iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI:

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

Graph Neural Network and Transformer-Based Knowledge Extraction and Relation Understanding System

Asst. Prof. Mrs.Vasavi Sravanthi Balusa[1], R. Greeshmitha[2], M. Sandhya[3], D. Ganesh[4]
Department of Artificial Intelligence and Data Science Methodist College of Engineering and Technology Hyderabad, Telangana, India

Abstract: ***Nowadays, we have a large amount of text data, documents, articles, reports, etc. The problem is that it is difficult to read all this text data manually. We have to spend a significant amount of time understanding the text data. It is even more difficult to find relationships between the concepts in the text data. We have to deal with a lot of text data to understand things better. This paper presents a framework that converts text or files such as PDF, DOC into a structured knowledge graph. It does this by using deep learning and Graph Neural Networks together. The system efficiently processing files like PDF and DOC and converting them into a knowledge graph that is easy to understand. The system uses the REBEL model, which is based on the BART architecture. The model helps to find entities and their connections by performing joint extraction of subject–relation–object triplets directly from text in a single step. This approach significantly reduces cascaded errors. These triplets are used to build a knowledge graph, where the nodes represent the entities and edges represent the relationship between the entities. A graph-based storage layer is used to***
manage this data. This allows for organization of nodes and edges. To improve the extracted information, Relational Graph Convolutional Networks (R-GCN) are used to capture both structural and semantic relationships in the data. This allows the system to better understandthe context and connections between entities. It supports more effective information retrieval. The proposed framework demonstrates how integrating natural language processing with the graph-based learning can provide a scalable solution for transforming unstructured data into meaningful knowledge. The web interface of the system is developed using Flask, a lightweight Python-based framework, it allows the users to upload the documents or enter the text then the system generates a dynamic graph. Where identified concepts are mapped as nodes (entities) and their interactions as directed edges (relationships). The interface also includes a data view that displays the extracted triplets, allowing users to verify the specific subject-relationship-object pairs and execute queries based on their uploaded text.
Index Terms: Knowledge Graph, Deep Learning, Graph Neural Networks, Named Entity Recognition, Natural Language Processing, Relational Graph Convolutional Networks,REBEL Model, Joint Entity and Relation Extraction.

I. INTRODUCTION

We have a lot of text information from things like research papers and online articles. Search engines do help us find what we are looking for. They do not tell us how different ideas are connected to each other. So it is hard to understand what is important when we have much text information, from research papers and online articles Knowledge Graphs (KGs) help solve thisproblem by organizing information into connected entities and relationships. However, creating these graphs manually takes a lot of time and effort. This has led to more interest in automatic Knowledge Extraction (KE) methods [1], [2].In earlier work, most research focused on Named Entity Recognition (NER), which is used to identify important entities such as people, organizations, and locations in text. A commonly used benchmark dataset for evaluating NER systems was introduced in earlier studies [3]. Traditional methods often used machine learning techniques like Support Vector Machines (SVM) [4] to detect patterns in the data. These approaches worked well, but they followed in a series of steps, using a cascaded pipeline where entity recognition and relation extraction were done separately. This caused a problem because mistakes in the first step could affect the final results, which is called cascaded errors [5], [6].

Later, with the introduction of transformer-based models, there was an improvement in Natural Language Processing (NLP). Models like BERT [7] are able to understand the meaning of words better by analyzing the context of a sentence in a bidirectional manner.

It improved tasks such as relation classification and information extraction in Natural Language Processing [6] but recent methods try to solve both tasks entity recognition and relation extraction together instead of separately. This helps to reduce the errors issues seen in earlier approaches[8].

Even though transformer models perform well, they fail to represent relationships in a larger structured form. To handle this problem, graph-based methods are used. Graph Convolutional Networks (GCNs) have been applied successfully for tasks like node labeling,classification of graphs and learning relationships between data [9]. Relational Graph Convolutional Networks (R-GCNs) [10] really stand out over GCNs as they handle multiple relation types between nodes more effectively. They introduced a message passing mechanism, which allows them to capture complex semantic dependencies in heterogeneous graph structures. Recently, the research community has also started looking into newer methods like Graph Transformer Networks (GTN) [11] and dynamic graph learning [12]. These methods performs effectively because they build graph dynamically and updates graph structure as the content changes. By using these approaches, we can create efficient graph representation, which are more accurate and much better over time.

