PROJET RISK MANAGEMENT

Realized by:

Yasmine Marzouk
Ben Amor Hamza
Roua Mrabet
Hedi Thameur
Slim Zouari
Brahime Abdelbeki

Content

Introduction

Problematic

Risk MisIdentification & Management

Delayed Risk Identification

Difficulty in quickly identifying potential risks due to limited or inefficient access to relevant project data

Unstructured Data Usage

Limitations in risk analysis caused by reliance on unstructured data, making it difficult to extract precise and actionable information.

Lack of Predictive Capability

Absence of mechanisms to predict future risks based on historical data, reducing the ability to anticipate and mitigate impacts on the project.

Business Objectives

Cost and Time Efficiency

Automation of decision-making regarding risk management process

Automated Recommendations for Risk Mitigation

User-Friendly Risk Monitoring Dashboard

Data science Objectives

Use automated deep learning to detect risks without manual effort

Create and implement advanced data models to automate risk management using machine learning and predictive analytics, providing accurate decision-making recommendations.

Build deep learning models to predict future risks using historical data; develop classification models to forecast likelihood and impact of risks.

Build a risk visualization system with dynamic dashboards; aggregate and present risk statuses from multiple data sources for easy monitoring.

Relative metrics for the entire project

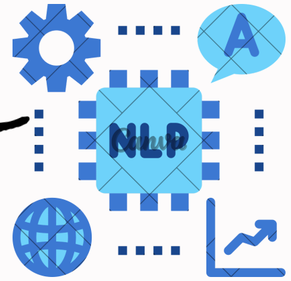
Smart metrics help make projects clear and effective. They make it easier to manage tasks and achieve goals on time

Proposed solution

PMI_RM

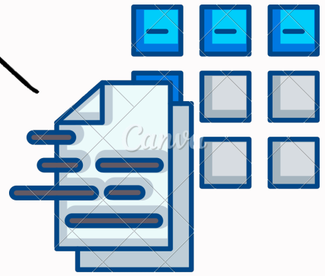
The PMI_RM Contains all the most important texts and sentences abourt projet RM’s Processes

PDF Data Extraction (library)

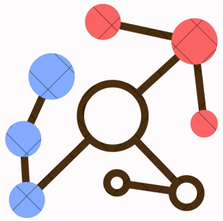


Using NLP processes for text semantic relationship analytics

Extract the necessary Keywords

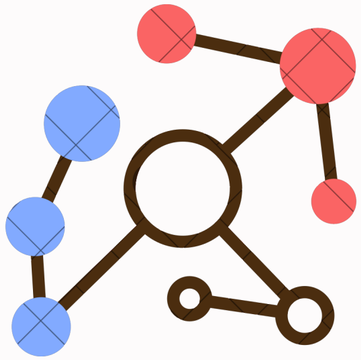


Creating coherent dataframes

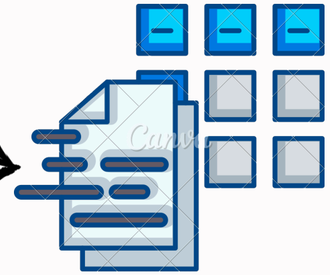


The Detailed General Process

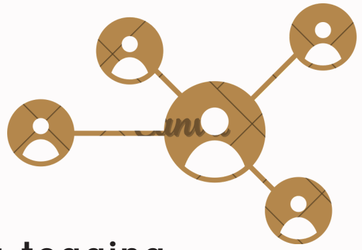
To make a graphe



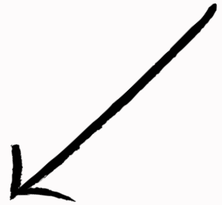
We have already a solution we use our dataframe grouped_df



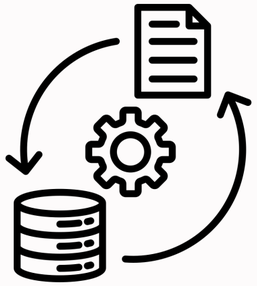
we studied the smeantic relationships



with post-tagging



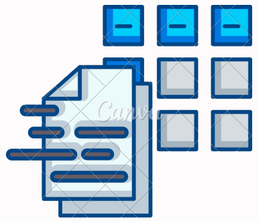
we Extracted the verbs



VERB

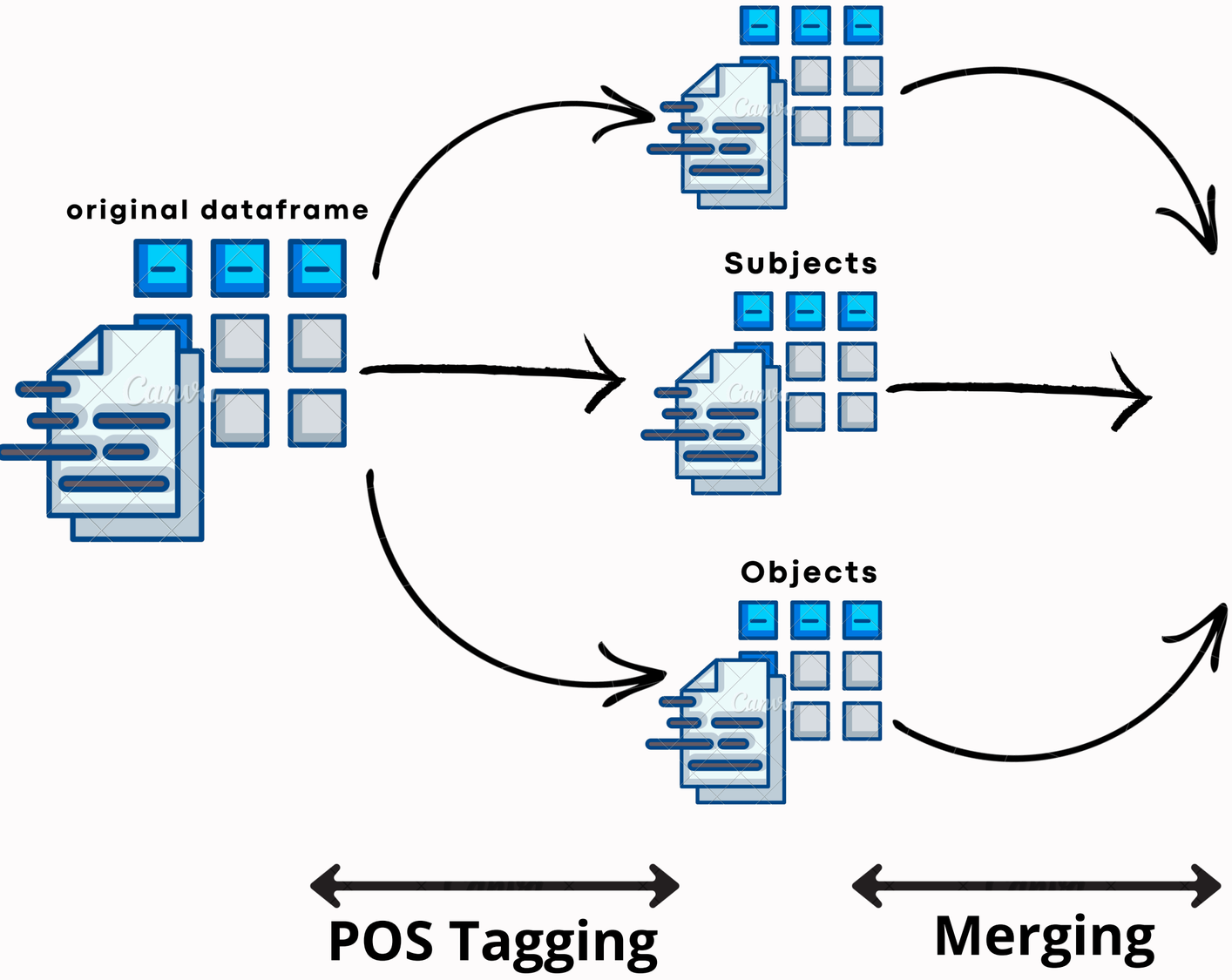
```
# Fonction pour extraire les verbes et leurs relations
def extract_verbs_relations(sentence):
```

```
Index: 0
  Verb: identify, Subject: , Object:
  Verb: manage, Subject: , Object:
  Verb: identify, Subject: , Object:
  Verb: aim, Subject: process, Object:
  Verb: identify, Subject: , Object: risk
  Verb: project, Subject: , Object: t
  Verb: identify, Subject: , Object: risk
  Verb: take, Subject: level, Object:
  Verb: identify, Subject: , Object: risk
  Verb: be, Subject: that, Object:
  Verb: fact, Subject: , Object:
  Verb: require, Subject: , Object: the
  Verb: identify, Subject: , Object:
  Verb: repeat, Subject: , Object: the
```



	verb	count
0	identify	67
5	be	54
7	require	36
87	provide	31
32	occur	30
..	...	...
155	go	1
157	stop	1
158	re	1
162	find	1
283	predict	1

