Augmenting Generative AI with Knowledge Graphs

Dennis Gannon, Professor Emeritus Luddy School of Informatics, Computing and Engineering.

Abstract.

Heterogeneous knowledge graphs link node entities such as people, places, artifacts by edges that correspond to semantic connections between the nodes. Together they can capture the total information about a topic of study such as European history or an enterprise like the government of a country. The problem with these graphs is that it can be difficult to extract the knowledge to satisfy a query without specialized skills. A large language model like GPT-4 can take vague natural language question and draw upon its vast exposure to text to formulate rather convincing responses to queries on any topic. Of course, the answer may be a brilliant hallucination. But it is possible to link these two tools together. In this note we illustrate how to tie a simple knowledge graph to a LLM. We use AutoGen, a multiagent toolkit from Microsoft Research for building applications from LLMs and the ACM paper citation data to build a graph query tool based on GPT-4.

Introduction

As an organization or enterprise grows, the knowledge needed to keep it going explodes. The shear complexity of the information sustaining a large operation can become overwhelming. Consider, for example the American Museum of Natural History Who does one contact to gain an understanding of the way the different collections interoperate? Relational databases provide one way to organize information about an organization, but extracting information from an RDBMS can require expertise concerning the database schema and the query languages. Large language models like GPT4 promise to make it easier to solve problems by asking open-ended, natural language questions and having the answers returned in well-organized and thoughtful paragraphs. The challenge in using a LLM lies in training the model to fully understand where fact and fantasy leave off.

Another approach to organizing facts about a topic of study or a complex organization is to build a graph where the nodes are the entities and the edges in the graph are the relationships between them. Next you train or condition a large language model to act as the clever frontend which knows how to navigate the graph to generate accurate answers. This is an obvious idea and others have written about it. Peter Lawrence discusses the relation to query languages like SPAQL and RDF. Venkat Pothamsetty has explored how threat knowledge can be used as the graph. A more academic study from Pan, et.al. entitled ‘Unifying Large Language Models and Knowledge Graphs: A Roadmap’ has an excellent bibliography and covers the subject well.

There is also obvious commercial potential here as well. Neo4J.com, the graph database company, already has a product linking generative AI to their graph system. “Business information tech firm Yext has introduced an upcoming new generative AI chatbot building platform combining large language models from OpenAI and other developers.” See article from voicebot.ai. Cambridge Semantics has integrated the Anzo semantic knowledge graph with generative AI (GPT-4) to build a system called Knowledge Guru that “doesn’t hallucinate”.

Our goal in this post is to provide a simple illustration of how one can augment a generative large language model with a knowledge graph. We will use AutoGen together with GPT4 and a simple knowledge graph to build an application that answers non-trivial English language queries about the graph content. The resulting systems is small enough to run on a laptop.

The Heterogeneous ACM Knowledge Graph

To illustrate how to connect a knowledge graph to the backend of a large language model, we will program Microsoft's AutoGen multiagent system to recognize the nodes and links of a small heterogeneous graph. The language model we will use is OpenAI's GPT4 and the graph is the ACM paper citation graph that was first recreated for a KDD cup 2003 competition for the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Its current form, the graph consists of 17,431 author nodes from 1,804 intuition nodes, 12,499 paper titles and abstracts nodes from 14 conference nodes and 196 conference proceedings covering 73 ACM subject topics. It is a snapshot in time from the part of computer science represented by KDD, SIGMOD, WWW, SIGIR, CIKM, SODA, STOC, SOSP, SPAA, SIGCOMM, MobilCOM, ICML, COLT and VLDB. The edges of the graph represent (node, relationship, node) triples as follows.