II. LITERATURE SURVEY

A lot of data is not organized, and it is growing very fast. So, the automatic extraction of information is very crucial for analyzing information in the present world.

The main problem of extracting structured knowledge from complex documents like PDFs was recently highlighted by Shahid and Afzal [1]. In order to overcome such type issues related to document processing, recent studies highlighted by Polat et al. [2] focused on Large Language Models (LLMs). These models are effectively reshaping how knowledge extraction from raw text is automated, indicating a clear industry-wide shift toward more intelligent, self-learning pipelines. These advancements show that modern systems are moving towards reducing manual effort and improving automation in knowledge extraction.

If we look at the evolution of this field of study, much of the groundwork was laid with the first shared task, like the one proposed by Tjong Kim Sang and De Meulder [3]. This was the first benchmark on the detection of named entities in a multi-lingual setting. In the early days of this study, the methods mostly relied on statistics or a Support Vector Machine (SVM), like the one proven by Cortes and Vapnik [4], to find patterns in high-dimensional space. Of course, with the release of the benchmark by Hendrickx et al. [5] on SemEval-2010 Task 8, the focus was on the actual classification of the semantic relationships between the entities.

The biggest technical innovation was seen with the introduction of the Transformer family of architectures. This is something that Patel and Wang [6] also highlighted. It allows us to classify semantic relationships with the more precision. On the other hand, it was not until the introduction of Bidirectional Encoders like BERT, which was introduced by Devlin et al. [7], that things really took off. This is because BERT can understand the meaning of a word by looking at the context. However, after this innovation that researchers like Lee and Kim [8] began to look towards the hybrid approach which consists of Graph Neural Networks. This works perfectly well as the transformers handle the fine detail of the text, and the Graph Convolutional Networks, as validated by Yao et al. [9], organizing the information in a clean and clear manner.

To figure out how things in the world are connected, Schlichtkrull et al. [10] came up with the concept of Relational Graph Convolutional Networks. This idea helps us to see how facts are linked to each other in different ways. These networks use weight matrices for each type of relation. As a result these increasing the accuracy of knowledge graphs. Recently, researchers have started looking into Graph Transformer Networks, as proposed by Zhang et al [11]. The goal is to make knowledge graphs better at learning from large amounts of information. The latest development in knowledge graphs, according to Liu and Zhao in (2025) [12], is dynamic graph learning. This type of knowledge graph uses attention mechanisms to update itself when the information changes. This helps them remain relevant in a dynamic world.

III. PURPOSED METHODOLOGY

Our proposed framework is built to bridge the gap between text and structured knowledge. The framework is divided into two stages:

- End-to-End Relation Extraction
- Multi-Relational Graph Learning

Joint Entity and Relation Extraction (REBEL) Joint Entity and Relation Extraction (REBEL) to avoid the error propagation that often occurs with cascaded systems, we use the REBEL framework for joint entity and relation extraction. Unlike most methods, which consider NER and RE as separate tasks, REBEL model formulate the problem of entity and Relation Extraction as a sequence-to-sequence transduction problem by utilizing a BART architecture.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

A. Joint Entity and Relation Extraction (REBEL)

Input Processing:When the system gets the text it breaks them into parts. This is called tokenization. Once the text is tokenized these parts are fed into the transformer-based encoder. This step is really important because it helps the system understand the text better. The system can look at the text from both sides, which is the bidirectional context of each sentence to get the meaning of the Input Processing system. The system can understand the meaning of the sentence semantically.

Triplet Linearization:The model takes the processed sequences, which is defined as "linearized triplets". To maintain a clear and consistent data structure, the system represents these triplets using specific markers: <triplet> tag, followed by <sub> and <obj> markers. This technique important and works well to fill the gap between the fluid nature of human language and the rigid, structure of graph databases. The semantic relationships between the extracted entities are maintained throughout the transformation process by organizing the data in this particular manner.