[284 rows x 2 columns]

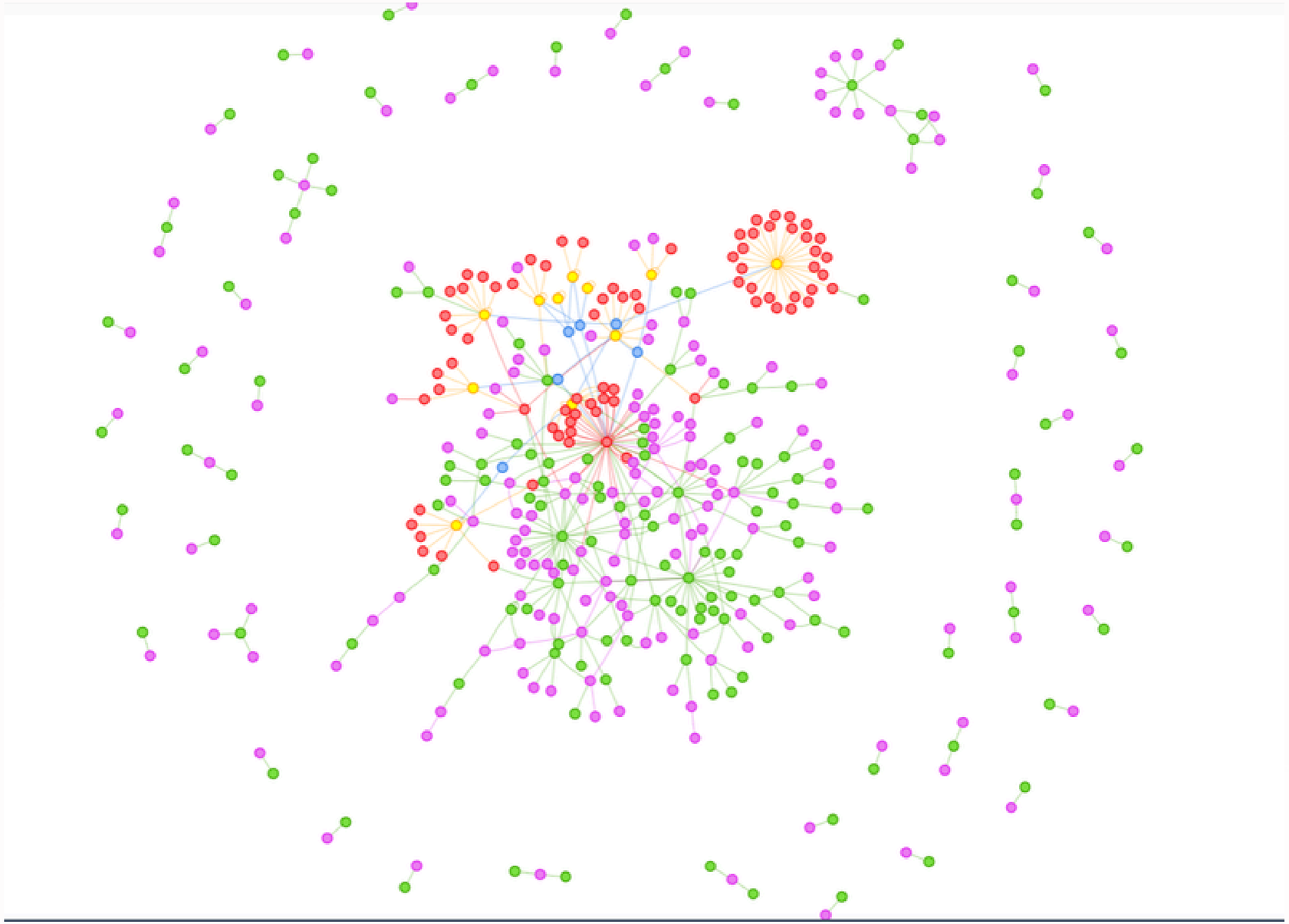


	Subject	Predicate	Object
0	process	take	input
1	description	require	uncertainty
2	people	use	judgment
3	enhance	have	strength
4	process	use	combination
...	...	...	...
352	breakdown structure	is a	tailor
353	risk cause	is a	group
354	project risk management	is a	structure
355	management plan	is a	base
356	risk strategy	is a	invoke

