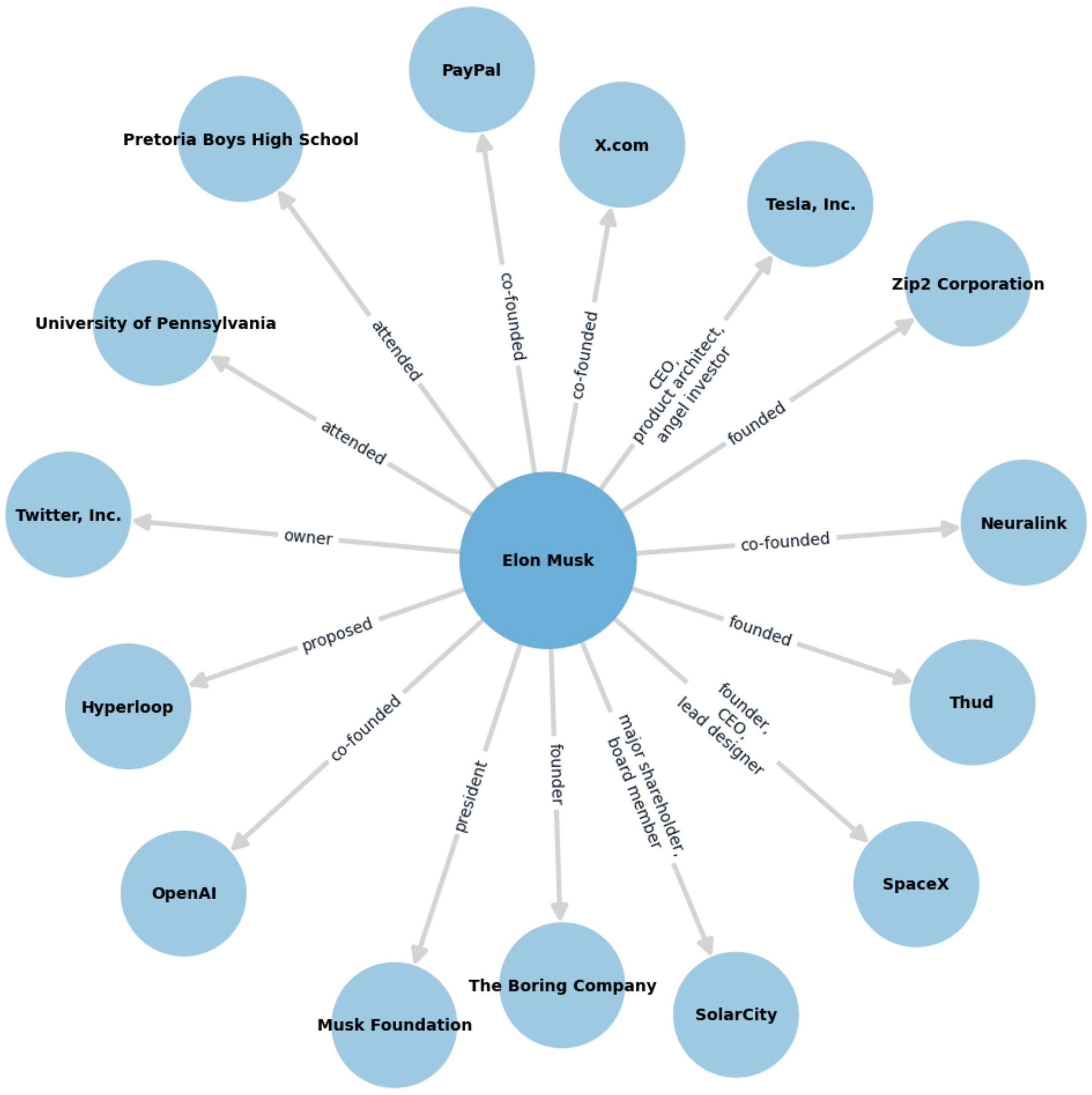
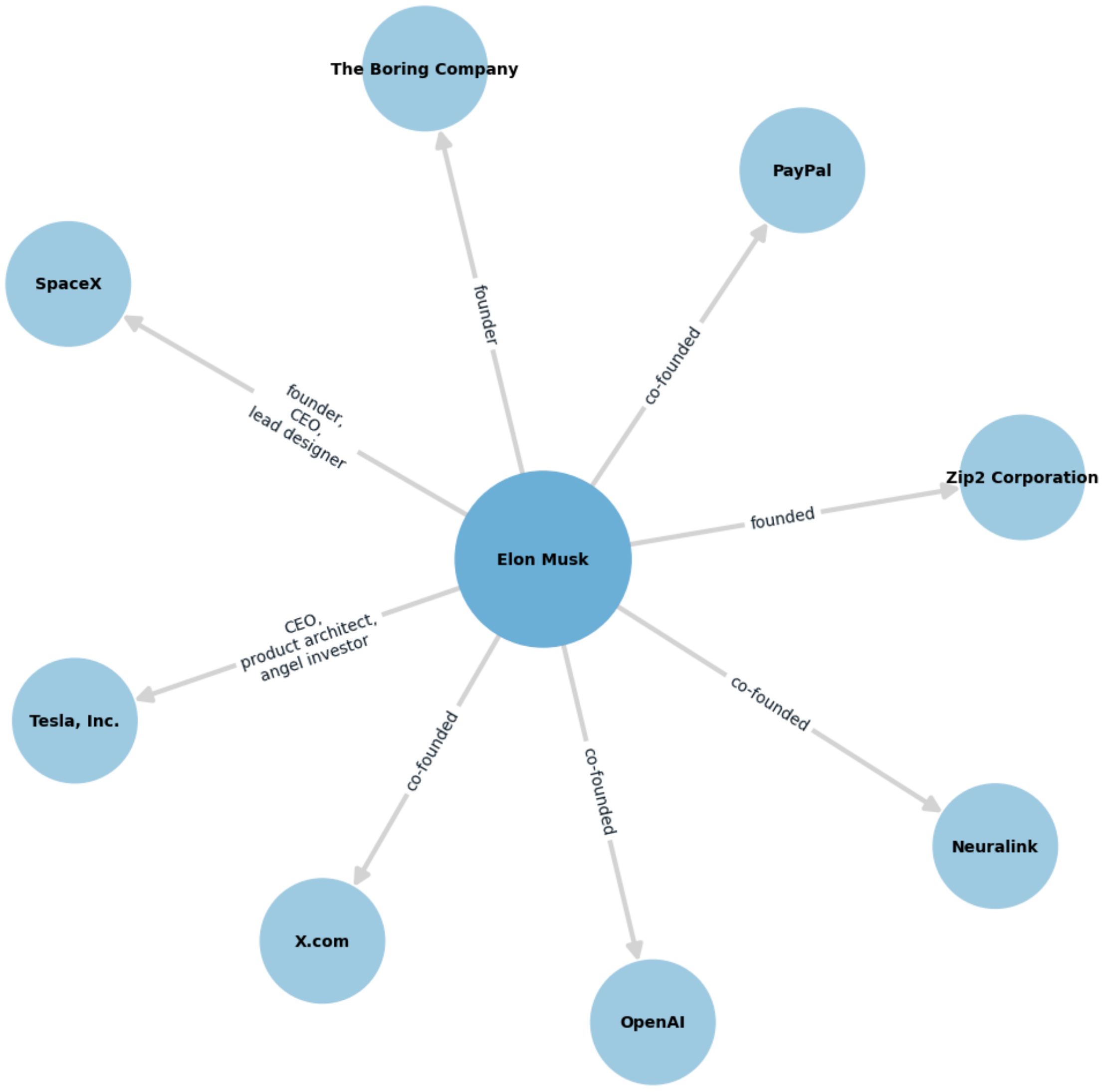


Figure 1: Knowledge Graph around the person `Yigit Ihlamur`



(a)

Figure 2: Knowledge Graph around the person `Elon Musk`



(b) Visualizing a subgraph of 9 nodes.

Figure 2: Knowledge Graph around the person `Elon Musk`