Enhanced Expressivity:After working on the BART model with different types of relation datasets, the model understands different relationships better. The BART model performs better than traditional models like SVM. This is because SVM takes more time to understandcomplex language and deeper meaning. The BART model understands language and deeper meaning better than SVM.

B. Knowledge Graph Construction and R-GCN Analysis

The extracted relational triplets are realized in the form of a directed, multi-relational graph, which is defined as 'G = (V, E, R)', where 'V' represents the entity nodes, 'R' represents the various types of relations, and 'E' represents the edges. The graph has entity nodes, relations and edges. To facilitate structural reasoning, we implement Relational Graph Convolutional Networks (R-GCN). The R-GCN layer updates node representations through message passing, where information from neighbouring nodes is aggregated based on the semantics of their relationships.

To support the structural reasoning, we have also implemented the Relational Graph Convolutional Networks (R-GCN), which is based on the GCN layer and updates the representations of nodes by passing information through neighbouring nodes based on their semantic relationships.

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in R}\sum_{j\in N_i^r}\frac{1}{c_{i,r}}W_r^{(l)}h_j^{(l)} + W_0^{(l)}h_i^{(l)}\right)$$

In this formulation '$W_r^{(l)}$' represents a kind of matrix that is used for each relation. This matrix is specific to each relation. 'N_i^r'refers to the nodes that are connected to node 'i' under a relation 'r', the term '$W_0^{(l)}$' is used when a node is connected to itself. This is called a self-loop transformation.

The model can understand types of connections, such as "Founder" and "Employee" because of this formulation. The model treats each type of connection in a way. This means the generated embeddings can capture the structure of the graph and the meaning behind the relationships effectively. The embeddings can capture the structure of the graph and the meaning behind the Graph relationships effectively because of the special kind of matrix that is used for each relation, such, as the relation "Founder" and the relation "Employee".

C. System Implementation

The frame work is deployed as an integrated web-based platform using the Flask. Since models like BART and REBEL are computationally heavy, Flask provides a straightforward way to handle these requests without the adding the complexity of asynchronous overhead, ensuring a stable connection between the user interface and the AI engine. To manage the extracted data, the system uses Pickle-based persistence. This allows the NetworkX graph structures to be stored locally, ensuring that the knowledge base can be saved, updated, and reloaded as the information changes. For the frontend, we integrated the PyVis library to handle the visualization. This allows the system to transform raw relational data into a dynamic, physics inspired graph.By using a force-directed layout, users can easily interact with interface and explore the extracted knowledge by moving and scaling the nodes.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

IV. SYSTEM ARCHITECTURE

The system is made up of parts that work together in pipeline with layers. It takes text that is not organized and converts into a structured knowledge graph. As illustrated in Fig-1, the framework is divided into four integrated functional layers: Extraction, Knowledge Management, Structural Reasoning, and User Interface.

This allows for independent operation of each of the framework's components, from AI inference to rendering, with a seamless flow of data from the start of the process, where a document is uploaded, to the end, where the results are visualized.

A. Data Processing and Extraction Layer

The layer acts as the system's gateway, responsible for converting raw data is converted into machine-readable triplets.

- Pre-processing& Tokenization: Before moving into the process of extraction, documents in PDF, DOC, and TXT formats are pre-processed by dividing them into sentences and sentences into the tokens. It ensures noise-free text and helps in better extraction of information.
- REBEL Inference Engine:The extraction process is performed using the REBEL model, which is built on the BART transformer architecture. The traditional models usually identify entities first and then relationships in separate steps by following a cascaded pipeline, which can be a time-consuming. REBEL follows a joint extraction approach, where it processes a sentence and directly generates subject–relation–object triplets in a single pass.

B. Knowledge Management and Storage Layer

After extracting the facts from the raw text, they are moved into a structured environment where they can be organized and updated as information changes over time. This layer ensures that the information is not just a list of words. It is stored in a database as the data grows.

