Se former autrement

HONORIS UNITED UNIVERSITIES

2024/ 2025

SPÉCIALITÉ : Data Science

Project AI & Cognition : Risk Management
AI- based Recommandation System

Réalisé par:

Hamza Ben Amor Ibrahime Abdelbeki

Yasmine Marzouk Rouaa Mrabet

Hedi Thameur Selim Zouari

Encadré par: Wiem Zouga

Dedication

This report is dedicated to the incredible efforts and unwavering dedication of our group members: Hamza Ben Amor , Yasmine Marzouk , Hedi Thameur , Slim Zouari, Rouaa Mrabet . Throughout the duration of this project, each member has displayed a remarkable level of commitment and synergy that has been crucial to our success.

The collaboration in our group was characterized by a profound mutual respect for each other's ideas and a consistent willingness to support one another. Our journey together was not just about meeting project deadlines but also about blending diverse perspectives and expertise to forge a comprehensive and coherent approach to solving complex problems. Every discussion and meeting was an opportunity to learn from each other and to push our collective creative boundaries.

We sincerely thank each group member for their hard work, innovative ideas, and the solidarity that helped transform challenges into opportunities for growth. This project would not have reached its full potential without the unique contributions of each team member. It is with great pride and gratitude that we share the success of this project with them.

Acknowledgments

Before we begin presenting this report, we would like to express our sincere gratitude to everyone who helped us complete our integrated project.

We would like to particularly thank **Ms. Wiem Zaouga**, our professor, for her patience, commitment, and guidance that led to the success of our project. her availability, dedicated supervision, and valuable advice were instrumental in our progress.

We also want to express our gratitude to all our professors at the institute for their high-quality coaching and to all the administrative staff for their dedication, hard work, and support in ensuring the success of our project.

Lastly, we warmly thank the jury members for the honor they have bestowed upon us by agreeing to evaluate our work. Please accept our sincere sentiments of gratitude and respect.

Contents

List of Figures

List of Tables

Acronyms

TDSP: **T**eam **D**ata **S**cience **P**rocess

CSV: **C**omma-**S**eparated **V**alues

NLP: **N**atural **L**anguage **P**rocessing

General Inroduction

In today's complex project environments, managing risks effectively is crucial to the success of any initiative. Traditional risk management methods, while useful, often struggle to predict risks accurately and offer timely recommendations. This academic project, part of the AI and Cognition program, aims to tackle this challenge by developing a recommendation system powered by Natural Language Processing (NLP), deep learning models, and ontology-based knowledge representation.
The project's goal is to assist users in identifying, analyzing, and mitigating potential risks in real-time. By leveraging advanced AI techniques, the system will not only predict risks but also suggest proactive measures, providing a comprehensive solution to enhance project risk management. The end result will be a deployable web application, accessible to users across different domains, contributing to better decision-making and risk mitigation strategies.

In this context, this project focuses on bridging the gap between traditional risk management practices and modern AI-driven solutions, providing businesses with the tools they need to anticipate and address risks effectively.

Our report is organized as follows:

Chapter 1: Project Overview

Chapter 2: Business Understanding

Chapter 3: Data Understanding and Preparation

Chapter 4: Modeling

Chapter 5: Deployment

Our report will conclude with a general conclusion that highlights the importance of the project.

CHAPTER 1

Project Overview

Introduction

In this chapter, we focus on the presentation of the project's context.At first, we will present the project framework introducing it and determining its goal.

The last part of this chapter is reserved for the definition of the methodology adopted for project execution, ensuring a systematic approach to achieving the project goals.

1.1 Project description

In this section, we will come over the scope of the project context and description.

1.1.1 Context of the project

As part of the university course for learning basic and advanced approaches and techniques in data science at the Esprit engineering school for the academic year 2024-2025 , we carried out an integrated project with the tunisian company (TenStep). The aim of this project is use natural language prossesing, machine learning and deep learning techniques to create an application that helps to an autmoatic intelligent recommandation and risk management

AI-system.

1.1.2 Problematic

The primary challenge in project risk management is the lack of a robust and automated system to identify, predict, and mitigate risks in real-time. Traditional methods rely heavily on human expertise and manual processes, which can result in delayed risk identification and inefficient response strategies. Furthermore, the increasing complexity and scale of modern projects—particularly in technology and software development—require more advanced approaches that leverage large datasets and predictive models to provide timely, accurate, and actionable insights for risk mitigation.

In this academic project, the goal is to develop an AI-powered risk management system that integrates Natural Language Processing (NLP), deep learning models, and ontology-based knowledge representation to automatically assess and recommend strategies for project risks. The system must be scalable and deployable in a web application to assist users, such as project managers and team leaders, in proactively managing risks throughout the project lifecycle.

1.1.3 The proposed solution

To address the problem of inefficient risk management in projects, this academic project proposes 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.

1.2 Methodology

To realize this work, we will utilize the Team Data Science Process methodology, proposed by the university, as our agile development approach.

1.2.1 Team Data Science Process Methodology

In October 2016, Microsoft introduced the Team Data Science Process as an agile, iterative data science methodology to deliver predictive analytics solutions and intelligent applications efficiently.

TDSP helps improve team collaboration and learning by suggesting how team roles work best together. It includes best practices and structures from Microsoft and other industry leaders to help toward successful implementation of data science initiatives. The goal is to help companies fully realize the benefits of their analytics program.

1.2.2 Team Data Science Process Steps

The figure below shows the steps of the Team Data Science Process methodology.

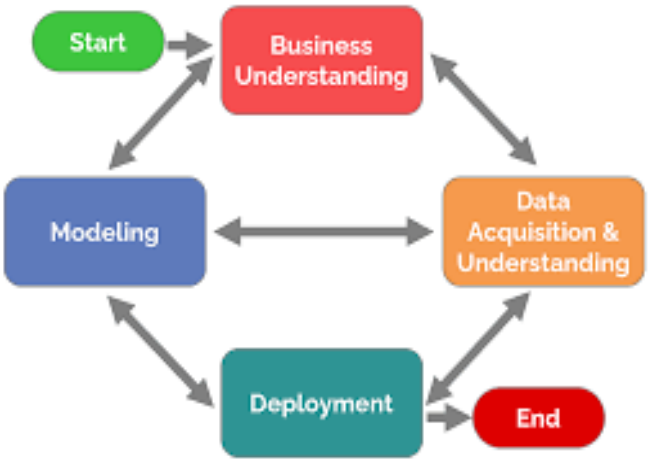

Figure 1: TDSP methodology

The lifecycle outlines the major stages that projects typically execute, often iteratively:

1. **Business Understanding :** define objectives and identify data sources.
2. **Data Acquisition and Understanding :** ingest data and determine if it can answer the presenting question (effectively combines Data Understanding and Data Cleaning from CRISP-DM).
3. **Modeling :** feature engineering and model training (combines Modeling and Evaluation).

4. **Deployment :** deploy into a production environment.

1.2.3 Team Data Science Process Roles

The figure below shows the Roles of the Team Data Science Process methodology.

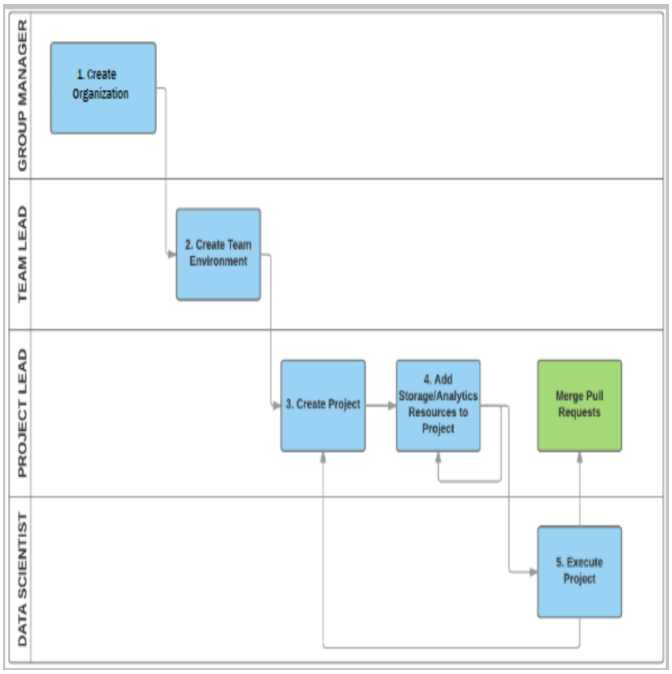

Figure 2: TDSP Roles

- **Group Manager :** Manages the entire data science unit in an enterprise. A data science unit might have multiple teams, each of which is working on multiple data science projects in distinct business verticals.

- **Team Lead :** Manages a team in the data science unit of an enterprise. A team consists of multiple data scientists. For a small data science unit, the Group Manager and the Team Lead might be the same person.

- **Project Lead :**Manages the daily activities of individual data scientists on a specific data science project.

- **Project Individual Contributors :** Data Scientists, Business Analysts, Data Engineers, Architects, and others who execute a data science project.

Conclusion

During the first chapter, we presented an overview of the project along with its context. We also defined the project's objectives, described the proposed solution, and outlined the proposed methodology. Moving forward, the second chapter will focus on the first step of our methodology.

CHAPTER 2

Business Understanding

Introduction

Business Understanding is a crucial phase in a data science project. This step corresponds to the first stage of the TDSP methodology where we will better understand the issues that the company aims to solve.

2.1 Business Objectives

The main business objectives of the AI-powered project risk management system focus on improving the efficiency and effectiveness of risk management in projects. Here are the business objectives:

- **Enhance Risk Detection Efficiency:** Objective: Streamline the identification of risks by automating the extraction of risk factors from project data, reducing reliance on manual processes. Value: Saves time and increases the accuracy of risk detection, leading to more proactive risk management.
- **Improve Risk Prediction Accuracy:** Objective: Use historical and real-time project data to accurately predict potential risks and their impact on project outcomes. Value:

Helps project managers anticipate risks more effectively, allowing them to take preventive measures before risks materialize.

- **Provide Data-Driven Risk Mitigation Strategies:** Objective: Offer personalized, actionable recommendations based on data analysis to mitigate identified risks. Value: Improves decision-making by providing clear strategies for addressing project risks, minimizing the likelihood of project failure.

- **Increase Scalability and Accessibility:** Objective: Deploy the solution as a web-based application, making it accessible to a wide range of users across different industries and project types. Value: Ensures that the solution can be easily adopted by various organizations and scaled to handle projects of all sizes and complexities.

2.2 Data Science Objectives

Once the business objectives are well-defined, it is now appropriate to translate them into data science objectives:

- **Automate Risk Extraction Using Natural Language Processing (NLP):** Develop NLP models to automatically extract and classify risks from unstructured textual data (e.g., project reports, emails).

- **Predict Project Risks Using Deep Learning Models:** Build and train deep learning models (LSTM, RNN) to forecast the likelihood and severity of potential project risks based on historical and real-time data.

- **Design a Recommendation System for Risk Mitigation:** Implement a recommendation system that provides personalized, data-driven strategies for mitigating identified risks based on previous project outcomes and risk characteristics.

- **Build a Scalable and Deployable AI System:** Create a scalable and deployable machine learning pipeline that can handle large datasets, integrate multiple data sources, and adapt to different types of projects.

Conclusion

These data science objectives provide the technical foundation for the business objectives, ensuring that the system is both powerful and practical, capable of addressing project risks in an automated, accurate, and scalable manner.

CHAPTER 3

Knowledge Retrieval Analysis

Introduction

Phase 1, titled "Knowledge Retrieval and Analysis", focuses on the acquisition and comprehension of knowledge from unstructured data sources. The primary goal of this phase is to extract valuable information from standard corpora, such as PMI RM, and to process this information for further analysis. By utilizing advanced text processing techniques, we aim to segment, categorize, and identify key concepts, their attributes, and relationships within the data. This involves analyzing patterns of term frequency to determine the most relevant and frequent concepts.

Throughout this phase, various natural language processing (NLP) tools and methods are employed to ensure effective data analysis. These include tokenization, part-of-speech (POS) tagging, stopword removal, lemmatization or stemming, and chunking for syntactic segmentation. Furthermore, we evaluate the efficiency of the knowledge extraction process through quantitative measures such as Levenshtein distance, TF-IDF (Term Frequency-Inverse Document Frequency), and F-measure.

Ultimately, this phase serves as the foundation for building a structured understanding of the unstructured data, which will guide subsequent phases of the project.

3.1 Preprocessing and Analysis of Unstructured Data

3.1.1 Extracting Data from Standard Corpora (PMI RM Standard)

Extracting data from standard corpora, such as PMI RM (Project Management Institute Risk Management Standard) , involves several important steps to ensure the relevant information is identified and prepared for further analysis.

- **Identification of Key Sections:** The first step is to locate and identify the relevant sections or chapters in the corpus that contain valuable knowledge. For instance, in PMI, this may include chapters on project risk management.
- **Text Parsing:** Once the relevant sections are identified, text parsing techniques are used to break down the structured content into manageable pieces. This involves dividing the text into paragraphs, sentences, or words, preparing it for more detailed analysis.
- **Entity and Concept Identification:** Using techniques such as Named Entity Recognition (NER), key concepts, terms, and entities related to project management (e.g., “risk”, “stakeholder”, “milestones”) are extracted. This step ensures that the core ideas and entities are recognized.
- **Segmentation and Labeling:** Segmentation involves breaking the text into smaller components (e.g., noun phrases, verb phrases) to capture specific details like roles, processes, and deliverables mentioned in the standards.
- **Term Frequency Analysis:** A frequency analysis, such as TF IDF, can then be applied to identify the most commonly mentioned terms and concepts within the standard corpus. This helps highlight the key focus areas of PMI RM.

3.1.2 Segmenting Data: Identifying Nouns, Verbs, Synonyms

- **Tokenization:** Split the text into individual tokens (words or phrases) to prepare for further processing.
- **Part-of-Speech (POS) Tagging:** Label each token with its grammatical category, such as noun, verb, adjective, etc.
- **Named Entity Recognition (NER):** Identify and categorize proper nouns and specific entities such as names, dates, and locations.
- **Synonym Identification:** The first step is to locate and identify the relevant sections or chapters in the corpus that contain valuable knowledge. For instance, in 7, this may include chapters on project risk management.
- **Chunking (Phrase Extraction):** Group words into syntactic units or phrases, such as noun phrases or verb phrases.
- **Lemmatization/Stemming:** Reduce words to their root form to handle different inflections.
- **Filtering and Categorization:** Filter out irrelevant terms (e.g., stopwords) and categorize words based on their identified roles (e.g., all nouns, all verbs).

3.1.3 Identifying Concepts, Their Attributes, and Relationships, and Calculating Concept Frequency

- **Dependency Parsing:** Analyze the grammatical structure of sentences to understand how words relate to each other and how concepts are connected.
- **Synonym Grouping:** Group concepts that have similar meanings or synonyms to avoid redundancy and create more accurate frequency counts.
- **Frequency Calculation (Term Frequency):** Use TF IDF to assess the importance of a concept not only by its frequency in one document but also by its rarity across multiple documents, giving more weight to unique or rare concepts.

- **Relationship Mapping and Visualization:** Create visual maps or graphs to represent the identified concepts, their attributes, and the relationships between them. This helps in understanding the knowledge structure and key interconnections.

3.1.4 Evaluating the Effectiveness of the Knowledge Extraction Process

- **Define Evaluation Metrics:** Choose appropriate metrics to evaluate the quality of knowledge extraction. Common metrics include precision, recall, F-measure, and accuracy.
- **Apply Levenshtein Distance:** Use Levenshtein Distance (or edit distance) to measure how close the extracted text is to the expected correct text. It calculates the number of single-character edits (insertions, deletions, or substitutions) required to transform one text into another.

3.2 Another Approach with LLM

3.2.1 Data Loading and Preprocessing

- **LangChain:** In this project, we used LangChain to facilitate efficient data loading and preprocessing. LangChain helped us handle the pipeline for retrieving unstructured data, cleaning it, and preparing it for chunking and further analysis.

3.2.2 Text Chunking:

- **Segmentation Criteria:** Since the data is too large to process in a single pass, we broke it down into smaller, manageable chunks using LangChain's TextSplitter.

3.2.3 Concept and Relationship Extraction:

- **Data Loading and Transformation:** Since the data is too large to process in a single pass, we broke it down into smaller, manageable chunks using LangChain's TextSplitter.
- **Understanding the DataFrame:** This data structure forms the basis for further processing to identify concepts and relationships.

3.2.4 Graph Construction:

- **Segmentation Criteria:** Since the data is too large to process in a single pass, we broke it down into smaller, manageable chunks using LangChain's TextSplitter.
- **Data Transformation with pd.melt:** Converts the DataFrame from wide to long format, stacking node1 and node2 under a single node column while retaining chunkid.
- **Creating Pairs of Nodes within the Same Chunk:** Performs a self-join on chunkid to find all pairs of nodes that co-occur in the same chunk.
- **Removing Self-loops:** Self-loops (edges from a node to itself) are identified and removed to prevent redundant relationships.
- **Removing Self Loop:** Identifies and removes rows where node1 equals node2, indicating self-loops.
- **Aggregating Edge Information:** Groups by node1 and node2, concatenates chunk ids where the pair appears together, and counts the number of co-occurrences.
- **Filtering Edges:** Removes edges where nodes only co-occur once (count != 1) to focus on stronger contextual relationships.
- **Assigning Edge Type:** Adds a new column edge and assigns the value "contextual proximity" to all edges.

3.2.5 Community Detection and Visualization:

- **Edge Betweenness and community structure:** The Girvan Newman algorithm detects communities by progressively removing edges from the original network. The connected components of the remaining network are the communities. Instead of trying to construct a measure that tells us which edges are the most central to communities, the Girvan Newman algorithm focuses on edges that are most likely "between" communities Sources and related content

Conclusion

In conclusion, the processes of extracting data from standard corpora such as PMI RM , segmenting this data into identifiable components, and evaluating the effectiveness of knowledge extraction are crucial steps in transforming unstructured information into structured knowledge. The extraction phase enables us to gather valuable insights from well-established frameworks in project management, ensuring that essential concepts, terms, and relationships are identified. Segmenting the data through tokenization, part-of-speech tagging, and synonym identification allows for a deeper understanding of the textual content and its grammatical structure. This segmentation paves the way for the identification of key concepts and their attributes, enabling the analysis of their frequency to determine relevance within the corpus. Moreover, evaluating the extraction process through defined metrics such as precision, recall, and human expert review ensures that the knowledge captured is not only accurate but also applicable in real-world scenarios. The iterative refinement of extraction techniques based on evaluation results further enhances the process, allowing for continuous improvement and adaptation to the data's evolving nature. Ultimately, these comprehensive steps work together to create a robust framework for knowledge retrieval and analysis, facilitating informed decision-making and enhancing the overall understanding of project management principles. By systematically executing each step, we can harness the full potential of unstructured data, transforming it into actionable knowledge that drives project success.

CHAPTER 4

Building content-based recommender system based on DL architecture

Introduction

In today's digital age, recommender systems have become essential tools in various industries, helping users navigate through large volumes of information and content by providing personalized suggestions. Whether in e-commerce, streaming services, or social media platforms, these systems enhance user experience, drive engagement, and ultimately contribute to business success. Among the different types of recommender systems, content-based recommendation approaches stand out by leveraging the intrinsic properties and features of the items themselves, rather than relying solely on user interactions or collaborative behaviors. This project focuses on building a content-based recommender system using a deep learning (DL) architecture. Unlike traditional content-based methods that rely on simple keyword matching or basic feature extraction, the deep learning approach enables more sophisticated and nuanced representations of content, leading to improved recommendation quality. Specifically, this approach leverages graph neural networks (GNNs), a cutting-edge architecture well-suited for modeling relationships within complex datasets. GNNs allow us to represent items and their relationships as a graph, capturing both the features of individual

items and the connections between them, which is crucial for content-rich and interconnected data. The system is built in several stages, starting with data loading and preprocessing, followed by text chunking and concept extraction, and culminating in the construction of a graph representation of the data. Within this graph, a series of layers—including message passing, attention, stacking, and pooling—process and aggregate information to generate embeddings that can be used to recommend similar items. The model's design aims to uncover latent patterns and relationships within the content, ultimately enhancing the relevance of recommendations and delivering a robust user experience

4.1 Initializing Node Embeddings and Input Features

In the context of building a content-based recommender system with a Graph Neural Network (GNN), initializing node embeddings and input features is a crucial step. This stage sets up the foundational representations that the GNN layers will later process, enabling the model to capture and learn from the relationships and content properties of each item in the dataset.