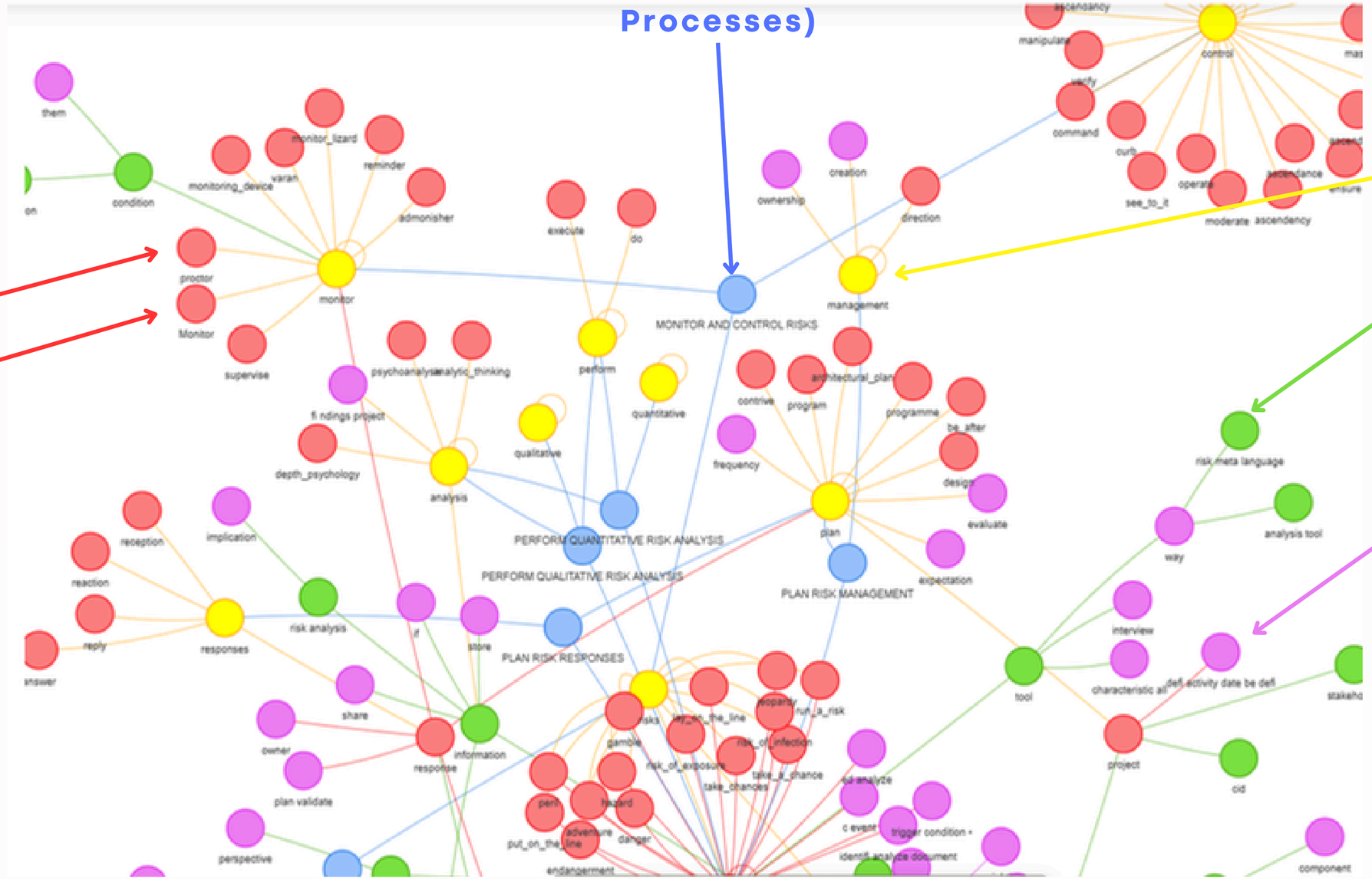
341 rows × 3 columns

We eliminated adverbs and determiners to improve the quality of the analysis and achieve more meaningful results.

Our Conceptual Graph

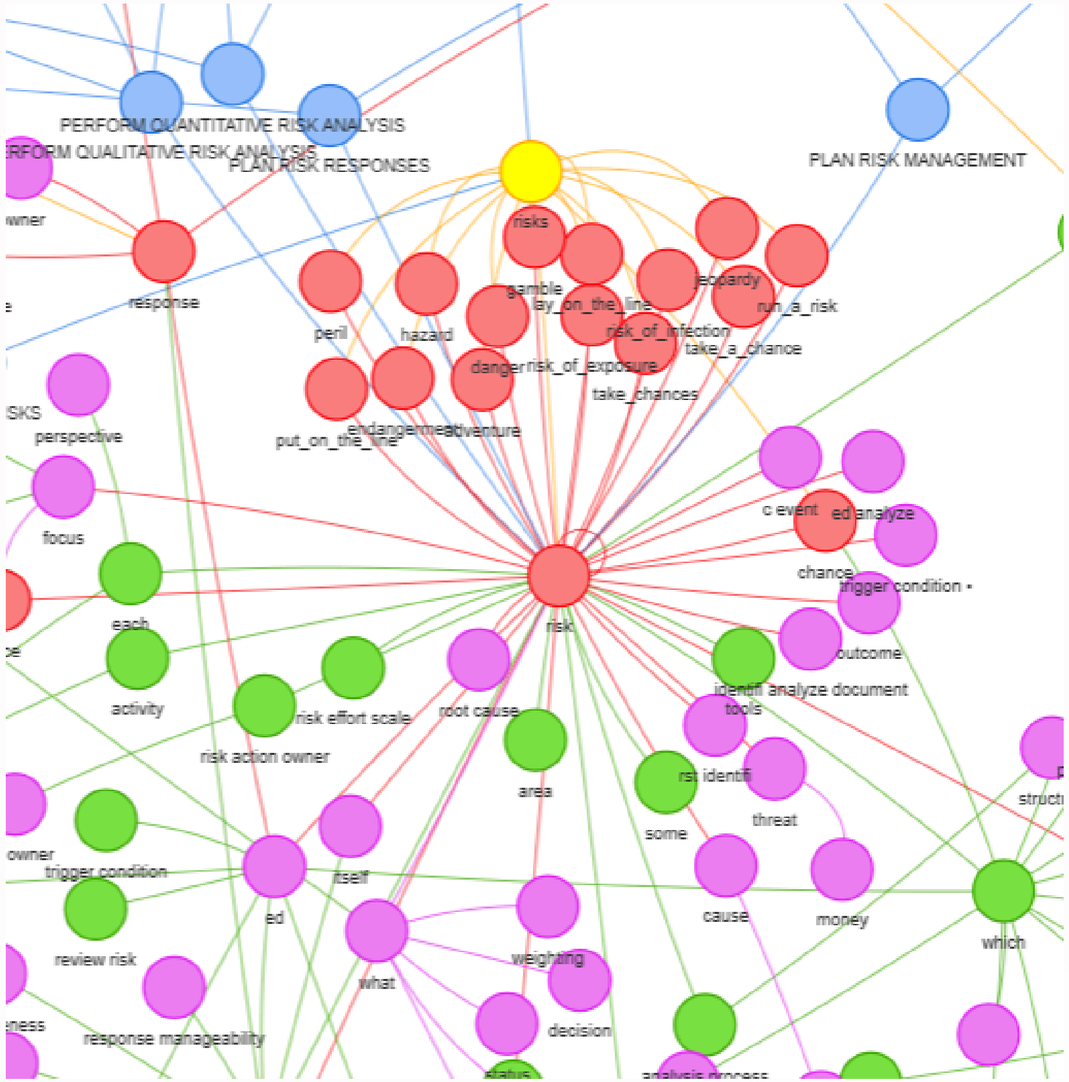


Our Conceptual Graph



Our Conceptual Graph

Each Edge represents semantic relationships between the nodes (verbs)



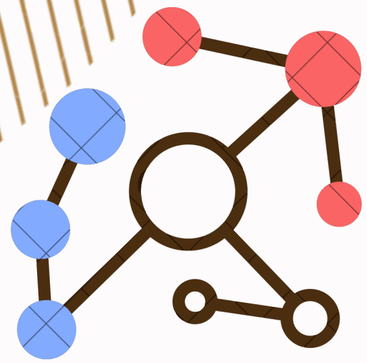
Graph Summary

```
# Calculer le degré de chaque nœud
```

```
Nœud: IDENTIFY RISKS, Degré: 2
Nœud: identify, Degré: 10
Nœud: risks, Degré: 18
Nœud: discover, Degré: 1
Nœud: key, Degré: 1
Nœud: distinguish, Degré: 1
Nœud: place, Degré: 3
Nœud: name, Degré: 1
Nœud: key_out, Degré: 1
Nœud: describe, Degré: 2
Nœud: risk_of_exposure, Degré: 2
Nœud: take_chances, Degré: 2
Nœud: jeopardy, Degré: 2
Nœud: run_a_risk, Degré: 2
Nœud: lay_on_the_line, Degré: 2
Nœud: chance, Degré: 3
Nœud: take_a_chance, Degré: 2
Nœud: peril, Degré: 2
Nœud: risk, Degré: 51
Nœud: put_on_the_line, Degré: 2
Nœud: adventure, Degré: 2
Nœud: risk_of_infection, Degré: 2
Nœud: gamble, Degré: 2
Nœud: danger, Degré: 2
Nœud: endangerment, Degré: 2
Nœud: hazard, Degré: 2
Nœud: MONITOR AND CONTROL RISKS, Degré: 3
Nœud: control, Degré: 30
Nœud: monitor, Degré: 13
Nœud: ascendency, Degré: 1
Nœud: assure, Degré: 2
Nœud: insure, Degré: 1
Nœud: dominance, Degré: 1
Nœud: hold_in, Degré: 1
```

```
#Top Les Meuilleurs degrés des nodes du graphe
```

```
Top 5 Nœuds les Plus Connectés:
risk: 51 connexions
control: 30 connexions
use: 25 connexions
plan: 15 connexions
monitor: 13 connexions
identify: 10 connexions
method: 10 connexions
analysis: 9 connexions
document: 8 connexions
response: 7 connexions
information: 7 connexions
```