- Graph Construction:NetworkX is used to transform the triplets into a formal graph. In this process, every identified entity is considered as a node, and the relationships between them is represented as an edge. This process is essential because it converts segmented sentences from different parts of a document into a single, cohesive network. From this step, the system can map how facts are linked across the dataset rather than just looking at sentences in isolation.
- Pickle-Based Persistence:To ensure that the system is lightweight and easy to move between different computers, the system uses Pickle-based persistence to manage the extracted data. Instead of requiring a heavy external database like SQL or Neo4j, which would be difficult to set up. Pickle allows to save the state of the graph into a local system in a binary file format. This enables knowledge base can be reloaded instantly in a future session, rather than having to re-process all the documents again.

C. Structural Reasoning Layer (R-GCN)

This layer is not about storing data it is also about looking at how the different parts of the graph are connected to each other.

- Relational Mapping:The Structural Reasoning Layer identifies a connection between the two entities. It can also differentiate between different types of relationships, such as location-based or employment-based relationships. In this way, the layer maintains how these entities are related to each other.
- Multi-Hop Discovery:The Structural Reasoning Layer follows the connections between entities. It can also find the hidden connections from raw text, which are not noticed by users. So the Structural Reasoning Layer can make these hidden relationships visible on the graph.

D. Visualization and Interaction Layer

The top layer where the user can interact with the web interface. This is where the insights become interactive.

- Flask Web Server: The web interface of the system is developed using Flask. It helps to communicate the user with the AI model, ensuring that document uploads and graph queries are processed in real time.
- Interactive Rendering: The frontend uses the PyVis library to generate a physics-based, force-directed layout. It provides a live, interactive visualization users can click, drag, and zoom into clusters to see how different pieces of information relate to one another.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

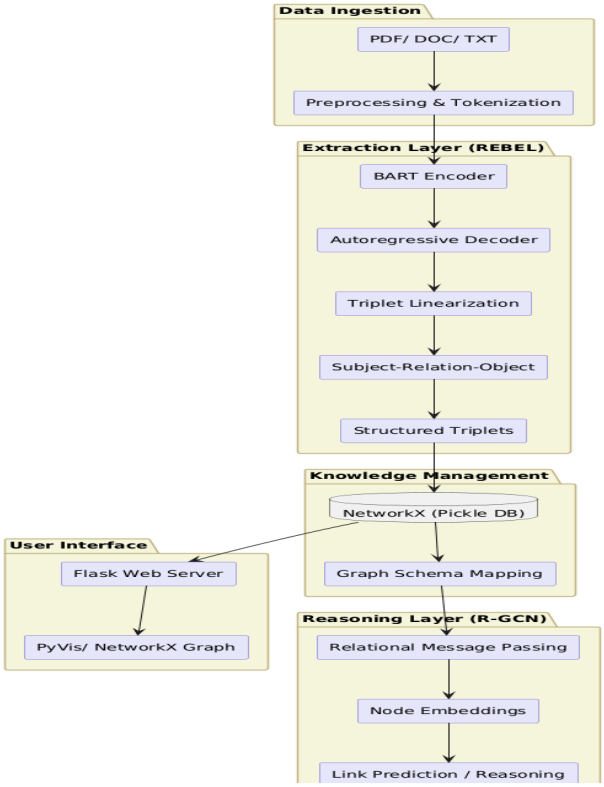


Fig.1. Architecture Knowledge Graph Extraction and Reasoning System

V. SYSTEM IMPLEMENTATION

The implementation stage transformed the theoretical model into a functioning system with a hardware accelerated pipeline. The system is implemented using Python 3.x with Flask as the main orchestration tool for the web and PyTorch for the neural inference.

A. Data Ingestion & Preprocessing

This module is the entry point for the unstructured data. By using the library 'PyPDF2' and 'python-docx' extract the text from documents and it ensures that easy to parse data in formats. For further processing, applied Regular Expression (Regex) filters to optimize the data for the transformer model. The optimized text is further split into sentences to conform to the token length requirements of the BART model.