4.1.1 Node Embedding Initialization

Node embeddings represent each node (or item) in the graph as a vector in a high-dimensional space. These embeddings capture initial information about the nodes, which serves as a starting point for learning more complex features through subsequent GNN layers. There are two primary ways to initialize node embeddings:

- **Pre-trained Embeddings:** pre-trained embeddings are available for the items (e.g., word embeddings for text-based content), these can be used as initial node representations. For instance, embeddings from models like Word2Vec or BERT can provide semantically rich representations, capturing meanings and relationships that have already been learned from large datasets.

4.2 Propagating Information Through Multiple Message-Passing and Attention Layers

In a Graph Neural Network (GNN), message passing and attention mechanisms are central to how information flows between nodes, enabling the network to capture relationships and dependencies across the graph. For a content-based recommender system, these layers allow the model to aggregate and learn from the features of related items, refining each node's representation based on its neighbors characteristics. Let's break down these two processes:

4.2.1 Message Passing Layer

The message-passing process is where each node gathers information from its neighboring nodes to update its own embedding. The primary goal of message passing is to create node embeddings that reflect both the node's own features and the features of its neighboring nodes. Here's how the process works:

- **Neighbor Aggregation:** For each node, the model aggregates features from its neighbors. This can involve simple aggregation operations like summation, averaging, or more complex transformations that depend on learned weights. Each neighbor's features are transformed by a weight matrix before being aggregated, allowing the model to learn how different features contribute to the node's updated representation.

- **Feature Transformation: :** The aggregated neighborhood information is combined with the node's own features, often using another weight matrix. This transformation allows the node embedding to encode both its individual properties and information from its neighbors. The combined features are then passed through a non-linear activation function, such as ReLU, to introduce complexity into the model.

- **Propagation Over Multiple Layers:** To capture information from nodes that are multiple "hops" away, multiple message-passing layers are used. Each layer allows nodes to gather information from neighbors further out in the graph, enabling them to build a more comprehensive understanding of the graph structure and their relative position within it.

4.2.2 Attention Mechanism in Message Passing

Incorporating attention mechanisms within message-passing layers allows the model to focus on the most relevant neighbors, assigning more importance to certain relationships over others. This is particularly useful when not all neighbors contribute equally to a node's representation — some nodes may be more relevant for recommendation purposes than others:

- **Computing Attention Weights:** For each pair of connected nodes, the model calculates an attention weight, which represents the importance of one node's information to the other. Attention weights can be computed by comparing the embeddings of the connected nodes.
- **Weighted Aggregation:** Instead of simply summing or averaging the features of neighboring nodes, the model applies attention weights to each neighbor's features before aggregation. This way, nodes that are more relevant or similar to the target node will have a greater influence on its updated representation.
- **Adaptive Focus on Neighbors:** With attention, the model adapts the influence of each neighbor based on context. For instance, if two nodes share a lot of similar features, the attention mechanism can amplify this relationship, emphasizing this neighbor's contribution during the aggregation process. This flexibility improves the model's ability to capture nuanced relationships.

4.3 Pooling or Aggregating Node Information to Create a Graph-Level Representation

Once node embeddings have been updated through message-passing and attention mechanisms, we need a way to combine or "pool" these individual node embeddings into a single, graph-level representation.

4.3.1 Pooling Techniques

- **Global Average Pooling:** This is the simplest approach, where we take the average of all node embeddings in the graph.
- **Attention Pooling** Using an attention mechanism, we can assign different weights to each node embedding during pooling. This approach allows the model to focus on the most important nodes (such as highly connected or contextually relevant nodes) when forming the graph representation, leading to a more targeted summary of the graph.

4.4 Processing the Pooled Representation Through Dense Layers

Dense layers allow the model to further process and transform this pooled information, enabling it to learn complex patterns and correlations that can be directly used for the recommendation task. In this phase, the model begins to move from representation to actionable insights, preparing to make predictions or recommendations based on the graph structure and content.

4.4.1 Structure of the Dense Layers

- **Multiple Dense (Fully Connected) Layers:** The pooled graph representation vector is passed through one or more dense layers. Each dense layer applies a linear transformation followed by a non-linear activation function, such as ReLU (Rectified Linear Unit), to introduce non-linearity and enable the model to learn complex patterns.
- **Output Layer:** The final dense layer outputs the values used for the recommendation task. For example: If the model needs to predict a score or relevance for each item, the output layer might have a single neuron with a linear activation. If the model is ranking items or classifying them into categories, a softmax or sigmoid activation might be used.

4.5 Training with a Suitable Loss Function and Optimization Method

The final stage in building the GNN-based content recommender system involves training the model to minimize error and maximize accuracy using a loss function and an optimization algorithm. This stage allows the model to learn from data iteratively, adjusting weights to improve predictions over time. Selecting the right loss function and optimization method is essential to ensuring that the model effectively learns relationships between content features and provides relevant recommendations.

4.5.1 Choosing the Loss Function

The loss function quantifies the error between the model's predictions and the actual values, guiding the model on how to adjust its weights to reduce this error

- **Mean Squared Error:** Used in regression-based recommendation tasks, MSE calculates the squared differences between the predicted scores and actual values (e.g., user ratings), penalizing larger errors more heavily.
- **Cross-Entropy Loss:** Commonly used for classification tasks, cross-entropy loss measures the error between the true labels and the model's predicted probabilities, making it suitable for binary or multi-class classification

4.5.2 Optimization Algorithm

The optimization algorithm updates the model's weights to minimize the loss function. In deep learning, the most common optimization method is Gradient Descent and its variations, which work by adjusting weights based on the loss gradient.

- **Stochastic Gradient Descent:** Used in regression-based recommendation tasks, MSE calculates the squared differences between the predicted scores and actual values (e.g., user ratings), penalizing larger errors more heavily.
- **Adam Optimizer:** Adam combines the advantages of both momentum and adaptive learning rate methods. It calculates adaptive learning rates for each parameter based

on the mean and variance of past gradients, making it popular for training GNNs due to its speed and efficiency.

CHAPTER 5

Project Challenges, Resolutions, and Deployment Approach

Introduction

In any project, the path to successful implementation is rarely without challenges. This chapter outlines the key obstacles encountered during the development of the recommender system for risk management, followed by the strategies and solutions employed to address them. Furthermore, it delves into the deployment process, detailing the steps taken to ensure the system's effective and seamless integration. By exploring these aspects, this section provides valuable insights into the complexities faced and the measures taken to overcome them, ensuring the project's operational success.

5.1 Challenges encountered

5.1.1 Model Accuracy Issues

One of the primary challenges faced during the project was the relatively low accuracy of the graph neural network (GNN) models. Despite implementing several advanced graph-based models, the accuracy remained suboptimal, with the following results observed: GNN: 0.23 accuracy GCN (Graph Convolutional Network): 0.21 accuracy GAT (Graph Attention Network): 0.1852 accuracy SAGE (GraphSAGE): 0.19 accuracy. These results were considerably lower than expected, especially considering the state-of-the-art nature of these models. A deeper investigation revealed that the models might not be properly tuned or may require more sophisticated feature engineering and preprocessing to enhance their performance. Additionally, challenges like overfitting and underfitting were observed in some of the models, requiring adjustments to hyperparameters, training strategies, and regularization techniques to improve accuracy.

5.1.2 Choosing the Most Appropriate Model

Selecting the best-performing model from a set of candidates proved to be a complex task. While the GNN family, including GCN, GAT, and GraphSAGE, each offered distinct advantages in capturing relationships within graph-structured data, none of them performed optimally in the specific context of the recommender system. The challenge lay in understanding the trade-offs between the models in terms of computational efficiency, scalability, and accuracy. Each model had its unique set of strengths: GCN is effective at capturing local neighborhood information but was limited in handling more complex relationships. GAT offered improved performance with its attention mechanism but still struggled with capturing broader patterns in the graph. GraphSAGE provided a scalable solution for large graphs but at the cost of lower accuracy on smaller datasets. After experimenting with various configurations, it became evident that fine-tuning these models required a better understanding of their behavior in relation to the specific data and domain of risk management.

5.1.3 Enhancing Model Outputs and Answers

Improving the quality of outputs or answers generated by the model posed another significant challenge. The answers generated by the model often lacked the level of depth

and reasoning expected for risk management tasks. The issue was that the model, despite capturing the structure of the graph, was not able to make more meaningful or contextually relevant predictions. To enhance the outputs, several approaches were considered:

Feature Engineering: Enhancing the node and edge features to better capture the nuances of the problem domain. Model Ensembling: Combining the predictions of multiple models to achieve a more robust and reliable output. Attention Mechanisms: Leveraging attention layers to better highlight the most influential nodes and relationships within the graph. Despite these efforts, finding the right combination of strategies to refine the model's output continued to present difficulties.

5.1.4 Integrating the Attention Layer and Reasoning Model

Another key challenge was the integration of an attention mechanism as an additional layer within the GNN architecture. While attention mechanisms have been proven to be effective in improving the performance of various neural networks, integrating them into a GNN proved to be challenging. Specifically, the difficulty stemmed from: Architectural Complexity: The complexity of combining the attention mechanism with the existing layers of the GNN, ensuring it did not disrupt the flow of message passing or information aggregation in the graph. Scalability Issues: The attention layer added computational overhead, making the model less scalable for large graphs, which is a crucial requirement for real-time recommender systems. Lack of Standardization: There is no one-size-fits-all solution for incorporating attention mechanisms into GNNs, which required extensive trial and error to find a configuration that worked well. These challenges highlighted the need for more advanced techniques or hybrid models that could more seamlessly integrate attention and reasoning layers with graph-based architectures.

5.1.5 Answering Specific Questions Using the Graph

A significant challenge in this project was designing a mechanism that could accurately answer specific questions using the information in the graph. The goal was to use similarity computation between the input question and the embeddings of nodes in the graph. The approach involved embedding both the question and the graph nodes into a vector space and then comparing the question's embedding with the embeddings of the graph nodes to find

the most relevant answers. However, several obstacles arose: Embedding Representation: Generating effective embeddings for both the input question and the nodes was non-trivial. The embeddings needed to capture the semantic meaning of the graph's structure while also representing the nuances of the user query. Similarity Computation: Calculating similarity between the question embedding and node embeddings effectively posed a challenge, as the embeddings did not always align well with the expected relevance of nodes to the question. Contextual Relevance: Ensuring that the correct nodes were selected based on the question context proved difficult. Without a deeper understanding of the graph's structure, the similarity computation was prone to returning irrelevant nodes or incomplete answers. Despite these challenges, refining the embedding generation process and exploring more advanced similarity measures were essential for improving the accuracy of the question-answering mechanism.

5.2 Solutions:

5.2.1 Building a Robust Model with Multiple Layers

To address the challenges with model accuracy and reasoning, we decided to build a more sophisticated model that integrates multiple layers to enhance its performance. This model combines several advanced components: Our Base Model: The foundational graph model was enhanced with additional layers, including attention mechanisms and multi-head attention techniques, to capture more intricate relationships within the graph data. Attention Layers: We introduced attention layers to better focus on the most relevant parts of the graph. To mitigate the issue of overfitting, we employed dropout techniques in these layers, which randomly dropped certain connections during training to prevent the model from memorizing the training data. This approach helped generalize the model better to unseen data. Linear Layer with Multi-Head Attention: A linear layer was incorporated to further refine the outputs, followed by eight multi-head attention mechanisms. This setup allowed the model to consider different aspects of the graph simultaneously, improving its ability to handle complex patterns and relationships within the data. By integrating these components, we

aimed to achieve a more accurate, efficient, and scalable model, capable of handling larger datasets and delivering more relevant results in the context of risk management.

5.2.2 Optimizing the Training Phase

The training phase of our model involved several critical steps to improve its efficiency and accuracy: Keyword-Based Training: We trained our model using keywords extracted from the data, which acted as important features that guided the model in identifying relevant patterns and relationships within the graph. This keyword-based approach allowed the model to focus on the most significant data points, improving both its accuracy and relevance. GPU Acceleration: To expedite the training process and handle the large volumes of data involved in graph-based learning, we utilized GPU acceleration. Training on GPUs significantly reduced the time needed for experimentation and fine-tuning, enabling us to explore various configurations and optimizations more efficiently. Fine-Tuning Attention Layers with GATConv: The Graph Attention Network (GAT) convolutional layers were a key component in enhancing the reasoning capabilities of our graph model. By fine-tuning these attention layers, we aimed to better capture the relationships between nodes and improve the model's ability to make context-aware predictions. The GATConv layers allowed the model to focus on the most relevant parts of the graph, while fine-tuning these layers helped optimize the reasoning process, ultimately leading to higher accuracy in graph-based predictions. Through these strategies, we were able to enhance the model's performance, ensuring it could learn more effectively from the graph data while mitigating issues like overfitting and underfitting.

5.3 Solutions:

The deployment phase of our project involved the integration of our graph-based recommendation system into a functional platform—specifically, a chatbot designed to assist with risk management tasks. The chatbot operates as an intelligent assistant that offers recommendations based on the graph model's insights, enabling users to receive context-sensitive advice for managing and mitigating risks in their projects.

5.3.1 Platform Overview

The core of the platform is a chatbot built using Flask, a lightweight web framework. The chatbot serves as the user interface, providing a conversational environment where users can ask questions related to risk management. Based on the graph model's data and the user's query, the chatbot generates risk management recommendations, helping users make informed decisions.

The chatbot is designed to interact with the ontology-based knowledge graph, which forms the backbone of the recommender system. This ontology, stored in an .owl (Web Ontology Language) file, is loaded at the start of each session, ensuring that the chatbot has access to up-to-date and structured information related to risk management. The chatbot then uses this knowledge base to generate answers and recommendations tailored to the specific needs of the user.

5.3.2 Optimizing the Training Phase

The training phase of our model involved several critical steps to improve its efficiency and accuracy: Keyword-Based Training: We trained our model using keywords extracted from the data, which acted as important features that guided the model in identifying relevant patterns and relationships within the graph. This keyword-based approach allowed the model to focus on the most significant data points, improving both its accuracy and relevance. GPU Acceleration: To expedite the training process and handle the large volumes of data involved in graph-based learning, we utilized GPU acceleration. Training on GPUs significantly reduced the time needed for experimentation and fine-tuning, enabling us to explore various configurations and optimizations more efficiently. Fine-Tuning Attention Layers with GATConv: The Graph Attention Network (GAT) convolutional layers were a key component in enhancing the reasoning capabilities of our graph model. By fine-tuning these attention layers, we aimed to better capture the relationships between nodes and improve the model's ability to make context-aware predictions. The GATConv layers allowed the model to focus on the most relevant parts of the graph, while fine-tuning these layers helped optimize the reasoning process, ultimately leading to higher accuracy in graph-based predictions. Through these strategies, we were able to enhance the model's performance, ensuring it could learn more effectively from the graph data while mitigating issues like

overfitting and underfitting.

5.4 Deployment

The deployment phase of our project involved the integration of our graph-based recommendation system into a functional platform—specifically, a chatbot designed to assist with risk management tasks. The chatbot operates as an intelligent assistant that offers recommendations based on the graph model's insights, enabling users to receive context-sensitive advice for managing and mitigating risks in their projects.

5.4.1 Platform Overview

The core of the platform is a chatbot built using Flask, a lightweight web framework. The chatbot serves as the user interface, providing a conversational environment where users can ask questions related to risk management. Based on the graph model's data and the user's query, the chatbot generates risk management recommendations, helping users make informed decisions. The chatbot is designed to interact with the ontology-based knowledge graph, which forms the backbone of the recommender system. This ontology, stored in an .owl (Web Ontology Language) file, is loaded at the start of each session, ensuring that the chatbot has access to up-to-date and structured information related to risk management. The chatbot then uses this knowledge base to generate answers and recommendations tailored to the specific needs of the user.

5.4.2 Workflow of the Chatbot System

The deployment process involved several key steps to ensure smooth interaction and accurate response generation:

- **Ontology Loading:** At the start of each session, the chatbot loads the ontology file . This file contains detailed information about various risk management concepts, structured in a graph format.

- **User Input Processing:** When a user submits a query, the chatbot first processes the input to understand the specific question and identify the relevant keywords. These

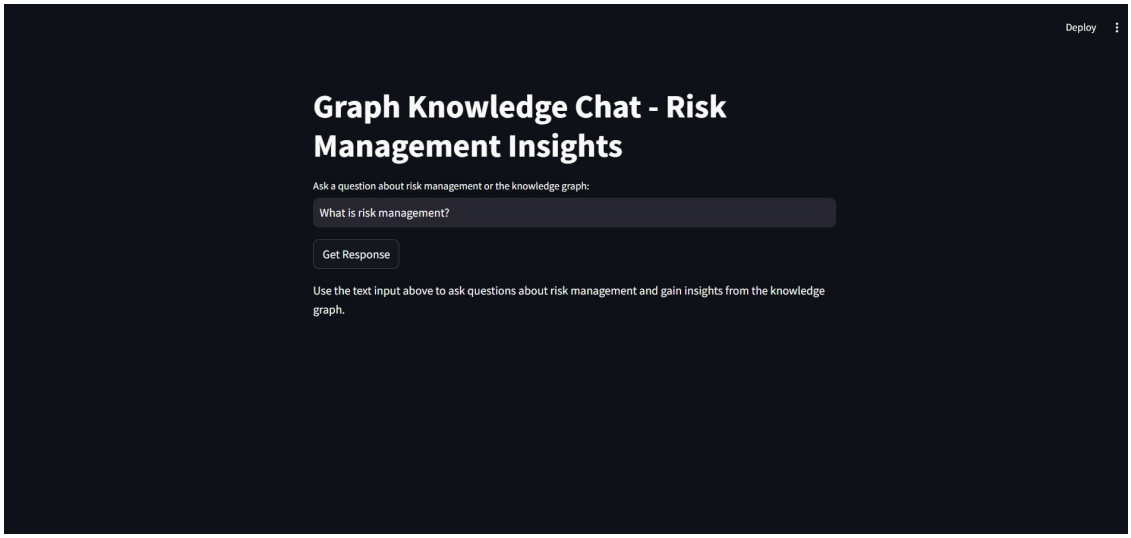

Figure 3: Data Warriors Team Chatbot For Risk Management

keywords are used to navigate the graph and find the most relevant nodes (i.e., concepts or entities related to risk management).

- **Recommendation Generation:** The chatbot utilizes the underlying graph model to generate recommendations. This process involves comparing the embeddings of the input query with the embeddings of the graph nodes, using similarity computations to identify the most relevant nodes and relationships. Based on these results, the chatbot presents a recommendation or an answer that aligns with the user's needs.

5.4.3 Integration Challenges

During the deployment process, several integration challenges were encountered:

- **Graph Integration with Chatbot:** One key challenge was ensuring that the graph-based model could efficiently interface with the chatbot's conversational interface. The system needed to provide quick and accurate responses in real-time, while also ensuring that the answers were contextually relevant to the user's question.

- **Ensuring Real-Time Performance:** As the graph model involved complex computations, ensuring real-time performance of the chatbot was a critical concern. The recommendation generation process had to be optimized to provide fast responses without compromising the quality of the answers.

- **Handling Ambiguities:** Risk management queries can be inherently ambiguous, and ensuring that the chatbot could effectively handle such ambiguities and provide accurate answers was a significant challenge. Techniques like semantic similarity and contextual understanding were employed to improve the chatbot's ability to interpret diverse queries.

5.4.4 Future Deployment Plans

To further scale the platform, future plans include:

- **Cloud Deployment:** Transitioning the platform to a cloud-based environment to ensure better scalability, reliability, and accessibility. This would allow users to access the chatbot from anywhere, and facilitate easier integration with other systems and data sources.

- **Continuous Improvement:** As new data becomes available, the system will be updated regularly to ensure that the recommendations remain relevant and accurate. Additionally, further enhancements to the graph model and chatbot interface will be explored to improve user experience and performance.

General Conclusion

In conclusion, the AI Cognition project addresses a critical need in project risk management by leveraging cutting-edge AI technologies such as Natural Language Processing (NLP), deep learning models, and ontology-based frameworks. Through the proposed solution, we aim to automate the detection, prediction, and mitigation of project risks, providing users with a scalable and deployable web-based application.

The project's key objectives are to improve the efficiency of risk detection, provide accurate risk predictions, and offer actionable recommendations for risk mitigation, ensuring that project managers are empowered with data-driven insights. The seamless integration of business objectives with data science solutions forms the foundation for this innovative system, which not only enhances decision-making but also reduces the likelihood of project failure.

By aligning these objectives with real-time monitoring, the system ensures that risks are managed proactively, resulting in more successful project outcomes. This solution promises significant value across industries, offering scalable and adaptable capabilities that make it a highly relevant and transformative tool in modern project management.

Bibliography

[**1**] TenStep , https://www.tenstep.tn/coaching-et-mentoring/ /

[**3**]TDSP, www.datascience-pm.com/tdsp/

[**4**] Presentation of AI and Cognition Project 2 PMBOK ,-Esprit

[**5**] python, www.w3schools.com/python/python_regex.asp

[**6**] VSCode, code.visualstudio.com/. [**7**] Python, www.python.org/