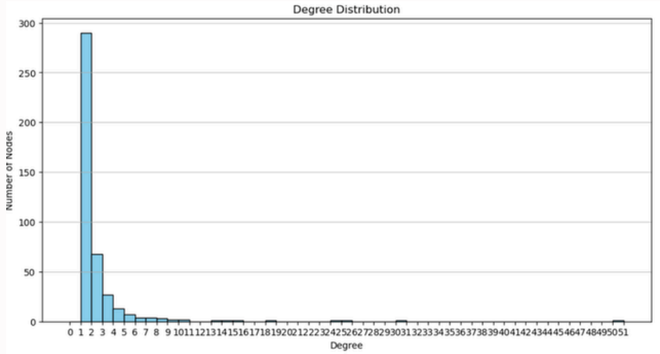
graph evaluation

Average Clustering

$$C_{\text{average}} = \frac{1}{N}\sum_{v=1}^{N} C_v$$

```
Number of Nodes: 428
Number of Edges: 449
Average Degree: 2.0981308411214954
Diameter: inf
Number of Components: 56
Size of Largest Component: 288
```

Nbre Edges > Nbre Nodes



Each Node 2 connections

Low dense Graph

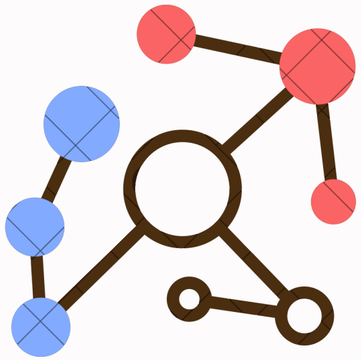
Infinite diametre ==>certain nodes are not connected to each other

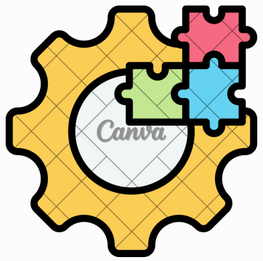
56 sub-graph connected to each other

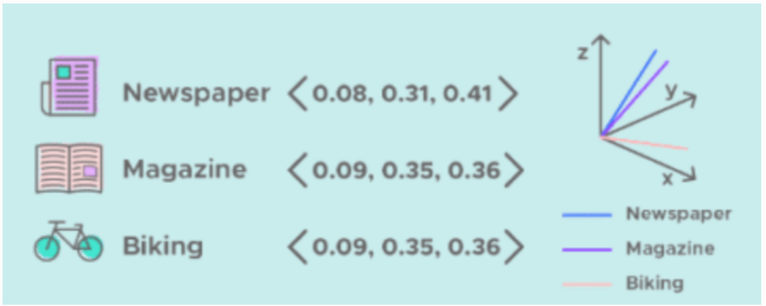
288 graph's nodes community well-connected

graph embedding

after graph evaluation

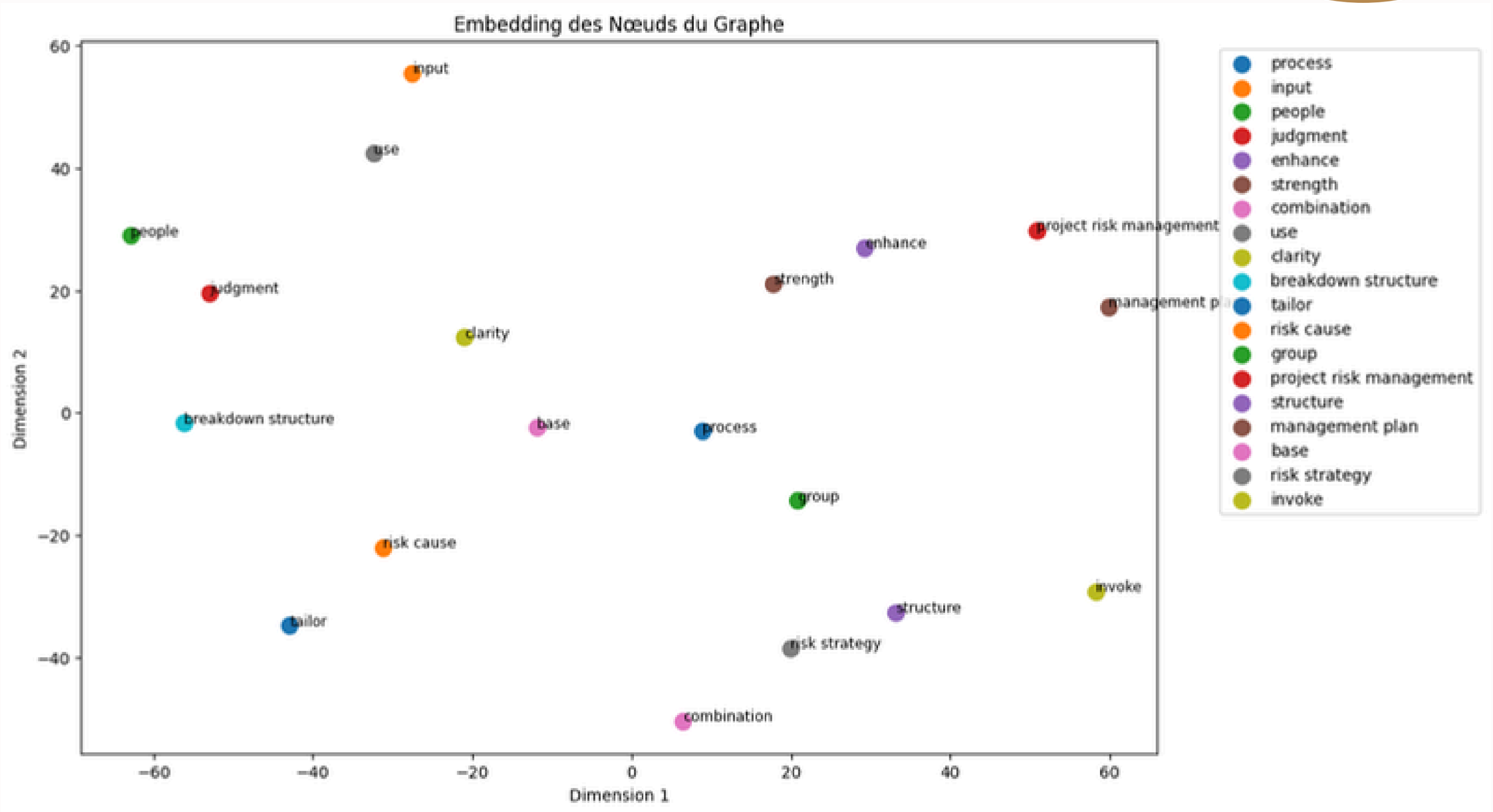




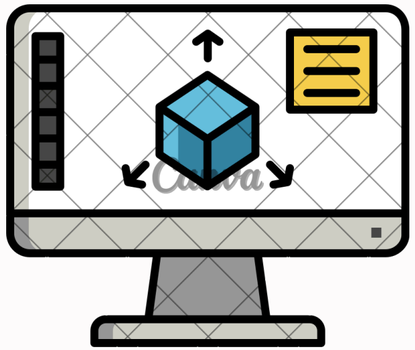


Node2Vec + Node2Text + Node2Edge

Embedding of nodes

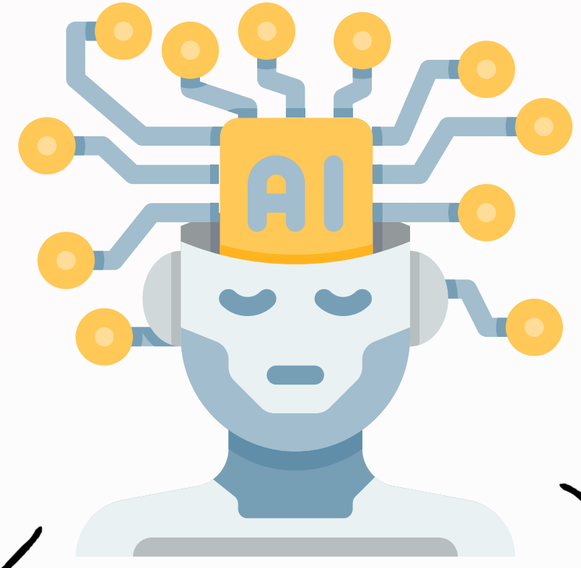


Modeling (The First classic approch)