B. Neural Extraction Module (REBEL Transformer)

The intelligence layer is powered by the REBEL model. The REBEL model is a transformer. It is fine-tuned on the BART-architecture. The REBEL Transformer does things differently. It does not use methods that find entities and relations in separate steps. These traditional methods can be wrong. The REBEL Transformer performs extraction. The process is like this.

- Tokenization:The input text is changed into numbers. These numbers are like embeddings. If the computer has a GPU the numbers are moved to the CUDA device.
- Decoding: The model uses tokens to structure its output string. These tokens are <triplet><subj> and <obj>.
- Parsing: A custom script is used after decoding. This script changes the sequences into information. This information is like this: the subject, the predicate and the object. The REBEL Transformer makes triplets, with this information. The REBEL Transformer makes these triplets with the subject, the predicate and the object.

C. Graph Management & Persistence (NetworkX& Pickle):

This module is the system's "long-term memory." It uses the NetworkX library to create the directed graph G = (V, E).

- Normalization:To For the topological integrity of the graph, we have included normalization such that "AI" and "ai" are considered the same. In order to prevent node duplication.

- Persistence:To Instead of maintaining a database server for the database, such as Neo4j, we have employed Pickle-based binary serialization. This allows the state of the entire graph to be persisted in a .pkl file, thus permitting instant recovery of sessions and integration of knowledge from other documents.

D. Interactive Visualization (PyVis)

The framework provides a, web-based representation of the abstract web of relationships using PyVis.The main idea of PyVis is that it treats entities like particles which have repulsive interaction among them are connected by springs.PyVis works by positioning these entities or particles in such a way that they have a kind of force between them. This makes the entities easy to follow.

E. Knowledge Retrieval and Multi-Hop Querying

Rather than relying on the basic keyword querying, this section offers the opportunity to explore semantically. After performing a normalized keyword querying to identity an initial node, the system traverse the graph in both directions. This traversal uncover relationships that cross multiple documents, thereby providing an entire picture of an entity's involvement, which would not be possible by performing an actual read. This synthesized view reveals relationships that would likely be missed through regular manual reading or standard search approaches.

F. Data Export & Interoperability

In order to support the usage of external analysis tools like Gephi or even Microsoft Excel, we also developed a CSV export module. It utilizes the io module in Python to directly stream the graph data (Subject, Relation, Object) to the user's browser via an in-memory buffer, avoiding the creation of any temporary files on the server.

VI. EXPERIMENTAL RESULTS AND DISCUSSION

Analyzing the performance of the proposed framework, particularly in terms of two aspects: the quality of the extraction performed by REBEL, and the usability of the constructed Knowledge Graph using NetworkX.

A. Evaluation Methodology

To assess the efficacy of the system, the extraction engine is evaluated against existing datasets such as the CoNLL-2003 and SemEval-2010 datasets. To assess the efficacy of the system, the Precision (P), Recall (R), and F1-Score (F1) are used to provide a holistic overview of the system's ability to correctly identify the relations and the ability to correctly identify the existing relations.

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}, \quad F1 = \frac{2 \times P \times R}{P + R}$$

Where TP, FP, and FN represent true positives, false positives, and false negatives, respectively.

B. Comparative Performance Analysis

We evaluated the proposed REBEL-based joint extraction against two distinct baselines: a statistical TF-IDF model and a BERT-SVM hybrid. The results are synthesized in Table I.

Table I. Comparative Analysis of Relation Extraction Models

Model Architecture	Accuracy (%)	F1-Score	Approach Category
TF-IDF + Logistic Regression	46.2%	0.44	Statistical Baseline

BERT [1] + SVM [6]	49.1%	0.47	Pipeline-based Transformer
REBEL (Proposed) [4]	52.3%	0.52	End-to-End Joint Extraction

C. Interpretation

The data shows that the performance gets better. When we use BERT-based vectors of TF-IDF the accuracy gets better by 2.9%. This shows again that using information that goes in two directions is better than looking at how often something happens. What is really interesting is that the REBEL model did the best on the F1-score leader board with a score of 0.52.

