GraphRAG Agentic Pipeline

HANOUZ Akram HORMI BOUAICHI Safa JOUVIN Jules

GraphRAG Agentic Pipeline

Limites des RAG traditionnels

- Recherche basée uniquement sur la similarité vectorielle
- Difficultés pour :
  - raisonnement multi-hop
  - relations complexes entre entités
  - synthèse inter-documents

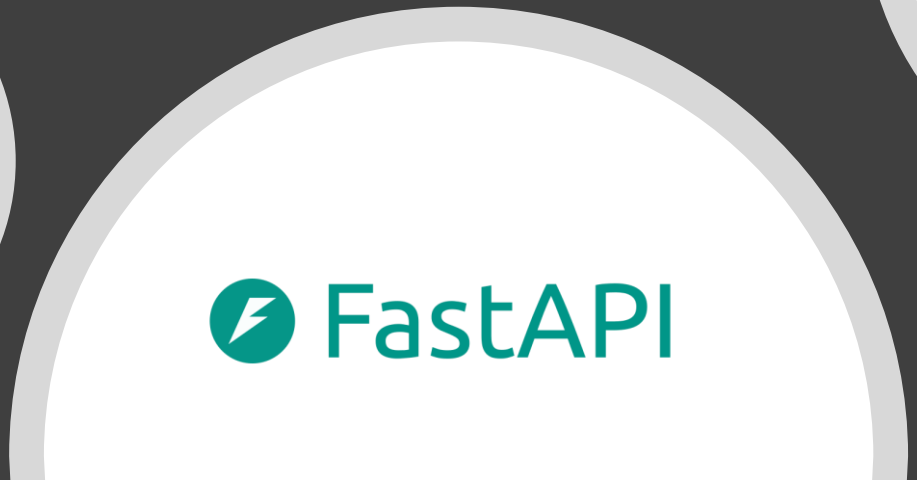
Idée clé : GraphRAG

- Construction automatique d'un **Knowledge Graph**
- Combinaison de :
  - recherche vectorielle (pertinence)
  - parcours de graphe (raisonnement)
  - Pipeline **agentique** et **local**

Architecture globale

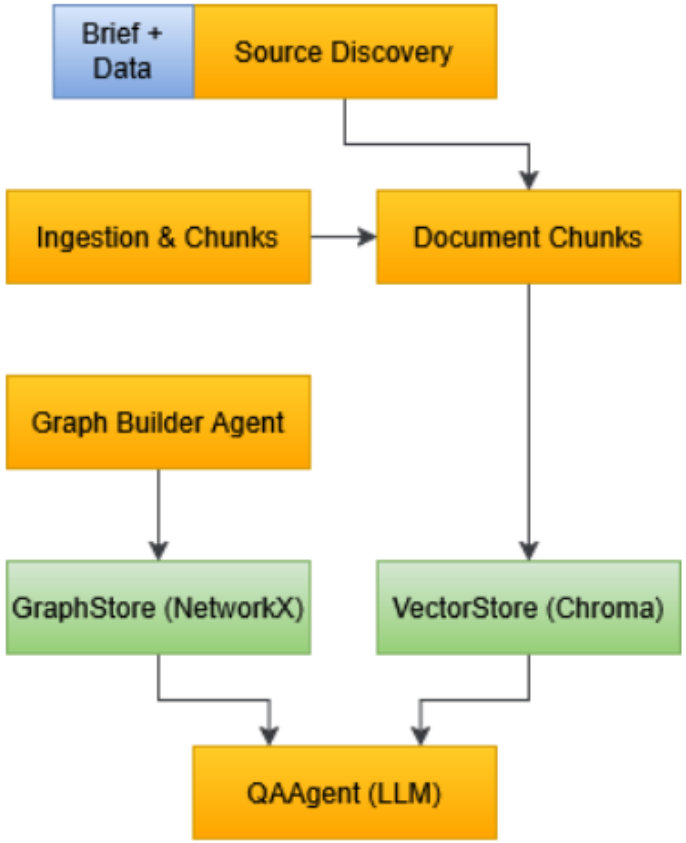
Vue d’ensemble

- Architecture modulaire en agents
- Backend : Python + FastAPI
- Frontend : Streamlit
- Deux stores complémentaires :
  - Knowledge Graph (NetworkX)
  - Vector Store (ChromaDB)



