* Graph Fusion } → Preferred

3 Step process of KGC :-

① Seed Concept Generation :- Identify key concepts from free-text data.
Tech used { ① Event detection ② Dependency parsing } → output → a set of seed concepts that serves as the foundation for the next step.

② Candidate Triple Extraction :- Extract KG triples.

③ KG fusion :- Merge & reconcile candidates triples into a consistent, final KG.
① Entity Merging
② Conflict Resolution
③ Novel triple.

* GRAG : ① Graph Retrieval & Pruning -
→ Input Query & Graph Indexing -

Detailed Steps {Pipeline}

① Step-1 :- Concept Generation

① Download NLTK Resources like (word net) for lemmatization

② Input raw text

③ Bertopic → for Embedding
→ UMAP for dimensionality Reduction
→ Countvectorizer for tokenization

④ Extract topics :- Unique list of Extracted Concepts.

TEXT2BENCH / Evaluation

(5) Normalize concept → to get base form

(6) Identify text relevant to each concepts

(7) Save output.

(2) Triplets Extraction

Extract triplets

→ Read concepts & ~~entities~~ abstracts.
→ Relation definition (ontology) → Play parameter
→ Prompt template
→ Generate triplets
→ Validate & ~~extr~~ filter triplets

(3) Knowledge Graph Fusion

→ triplets + Relation definition } Inputs
→ Create concept mapping (concept to id or id to concept)
or
[redacted]
→ Constructs Graph.

[redacted] Graph

→ Initialize prompt template (prompt-fusion)

→ Iterate over each concept
  ↓
  ① Generate sub graph
  ② Combines text with relation definition
  ③ Send this info to LLM for fusion.

→ Validate outputs → only valid relations (ontology)

→ Save outputs

---

* **HIPPO RAG** :-

Problem :- 1) Vanilla RAG
→ DB regard ~~chunks~~ relationships blw chunks

2) Graph RAG
→ Rigid Schema (Ontology) for their KG construction. (labour intensive)
→ Integrating KG with RAG is challenging.
→ Usually 2 layer mechanism. (KG & Vector Database)

3) **Hippo RAG** → KG + RAG + PPR.
↓
Personalised Pagerank Algorithm
{No, cypher or SparQL language to query DB}

PPR → does traversal in the graph to identify relevant nodes & the corresponding passages.

Open IE

Steps :- ① Offline Indexing

→ Construct KG from Corpus { No restriction }

→ Create N×N (KG) matrix

N×P (matrix)

Nodes
$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$
Nodes.

Nodes
$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$
Paragraphs

② Online Retrieval

→ Extract entities from Queries.

→ Link to KG nodes using (cosine similarity)

→ Leverage PPR (Personalized Page rank)

→ Retrieve relevant nodes (subject), & corresponding objects (nodes) (object)

Using relationships.

→ then use [N×P] matrix to identify relevant passages (chunks) to feed to LLM to generate the answer.