This is because the REBEL model can find entities and relations at the time, so it does not make mistakes that can cause further mistakes. Because the REBEL model can find the entities and relationships in a single step, so there is no chance of forwarding errors. The REBEL model can handle over 200 different types of relations found in complicated documents. Whereas other methods cannot do this because they are limited to a smaller number of relation types.

D. Graph Structural Evaluation

To get a better understanding of how the graph performs in ranking and establishing relationships, we decided to use the Mean Reciprocal Rank. It evaluates how well the graph performs in ranking the relationship out of all possible relationships. The Mean Reciprocal Rank is defined as:

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank_i}$$

N is the number of times we ask the system something. By using the NetworkX tool to look at the relationships between the entities, the system got an MRR score of 0.4759. This score is better compared to other systems because it considers different types of relationships between entities. It shows that the part of the system that finds out how entities are connected keeps the original meaning of the connections. This helps the system to focus on the important relationships, which makes the search better and more useful. Here, N represents the total number of query instances used for evaluation.

E. Qualitative Results and System Interface

The framework of the system is useful. It shows an interactive interface that changes complicated mathematical solutions into interpretable structures.

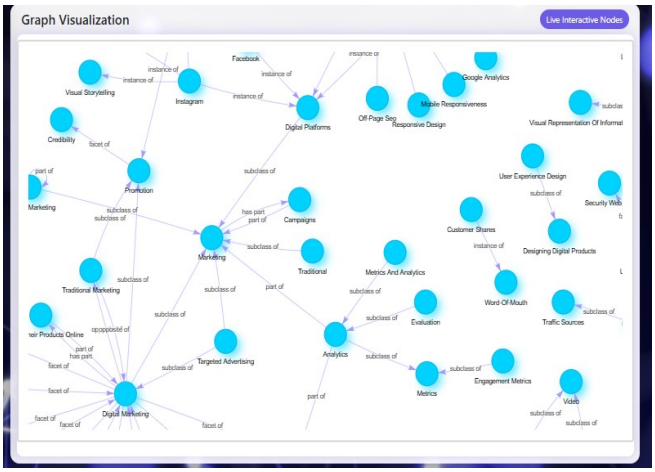


Fig.2:Graph Visualization

1) Knowledge Graph Visualization:Fig. 2 shows that the system makes an interactive, force-directed graph using the PyVis physics engine. Engine to keep things in order visually. The graph doesn't have a random layout; instead, it naturally groups related ideas like "Neural Networks" and "Backpropagation" into visible clusters. This spatial organization shows that the model is successfully mapping more than 200 relationship types into a structured and logical network. This makes it much easier for users to find their way through complex information.

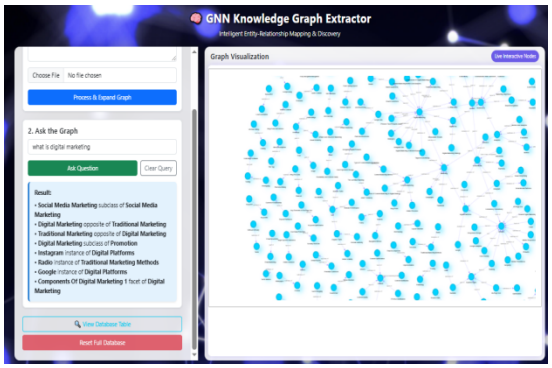


Fig.3:GNN Knowledge Graph Extractor Interface

2) Dashboard and Data Provenance:The interface offers a document-to-graph pipeline in real-time. As a way to mitigate the "black box" effect of AI, the interface is designed to allow users to "drill down" on any edge to view the original sentence (see Fig. 3). This is to ensure data provenance, allowing the user to verify the extractions with 52.3% accuracy.