Agents principaux

SourceDiscoveryAgent	sélection intelligente des documents (hybride)
IngestionAgent	normalisation des formats
GraphBuilderAgent	extraction entités / relations
EmbeddingAgent	embeddings pour la recherche
QAAgent	réponses avec citations + chemins de graphe



Traitement des données & graphe

De texte à graphe

- Ingestion de PDFs, CSV, HTML, TXT…
- Chunking :
  - 400 tokens
  - overlap de 50 tokens
- Extraction LLM → JSON structuré
- Fusion des entités entre documents
- Traçabilité complète jusqu’au chunk source

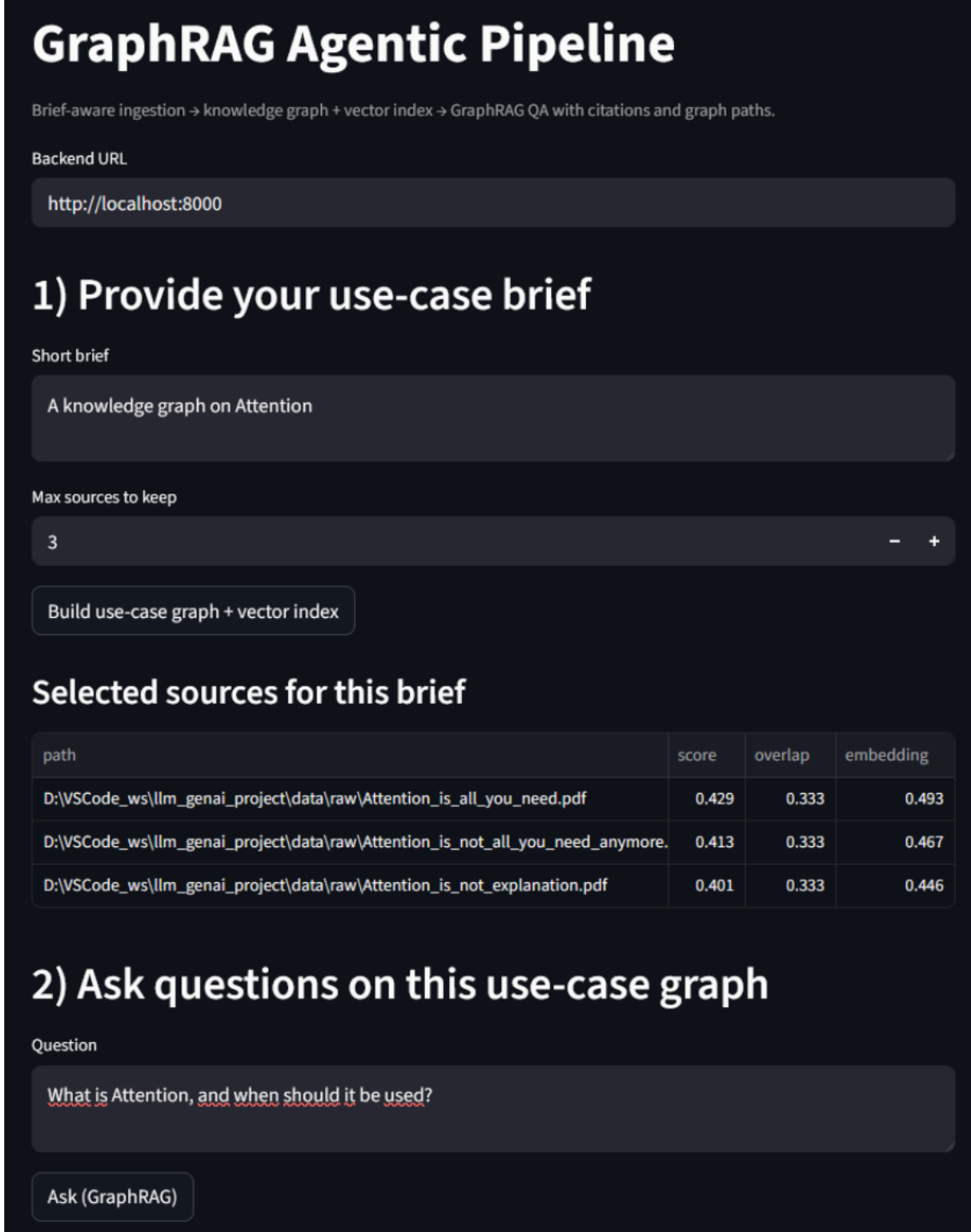
Choix techniques

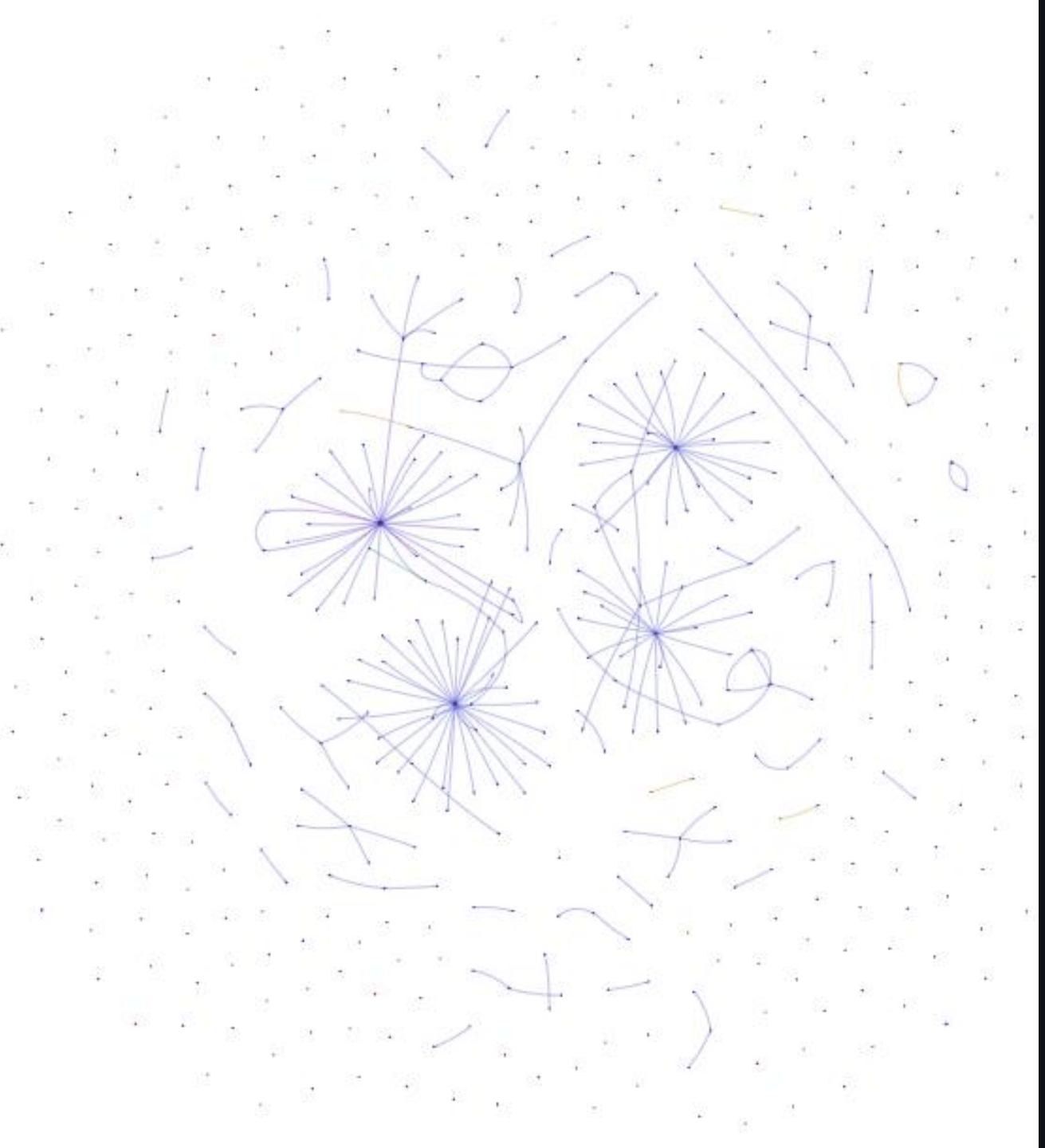
- **Ollama (local-first)**
  - pas de coûts API
  - données privées
- **LLaMA 3**
  - bon compromis performance / latence
- **nomic-embed-text**
  - embeddings optimisés pour le retrieval
- **NetworkX**
  - simple, suffisant pour un graphe documentaire

Démonstration & interface

Streamlit UI

- Saisie d’un brief utilisateur
- Visualisation des sources sélectionnées
- Chat de questions-réponses
- Réponses enrichies avec :
    - citations
    - chemins dans le graphe
- Visualisation interactive du Knowledge Graph





Answer

A long text!

What is attention?

In the context of natural language processing (NLP) and deep learning, attention refers to a mechanism that allows a model to focus on specific parts of an input sequence or document when making predictions. Attention helps the model to selectively weigh the importance of different elements in the input, such as words or tokens, based on their relevance to the task at hand.

When should attention be used?

Attention is particularly useful in situations where:

1. **Long-range dependencies**: The input sequence has long-range dependencies, meaning that the relationships between distant parts of the sequence are important for making predictions.
2. **Variable-length inputs**: The input sequences have varying lengths, and the model needs to adapt its processing to accommodate different lengths.
3. **Selective focus**: The model needs to selectively focus on specific parts of the input sequence or document based on their relevance to the task.

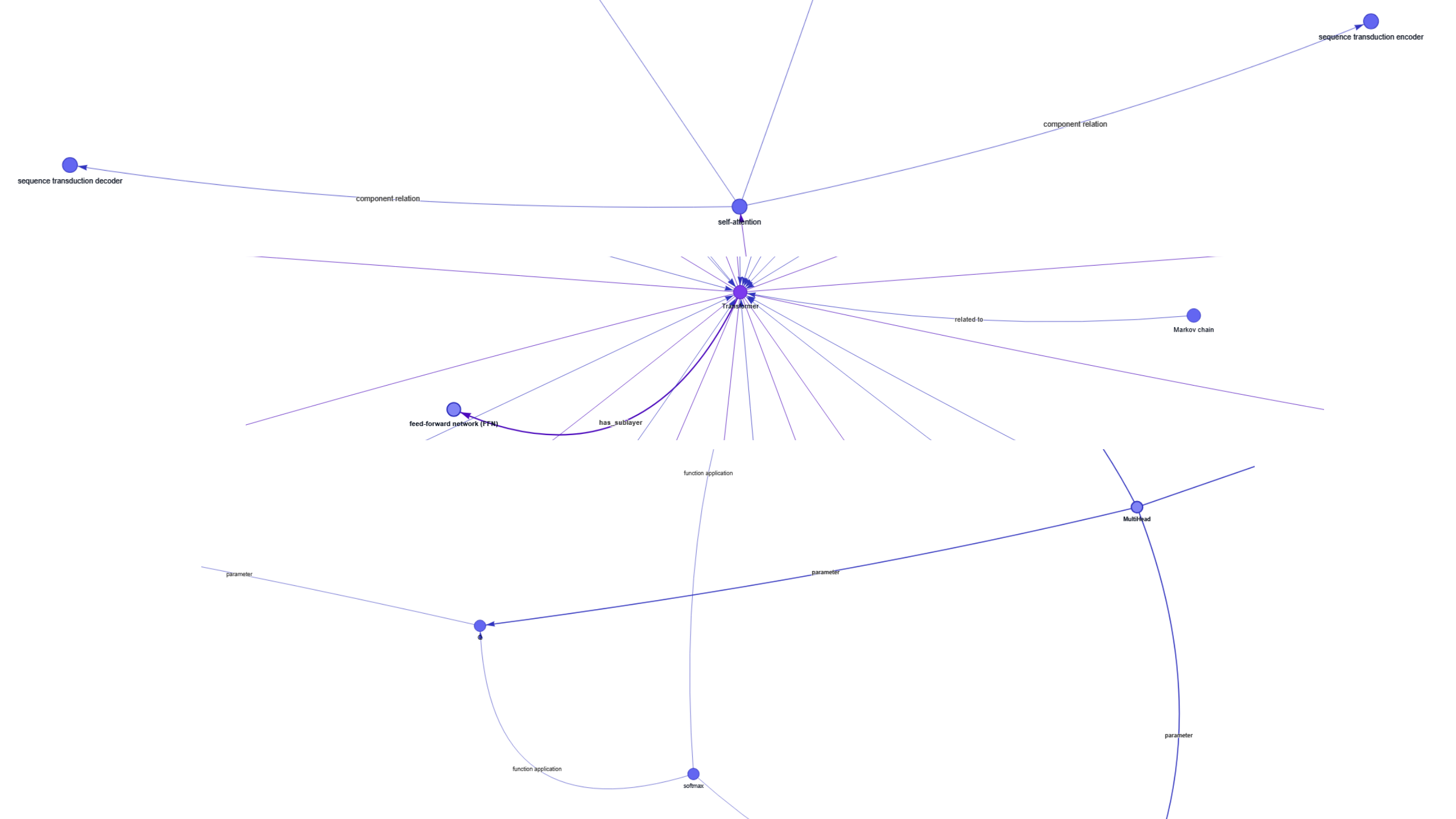
Some examples where attention is commonly used include:

1. **Machine translation**: Attention helps translate sentences by focusing on relevant words and phrases in both the source and target languages.
2. **Question answering**: Attention enables models to focus on relevant parts of a passage when answering questions.
3. **Text classification**: Attention helps classify text based on specific features or keywords.

In summary, attention is a powerful mechanism that allows models to selectively focus on important parts of an input sequence or document, making it particularly useful in situations where long-range dependencies, variable-length inputs, and selective focus are relevant.

Citations

- `D:\VSCode_ws\llm_genai_project\data\raw\Attention_is_not_explanation.pdf@7000-7400`
- `D:\VSCode_ws\llm_genai_project\data\raw\Attention_is_all_you_need.pdf@5950-6113`
- `D:\VSCode_ws\llm_genai_project\data\raw\Attention_is_not_explanation.pdf@0-400`
- `D:\VSCode_ws\llm_genai_project\data\raw\Attention_is_not_explanation.pdf@5950-6350`
- `D:\VSCode_ws\llm_genai_project\data\raw\Attention_is_not_explanation.pdf@2450-2850`


sequence transduction encoder
component relation
sequence transduction decoder
component relation
self-attention
Transformer
related to
Markov chain
feed-forward network (FFN)
has_sublayer
function application
MultiHead
parameter
parameter
parameter
function application
softmax

Évaluation & résultats

Évaluation du système

- Basée sur GraphRAG-Bench (sample de 735 questions)
- Utilisation du framework Ragas
- Métriques :
    - couverture
    - grounding
    - faithfulness
    - pertinence
- Résultats :
    - ~23 % de succès strict
    - hallucinations encore présentes
- Latence médiane : ~10,5 s

Limitation

- Très long a évolué
    - Evaluer sur seulement ¼ du benchmark
    - Génération d'un GraphRag très limité

Non représentatif de nos vrais résultats

Limites & Conclusion

Limites et perspectives

- Chunking non sémantique
- Génération parfois trop verbeuse
- Scalabilité limitée

Améliorations futures

- Chunking sémantique
- Décomposition des requêtes
- Contraintes de génération
- Migration vers une vraie base de graphes