Some challenges we have faced(1)

Models Accuracy



GAT : 0.1852

GNN :0.23

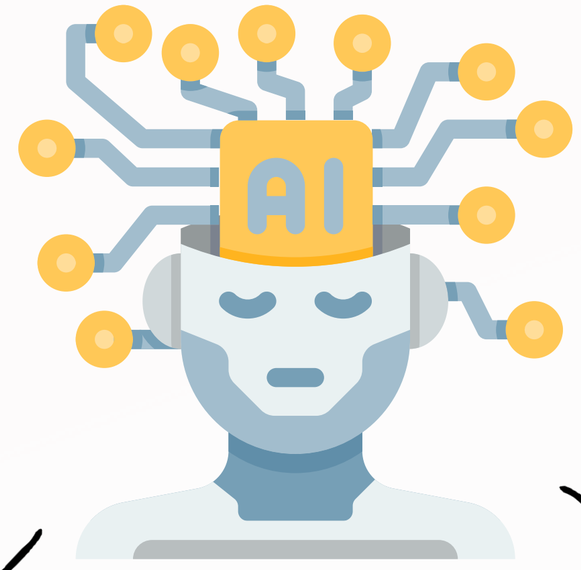
GCN:0.21

SAGE:0.19

These accuracies are suffisant to builde a robust graph to deal with, and make some reasoning and logical predictions

Some challenges we have faced(2)

Choosing the most appropriate model



GAT

GNN

GCN

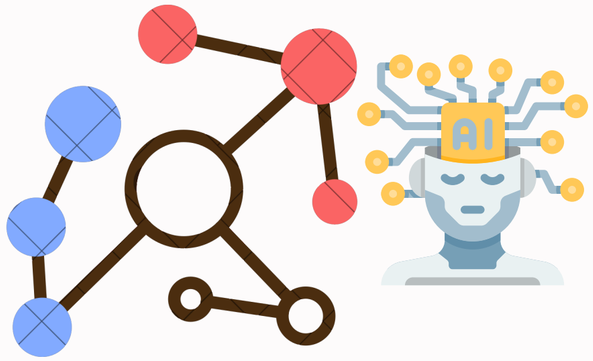
SAGE

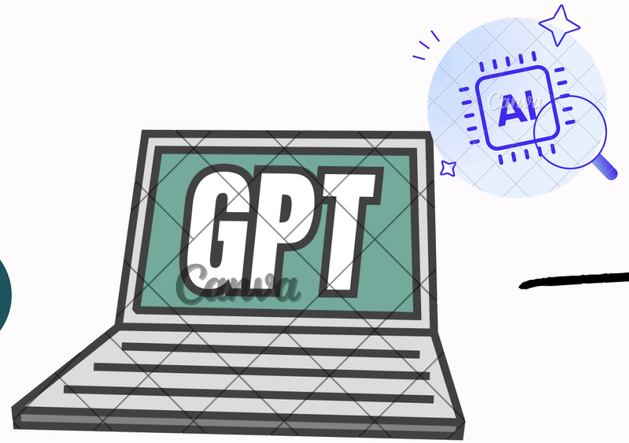
Difficulty to choose the most appropriate model for our task , and to merge a certain combination of certain layers in a graph model

Another Challenge

how to enchance the outputs/answers from our model?

Our Graph model





GPT-2 Pre-trained Model

We give the question and the context nodes from the graph

```
question = "How does the root cause affect risk management?"
context_nodes = ["root cause", "risk effort scale", "event", "some"]
```

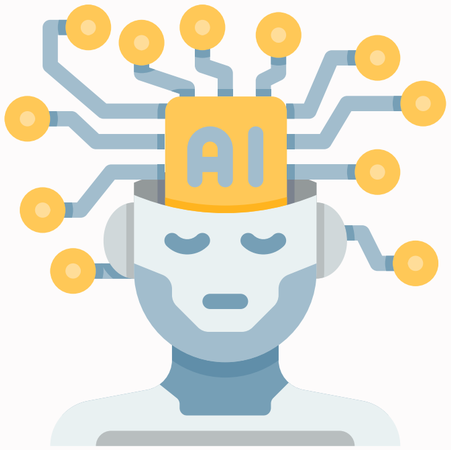
```
response = generate_response(question, context_nodes)
print(response)
```

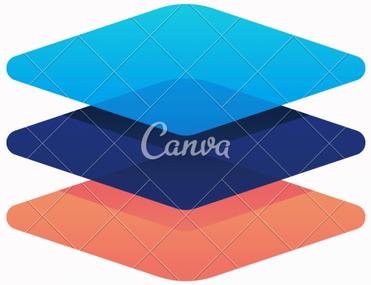
```
Question: How does the root cause affect risk management?
Context: Here are some important factors in risk management: root cause, risk effort scale, event, some
Answer: The root causes of risk are:
1. The risk of a disease or injury.
2. A person's ability to control their own health. This is the most important factor. It is important to understand that the ri
sk is not always the same for all people. For example, if you have a high risk for heart disease, you may have higher risk than
others. If you are a smoker, your risk may be higher than other people, but you will have lower risk. In addition, the risks of
certain diseases are different for different people and different groups
```

The Result

Some challenges we have faced(3)

Attention Layer and Reasoning Model





Attention Layer

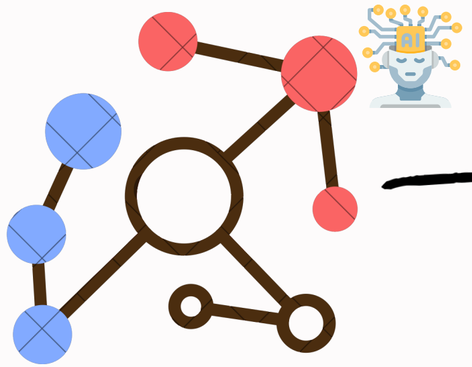
which is used to weigh the information from different neighbors according to their relative importance. Attention allows the model to give more weight to relevant connections for each task by adapting the weighting of messages sent by each neighbor

Difficult to integrate as a layer in a specific model GNN

Another Challenge

How to answer from a specifique question using our graph?

Our Graph model



Smiliraty computing of our graph's nodes

```
# Calculer la similarité cosinus entre les embeddings des nœuds
similarity_matrix = cosine_similarity(node_embeddings)
```

22

Then we compare the embedding of the imput question and the nodes's embeddings

```
# Déplacer la question vers le device et calculer l'embedding de la question
question = "How is the risk management ?"
question_inputs = graph_processor.tokenizer(question, return_tensors='pt', padding=True, truncation=True, max_length=512)

# Évaluation du modèle BERT pour obtenir l'embedding de la question
with torch.no_grad():
    question_outputs = graph_processor.bert_model(**question_inputs)

# Projection de l'embedding BERT vers l'espace latent, suivi du détachement pour conversion en numpy
question_embedding = graph_processor.projection(question_outputs.last_hidden_state[:, 0, :]).detach().cpu().numpy()
```

We display the closest nodes to the input question

```
Nœud le plus proche 1: Index = 325, Similarité = -0.0059
Nœud le plus proche 2: Index = 228, Similarité = -0.0059
Nœud le plus proche 3: Index = 291, Similarité = -0.0059
Nœud le plus proche 4: Index = 296, Similarité = -0.0059
Nœud le plus proche 5: Index = 398, Similarité = -0.0059
```

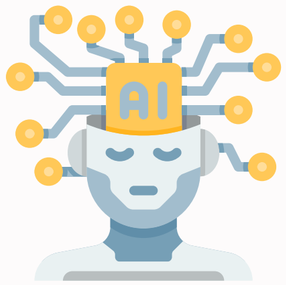
The answer of the question

[identifie ,root cause ,risk effort scale , event]