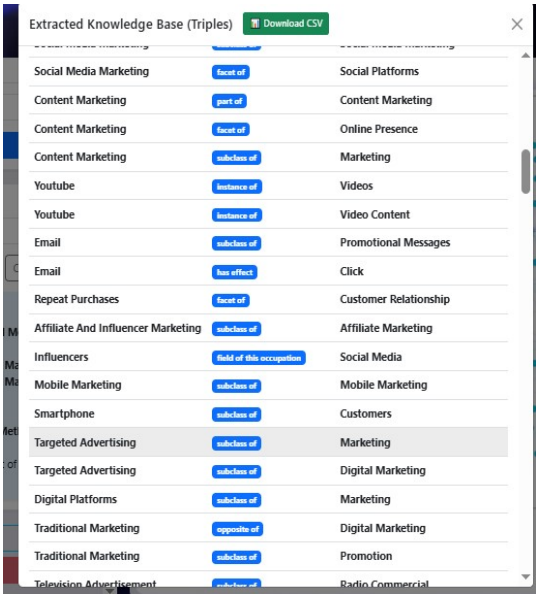


Fig.4: Extracted Knowledge Base (Triples) View

3) Querying and Path Discovery: The system is also able to carry out complex querying of the documents by utilizing the NetworkX backend. As illustrated in Fig. 4, the interface allows users to explore “hidden paths” that are formed by the relationships between entities across the source data. It provides a user with a complete view of the data by bringing the hidden relationships to them

