

Graph Fusion

} → preferred

3 Step process of KGC :-

Seed Concept Generalisation :-

Identify key concepts from free-text data.

① Event detection

② Dependency parsing

} → output → a set of seed concepts that serves as the foundation for the next step.

Candidate Triple Extraction :-

Extract KG triples.

KG fusion :-

Merge & reconcile candidate triples into a consistent, final KG.

① Entity Merging

② Conflict Resolution

③ Novel triple.



Detailed Steps { Pipeline }

Step-1 :- Concept Generation

① Download NLP resources like (word net) for lemmatization

② Input raw text

③ Bertopic → for Embedding
→ UMAP for dimensionality reduction
→ Counter for tokens

④ Extract topics :- Unique list of Extracted Concepts

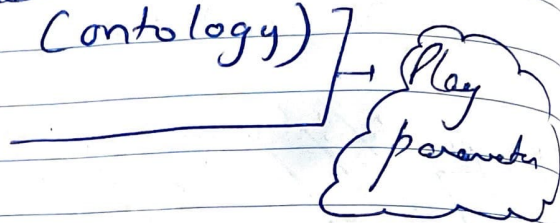
TEXT BENCH + Evaluation

- ⑤ demarize concept → to get base form
- ⑥ Identify text relevant to each concept.
- ⑦ Save output.

② Triplets Extraction

Extract triplets.

- Read concepts & ~~extra~~ abstracts.
- Relation definitions (ontology)
- Prompt template
- Generate triplets
- Validate & ~~extra~~ filter triplets.



③ Knowledge Graph Fusion

- triplets + Relation definitions } Input
- Create concept mapping (concept to id or id to concept)

or

~~Constructs Graph~~

- Constructs Graph

~~Constructs 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 output

Hippo RAG

Problem :- 1) Verilla RAG

→ DB regard ~~changes~~ relationships b/w chunks

2) Graph RAG

→ Rigid Schema (Ontology) for their KG construction (Jabian 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.

Steps:- ① Offline Indexing

Open 12

→ Construct KG from Corpus {No restrictions}

→ Create $N \times N(KG)$ matrix $N \times P$ (matrix)

Nodes

1	0	1	0	1
0	1	0	1	0
0	0	1	0	0
1	0	0	1	0
0	0	1	0	1

Nodes

Nodes

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

Paragraphs

② Online Retrieval

→ Extract entities from Query.

→ Link to KG nodes using (cosine similarity)

→ Leverage PPR (Personalized Page Rank)

→ Retrieve relevant nodes & corresponding objects ^{nodes} _{subject} ^{objects} _{object} using relationships.

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