- ('paper', 'written-by', 'author')
- ('author', 'writing', 'paper')
- ('paper', 'citing', 'paper')
- ('paper', 'cited-by', 'paper')
- ('paper', 'is-about', 'subject')
- ('subject', 'has', 'paper')
- ('paper', 'venue', 'conference')
- ('paper', 'in', ';proceedings')
- ('proceedings', 'of-conference', 'conference')
- ('author', 'from', 'institution')

Figure 1 illustrates the relations between the classes of nodes. (This diagram is also known as the metagrapah for the heterogeneous graph.) Within each class the induvial nodes are identified by an integer identifier. Each edge can be thought of as a partial function from one class of nodes to another. (It is only a partial function because a paper can have multiple authors and some papers are not cited by any other.)

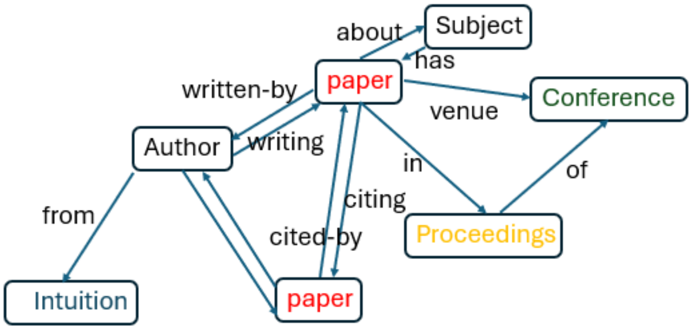


Figure 1. Relations between node classes. We have not represented every possible edge. For example, proceedings are "of" conferences, but many conferences have a proceeding for each year they are held.

Connecting the Graph to GPT4 with AutoGen.

Autogen is a system that we have described in a previous post, so we will not describe it in detail here. However the application here is easy to understand. We will use a system of two agents.

1. A UserProxyAgent called user_proxy that is capable of executing the functions that can interrogate our ACM knowledge graph. (It can also execute Python program, but that feature is not used here.)
2. An AssistantAgent called the graph interrogator. This agent takes the English language search requests from the human user and breaks them down into operations that can be invoked by the user_proxy on the graph. The user_proxy executes the requests and returns the result to the graph interrogator agent who uses that result to formulate the next request. This dialog continues until the question is answered and the graph interrogator returns a summary answer to the user_proxy for display to the human.

The list of graph interrogation functions mirrors the triples that define the edges of the graph. They are:

- find_author_by_name(string)
- find_papers_by_authors (id list)
- find_authors (id list)
- paper_appeared_in (id list)
- find_papers_cited_by (id list)
- find_papers_citing (id list)
- find_papers_by_id (id list)
- find_papers_by_title (string)
- paper_is_about (id list)
- find_papers_with_topic (id list)
- find_proceedings_for_papers (id list)
- find_conference_of_proceedings (id list)
- where_is_author_from (id list)

Except for *find_author_by_name* and *find_papers_by_title* which take strings for input, the others all take graph node id lists. They all return nod id lists or list of (node id, strings) pairs. It is easiest to understand the dialog is to see an example. Consider the query message

> ***Msg = 'Find the authors and their home institutions of the paper "A model for hierarchical memory".'***

We start the dialog by asking the user_proxy to pass this to the graph_interrogator.

user_proxy.initiate_chat(graph_interogator, message=msg)

The graph interrogator agent responds to the user proxy with a suggestion for a function to call.

```
graph_interrogator (to user_proxy):
***** Suggested function Call: find_papers_by_title *****
Arguments: {"papers":"A model for hierarchical memory"}
```

What follows Is the dialog.

```
user_proxy (to graph_interrogator):
***** Response from calling function "find_papers_by_title" *****
[(9665, 'A model for hierarchical memory')]
```

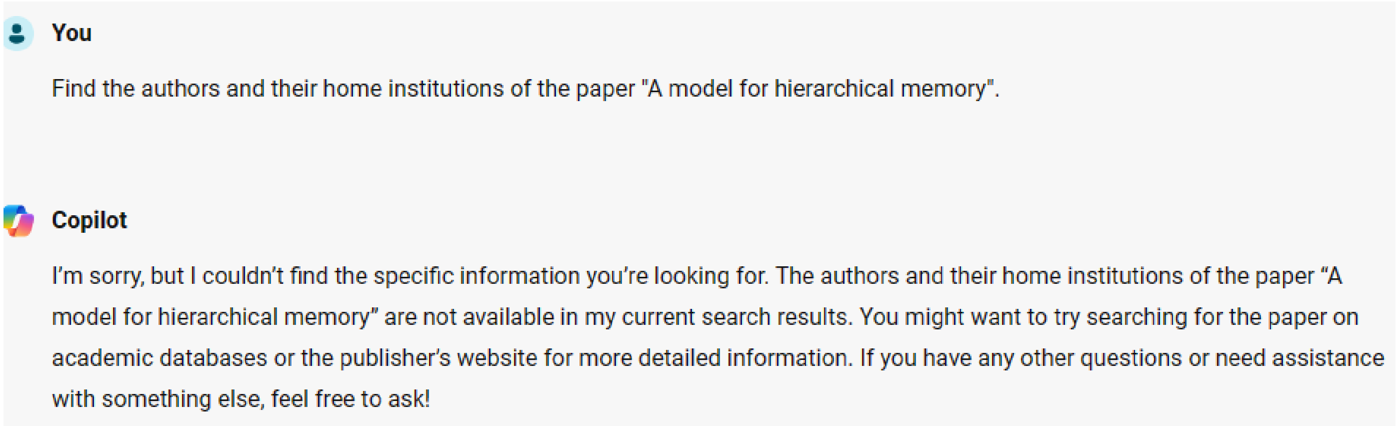
The graph_interrogator now knows the ID of the paper node. It then makes the next suggestion.

```
graph_interrogator (to user_proxy):
***** Suggested function Call: find_authors *****
Arguments: {"papers":"9665"}

user_proxy (to graph_interrogator):
***** Response from calling function "find_authors" *****
{(4115, 'M. Snir'), (5427, 'Ashok K. Chandra'), (3105, 'B. Alpern'), (281,
'A. Aggarwal')}

graph_interrogator (to user_proxy):
***** Suggested function Call: where_is_author_from *****
Arguments: {"authors":"4115, 5427, 3105, 281"}

user_proxy (to graph_interrogator):
***** Response from calling function "where_is_author_from" *****
['University of Illinois at Urbana-Champaign', 'Stanford University', 'The
Hebrew University of Jerusalem', 'Cornell University', 'IBM', 'New York
University', 'University of New Mexico']
```

Finally, the graph_interrogator responds with the summary:

```
graph_interrogator (to user_proxy):
The authors of the paper "A model for hierarchical memory" and their home
institutions are as follows:

- M. Snir is affiliated with the University of Illinois at Urbana-
Champaign.
- Ashok K. Chandra is affiliated with Stanford University.
- B. Alpern's affilations include IBM and The Hebrew University of
Jerusalem.
- A. Aggarwal is affiliated with multiple institutions: Cornell
University, IBM, New York University, and the University of New Mexico.
```

To compare this to GPT-4 based Microsoft Copilot in “precise” answer mode, we get:



You

Find the authors and their home institutions of the paper "A model for hierarchical memory".

Copilot

I'm sorry, but I couldn't find the specific information you're looking for. The authors and their home institutions of the paper "A model for hierarchical memory" are not available in my current search results. You might want to try searching for the paper on academic databases or the publisher's website for more detailed information. If you have any other questions or need assistance with something else, feel free to ask!

Asking the same question in "creative" mode, Copilot lists four papers, one of which is correct and has the authors' affiliation as IBM which was correct at the time of the paper's writing. The other papers are not related.

(Below we look at a few more example queries and the responses. We will skip the dialogs. The best way to see the details is to try this out for yourself. The entire graph can be loaded on a laptop and the AutoGen program runs there as well. You will only need an OpenAI account to run it, but it may be possible to use other LLMs. We have not tried that. The Jupyter notebook with the code and the data are in the GitHub repo.)

Here is another example:

```
msg = '''find the name of authors who have written papers that cite paper "Relational learning via latent social dimensions". list the conferences proceedings where these papers appeared and the year and name of the conference where the citing papers appeared.'''
```

```
user_proxy.initiate_chat(graph_interrogator, message=msg)
```

Skipping the detail of the dialog, the final answer is

```
graph_interrogator (to user_proxy):
The authors who have written papers that cite the paper "Relational
learning via latent social dimensions" are:

- Hanghang Tong
- J. Hopcroft
- Zohreh Nazeri
- Lei Tang
- Piotr Indyk
- Jennifer Neville
- Byron Dom
- Brian Gallagher
- Christos Faloutsos
- Brian Kulis
- Vijay K. Narayanan
- Tina Eliassi-Rad
- Jianping Zhang
- Bart Selman
- Suju Rajan
- Soumen Chakrabarti
- Huan Liu
- David D. Jensen
- Omar Khan

The conference proceedings where these papers appeared are:

- Proceedings of the tenth ACM SIGKDD international conference on
Knowledge discovery and data mining
- Proceedings of the 18th international conference on World wide web
- Proceedings of the ninth ACM SIGKDD international conference on
Knowledge discovery and data mining
- Proceeding of the 14th ACM SIGKDD international conference on Knowledge
discovery and data mining
- Proceedings of the 1998 ACM SIGMOD international conference on
Management of data

The names of the conferences where the citing papers appeared and their
years are as follows:

- SIGMOD - Associated with the Proceedings of the 1998 ACM SIGMOD
international conference on Management of data
- KDD - Associated with multiple proceedings such as the tenth, ninth, and
14th ACM SIGKDD international conference on Knowledge discovery and data
mining
- WWW - Associated with the Proceedings of the 18th international
conference on World wide web
```

The failing here is that the graph does not have the year of the conference.

Here is another example:

```
msg = '''find the topics of papers by Lawrence Snyder and find five other papers on the same topic.  List the titles and proceedings each appeared in. '''

user_proxy.initiate_chat(graph_interogator, message=msg)
```

```
graph_interrogator (to user_proxy):

The topics of papers by Lawrence Snyder are related to the topic with the
ID 'D.4'. Here are five other papers on the same topic along with their
titles and the proceedings they appeared in:

1. "Experience with processes and monitors in Mesa (Summary)" - Appeared
in the proceedings of the seventh ACM symposium on Operating systems
principles (SOSP).
2. "Evaluating synchronization mechanisms" - Appeared in the same
proceedings of SOSP.
3. "WFS a simple shared file system for a distributed environment" - Also
appeared in the proceedings of SOSP.
4. "A mechanism for information control in parallel systems" - Again,
appeared in the proceedings of SOSP.
5. "Systems aspects of The Cambridge Ring" - Appeared in the proceedings
of SOSP.
```

Note: The acm topic for Snyder’s paper is “Operating Systems” and that is ACM topic D.4.

Final Thoughts

This demo is, of course, very limited. Our graph is very small. It only covers a small fraction of ACM’s topics and scope. One must then ask how well this scale to a very large KG. In this example we only have a dozen edge types. And for each edge type we needed a function that the AI can invoke. These edges correspond to the verbs in the language of the graph and a graph big enough to describe a complex organization or a field of study may require many more. Consider for example a large natural history museum. The nodes of the graph may be objects in the collection and the categorical groups in which they are organized, their location in the museum, the historical provenance of the pieces, the scientific importance of the piece and many more. The edge “verbs” could be extremely large and reflect the way these nodes relate to each other. The American Natural History Museum in New York has many on-line databases that describe its collections. One could build the KG by starting with these databases and knitting them together. This raises an interesting question. Can an AI solution create a KG from the databases alone? In principle, it is possible to extract the data from the databases and construct a text corpus that could be used to (re)train a BERT or GPT like transformer network. Alternatively, one could use a named entity recognition pipeline and relation extraction techniques to build the KG. One must then connect the language model query front end. There are probably already start-ups working on automating this process.