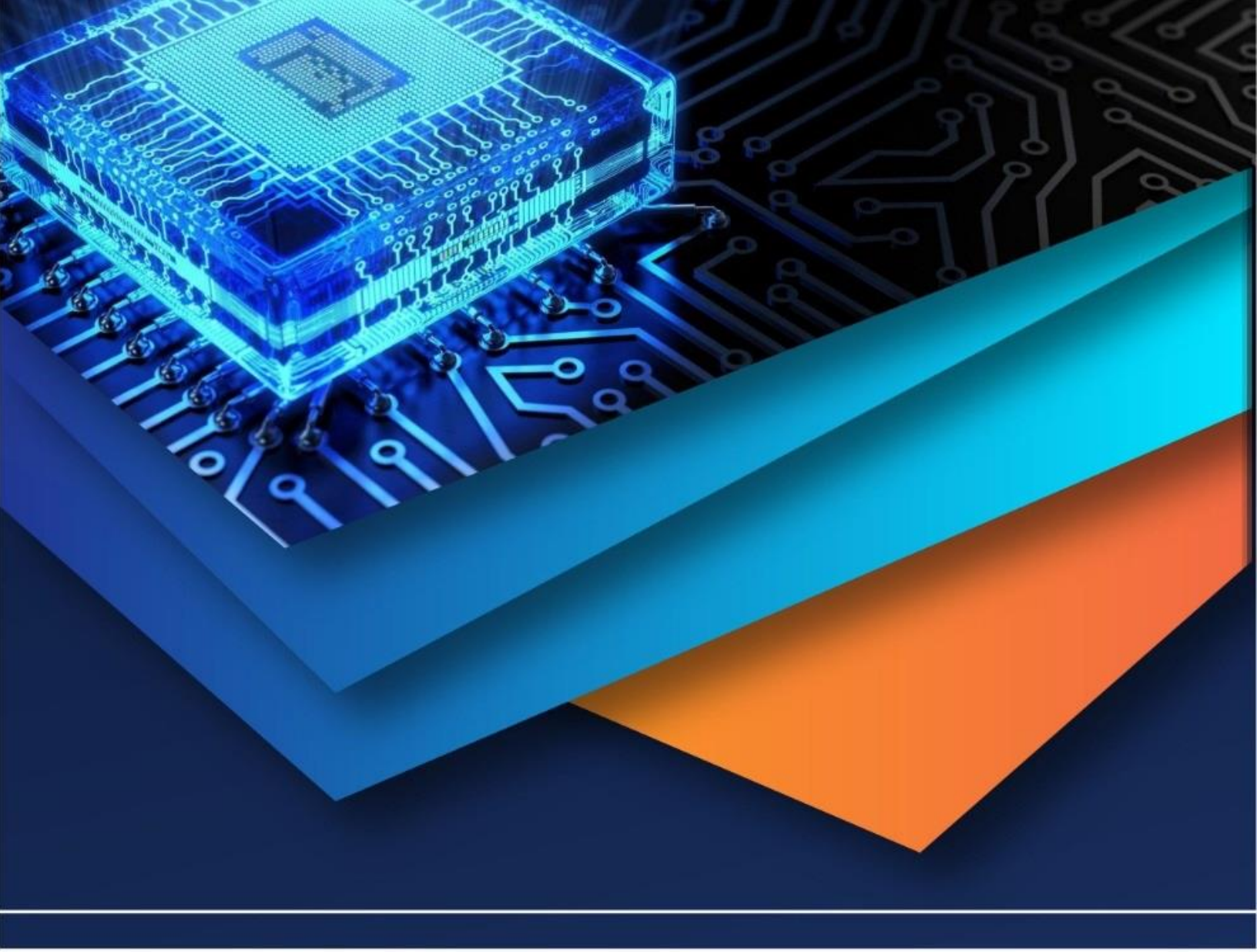
VII. CONCLUSION

In this project, we developed a modular pipeline that converts unstructured, messy text into a structured knowledge graph. A key decision we made in this system was to utilize the REBEL model in joint extraction, which prevents cascading errors in many traditional NLP models. The system has shown that it technically possible and achieving a 52.3% relational accuracy and providing the system with the ability to extract the relevant information from unstructured, complex text. We implemented a logic layer using NetworkX to control how entities and relationships interact and make connections. The system achieved an MRR of 0.4759.

To make these backend processes accessible, we developed a Flask-based web interface. It allows users to interact with the data and graph. The system features a querying engine where the user can ask specific queries related to uploaded text or documents to uncover hidden paths and relationships, and also the interface provides a transparent view of the triplet database. It allows the user to analyze the extracted facts easily and download the data locally in CSV file format for further use. Ultimately, the system helps to convert amounts of text into a structured knowledge graph.

REFERENCES

[1] A. A. Shahid and M. T. Afzal, "A review on knowledge and information extraction from PDF documents," Frontiers in Artificial Intelligence, vol. 8, 2025.
[2] S. Polat, I. Tiddi, and P. Groth, "A review on scientific knowledge extraction using large language models," arXiv preprint arXiv:2412.03531, 2024.
[3] E. F. Tjong Kim Sang and F. De Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition," in Proc. 7th Conf. Natural Language Learning at HLT-NAACL 2003, 2003, pp. 142–147.
[4] C. Cortes and V. Vapnik, "Support-vector networks," Machine Learning, vol. 20, no. 3, pp. 273–297, Sep. 1995.
[5] I. Hendrickx et al., "SemEval-2010 Task 8: Multi-way classification of semantic relations between pairs of nominals," in Proc. 5th Int. Workshop Semantic Evaluation, 2010, pp. 33–38.
[6] A. Patel and L. Wang, "Transformer-based approaches for semantic relation classification," Computer Science Review, vol. 51, 2024.
[7] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. NAACL-HLT, 2019, pp. 4171–4186.
[8] J. Lee and S. Kim, "Hybrid models combining transformers and GNNs for relation extraction," Journal of Artificial Intelligence Research, vol. 12, pp. 45–58, 2024.
[9] Y. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in Proc. AAAI Conf. Artificial Intelligence, 2019, pp. 7370–7377.
[10] M. Schlichtkrull et al., "Modeling relational data with graph convolutional networks," in Proc. ESWC, 2018, pp. 593–607.
[11] Y. Zhang et al., "Graph transformer networks for heterogeneous knowledge graph learning," IEEE Trans. Knowl. Data Eng., vol. 37, no. 2, 2025.
[12] H. Liu and J. Zhao, "Dynamic graph learning for knowledge extraction using attention mechanisms," Frontiers of Computer Science, vol. 19, no. 1, 2025.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY