Retrieval-Augmented Generation on LLM

Based on graph database

The core discussion: node embedding

What we have:

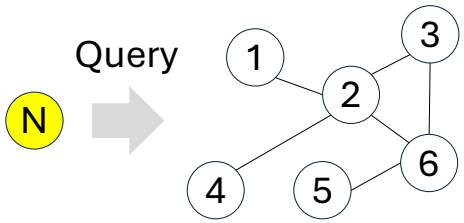
Content of Node:
Domain/knowledge agnostic string

Edge: domain agnostic, discrete or numeric edge

What we need:

Method to calculate distance/category of edges from unseen node.

Concept of flow:



$P_{edge1} = f_1(N, 1)$ $Dst_1 = f_2(N, 1)$
$P_{edge2} = f_1(N, 2)$ $Dst_2 = f_2(N, 2)$
....

F(a,b): <node_a_vector>*<node_b_vector>^T
<node_x_vector> = node embedding + weighted neighbor factor

Main discussion focus:

Node embedding: embedding strategy, node vectorization, ...
Weighted neighbor: sampling strategy, edge vectorization, ...

Vectorization method Categorization:

ML embedding:

- Node2Vec (2016, https://arxiv.org/abs/1607.00653)
- GraphSAGE, GCN, Loss = self-embedding similarity + negative neighbor penalty (2018, https://arxiv.org/abs/1706.02216)
- HashGNN, “hash” layer to project high dimension vector to low for training and inference (2020, https://arxiv.org/abs/2003.01917)

Graph DB / engine for searching

Graph searching engine	Support backend	Commercial	Introduction
Stellargraph	Local inference	no	A community maintained tool for graph ML, including classification of node, edge, and entire graph, edge prediction, etc.
LangGraph.js	Local inference	no	A community maintained tool for graph ML, alternative of Stellargraph.
neo4j	SaaS	yes	A company maintained SaaS service, provide ML based knowledge graph construction and graph based AI-QA system.

GDB: practical implementation

Stellargraph	LangGraph.js	neo4j

0. Instantiate graph
`G = sg.StellarGraph(nodes, edges)`
1. Prepare model
`G_test, G_train = train_test_split(G)`
`graphsage = GraphSAGE(...)`
2. Determine training task
`prediction = link_classification(...)`
`optimizer=keras.optimizers.Adam`
`model.compile...`
3. Predict
`node -> graphsage -> model -> prediction`
	0. Instantiate graph
	`workflow = StateGraph(MessagesState).add_node(...).addEdge(...)`
	1. Load model and tools
	`model = ChatOpenAI`
	`tools = [TavilySearchResults(max_results=1)]`
	2. Compile graph
	`app = workflow.compile(checkpointer=checkpointer)`
	3. Inference
	`app.invoke( {"messages": [HumanMessage(content="what is the weather in sf")]}, config={"configurable": {"thread_id": 42}} )`
		0. buy Graph SaaS from neo4j
		1. Use API to query