It seems logique

The Solution we made 1

Build a massive model with all different layers



Our Model

=

```
class GATQAModel(nn.Module)
```

+

```
class GraphQAProcessor
```

2 Attention Layers

```
self.gat1 = GATConv(in_channels, hidden_channels, heads=num_heads, dropout=0.3)
self.gat2 = GATConv(hidden_channels * num_heads, hidden_channels, heads=1, dropout=0.3)
```

+ Dropout Technics to reduce overffiting

Linear Layer

```
self.linear = nn.Linear(hidden_channels, hidden_channels)
```

+ 8 Multi-head attentions

Used for data training and processing

- Tokenizer from 'bert-base-uncased'
- A pre-trained model 'bert-base-uncased'

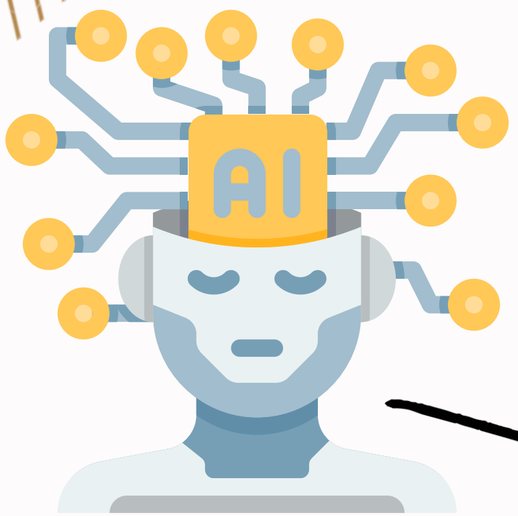
Used for Graph to pytorch Geometric conversion

- Node Embeddings

The Solution we made 2

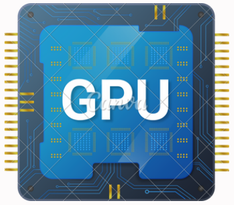
The Training phase of our model

Our Graph Model



Train our mode with keywords extracted form the previous data

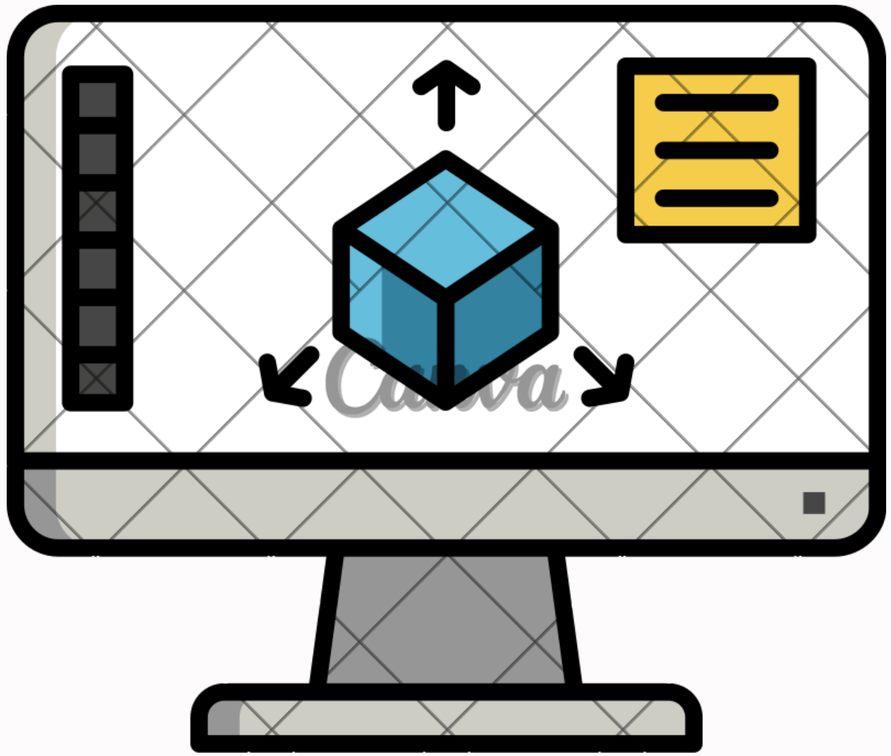
We Train our model using GPUs to accelerate the training phase




```python
# Fonction pour tester la disponibilité du GPU
def test_gpu_availability():
    """Test if GPU is available"""
    return torch.cuda.is_available()
```

We manipulated and fine-tuned the attention layers using GATConv to upgrade the reasoning accuracy of our graph model

Modeling (The Second approch)



Introduction to new pre-trained advanced model for graphe building

Neo4j

Neo4j is a highly scalable, native graph database designed to leverage data relationships as first-class entities. Unlike traditional relational databases, which use tables to store data, Neo4j uses nodes, relationships, and properties to represent and store connected data. Nodes represent entities, relationships connect nodes, and properties store relevant information about these nodes and relationships.

Key Features of Neo4j:

Graph-based Storage: Neo4j stores data as a graph directly on disk, allowing it to model complex, interconnected data naturally.

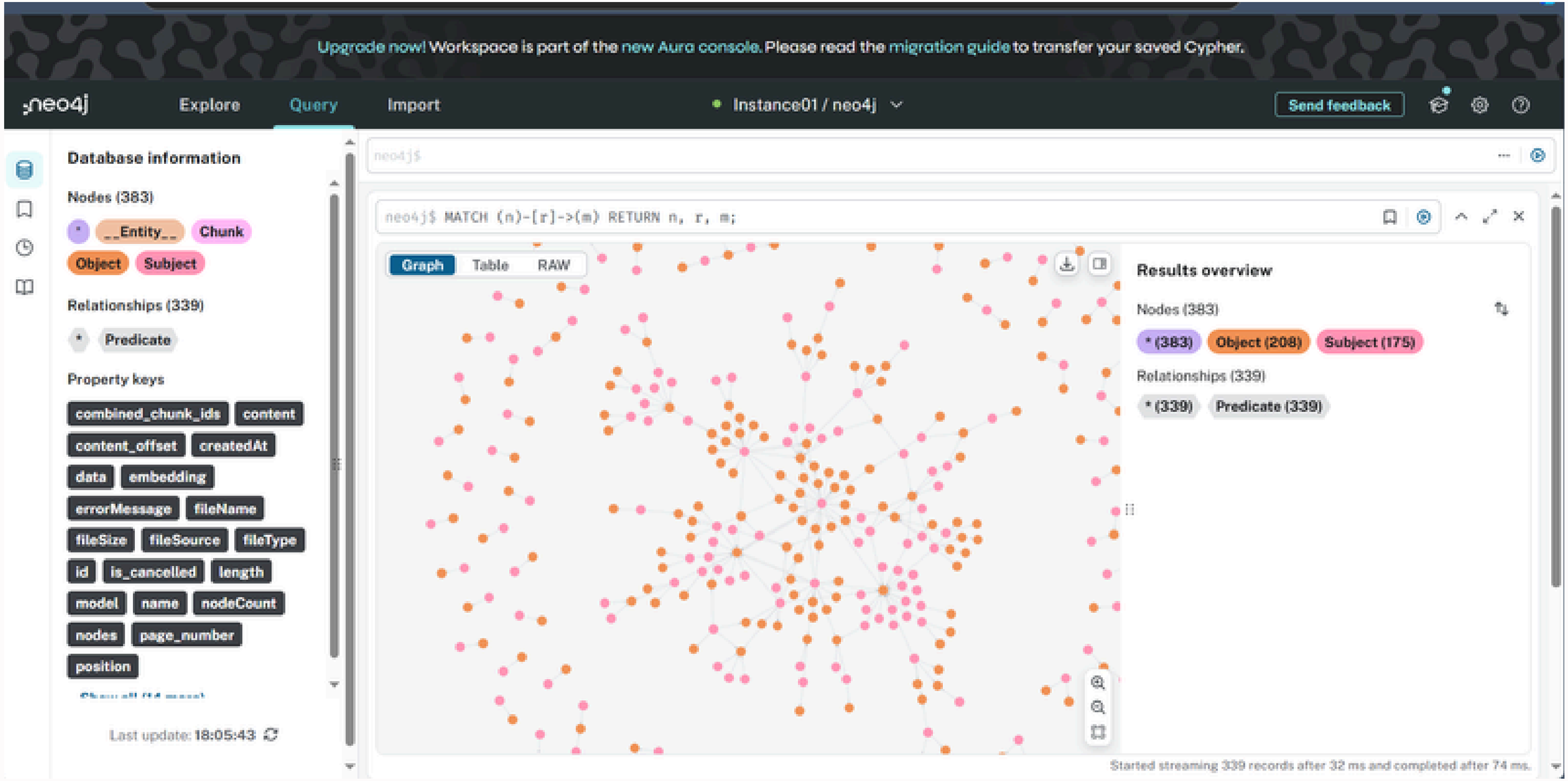
Cypher Query Language: It uses Cypher, a powerful and easy-to-read graph query language, to traverse, analyze, and manipulate the graph.

High Performance for Connected Data: Neo4j excels at handling highly connected data and complex queries, often performing significantly faster than traditional databases on such workloads.

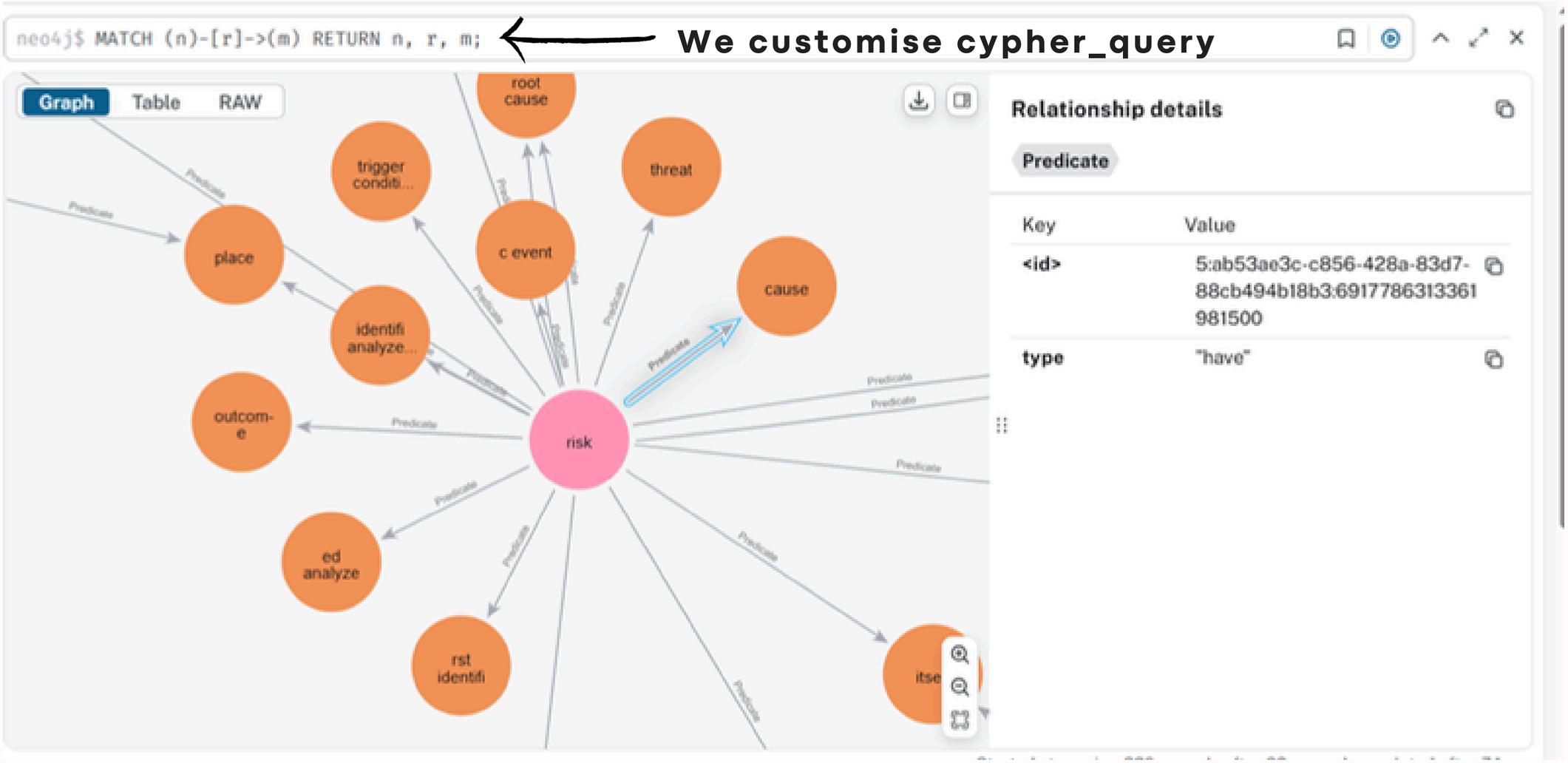
Scalability and Flexibility: Neo4j is scalable across large datasets, making it suitable for applications in recommendation engines, fraud detection, network management, knowledge graphs, and more.

neo4j

Our Conceptual Graph



Our Conceptual Graph neo4j



Our Conceptual Graph neo4j

Our DataFrame

	Subject	Predicate	Object
0	process	take	input
1	people	use	judgment
2	enhance	have	strength
3	process	use	combination
4	use	ensure	clarity
...	...	...	...
334	breakdown structure	is a	tailor
335	risk cause	is a	group
336	project risk management	is a	structure
337	management plan	is a	base
338	risk strategy	is a	invoke

339 rows × 3 columns

```python
def insert_into_neo4j(tx, subject, predicate, obj):
    # Cypher query to create or merge nodes and relationships
    query = """
    MERGE (s:Subject {name: $subject})
    MERGE (o:Object {name: $object})
    MERGE (s)-[r:Predicate {type: $predicate}]->(o)
    """
    # Execute query with provided subject, predicate, and object
    tx.run(query, subject=subject, predicate=predicate, object=obj)
```

We inserted nodes and edges into our neo4j Graph

```
Node properties:
Subject {name: STRING}
Object {name: STRING}
Relationship properties:
Predicate {type: STRING}
The relationships:
(:Subject)-[:Predicate]->(:Object)
```

We used another LLM model integrable into our graph neo4j model for high ranked graph outputs

```python
class ChatGoogleGenerativeAI(BaseLLM, BaseModel):
    model: str
    google_api_key: str
    temperature: float = 0.0
    url: str = Field(default="https://generativelanguage.googleapis.com/v1beta/models/gemini-1.5-flash-latest:generateContent"
```

```python
def _generate(self, prompts: list[str], stop: list[str] = None) -> LLMResult:
    # Prepare the payload for the API request
    data = {
        "contents": [{"parts": [{"text": prompt}]} for prompt in prompts]
    }

    # Define the headers
    headers = {'Content-Type': 'application/json'}

    # Send the request to the Google Generative API
    response = requests.post(self.url, params={'key': self.google_api_key}, headers=headers, data=js

    # Handle the response
    if response.status_code == 200:
        response_json = response.json()
        print(f"API Response: {response_json}")  # Debugging line to see the response

        # Check if there are results in the response
        if 'results' in response_json and response_json['results']:
            texts = [{"text": content.get('content', 'No content returned')} for content in response
            generated_cypher = texts[0].get('text', '').strip()

            # Validate that the generated Cypher is not empty or invalid
            if generated_cypher:
                return LLMResult(generations=[texts])
            else:
                # Handle case where the generated Cypher is invalid
                return LLMResult(generations=[{"text": "Generated Cypher query was empty."}])
        else:
            # Handle case where no results are found
            return LLMResult(generations=[{"text": "No results returned from the API."}])
    else:
        # If the API call failed, handle the error
        error_message = f"Error {response.status_code}: {response.text}"
        return LLMResult(generations=[{"text": error_message}])
```

We used this function to deal with gemini api (llm model) and to generate a cyber query accoding to user input

```python
# Assuming `graph` is already defined

# Assuming `graph` is already defined
llm = ChatGoogleGenerativeAI(model="gemini", google_api_key="AIzaSyDpcsj6GIUQkrfS-TzdpKaA58dxqO3hVVQ",

chain = GraphCypherQAChain.from_llm(
    graph=graph,
    llm=llm,
    verbose=True,
    allow_dangerous_requests=True  # Set this to acknowledge potential risks
)
```

We used langchain library to answer the input question using the inference technique in the knowledge graph for better entity extractions and their neighbors in the graph

An Exemple for corresponding question-Cypher-Query

```python
# Fonction pour générer les requêtes
def generate_cypher_queries(df):
    queries = []

    for _, row in df.iterrows():
        subject = row['Subject']
        predicate = row['Predicate']
        obj = row['Object']

        # Générer la question et la requête Cypher en fonction du prédicat
        if predicate == 'take':
            queries.append({
                "question": f"What does {subject} take?",
                "query": f"MATCH (p:{subject.capitalize()})-[:TAKES]->(o:{obj.capitalize()}) RETURN p.name"
            })
```

Another Exemple

```
# Exécution de la requête
with driver.session() as session:
    query = """
    MATCH (p:Entity)-[r:RELATION]->(input)
    WHERE r.type = 'use' AND p.name = 'process'
    RETURN p.name, input.name
    """
    result = session.run(query)
    print(result.data())  # Afficher les résultats
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_7468\2083389073.py:2: DeprecationWarning:

Using a driver after it has been closed is deprecated. Future versions of the driver will raise an error.
```

```
[{'p.name': 'process', 'input.name': 'combination'}]
```

```
# Sélectionner des exemples en fonction de la question
selected_examples = example_selector.select_examples({"question": "What does process use?"})

# Extraire la requête Cypher de l'exemple sélectionné
cypher_query = selected_examples[0]['query']

# Connexion à Neo4j
driver = GraphDatabase.driver("neo4j+ssc://b249df7a.databases.neo4j.io", auth=("neo4j", "UWaK4Bf7fgFMko-w
```

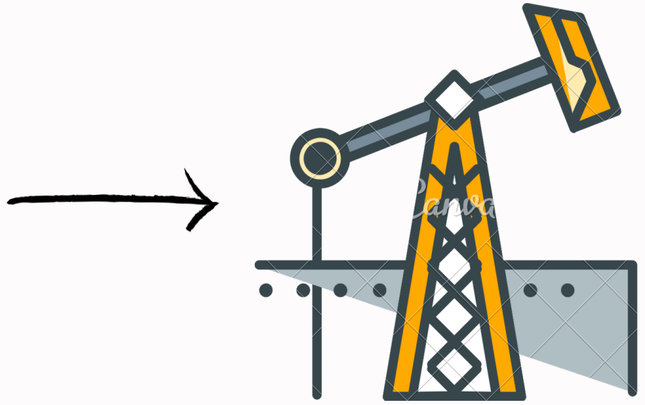
```
Résultats de la requête Cypher:
[{'p.name': 'process'}]
```

Another Challenge

How to enrich our graph by training another use cases data

We extracted several uses cases from the official PMI Web Site

Project Management Institute.



aramco

AstraZeneca

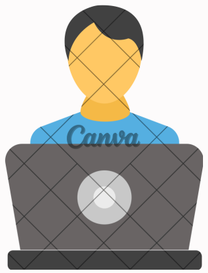
Fighting a Global Pandemic
(Risk Mangement Project Use Case)

Offshore Construction in Crowded Waters (Risk Mangement Project Use Case)

Another Challenge

How to make these prompts dynamique according to each user's input

User



Enter a prompt

Similarity

we used a library called :

```
SemanticSimilarityExampleSelector
```

We are going to calculaty the similarity from given output to make the most simiar dynamique cypher query from neo4j graph

```
example_selector.select_examples({"question": "What does process use?"})
```

```
{'query': "MATCH (p:Entity)-[r:RELATION]->(input) WHERE r.type = 'use' AND p.name='process' RETURN p.name",
 'question': 'What does process use?'},
```

Each word of the given question will be interptreted as a single node/entity to be search in the whole knowledge graph

In order to get an appropriter Cypher Query for better responses

Critique of Existing Approaches(1/3)

Traditional Approaches

SAP SECURITY & GRC ®

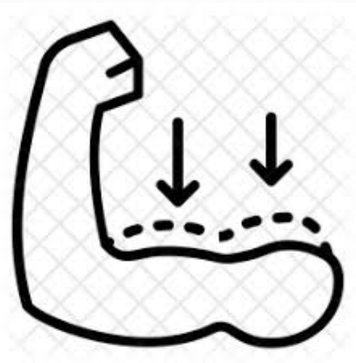
ARCHER

RiskWatch

Critique of Existing Approaches(2/3)
Solutions based on AI & Chatbot

Critique of Existing Approaches(3/3)

These platforms are typically robust, well-integrated, and offer a wide range of features for risk management, compliance, and auditing.



Current solutions often have lack accessibility for SMEs, are not designed for simplified use (such as a chatbot interface), and are costly

Value Proposition

- Our AI-powered risk management chatbot provides real-time insights, reduces human error, and enhances decision-making efficiency.
- Our solution will be able to handle complex scenarios and its user-friendly design tailored to SMEs.

Poster


14 November
DATA-WARRIORS
Empowering Smart Risk Management
"Future-Proof Your Risk Strategy"
PLAY NOW
Intelligent
Efficient
Resilient

Flyers


DATA-WARRIORS
WHY US ?
WE PROPOSE THE DEVELOPMENT OF AN AI-POWERED RISK MANAGEMENT SYSTEM THAT INTEGRATES SEVERAL ADVANCED TECHNOLOGIES AND METHODOLOGIES. THE SYSTEM WILL AUTOMATICALLY IDENTIFY, PREDICT, AND RECOMMEND STRATEGIES FOR MANAGING PROJECT RISKS, MAKING IT A VALUABLE TOOL FOR PROJECT MANAGERS AND DECISION-MAKERS.
OUR MISSION
TO EMPOWER BUSINESSES WITH A PROACTIVE, AI-DRIVEN RISK MANAGEMENT TOOL THAT SIMPLIFIES COMPLEX PROCESSES, ENHANCES DECISION-MAKING, AND PREVENTS FUTURE RISKS.


DATA-WARRIORS
HOW IT WORKS ?
1. INPUT DATA: PROJECT MANAGERS DESCRIBE THEIR CURRENT SITUATION OR SPECIFIC CHALLENGES.
2. ANALYSIS BY REASONING MODEL: THE CHATBOT'S REASONING ENGINE EVALUATES THE INFORMATION USING PRE-DEFINED KNOWLEDGE AND ADVANCED ALGORITHMS.
3. RISK PREDICTION AND STRATEGY RECOMMENDATION: BASED ON THE ANALYSIS, IT PROVIDES ACTIONABLE INSIGHTS AND STRATEGIES TO MITIGATE OR MANAGE RISKS.
KEY FEATURES
• REAL-TIME RISK ASSESSMENT: INSTANTLY ANALYZE AND INTERPRET POTENTIAL RISKS AS THEY ARISE.
• PREDICTIVE ANALYTICS: USES ADVANCED ALGORITHMS TO PREDICT FUTURE RISKS, GIVING DECISION-MAKERS A PROACTIVE ADVANTAGE.
• NATURAL LANGUAGE PROCESSING (NLP): UNDERSTANDS AND PROCESSES USER INPUT NATURALLY, ALLOWING SEAMLESS INTERACTION FOR NON-TECHNICAL USERS.
• SCALABLE AND ADAPTABLE: DESIGNED FOR BOTH SMES AND LARGE ENTERPRISES, PROVIDING TAILORED SOLUTIONS THAT GROW WITH YOUR BUSINESS NEEDS.

Conclusion

Thank you!