Final Year BTech Project-II Report

On

SCHOLARGRAPH

For the Degree of

Bachelor of Technology

In

Computer Science and Engineering

Submitted by

21510007 – Neha Kundan Sonkamble

21510037 – Sourabh Yashwant Chaugule

21510102 – Maria Sarfraz Shaikh

21510119 – Trilok Vikram Kulkarni

Under the Guidance of

Mr. Shailesh Patil

Department of Computer Science and Engineering,

Walchand College of Engineering, Sangli.

(*An Autonomous Institute*)

AY 2024-25


WALCHAND COLLEGE OF ENGINEERING

Walchand College of Engineering, Sangli.

(*An Autonomous Institute*)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Project Report entitled, **" SCHOLARGRAPH"** submitted by

21510007 – Neha Kundan Sonkamble

21510037 – Sourabh Yashwant Chaugule

21510102 – Maria Sarfraz Shaikh

21510119 – Trilok Vikram Kulkarni

to **Walchand College of Engineering, Sangli**, India, is a record of bonfire Project work of course **"PROJECT-II (6CS492)"** carried out by him/her under my/our supervision and guidance and is worthy of consideration for the award of the degree of Bachelor of Technology in Computer Science & Engineering during the academic year ***2024-25***.

Mr. Shailesh Patil Guide		External Examinar

Mrs. Dr. M. A. Shah

Head

Department of Computer Science and Engineering

Declaration

I hereby declare that work presented in this project report titled **"SCHOLARGRAPH"** submitted by us in the partial fulfilment of the requirement of the award of the degree of Bachelor of Technology (B.Tech) Submitted in the Department of Computer Science & Engineering, Walchand College of Engineering, Sangli, is an authentic record of my project work carried out under the guidance of Mr. Shailesh Patil.

21510007 – Neha Kundan Sonkamble

21510037 – Sourabh Yashwant Chaugule

21510102 – Maria Sarfraz Shaikh

21510119 – Trilok Vikram Kulkarni

Date: 13/05/2025

Place: Sangli.

Acknowledgement

We would like to express our sincere gratitude to all those who supported and guided us throughout the successful completion and implementation of this project. We are especially thankful to our Director, Prof. U. A. Dabade, and our Head of the Department, Dr. M. A. Shah (CSE), for their continuous encouragement, valuable advice, and for providing the resources necessary to carry out this project. Our heartfelt appreciation goes to our project guide, Mr. Shailesh Patil, for his dedicated mentorship, insightful suggestions, and constant motivation, which were vital to the progress and successful execution of our work. We also wish to thank our families and friends for their unconditional support, patience, and encouragement during every stage of this journey. Your faith in us gave us the strength to overcome challenges and stay committed to our goals.

Abstract

In the information age, researchers and students are constantly confronted by the increasing volume of academic literature on the internet. Thousands of research papers by disciplines are produced daily, making it hard to keep up to date, increasing relevant information, or even start an intensive literature review. This leads to information overload whereby the volume of resources makes it difficult to work instead of strengthen the work. The previous methodologies of conducting literature reviews time-consuming and hard when that comes to the processing of bulk data.

ScholarGraph is a solution to address these modern scholarly problems. It is a semi-automatic platform that facilitates the recovery of research papers, analysis, and visualization of papers. Using a set of technologies that encompass Natural Language Processing (NLP), web scraping, and knowledge graph construction, ScholarGraph allows users to input a topic of interest and retrieve relevant research papers from sources such as Google Scholar with the help of automation. By doing various operations allow us to for extract key terms, phrases, and entities from the papers.

Once the primary information is extracted, advanced methods such as TF-IDF, cosine similarity, and cooccurrence matrices and other utilized to determine relations between words and increase the content of the research. The resultant data is stored in graph database for better visualization. This visual representation assists users to better see how varying concepts connect to each other and find research gaps areas or groups of knowledge.

ScholarGraph offers an intuitive front end with interactive navigation capabilities over such knowledge graphs. It is possible for researchers to ask terms, explore the relationships and retrieve overview of the research part of interest. Coupled with the intelligent data extraction and visualization features, this work significantly reduces effort levels in preliminary stages of research and helps the user make more rapid, educated decisions about his area of study.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviations	Abbreviations Description
NLP	Natural Language Processing
BERT	Bidirectional Encoder Representations from Transformers
NER	Name Entity Relation
AI	Artificial Intelligence
DGL	Deep Graph Library
TF-IDF	Term Frequency-Inverse Document Frequency
LSA	Latent Semantic Analysis
LDA	Latent Dirichlet Allocation
API	Application Programming Interface
NLTK	Natural Language Toolkit
UI	User Interface
OpenIE	Open Information Extraction
GraphQL	Graph Query Language
SerpAPI	Search Engine Results Page API
RestAPI	Representational State Transfer Application Programming Interface

JSON	JavaScript Object Notation

CONTENTS

CHAPTER 1: INTRODUCTION **Page Number**

CHAPTER 2: LITERATURE REVIE

CHAPTER 3: PROPOSED METHODOLOGY

CHAPTER 4: IMPLEMENTATION

CHAPTER 5: RESULTS AND ANALYSIS

CHAPTER 6: CONCLUSION AND FUTURW SCOPE

CHAPTER 1: INTRODUCTION

1. **Project Overview**

ScholarGraph gathers information primarily from scholarly articles based on the topics or keywords entered by the user. This data is sourced from platforms such as Google Scholar using APIs or scraping tools. The model gets the desired research articles. The extracted content typically includes data from abstract till reference. When available, the system can also access the complete text of open-access articles in PDF formats. After collection, the data undergoes a preprocessing stage involving natural language

processing (NLP) techniques. These include removing irrelevant terms which are known as Stopwords, normalize the words through lemmatization, at last forming tokens to do the process

To identify key concepts and relationships within the text the ScholarGraph uses the NLP models such as spaCy to detect named entities, domain-specific terms, and important keywords. It then examines how these elements co-occur and are related semantically which is then converting process containing these insights into a graph-based structure where each concept is a node and each relationship forms an edge. The output data that we get is then stored in the structured formats like JSON, txt or CSV and is then send into graph databases like Neo4j for better visualization. This leads to the creation of a dynamic knowledge graph that helps researchers explore related ideas by uncovering the hidden patterns, and gain deeper insights from complex academic literature in an intuitive and visual manner.

1.2 Problem Statement

The explosive growth of scholarly literature has rendered it ever more challenging to find and process important information manually. Conventional search strategies do not come close with the size of research currently available and produce information overload the researcher. Moreover, currently used tools do not give the desired. This project seeks to solve these challenges through creating a system that computerizes the searching, fetching, analysis, and visualization of research material, providing an integrated, user-oriented method of literature review and knowledge discovery. Making the task of the research on a topic fast, reliable and less time consuming.

1.3 Objectives

- To automate searching and retrieving full-text research papers from websites such as Google Scholar through APIs.
- To preprocess and extract information from academic texts by applying NLP methods.
- To recognize significant entities, concepts, and terms via tokenization, lemmatization, n-gram extraction, and Named Entity Recognition.
- To examine term relationships through methods such as co-occurrence matrices, TF-IDF, and similarity metrics.
- To apply topic modeling with LSA and LDA for theme revelation and research patterns.
- To develop and store terms relationships and visualize that relations through graph databases like Neo4j.
- To design an interactive UI that allows users to visualize the resulting knowledge graph and navigate findings.

1.4 Scope of the Project

- ScholarGraph is particularly targeted towards the research and academic field. It aims at streamlining the initial phases of the research process, particularly during literature review and topic discovery.

- The system will operate on publicly available research papers in PDF or HTML format obtained from websites like Google Scholar using APIs like SerpAPI.

- It's designed to pull and work with information like abstract, keywords, content details, and associations between research concepts — mainly the text data of papers.

- ScholarGraph applies natural language processing (NLP) to detect important terms and interactions, and subsequently projects this information onto a visual, interactive knowledge graph for easy comprehension.

- The scope is restricted to user-defined areas of research (e.g., maize crop disease, climate change, cancer etc). Although the system is generalizable, its current prototype might excel most in areas with an good amount of available literature.

- The system does not try to summarize complete research papers, evaluate the quality of research, or interpret results and methods for the researcher.

- ScholarGraph is designed to be a facilitative tool — a research assistant that speeds up the discovery of relevant literature and uncovers implicit connections, but still relies on the user's own experience and judgment for interpretation and critical assessment.

- It is not a paper writer or citation manager. Rather, its purpose is to equip users with a more coherent and quicker comprehension of their subject of choice through AI-generated insights.

1.5 Motivation

- Scholarly research is expanding at a very fast rate, which makes it challenging for students, scholars, and researchers to keep the track of research with the enormous volume of published papers.
- Searching, reading, and analyzing papers manually is time consuming, intellectually exhausting, and sometimes inefficient and particularly when working with interdisciplinary or unknown subjects.
- Available search utilities (such as Google Scholar) assist in identifying papers but do not aid in structuring, summarizing, or relating the principal concepts of those papers.
- There is an evident need for an intelligent system that retrieves not just academic content but also processes it, determines key concepts, and provides it in a comprehensible manner.
- ScholarGraph is inspired by these challenges and to overcome those. It seeks to simplify and automate the first steps of research, allowing users to quickly understand core basic themes, key entities, and their interrelations within a subject.

- The visual knowledge graph produced by the system enables users to navigate interrelated concepts and discover patterns that may be lost in conventional reading.
- By decreasing the effort involved in literature reviews, ScholarGraph enables researchers to spend more time on critical thinking, creativity, and innovation.
- This project also demonstrates the increased potential of using the NLP to make academic workflows easier, more accessible, and more organized.

1.6 Significance

ScholarGraph is more than it looks, that gives a better way of imagining the academic content. With the current research working state the, students and researchers are under deep pressured by the amount of published work. Sorting through several hundred papers to find the work that's actually relevant is intensive to do. ScholarGraph solves this problem by automating data download process and visualization to provide an intelligent and user-friendly means of performing literature reviews and extracting insights.

Here's how this project makes a change in workload that student and researcher:

1. Enhances Research Productivity

No longer will researchers spend hours fetching and browsing through papers by hand. Now, in mere seconds, they have access to a mapped view of the knowledge universe. ScholarGraph speeds up research and preserves invaluable time that is more useful to invest in thinking and analyzing.

2. Uncovers Latent Relations

Through constructing a knowledge graph that visually relates significant terms and concepts between various papers, the system reveals relations that may otherwise be overlooked by standard reading. It assists scholars in detecting concurrent themes, proofs, or even literature gaps with much potential to explore.

3. Encourages Targeted Research

ScholarGraph enables users to limit their search to highly specific domains or sub-topics, providing a more focused and relevant set of insights. This is particularly helpful for novice researchers who require direction in exploring unfamiliar areas.

4. Uses AI in a Meaningful Context

The project is an applied use of advanced technologies such as NLP, topic modeling, and graph databases. It illustrates how AI can augment human knowledge instead of replacing it, assisting researchers as a clever assistant, not a replacement.

5. Facilitates Cross-Disciplinary Work

The capacity to think about how things relate can provoke cross-disciplinary ideas. An agricultural researcher, for instance, could learn valuable strategies from computer science or economics because of keyword co-occurrences—something a manual review is less likely to produce.

6. Scalable and Adaptable

While the initial development is aimed at particular areas, the ScholarGraph architecture is adaptable enough to scale to different areas—medical research, engineering, social sciences, and so on. This positions it as a future-proofed tool with extensive applications.

In essence, ScholarGraph makes research in academia smarter, quicker, and more perceptive. It aligns with the changing requirements of the research world and provides new avenues for knowledge discovery.

CHAPTER 2: LITERATURE REVIEW

1. **Literature Review**

Paper Title	Platform	Key findings	Research gap
(2015)	ReVerb, OpenIE	The paper advances OpenIE by improving triple extraction from unstructured text, enhancing accuracy, scalability, and domain independence. It explores applications in knowledge bases, question answering, and text summarization.	Contextual understanding, adapting to domain-specific texts, and enhancing integration with knowledge graphs can be expanded on.
	BERT-based NER, SpaCy	This paper discusses Named Entity Recognition (NER), a crucial NLP task for identifying entities like names, locations, and dates in text. It explores rule-based, machine learning, and deep learning approaches, highlighting their applications in search engines, question answering, and content categorization.	Existing NER models struggle with domain adaptation, fine-grained entity recognition, and handling noisy text, limiting their effectiveness in diverse real-world applications.
		This paper presents a large-scale, heterogeneous academic knowledge graph covering millions of publications, authors, and affiliations. It discusses MAG's data structure, citation relationships, and applications in academic research and trend analysis. The	This paper lies in the need for improved data completeness, accuracy, and

	Microsoft AI	paper highlights MAG's role in enhancing scholarly discovery through machine learning and bibliometric studies.	real-time updates. Additionally, there is a lack of advanced methods for handling entity disambiguation and evolving academic trends.
	NLP, Graph Databases, AI Models	The paper presents an extensive RDF knowledge graph derived from OpenAlex, modelling the global research ecosystem. With over 26 billion triples, it interlinks publications, authors, institutions, and concepts, enabling advanced semantic analysis. The work enhances scientific data interoperability and supports applications in scholarly discovery, analytics, and recommendation systems.	The lack of dynamic, open-access knowledge graphs that can effectively capture evolving scholarly data. Existing solutions are either static, fragmented, or lack interoperability, limiting their ability to support advanced research analytics and discovery.
Open Information Extraction for Scholarly Texts [14] (2015)	ReVerb, OpenIE	Advanced OpenIE methods for extracting triples from unstructured scholarly text, enhancing accuracy, scalability, and domain independence. Applied to knowledge bases, question answering, and summarization.	Contextual understanding, domain adaptation, and improved integration with knowledge graphs are still needed.

Entity Recognition in Scientific Literature [15] (2018)		Surveyed rule-based, ML, and deep learning approaches for NER in scientific texts. Emphasized applications in search engines, content categorization, and QA systems.	Current NER models face challenges with domain adaptation, fine-grained entity detection, and handling noisy, unstructured data.
	SpaCy, Stanford Dependency Parser	Combined co-occurrence analysis with dependency parsing to extract accurate relations in biomedical texts, improving over co-occurrence-only methods.	Does not utilize similarity scores (e.g., Jaccard) for enhancing weak relations; lacks integration into graph visualization platforms.
	BERT, Neo4j, DGL	Utilized deep learning for entity linking and relation extraction to build scholarly knowledge graphs, improving accuracy.	User-centric visualization, interactive graph exploration, and automated keyword clustering are underexplored.
	Neo4j, GraphQL	Demonstrated that graph-based exploration significantly improves literature review efficiency compared to traditional search methods.	Highlights need for automated n-gram extraction, TF-IDF scoring, clustering, and similarity-based relation mapping for comprehensive exploration.

Table. 1: Literature Review

1. **Research Gap**

Even with the advent of a number of research tools and platforms over the past few years, there is still a visible lack of systems offering an integrated, end-to-end solution for initial-stage academic investigation. Current platforms like Microsoft Academic Graph (MAG), AMiner, and Semantic Scholar have made significant contributions to the structuring of scholarly data. Yet, the majority of these systems are built upon structured metadata like author names, citations, and journal sources. They frequently don't take time to go deep into the complete textual content of research articles, particularly when it comes to eliciting conceptual associations or thematic frames from the paper body itself.

A second important limitation is accessibility. Most of today's tools, although very powerful, are intended for experienced users' data scientists, digital librarians, or institutions with technical staff to manage APIs and bespoke scripts. For most students or junior researchers, particularly those without advanced programming skills, these platforms can be daunting or impossible to use. There is little in the way of tools that make the research review process meaningful for these users.

Moreover, most existing platforms are not truly real-time or user-driven. They offer access to predefined datasets and static knowledge graphs, which are useful for retrospective analysis but lack the flexibility to handle novel research queries or emerging fields. For example, a student wishing to explore a niche or interdisciplinary topic may find that existing tools either do not support that topic or lack relevant papers. Conversely, these systems do not often enable the users to customize the graph structures according to particular requirements, nor do they provide visualization tools that can be manipulated and read by the user as they see fit.

ScholarGraph specifically overcomes these limitations by providing an interactive, semi-automated tool where researchers can begin with a basic topic input and get the full pipeline of research paper retrieval, natural language analysis, and relationship visualization as knowledge graphs. It operates in real-time, retrieving pertinent papers via live queries (e.g., through SerpAPI to Google Scholar), processes unstructured data with NLP methods (like lemmatization, named entity recognition, and topic modeling), and renders the processed data in an easily consumable visual format.

This method closes the usability gap since it provides a low-code/no-code interface that is not dependent on technical knowledge, and it a

CHAPTER 3: PROPOSED METHODOLOGY

3.1 System Design

The system has been proposed as an NLP pipeline that converts huge amounts of unstructured text into a structured knowledge graph. These graph helps the researcher to get a better insight into the key data and their relationship. The last product is an interactive graph database (Neo4j) that can be interacted directly and can be queried through a REST API.

Key Components and Workflow

1. Topic input and pdfs gathering: When a user specifies a research topic the relevant Paper Extraction depending on the topic, a collection of related scientific papers are gathered from well-known sources. These documents are used as the starting point of the knowledge exploration.

1. Text preprocessing: The model uses the PyMuPDF (fitz) for extracting textual data from the PDF files. Basic text cleaning is done to eliminate unwanted words and taking the part between the abstract till reference. The cleaned text is then tokenized, lemmatized and stopwords along with unwanted words.

1. b. N-Gram Generation and TF-IDF Calculation: With help of the NLTK the system generates n-grams ranging from unigrams to desired n-gram by keeping threshold to get the n-gram who pass that threshold and also by removing the redundant n-grams. The top n-grams are taken and a tf – idf score is assigned to them.

1. Filtering using Similarity Scores: A co-occurrence matrix is built to determine the frequency of occurrence of n-grams together in documents. Jaccard similarity scores are derived from this in order to measure how each n-gram are related or similar to each other. A threshold of filtering is used to keep only the most significant relationships.

1. Dependency Parsing and Relation Extraction: Dependency between each n-gram is identified and a relation is found out. With that dependency parsing gives us the data how n-grams are connected to

one another and what relation do they follow with one another.

1. Named Entity Recognition (NER): All entities in the relation set are passed through an NER model to label with proper value. This shows the difference between similarly looking terms and labels nodes in the graph correctly.

1. Graph Generation and UI interface: The dependent data is stored in graph database to create a interactive knowledge graph. The UI interface is connected to that knowledge graph to extract the related data about the desired keyword that user want about his research topic.

1. **General Methodology**

Data Collection:

The process starts with data collection, where the researcher inputs a research topic. Google API is used to get the required papers to keep in corpus. The downloaded papers are stored in the folder for further processing. These serves as the foundation for upcoming analysis. To run the process quickly the UI is made where user can define their research topic along with the papers count that required. By that can get the desired papers enabling quick and efficient building of the corpus.

Data Preprocessing:

Now after that on the gather files the text is extracted and with the help of NLTK (Natural Language Processing Tool Kit) downloads like punkt to tokenize, wordnet to lemmatization and stop words to remove the stop words the task of tokenization, lemmatization and stop words removal is performed respectively. Along with that the unwanted words set is kept by which we remove all the unwanted words. Then the n-grams with how much they occur in all the corpus are identified along with that a threshold is kept . Every n-gram which have frequency higher than that threshold is kept with each n-gram having different frequency threshold they have to watch for , then we remove the redundant n-grams which are the subsets of the higher n-gram by reducing the redundant n-gram count (frequency of redundant n-gram – frequency of higher n-gram) and when that redundant n-gram count reaches zero we remove them making the n-grams more perfect to use without any impurity , after that the cleaned n-grams are gathered and from that top 1500 most frequent occurring n-grams are taken which helps in gathering the most relevant n-grams from the corpus.

TF – IDF and Cooccurrence matrix:

These n-grams are then made to Term Frequency-Inverse Document Frequency (TF-IDF) so that their frequency with each paper is identified and a score is kept. This filter is again used to get the better viewed n-grams. Now by using that a co-occurrence matrix is prepared with n-gram, paper and counts of frequency of that n-gram with that paper.

Visualization:

Once the key entities and frequent terms are identified, visual representation of that data is created to provide clear insights about that data. A WordCloud is generated by combining all relevant text into a single string, where the font size increases with increasing frequency of the n-grams.

LSA LDA:

After the n-grams are filtered, LSA (Latent Semantic Analysis) and LDA (Latent Dirichlet Allocation) is performed. LSA begins by using the tf-idf score to know the importance of each in the corpus. Then that is reduces the complexity using the mathematical technique called Truncated Singular Value Decomposition (SVD). This gives the words group with similar context, which helps in imagining the building structure that we will use. The result is top n terms for each topic describing topic. For LDA the approach takes a different way by treating each document as a mix of topics then tries to figure out which topic is likely to explain the words in the document.

Calculating the Similarity Between N-Grams:

The similarity between two n-grams occurring in the top filtered set is used to determine how often they co-occur across the corpus. This is achieved using Jaccard Similarity, which measures how closely one n-gram is associated with another within the papers. To calculate this, the n-grams from a single paper are considered as a set. Let A be the set of target n-grams in one paper, and B be the set of n-grams in another paper. The Jaccard Similarity between these two sets is computed using the formula: $J(A, B) = |A \cap B| / |A \cup B|$

A higher Jaccard Similarity score indicates that the n-gram appears frequently across multiple papers, thus reflecting strong similarity or co-occurrence. Conversely, a lower score suggests limited overlap, representing weaker association or less frequent occurrence between the n-grams.

Dependency parsing:

This step analysis how the n-gram are connected by the use of cooccurrence matrix and Jaccard Similarity along with the LSA and LDA topic. Now a another Jaccard similarity are taken which is ngram with other ngram then that gives the value of similarity and that is done by using ngram with papers Jaccard similarity that we got above. Here a threshold hold is kept above which the Jaccard similarity data is extracted and used. Now on that data we use spaCy model to perform dependency parsing and gives the Subject Verb Object (SVO) relationship to achieve the entities and their relation. The parsing gives a overview of which word in entity is subject, which are verb and which are object. That helps in defining proper relationships and storing them in proper csv to be used in further.

NER:

As soon as we get the dependency parsing, we proceed the step to get NER. Named Entity Recognition is used to find and label important terms in the text, that help the in understanding the meaning of text even better way. To do that a transformer model is used which will identify the entities and there relation and tag them. The given entities are listed and then a proper value to each entity is given and the proper relation is discovered between each one of them and given to them.

Storage and UI:

Finally, the nodes obtained from dependency parsing and the values got by NER are added to Neo 4j data base. The Named Entity Recognition (NER) improves the understanding by which the proper relationships are established between the nodes to ensure the data is structured in a user-friendly manner. The nodes and graph present there can be interacted with. In addition, appropriate queries are defined to facilitate efficient querying and processing with the data. A Ui is developed with flask containing the Rest API and connected to the Neo4j database. With this UI user can gather data about any node they want.

Block Diagram:

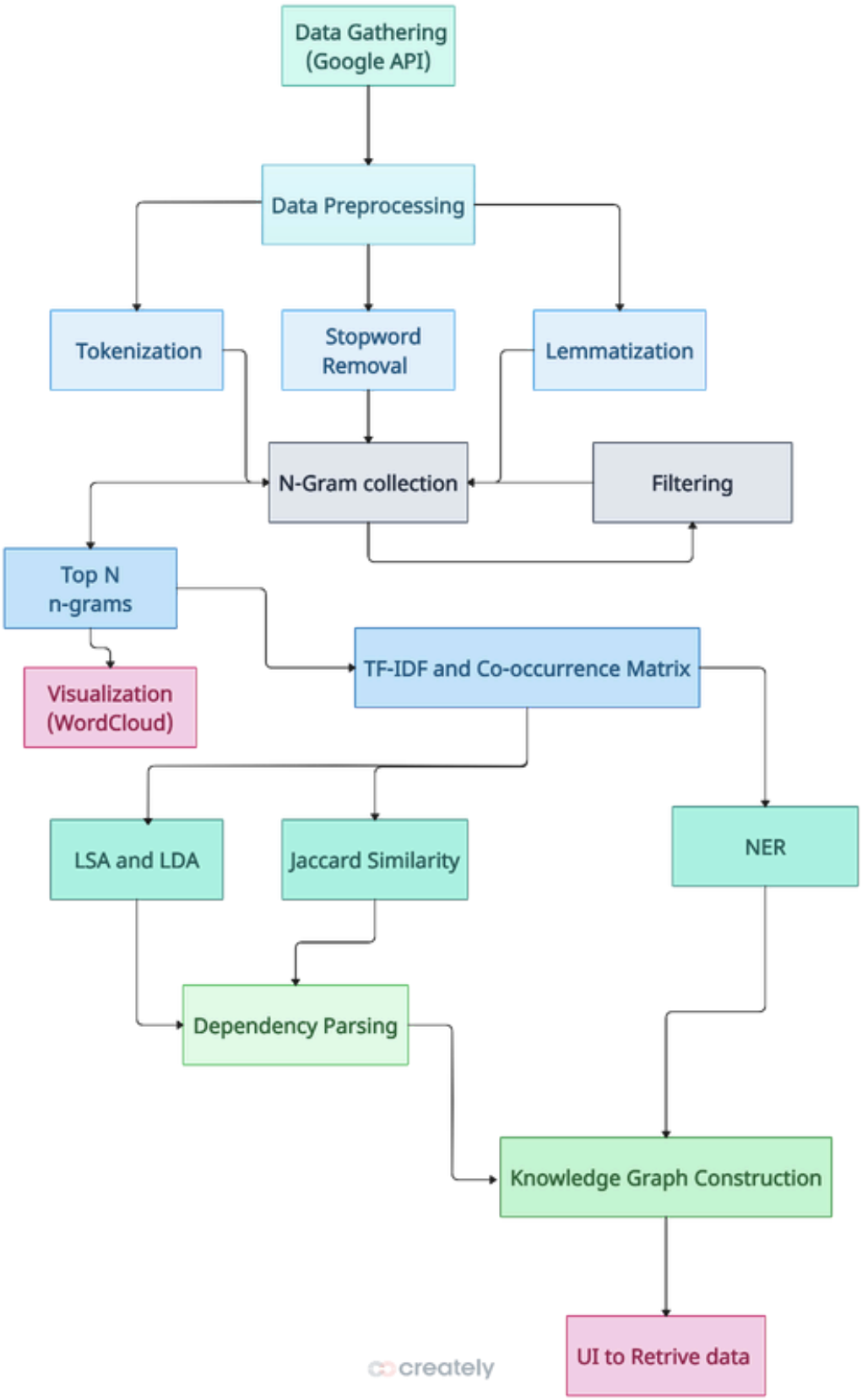


Fig. 1 : Block Diagram

1. **UML Diagram and Flowchart**

UML Diagram:

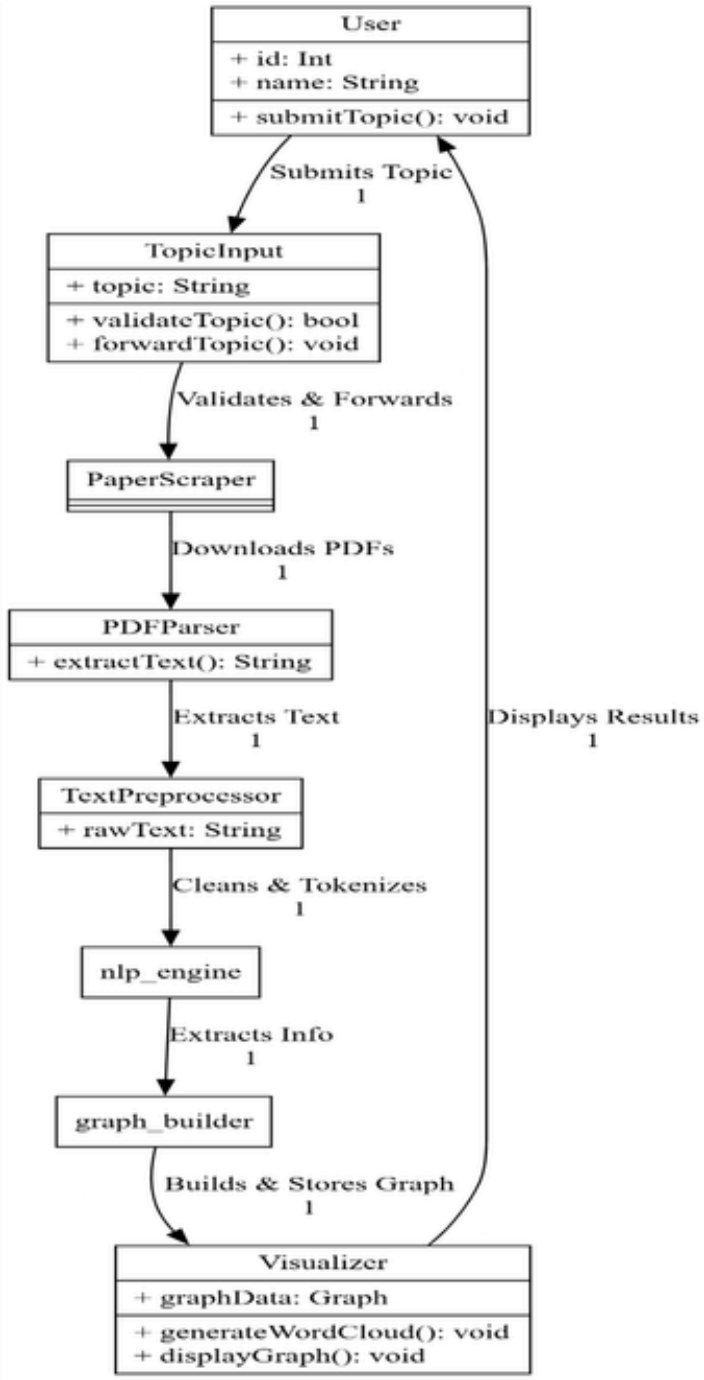


Fig. 2 : UML Diagram

Flowchart:

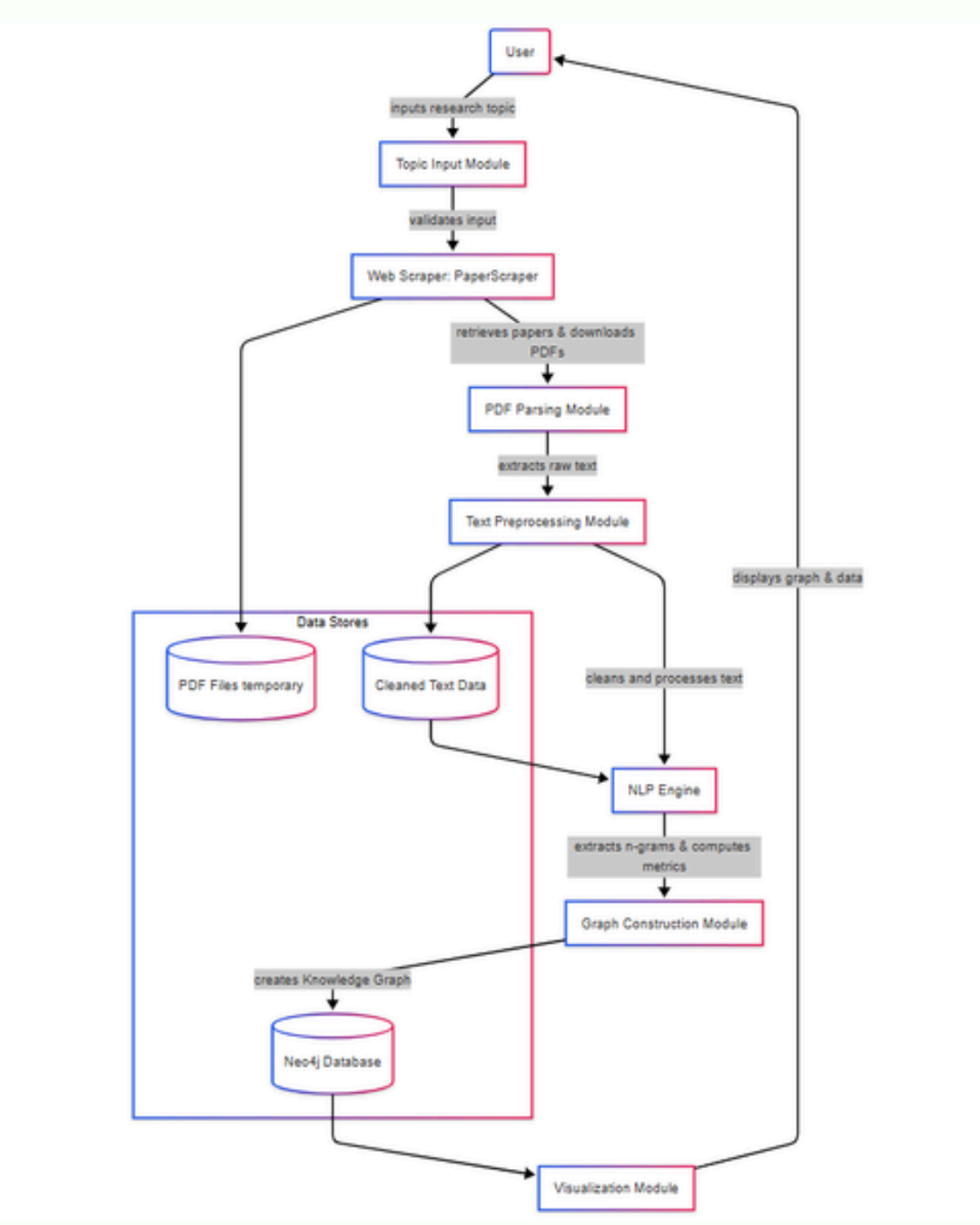


Fig. 3 : Flowchart

3.4 Modules / Components

The project has been divided into a number of functional modules that perform different tasks in the Natural Language Processing pipeline. The modules are executed in sequence to harvest useful biomedical data from raw text and store them in a graph database in structured form for later querying and analysis. The primary components of the system and their functionalities are described below:1. Research Topic and Paper Acquisition ModuleFunctionality:This is the gateway to the system. The user provides a research topic they are interested in and on the basis of which a list of related papers is retrieved either from a pre-stored database or web repositories. The aim is to collect a corpus for downstream processing

2. Text Extraction and Cleaning ModuleFunctionality:This step pulls raw text out of the research papers, usually PDFs, with PyMuPDF (fitz). After extraction, the text is preprocessed to unnecessary items like: Page numbers, headers, footers, Citations, URLs, and special characters the non-informative elements such as references or metadata too. The result is cleaner text for analysis.

3. Preprocessing ModuleFunctionality:Preprocesses the cleaned text for analysis through various NLP methods:Tokenization: Convert text into tokens to use in next step.Stopword Removal: Removes frequent and unnecessary wordsUnwanted Word Filtering: Removes pre-defined unwanted wordsLemmatization: Converts words to their base form for uniformity

4. N-Gram and TF-IDF ModuleFunctionality:This module extracts ngrams (1-gram to 4-gram) from preprocessed tokens in order to encode informative terms. The relevance of each n-gram is computed using TF-IDF (Term Frequency–Inverse Document Frequency) to filter the unrelated terms. Output is given as a top-ranked list of relevant n-grams that have to be used.

5. Co-occurrence and Similarity Filtering ModuleFunctionality:Builds a co-occurrence matrix representing how frequently ngrams occur together within documents. On that basis, Jaccard Similarity is computed between sets of n-grams from various papers:formula: $J(A, B) = |A \cap B| / |A \cup B|$6. Dependency Parsing and Relation Extraction ModuleFunctionality:Named Entity Recognition (NER) is used to assign a label to each word that is recognized. This guarantees that every term is put in its proper position within the graph based on it proper idea.

7. Named Entity Recognition (NER) ModuleFunctionality:Named Entity Recognition (NER) is used to assign a label to each word that is recognized.This guarantees that every term is put in its proper position within the graph based on it proper idea.

8. Module of Graph ConstructionFunctionality:All terms and relations extracted are stored in graph database that is a Neo4j graph database:

- Nodes are used to represent entities (words)
- Edges are used to represent relations between those entities

9. Module of API and Query InterfaceFunctionality:A RESTful API, implemented with Flask which links the graph database to a user interface. These helps user to Search for desired term and get the data.

3.5Technologies Used

a. Programming Language: Python 3.10+
b. Libraries:
    - spaCy : for tokenization, dependency parsing and NER
    - Pandas and Numpy : To work on CSV data
    - NLTK (Natural Language Toolkit) : for text preprocessing, Stopword removal, tokenization and ngram generation.
    - Scikit-learn: Get tf-idf score
    - PyMuPDF (fitz) : Extract textual content from research paper
    - Flask: Build RestAPI backend and connect to graph database
c. Database:

Neo4j: To store knowledge graph consisting of nodes.

a. Development Environment:
    - Jupyter Notebook
    - Google Colab
    - Anaconda / pip

CHAPTER 4: IMPLEMENTATION

4.1 Development process

1. Requirement GatheringWe started with the identification of the main problem: researchers spend considerable time looking for academic documents manually and attempting to make sense of the vast body of information. It was evident that we had to automate this process. Our goal revolves around creating a tool that would automatically fetch research papers, extract relevant information from them, and display it visually in an intuitive and meaningful manner. We were going to employ Natural Language Processing (NLP) for text understanding and knowledge graphs for visualization.

2. Technology Stack Selection: Once the objectives were established, we chose the appropriate tools to get it done. Python was used for the backend due to its great libraries for data manipulation and NLP. React was chosen to develop a better and robust web application that would act as our interface. For storing and managing relationships between research terms and topics, we chose Neo4j a graph database famous for working with connected data gracefully.

3. Paper Retrieval ModuleWe then developed a module that retrieves research papers. We utilized GoogleAPI, which interfaced with Google, to automatically query and retrieve links and papers. We also ensured that users could feed in their topic or keywords themselves, so that the system would customize the search accordingly.

4. Text PreprocessingOnce the papers were gathered, we prepared and cleaned the text data. Through **spaCy**, we tokenized the text into words and phrases, eliminated stopwords (such as “and”, “the”, and so on), and lemmatized words such that variations such as “studying” and “study” were given equal treatment. We also extracted n-grams (multi-word common terms), named entities (such as locations, illnesses, chemicals), and subject-verb-object (SVO) relations to give a better understanding of the data.5. Information Analysis & ExtractionAfter the text was pre-processed, we concentrated on extracting what was most important. We employed TF-IDF to discover words that were significant throughout the collection of papers. Subsequently, we examined how frequently words co-occurred and how semantically

related they were. To move further, we utilized topic modelling based on algorithms such as LDA (Latent Dirichlet Allocation) and LSA (Latent Semantic Analysis) to discover themes and groups of connected ideas in various papers.

6.Knowledge Graph BuildingWith all this data in hand, we began constructing the knowledge graph. Each significant term or paper was a node in the graph, and the connections among them (such as co-occurrence or similarity) were edges. We utilized Neo4j to build and store these graphs. We also weighted the edges based on how weak or strong the connections were, so users wouldn't just view connections but also understand how significant those connections were.

7.VisualizationThe data is then entrusted in graph-based database which is Neo 4j which gives a better insight of the relations present. View each node and get a path connecting two different nodes to get a better idea of the relations and help the user visualize the data.

4.2 Code Snippets

Pre-processing:

```
nlp = spacy.load("en_core_web_sm")
standard_stopwords = set(stopwords.words('english'))

unwanted_set = set(u.strip().lower() for u in unwanted_words)
all_stopwords = unwanted_set.union(standard_stopwords)

min_word_length = 3
thresholds = {1: 150, 2: 100, 3: 54, 4: 25}
input_folder = "Cancer2/"
output_folder = "25-3-2025_14_29_3_5/"
ngram_file_path = os.path.join(output_folder, "ngrams.txt")

def extract_text_from_pdf(file_path):
    doc = fitz.open(file_path)
    return " ".join(page.get_text("text") for page in doc)

def is_valid_alphanumeric(word):
    return bool(re.match(r'^[a-zA-Z]+\d+$', word)) or bool(re.match(r'^\d+[a-zA-Z]+$', word))

def is_broken_word(word):
    if is_valid_alphanumeric(word):
        return False
    return (
        len(word) < min_word_length or
        word.endswith('-') or
        word.isdigit() or
        not re.match(r'^[a-zA-Z0-9]+$', word)
    )

def valid_token(w):
    return not is_broken_word(w)

def clean_and_lemmatize(text):
    tokens = word_tokenize(text.lower())
    tokens = [w for w in tokens if valid_token(w)]
    doc = nlp(" ".join(tokens))
    lemmas = [token.lemma_.lower() for token in doc]
    return [lemma for lemma in lemmas if lemma not in all_stopwords]
```

Fig. 4 : Preprocessing Code

Taking most occurring n-grams which are top 1500 :

```
def save_top_clean_ngrams(filtered_ngrams, filename="top_1500_ngram-2_priority.txt", top_n=1500):
    with open(filename, "w", encoding="utf-8") as f:
        for n in range(1, 5):
            f.write(f"\nTop {top_n} {['Unigrams', 'Bigrams', 'Trigrams', 'Fourgrams'][n-1]}:\n")
            top_ngrams = filtered_ngrams[n].most_common(top_n)
            seen = set(ngram for ngram, _ in top_ngrams)
            alphanum_priority_ngrams = [
                (ngram, count) for ngram, count in filtered_ngrams[n].items()
                if (contains_valid_alphanum(ngram) or contains_priority_keyword(ngram)) and ngram not in seen
            ]
            combined = top_ngrams + alphanum_priority_ngrams
            for ngram, count in combined:
                f.write(f"{' '.join(ngram)}: {count}\n")

top_ngrams_priority_path = os.path.join(output_dir, "top_1500_ngram-2_priority.txt")
save_top_clean_ngrams(ngram_data, filename=top_ngrams_priority_path, top_n=1500)

print("✅ Cleaned and Top Priority n-gram files saved!")
```

Fig. 5 : Top 1500 n-grams code

Jaccard Similarity:

```
def jaccard_similarity(set1, set2):
    intersection = len(set1.intersection(set2))
    union = len(set1.union(set2))
    return intersection / union if union != 0 else 0.0

ngram_sets = [set(ng.split()) for ng in cleaned_ngrams]
jaccard_matrix = [
    [jaccard_similarity(set1, set2) for set2 in ngram_sets]
    for set1 in ngram_sets
]

jaccard_df = pd.DataFrame(jaccard_matrix, index=cleaned_ngrams, columns=cleaned_ngrams)
jaccard_df.to_csv(output_file)

print(f"✅ Jaccard similarity matrix saved at: {output_file}")
```

Fig. 6 : Jaccard Similarity code

TF-IDF:

```python
import os
import zipfile
from PyPDF2 import PdfReader
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd

def extract_text_from_pdf(pdf_path):
    """Extracts text from a PDF file."""
    text = ""
    try:
        with open(pdf_path, "rb") as file:
            reader = PdfReader(file)
            for page in reader.pages:
                if page.extract_text():
                    text += page.extract_text() + "\n"
    except Exception as e:
        print(f"Error extracting {pdf_path}: {e}")
    return text


zip_path = "/content/Cancer_files.zip"
extract_path = "/content/Cancer_files/"


with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)

tf_idf_folder = os.path.join(extract_path, "")
file_list = os.listdir(tf_idf_folder)

documents = {}
for file in file_list:
    file_path = os.path.join(tf_idf_folder, file)
    text = extract_text_from_pdf(file_path)
    if text.strip():
        documents[file] = text


vectorizer = TfidfVectorizer(stop_words='english', max_features=1000)
tfidf_matrix = vectorizer.fit_transform(documents.values())


feature_names = vectorizer.get_feature_names_out()
df_tfidf = pd.DataFrame(tfidf_matrix.toarray(), index=documents.keys(), columns=feature_names)


df_tfidf.to_csv("TF_IDF_results.csv")
print("TF-IDF computation completed. Results saved in 'TF_IDF_results.csv'.")
```

Fig. 7 : TF – IDF code

Dependency Parsing:

```python
ngram_frequencies = ngrams_freq.set_index("ngram")["frequency"]

def get_dependencies(ngram):
    doc = nlp(ngram)
    dependencies = []
    for token in doc:
        dependencies.append((token.text, token.dep_, token.head.text))
    return dependencies

ngrams_freq['dependencies'] = ngrams_freq['ngram'].apply(get_dependencies)

ngram_edges = ngram_jaccard.where((ngram_jaccard > 0.33) & (ngram_jaccard < 1.0))
ngram_edges = ngram_edges.stack().reset_index()
ngram_edges.columns = ["source_ngram", "target_ngram", "similarity"]
ngram_edges["source_freq"] = ngram_edges["source_ngram"].map(ngram_frequencies)
ngram_edges["target_freq"] = ngram_edges["target_ngram"].map(ngram_frequencies)

ngram_edges_with_dependencies = ngram_edges.merge(ngrams_freq[['ngram', 'dependencies']], left_on='source_ngram', right_on='ngram

ngram_edges_with_dependencies = ngram_edges_with_dependencies.drop(columns=['ngram_y'], errors='ignore')
ngram_edges_with_dependencies = ngram_edges_with_dependencies.rename(columns={'ngram_x': 'source_ngram', 'dependencies': 'source_

ngram_edges_with_dependencies = ngram_edges_with_dependencies.merge(ngrams_freq[['ngram', 'dependencies']], left_on='target_ngram

ngram_edges_with_dependencies = ngram_edges_with_dependencies.drop(columns=['ngram_y'], errors='ignore')
ngram_edges_with_dependencies = ngram_edges_with_dependencies.rename(columns={'ngram_x': 'target_ngram', 'dependencies': 'target_

ngram_edges_with_dependencies.to_csv("25-3-2025_14_29_3_5/medical_ngram_dependencies_with_parsing_prio.csv", index=False)

ngram_paper = ngram_paper.apply(pd.to_numeric, errors='coerce')
ngram_paper_long = ngram_paper[ngram_paper > 0].stack().reset_index()
ngram_paper_long.columns = ["ngram", "paper", "score"]
ngram_paper_long["ngram_freq"] = ngram_paper_long["ngram"].map(ngram_frequencies)

ngram_paper_long.to_csv("25-3-2025_14_29_3_5/ngram_paper_links_prio.csv", index=False)

print(" Files saved: medical_ngram_dependencies_with_parsing.csv & ngram_paper_links.csv")
```

Fig. 8 : Dependency Parsing code

Enter data in neo4j:

```
uri = "bolt://localhost:7687"
username = "neo4j"
password = "sourabh@123"

driver = GraphDatabase.driver(uri, auth=(username, password))

file_path = "25-3-2025_14_29_3_5/medical_ngram_dependencies_with_parsing.csv"
df = pd.read_csv(file_path)

df['source_dependencies'] = df['source_dependencies'].apply(ast.literal_eval)
df['target_dependencies'] = df['target_dependencies'].apply(ast.literal_eval)

def create_ngram_graph(tx, source, target, similarity, source_deps, target_deps):
    tx.run("""
        MERGE (src:Ngram {text: $source})
        MERGE (tgt:Ngram {text: $target})
        MERGE (src)-[s:SIMILAR_TO]->(tgt)
        SET s.similarity = $similarity
    """, source=source, target=target, similarity=similarity)

    for word, dep, head in source_deps:
        tx.run("""
            MATCH (n:Ngram {text: $ngram})
            MERGE (t:Token {text: $word})
            MERGE (h:Token {text: $head})
            MERGE (t)-[:DEPENDS_ON {type: $dep}]->(h)
            MERGE (n)-[:CONTAINS]->(t)
        """, ngram=source, word=word, dep=dep, head=head)

    for word, dep, head in target_deps:
        tx.run("""
            MATCH (n:Ngram {text: $ngram})
            MERGE (t:Token {text: $word})
            MERGE (h:Token {text: $head})
            MERGE (t)-[:DEPENDS_ON {type: $dep}]->(h)
            MERGE (n)-[:CONTAINS]->(t)
        """, ngram=target, word=word, dep=dep, head=head)
```

Fig. 9: Neo4j data entry code

4.3 Dataset Description

The primary dataset for this project was research paper metadata and abstracts, which were gathered automatically from Google Scholar via API. Rather than using an existing academic database, we built a dynamic and live dataset from actual, real-time user-entered search topics. This made ScholarGraph responsive and versatile to various research topics.

Source of DataThe major sources of data were Google Scholar and arXiv, both of which are extensively utilized scholarly search engines. We utilized the API platform to programmatically access this data. By passing search queries through the API, we could retrieve research paper name, PDFs , Direct links to full papers

Structure of the DatasetEach retrieved paper provided multiple fields of desired information. The most important ones used in our processing pipeline were:Title: The name of the research paperAbstract: A summary of the paper, which formed the main body of text for our NLP tasksDOI or URL:
A distinctive link for citing or viewing the paper onlineThese aspects were important since they enabled us to conduct both metadata-driven analysis (e.g., author trends or publication
timelines) and content analysis (on the abstract).Data FormatAPI's raw data came in JSON format, which is simple to parse and extract a single field from. After processing, we stored the cleaned data in various formats:For abstracts and keyword extractionCSV: For structuring keyword pairs, relationships, and term frequency exportJSON: For consumption within visualization and topic modellingVolume of DataTo illustrate ScholarGraph's functionality, we restricted our dataset to a reasonable size during development. On average, we processed 500+ papers per topic or domain provided. This kept the knowledge graph tidy and comprehensible without compromising meaningful insights. Domains we tested included topics such as maize crop diseases, machine learning usage, and medical text processing.

Usage in the ProjectThe dataset was utilized for several purposes:The abstract text was utilized for NLP processing, including tokenization, entity recognition, topic modeling, and semantic similarity analysis. Metadata was utilized to label and annotate the knowledge graph so that users could comprehend the context of each node or relationship.

4.4 Challenges Encountered

- Most research papers were received in varying forms, which was challenging to read and extract useful information from. Some had incomplete abstracts, some had unusual structures, and others were simply difficult to process automatically.

- The text within such papers also used to have excess characters, e.g., special symbols, formulas, or control marks. Such excess symbols also made it difficult for the computer program to identify what the paper was actually expressing.

- When we attempted to display too many keywords or topics in the graph, it became cluttered. The graph began to appear messy and confusing rather than useful, making it difficult for users to

concentrate on the essentials.

- Not all the papers that were retrieved from online sources pertained to the topic typed in by the user. At times, the system retrieved papers that appeared related through keywords but weren't useful in reality.

- Whenever we processed a large number of papers simultaneously, the system would slow down. Processing large volumes of text and creating graphs for all of them consumed more resources and time than anticipated.

4.5 Tools and Platform

- Programming Language:
  - Python 3.10 for backend, data processing, and NLP
- Frameworks:
  - spaCy and NLTK for NLP
  - Scikit-learn for TF-IDF and LDA
  - Gensim for topic odelling
- Database:
  - Neo4j (community edition) for graph database storage
- Visualization:
  - (HTML + JS frontend) for data visualization using UI.
- APIs:
  - GoogleAPI for fetching research paper metadata
- Development Tools:
  - VS Code as the primary IDE
  - Jupyter Notebook to run the Python code
  - Git for version control
  - Neo4j to store the graphical data
- Hardware:
  - Development done on a standard Windows/Linux laptop with space of 8 to 10 GB RAM.

CHAPTER 5: RESULTS AND ANALYSIS

5.1 Testing Methodology:

To ensure that ScholarGraph functioned correctly and delivered key insight and results, we followed a structured testing methodology. Our objective was to verify that each component of the system performed as expected.

1. Unit Testing

Unit testing involved the testing each component of the system individually to ensure correct functionality:

- We tested whether the web scraper could accurately retrieve relevant research papers from online sources based on a user provided search topic. So to do that we have put variety of topics in web scrapper UI and tried to get the desired output that we need.
- The ngrams extraction module was tested to ensure it correctly identified significant words and data from the collected papers. To proceed we created and saved data in csv to verify them manually.
- We validated that the graph visualization tool correctly displayed the keywords along with their relationships. And to these we have used cypher query language to get the relevant data, node or path between them is used and studied.

These individual tests helped us to identify and fix issues we encounter.

2. Integration Testing

Once individual components passed unit tests, we moved on to integration testing:

- We tested whether the papers gathered by the scraper were correctly passed on to the NLP (Natural Language Processing) module. And that is done by checking the tokens created, ngrams identified and even by seeing corpus folder.
- We checked whether the ngrams extracted by NLP were correctly displayed in the graph structure. Which is done by viewing the csv which contain all ngrams , parsed ngrams and other also by using cypher query and searching random nodes to check the availability.
- We check the API connected to graph data base work correctly. That is checked through a process of randomly searching the search on UI and verifying the data we get. Integration testing helped validate the complete workflow of project.

3. User Testing

After technical testing, we performed user testing by involving participants. These users interacted with the system in real-time to simulate practical usage:

- Users typed in a research topic in the search interface.
- They reviewed the papers and keywords returned by the system.
- They explored the graph visualization to understand how different terms and subjects were interconnected.

Through user testing we got feedback and helped us in improving the model.

5.2 Test Case:

Test Number	Input	Desired Output	Description	Result
Test_1	Topic: Cancer Number: 5	Papers related to cancer	In UI topic is entered to get the papers related to topic	Passed
Test_2	Ngrams quality checking	Get relevant and necessary ngrams	Seeing the ngrams quality and how much they have worked	Passed
Test_3	Entered query to find any random nodes and their connection	Get nodes and a proper connection between them	Checking the knowledge graph	Passed
Test_4	Zooming and expanding nodes in graph	Zoom the graph and after clicking on nodes the connected nodes to main node should be displayed	Interactivity of the graph	Passed

Table 2 : Test Case

5.3. Results and Analysis

1. Corpus Extraction: The desired topic corpus is downloaded using the web Scrapper

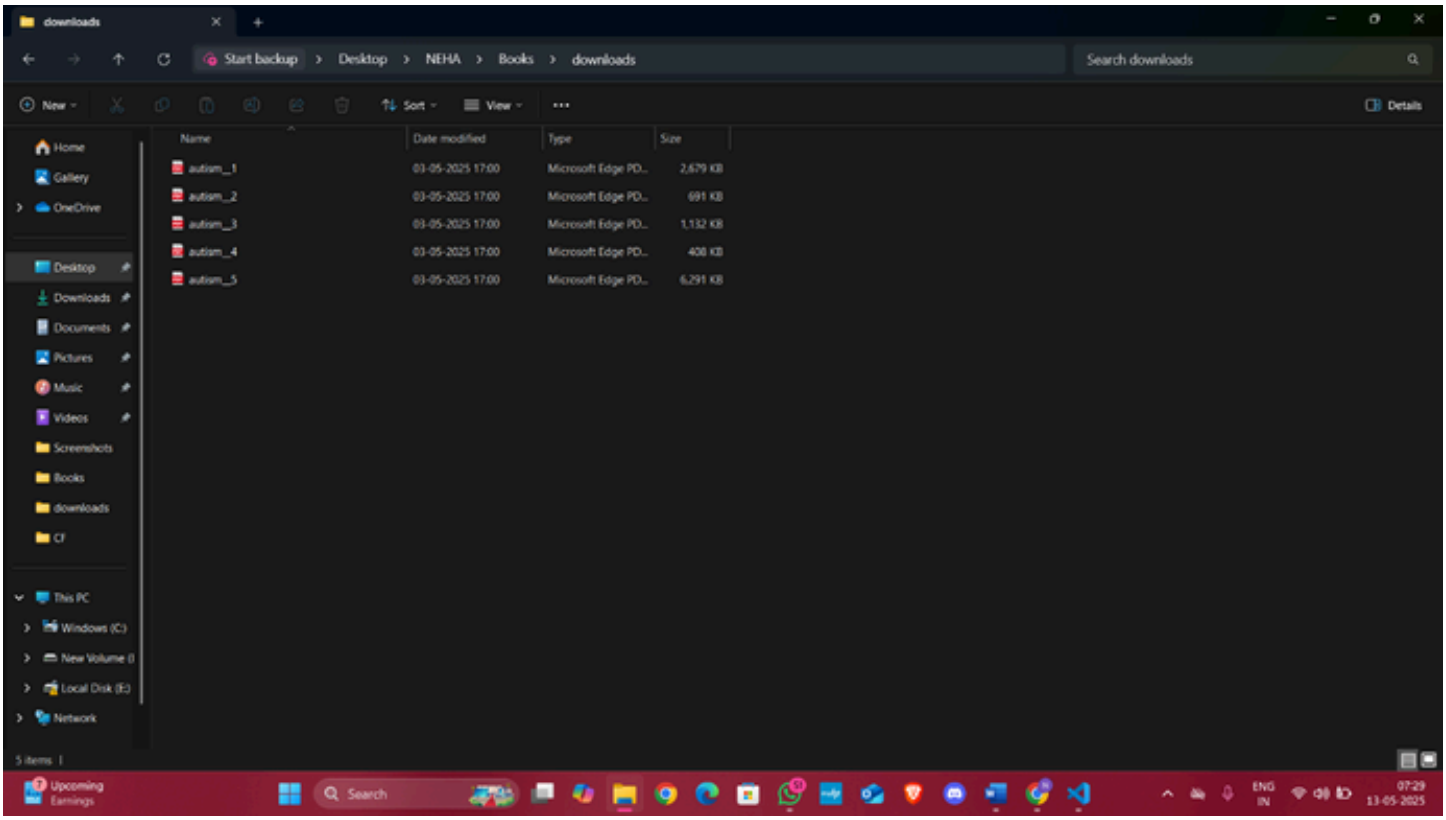


Fig. 10: Corpus Extraction (1)

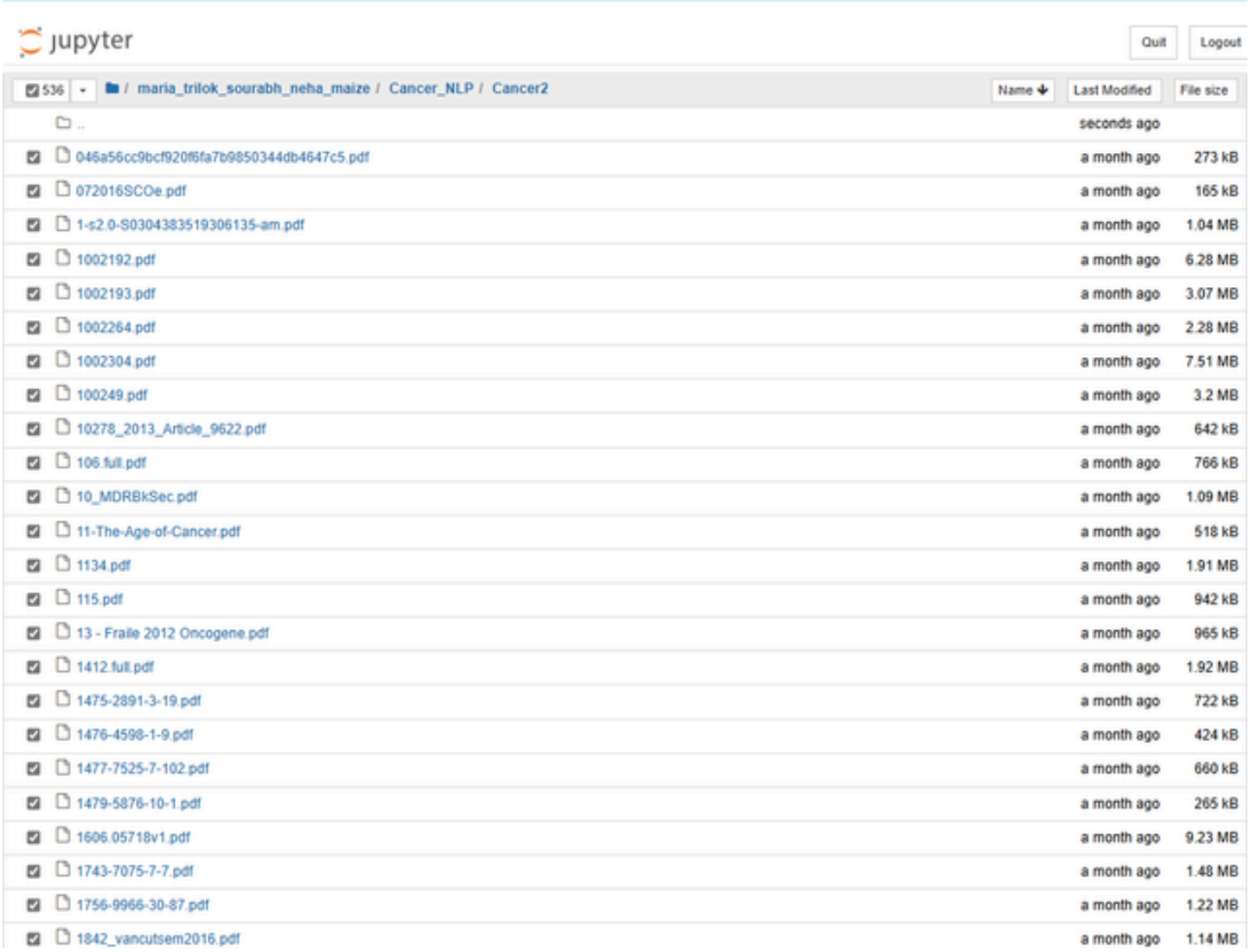


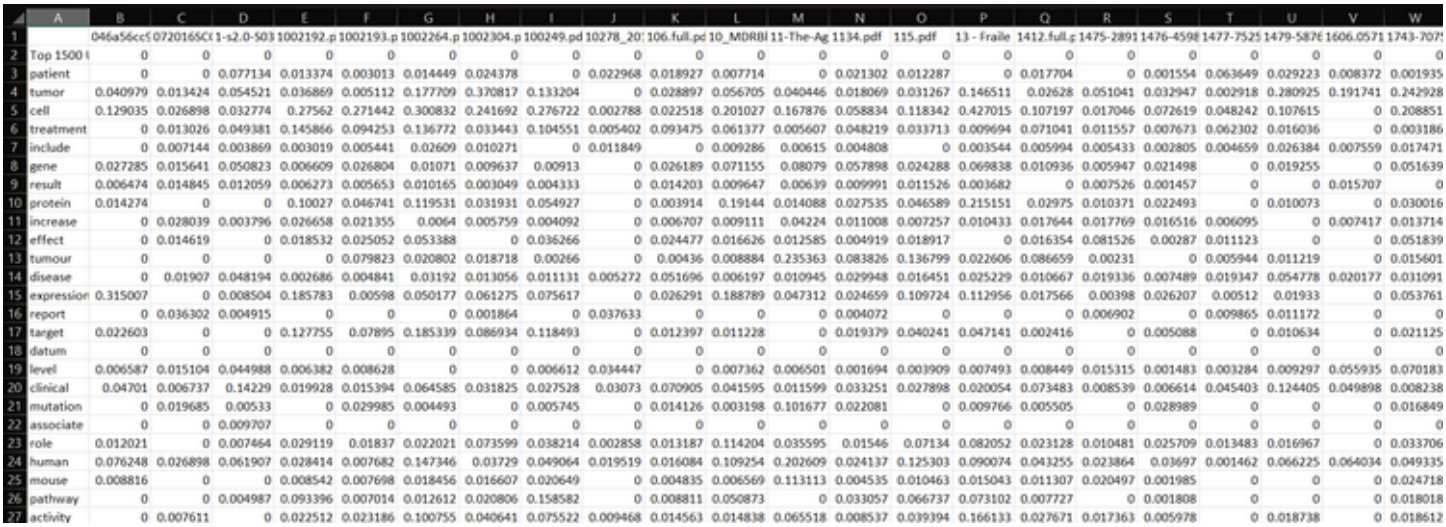
Fig. 11: Corpus Extraction (2)

2. N-gram Gathering: Most occurring n-grams are collected

ngram	total_frequency
patient	3878
tumor	15008
cell	27830
treatment	9646
include	2033
gene	6396
result	1576
protein	4968
increase	2738
effect	2794
tumour	7572
disease	7141
expression	8239
report	1487
target	2735
level	1696
clinical	8391
mutation	2271
associate	131
role	4743
human	7941
mouse	1966
pathway	3215
activity	4635
tissue	3451
case	2016

Fig. 12: N-gram Gathering

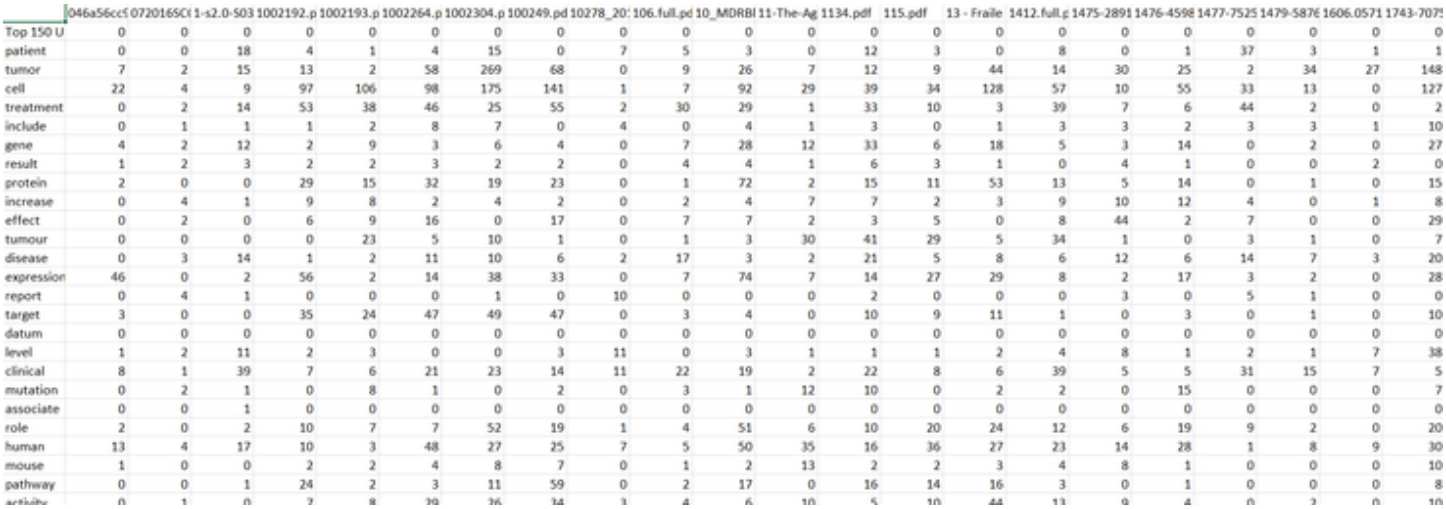
3. TF -IDF: TF – IDF score for each n-gram corresponding to papers are given



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1		046a56cc5	072016SC(	1-s2.0-S03	1002192.p	1002193.p	1002264.p	1002304.p	100249.pd	10278_20:	106.full.pd	10_MDRBI	11-The-Ag	1134.pdf	115.pdf	13 - Fraile	1412.full.g	1475-2891	1476-4598	1477-7525	1479-5876	1606.0571	1743-7075
2	Top 1500 U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	patient	0	0	0.077134	0.013374	0.003013	0.014449	0.024378	0	0.022968	0.018927	0.007714	0	0.021302	0.012287	0	0.017704	0	0.001554	0.063649	0.029223	0.008372	0.001935
4	tumor	0.040979	0.013424	0.054521	0.036869	0.005112	0.177709	0.370817	0.133204	0	0.028897	0.056705	0.040446	0.018069	0.031267	0.146511	0.02628	0.051041	0.032947	0.002918	0.280925	0.191741	0.242928
5	cell	0.129035	0.026898	0.032774	0.27562	0.271442	0.300832	0.241692	0.276722	0.002788	0.022518	0.201027	0.167876	0.058834	0.118342	0.427015	0.107197	0.017046	0.072619	0.048242	0.107615	0	0.208851
6	treatment	0	0.013026	0.049381	0.145866	0.094253	0.136772	0.033443	0.104551	0.005402	0.093475	0.061377	0.005607	0.048219	0.033713	0.009694	0.071041	0.011557	0.007673	0.062302	0.016036	0	0.003186
7	include	0	0.007144	0.003869	0.003019	0.005441	0.02609	0.010271	0	0.011849	0	0.009286	0.00615	0.004808	0	0.003544	0.005994	0.005433	0.002805	0.004659	0.026384	0.007559	0.017471
8	gene	0.027285	0.015641	0.050823	0.006609	0.026804	0.01071	0.009637	0.00913	0	0.026189	0.071155	0.08079	0.057898	0.024288	0.069838	0.010936	0.005947	0.021498	0	0.019255	0	0.051639
9	result	0.006474	0.014845	0.012059	0.006273	0.005653	0.010165	0.003049	0.004333	0	0.014203	0.009647	0.00639	0.009991	0.011526	0.003682	0	0.007526	0.001457	0	0	0.015707	0
10	protein	0.014274	0	0	0.10027	0.046741	0.119531	0.031931	0.054927	0	0.003914	0.19144	0.014088	0.027535	0.046589	0.215151	0.02975	0.010371	0.022493	0	0.010073	0	0.030016
11	increase	0	0.028039	0.003796	0.026658	0.021355	0.0064	0.005759	0.004092	0	0.006707	0.009111	0.04224	0.011008	0.007257	0.010433	0.017644	0.017769	0.016516	0.006095	0	0.007417	0.013714
12	effect	0	0.014619	0	0.018532	0.025052	0.053388	0	0.036266	0	0.024477	0.016626	0.012585	0.004919	0.018917	0	0.016354	0.081526	0.00287	0.011123	0	0	0.051839
13	tumour	0	0	0	0	0.079823	0.020802	0.018718	0.00266	0	0.00436	0.008884	0.235363	0.083826	0.136799	0.022606	0.086659	0.00231	0	0.005944	0.011219	0	0.015601
14	disease	0	0.01907	0.048194	0.002686	0.004841	0.03192	0.013056	0.011131	0.005272	0.051696	0.006197	0.010945	0.029948	0.016451	0.025229	0.010667	0.019336	0.007489	0.019347	0.054778	0.020177	0.031091
15	expression	0.315007	0	0.008504	0.185783	0.00598	0.050177	0.061275	0.075617	0	0.026291	0.188789	0.047312	0.024659	0.109724	0.112956	0.017566	0.00398	0.026207	0.00512	0.01933	0	0.053761
16	report	0	0.036302	0.004915	0	0	0	0.001864	0	0.037633	0	0	0	0.004072	0	0	0	0.006902	0	0.009865	0.011172	0	0
17	target	0.022603	0	0	0.127755	0.07895	0.185339	0.086934	0.118493	0	0.012397	0.011228	0	0.019379	0.040241	0.047141	0.002416	0	0.005088	0	0.010634	0	0.021125
18	datum	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	level	0.006587	0.015104	0.044988	0.006382	0.008628	0	0	0.006612	0.034447	0	0.007362	0.006501	0.001694	0.003909	0.007493	0.008449	0.015315	0.001483	0.003284	0.009297	0.055935	0.070183
20	clinical	0.04701	0.006737	0.14229	0.019928	0.015394	0.064585	0.031825	0.027528	0.03073	0.070905	0.041595	0.011599	0.033251	0.027898	0.020054	0.073483	0.008539	0.006614	0.045403	0.124405	0.049898	0.008238
21	mutation	0	0.019685	0.00533	0	0.029985	0.004493	0	0.005745	0	0.014126	0.003198	0.101677	0.022081	0	0.009766	0.005505	0	0.028989	0	0	0	0.016849
22	associate	0	0	0.009707	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	role	0.012021	0	0.007464	0.029119	0.01837	0.022021	0.073599	0.038214	0.002858	0.013187	0.114204	0.035595	0.01546	0.07134	0.082052	0.023128	0.010481	0.025709	0.013483	0.016967	0	0.033706
24	human	0.076248	0.026898	0.061907	0.028414	0.007682	0.147346	0.03729	0.049064	0.019519	0.016084	0.109254	0.202609	0.024137	0.125303	0.090074	0.043255	0.023864	0.03697	0.001462	0.066225	0.064034	0.049335
25	mouse	0.008816	0	0	0.008542	0.007698	0.018456	0.016607	0.020649	0	0.004835	0.006569	0.113113	0.004535	0.010463	0.015043	0.011307	0.020497	0.001985	0	0	0	0.024718
26	pathway	0	0	0.004987	0.093396	0.007014	0.012612	0.020806	0.158582	0	0.008811	0.050873	0	0.033057	0.066737	0.073102	0.007727	0	0.001808	0	0	0	0.018018
27	activity	0	0.007611	0	0.022512	0.023186	0.100755	0.040641	0.075522	0.009468	0.014563	0.014838	0.065518	0.008537	0.039394	0.166133	0.027671	0.017363	0.005978	0	0.018738	0	0.018612

Fig. 13: TF – IDF Score

4. Co-occurrence Matrix: Co-occurrence of the n-gram in papers



	046a56cc5	072016SC(	1-s2.0-S03	1002192.p	1002193.p	1002264.p	1002304.p	100249.pd	10278_20:	106.full.pd	10_MDRBI	11-The-Ag	1134.pdf	115.pdf	13 - Fraile	1412.full.g	1475-2891	1476-4598	1477-7525	1479-5876	1606.0571	1743-7075
Top 150 U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
patient	0	0	18	4	1	4	15	0	7	5	3	0	12	3	0	8	0	1	37	3	1	1
tumor	7	2	15	13	2	58	269	68	0	9	26	7	12	9	44	14	30	25	2	34	27	148
cell	22	4	9	97	106	98	175	141	1	7	92	29	39	34	128	57	10	55	33	13	0	127
treatment	0	2	14	53	38	46	25	55	2	30	29	1	33	10	3	39	7	6	44	2	0	2
include	0	1	1	1	2	8	7	0	4	0	4	1	3	0	1	3	3	2	3	3	1	10
gene	4	2	12	2	9	3	6	4	0	7	28	12	33	6	18	5	3	14	0	2	0	27
result	1	2	3	2	2	3	2	2	0	4	4	1	6	3	1	0	4	1	0	0	2	0
protein	2	0	0	29	15	32	19	23	0	1	72	2	15	11	53	13	5	14	0	1	0	15
increase	0	4	1	9	8	2	4	2	0	2	4	7	7	2	3	9	10	12	4	0	1	8
effect	0	2	0	6	9	16	0	17	0	7	7	2	3	5	0	8	44	2	7	0	0	29
tumour	0	0	0	0	23	5	10	1	0	1	3	30	41	29	5	34	1	0	3	1	0	7
disease	0	3	14	1	2	11	10	6	2	17	3	2	21	5	8	6	12	6	14	7	3	20
expression	46	0	2	56	2	14	38	33	0	7	74	7	14	27	29	8	2	17	3	2	0	28
report	0	4	1	0	0	0	1	0	10	0	0	0	2	0	0	0	3	0	5	1	0	0
target	3	0	0	35	24	47	49	47	0	3	4	0	10	9	11	1	0	3	0	1	0	10
datum	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
level	1	2	11	2	3	0	0	3	11	0	3	1	1	1	2	4	8	1	2	1	7	38
clinical	8	1	39	7	6	21	23	14	11	22	19	2	22	8	6	39	5	5	31	15	7	5
mutation	0	2	1	0	8	1	0	2	0	3	1	12	10	0	2	2	0	15	0	0	0	7
associate	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
role	2	0	2	10	7	7	52	19	1	4	51	6	10	20	24	12	6	19	9	2	0	20
human	13	4	17	10	3	48	27	25	7	5	50	35	16	36	27	23	14	28	1	8	9	30
mouse	1	0	0	2	2	4	8	7	0	1	2	13	2	2	3	4	8	1	0	0	0	10
pathway	0	0	1	24	2	3	11	59	0	2	17	0	16	14	16	3	0	1	0	0	0	8
activity	0	1	0	2	8	39	36	34	3	4	6	10	5	10	44	13	9	4	0	3	0	10

Fig. 14: Cooccurrence Matrix

5. LSA and LDA: Related words with topics are given with clustered of those documents are also there

a. LSA:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Topic	Topic_Name	Term1	Term2	Term3	Term4	Term5	Term6	Term7	Term8	Term9	Term10	
2	Topic 1	Diagnosis	cancer	breast	cell	cells	lung	res	prostate	stem	lines	patients	
3	Topic 2	Cancer Cell Behavior	cell	squamous	neck	carcinoma	lung	head	small	oxygen	species	reactive	
4	Topic 3	Expression Analysis	female	male	lung	rates	bronchus	incidence	small	rectum	prostate	cancers	
5	Topic 4	Tumor Characteristics	biomarkers	prev	epidemiol	oxygen	reactive	species	ros	risk	prostate	nitrogen	
6	Topic 5	Tumor Biology	cells	female	male	oxygen	species	reactive	stem	ros	colon	bronchus	
7	Topic 6	Cancer Therapy	lung	small	cells	cancer	stem	pancreatic	nsclc	prostate	patients	trial	
8	Topic 7	Immune Response	harvard	medical	school	boston	hospital	breast	reactive	oxygen	species	lung	
9	Topic 8	Metastasis & Migration	oncol	radiat	biol	phys	cells	stem	trial	lancet	harvard	medical	
10	Topic 9	Clinical Trials	reactive	oxygen	species	lung	breast	oncol	patients	radiat	ros	biol	
11	Topic 10	Gene Expression	prostate	res	biochem	biophys	commun	radiat	biol	phys	lines	oncol	
12	Topic 11	Side Effects	prostate	trial	lancet	phase	incidence	rates	united	cancer	patients	neck	
13	Topic 12	Survival & Prognosis	lancet	trial	phase	randomise	biochem	biophys	commun	controlled	res	ovarian	
14													

Fig. 15 : LSA

b. LDA:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Topic	Topic_Name	Term1	Term2	Term3	Term4	Term5	Term6	Term7	Term8	Term9	Term10	
2	Topic 1	Diagnosis	cancer	thyroid	pancreatic	erentiated	patients	dii¬€	advanced	intraducta	mucinous	papillary	
3	Topic 2	Cancer Cell Behavior	cancer	lung	cell	small	thyroid	clinical	harvard	medical	patients	sons	
4	Topic 3	Expression Analysis	cell	neck	squamous	carcinoma	head	cancer	phase	lancet	trial	oncol	
5	Topic 4	Tumor Characteristics	cancer	breast	cells	res	biol	oncol	radiat	women	reactive	oxygen	
6	Topic 5	Tumor Biology	copy	proofs	uncorrecte	prepublica	lymph	node	cancer	sentinel	biopsy	learning	
7	Topic 6	Cancer Therapy	female	male	cancer	lung	bronchus	cancers	colon	rates	death	rectum	
8	Topic 7	Immune Response	colon	bronchus	lung	rectum	leukemia	female	cancers	cancer	pharynx	male	
9	Topic 8	Metastasis & Migration	cancer	breast	cell	res	cells	lines	ovarian	human	patients	risk	
10	Topic 9	Clinical Trials	mutation	inactivatin	atlas	genome	cancer	tcga	death	mononucl	peripheral	cell	
11	Topic 10	Gene Expression	cancer	rates	incidence	breast	status	nation	death	united	report	states	
12	Topic 11	Side Effects	cancer	biomarker	epidemiol	prev	cells	prostate	stem	risk	breast	cell	
13	Topic 12	Survival & Prognosis	target	antitumor	ther	explor	cancer	india	cells	bar	scale	uteri	
14													

Fig. 16: LDA

c. Clustering:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Unnamed: 0	LSA_immune	LSA_surviv	LSA_tumo	LSA_clinic	LSA_cance	LSA_tumo	LSA_diagn	LSA_side e	LSA_meta	LSA_gene	LSA_cance	LSA_expre	DBSCAN_Cluster	KMeans_Cluster	Agglomerative_Cluster	
2	046a56cc9bcf920f6fa7b9850344db4(	0	0	0	0	0	0	0	0	0	0	0	1	0	2	0	
3	072016SCOe.pdf	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	
4	1-s2.0-S0304383519306135-am.pdf	0	0	0	0.036302	0	0	0.998645	0	0	0.037288	0	0	0	1	1	
5	1002192.pdf	0	0	0	0	0	0	0	0	0	0.291741	0.956497	0	0	4	0	
6	1002193.pdf	0	0	0	0.385319	0	0	0.321207	0	0	0	0.865075	0	0	4	0	
7	1002264.pdf	0.5027507	0	0	0.351308	0	0	0.041836	0	0	0	0.788717	0	0	4	0	
8	1002304.pdf	0.8050672	0	0	0.562558	0	0	0	0	0	0.165096	0.090214	0	0	3	3	
9	100249.pdf	0	0	0	0.393247	0	0	0.218544	0	0	0.134642	0.882874	0	0	4	0	
10	10278_2013_Article_9622.pdf	0	0	0	0.705357	0	0	0.587994	0	0	0	0.395897	0	0	3	4	
11	106.full.pdf	0	0	0	0.976547	0	0	0.135677	0	0	0.167178	0	0	0	3	4	
12	10_MDRBkSec.pdf	0	0	0	0	0	0.211865	0	0	0	0.770372	0.601365	0	0	0	2	
13	11-The-Age-of-Cancer.pdf	0	0	0	0	0	0	0	0	0	1	0	0	0	0	2	
14	1134.pdf	0	0	0	0.131817	0	0	0.98896	0	0	0.067699	0	0	0	1	1	
15	115.pdf	0.3136483	0	0	0.657506	0	0	0.182702	0	0	0.450242	0	0.482922	0	3	4	
16	13 - Fraile 2012 Oncogene.pdf	0	0	0	0.563029	0	0	0	0	0	0.57832	0.42135	0.413531	0	3	2	
17	1412.full.pdf	0.6752298	0	0	0.386044	0	0	0.035757	0	0	0.044059	0.625951	0	0	4	3	
18	1475-2891-3-19.pdf	0	0	0	0.122554	0	0	0.204326	0	0	0.125883	0.963008	0	0	4	0	
19	1476-4598-1-9.pdf	0	0	0	0	0	0	0.376008	0	0	0.926617	0	0	0	0	2	
20	1477-7525-7-102.pdf	0	0	0	0.567005	0	0	0.630217	0	0	0	0.530407	0	0	3	4	
21	1479-5876-10-1.pdf	0.7343268	0	0	0	0	0	0.427749	0	0	0.527063	0	0	0	0	3	
22	1606.05718v1.pdf	0	0	0	0	0	0	0	0	0	0	0	0	-1	2	0	
23	1743-7075-7-7.pdf	0	0	0	0	0	0	0	0	0	0.836252	0.548345	0	0	0	2	
24	1756-9966-30-87.pdf	0	0	0	0.826421	0	0	0	0	0	0.084887	0.556616	0	0	3	4	
25	1842_vancutsem2016.pdf	0	0	0	0.563915	0	0	0.822652	0	0	0.072404	0	0	0	1	1	
26	189219927.pdf	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	
27	18_2011_Article_672.pdf	0	0	0	0	0	0	0	0	0	0	0	0	-1	2	0	

Fig. 17: Clustering

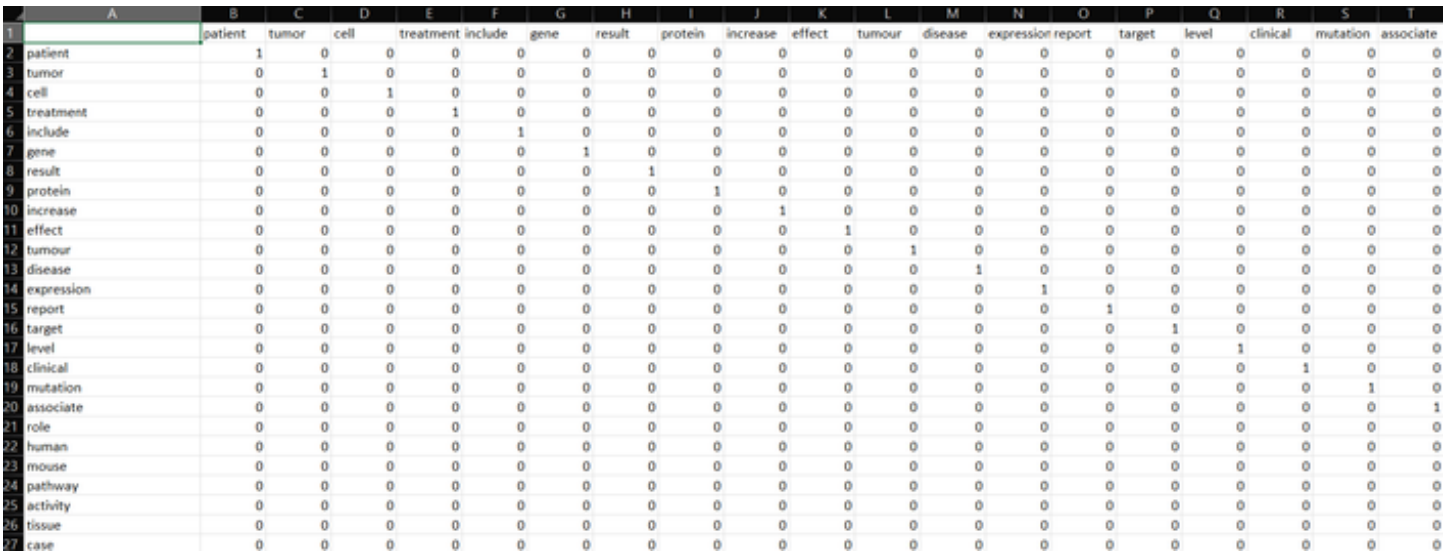
6. Jaccard Similarity: How familiar a n-gram with paper and similarity between n-gram with another n-gram

a. Jaccard Similarity of n-gram with papers:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1		patient	tumor	cell	treatment	include	gene	result	protein	increase	effect	tumour	disease	expression	report	target	level	clinical	mutation	associat
2	046a56cc9bcf920f6fa7b9850344db4647c5.pdf	0	7	22	0	0	4	1	2	0	0	0	0	46	0	3	1	8	0	
3	072016SCOe.pdf	0	2	4	2	1	2	2	0	4	2	0	3	0	4	0	2	1	2	
4	1-s2.0-S0304383519306135-am.pdf	18	15	9	14	1	12	3	0	1	0	0	14	2	1	0	11	39	1	
5	1002192.pdf	4	13	97	53	1	2	2	29	9	6	0	1	56	0	35	2	7	0	
6	1002193.pdf	1	2	106	38	2	9	2	15	8	9	23	2	2	0	24	3	6	8	
7	1002264.pdf	4	58	98	46	8	3	3	32	2	16	5	11	14	0	47	0	21	1	
8	1002304.pdf	15	269	175	25	7	6	2	19	4	0	10	10	38	1	49	0	23	0	
9	100249.pdf	0	68	141	55	0	4	2	23	2	17	1	6	33	0	47	3	14	2	
10	10278_2013_Article_9622.pdf	7	0	1	2	4	0	0	0	0	0	0	2	0	10	0	11	11	0	
11	106.full.pdf	5	9	7	30	0	7	4	1	2	7	1	17	7	0	3	0	22	3	
12	10_MDRBkSec.pdf	3	26	92	29	4	28	4	72	4	7	3	3	74	0	4	3	19	1	
13	11-The-Age-of-Cancer.pdf	0	7	29	1	1	12	1	2	7	2	30	2	7	0	0	1	2	12	
14	1134.pdf	12	12	39	33	3	33	6	15	7	3	41	21	14	2	10	1	22	10	
15	115.pdf	3	9	34	10	0	6	3	11	2	5	29	5	27	0	9	1	8	0	
16	13 - Fraile 2012 Oncogene.pdf	0	44	128	3	1	18	1	53	3	0	5	8	29	0	11	2	6	2	
17	1412.full.pdf	8	14	57	39	3	5	0	13	9	8	34	6	8	0	1	4	39	2	
18	1475-2891-3-19.pdf	0	30	10	7	3	3	4	5	10	44	1	12	2	3	0	8	5	0	
19	1476-4598-1-9.pdf	1	25	55	6	2	14	1	14	12	2	0	6	17	0	3	1	5	15	
20	1477-7525-7-102.pdf	37	2	33	44	3	0	0	0	4	7	3	14	3	5	0	2	31	0	
21	1479-5876-10-1.pdf	3	34	13	2	3	2	0	1	0	0	1	7	2	1	1	1	15	0	
22	1606.05718v1.pdf	1	27	0	0	1	0	2	0	1	0	0	3	0	0	0	7	7	0	
23	1743-7075-7-7.pdf	1	148	127	2	10	27	0	15	8	29	7	20	28	0	10	38	5	7	
24	1756-9966-30-87.pdf	3	3	85	31	14	46	8	23	3	3	28	11	28	4	17	2	37	1	
25	1842_vancutsem2016.pdf	2	1	7	45	1	8	0	2	3	0	28	27	2	1	2	0	21	6	
26	189219927.pdf	2	0	0	17	8	0	1	0	9	4	0	3	1	0	0	1	1	0	
27	18_2011_Article_672.pdf	10	41	87	3	1	2	1	1	0	0	8	0	18	0	4	2	2	0	

Fig. 18 : Jaccard Similarity of n-gram with paper

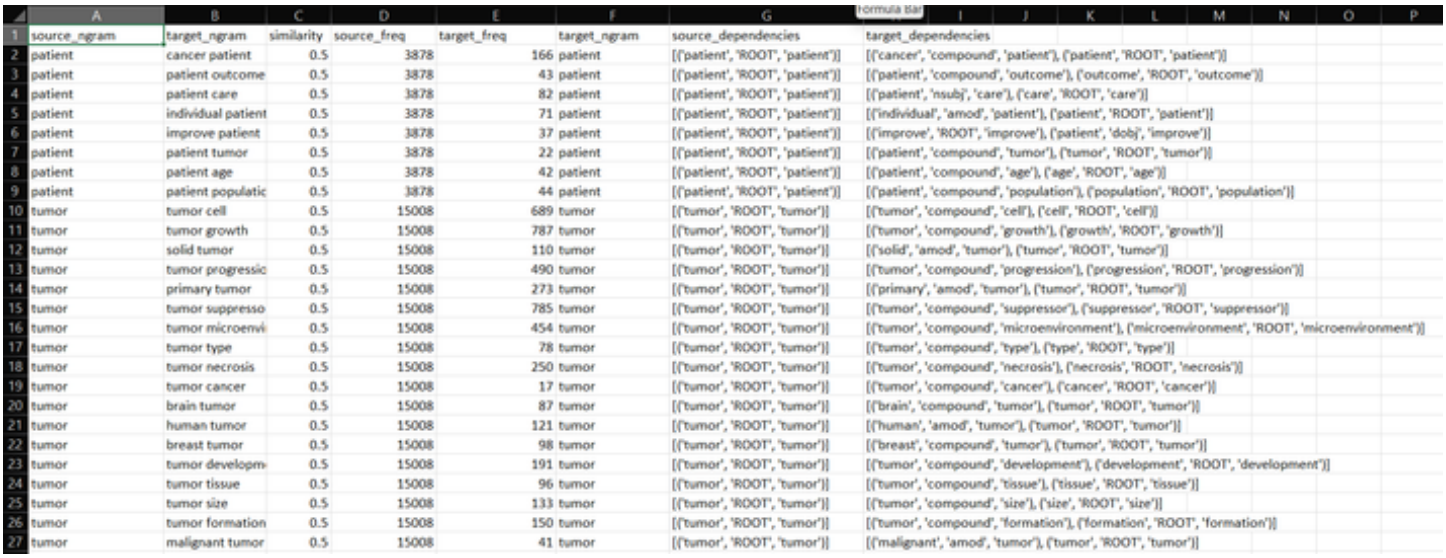
b. Jaccard Similarity of n-gram with n-gram:



	patient	tumor	cell	treatment	include	gene	result	protein	increase	effect	tumour	disease	expression	report	target	level	clinical	mutation	associate
patient	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
tumor	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
cell	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
treatment	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
include	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
gene	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
result	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
protein	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
increase	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
effect	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
tumour	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
disease	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
expression	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
report	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
target	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
level	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
clinical	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
mutation	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
associate	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
role	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
human	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
mouse	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
pathway	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
activity	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
tissue	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
case	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 19: Jaccard Similarity of n-gram with n-gram

7. Dependency Parsing with help of Jaccard Similarity: Dependency parsing between the entities



source_ngram	target_ngram	similarity	source_freq	target_freq	target_ngram	source_dependencies	target_dependencies
patient	cancer patient	0.5	3878	166	patient	[('patient', 'ROOT', 'patient')]	[('cancer', 'compound', 'patient'), ('patient', 'ROOT', 'patient')]
patient	patient outcome	0.5	3878	43	patient	[('patient', 'ROOT', 'patient')]	[('patient', 'compound', 'outcome'), ('outcome', 'ROOT', 'outcome')]
patient	patient care	0.5	3878	82	patient	[('patient', 'ROOT', 'patient')]	[('patient', 'nsubj', 'care'), ('care', 'ROOT', 'care')]
patient	individual patient	0.5	3878	71	patient	[('patient', 'ROOT', 'patient')]	[('individual', 'amod', 'patient'), ('patient', 'ROOT', 'patient')]
patient	improve patient	0.5	3878	37	patient	[('patient', 'ROOT', 'patient')]	[('improve', 'ROOT', 'improve'), ('patient', 'dobj', 'improve')]
patient	patient tumor	0.5	3878	22	patient	[('patient', 'ROOT', 'patient')]	[('patient', 'compound', 'tumor'), ('tumor', 'ROOT', 'tumor')]
patient	patient age	0.5	3878	42	patient	[('patient', 'ROOT', 'patient')]	[('patient', 'compound', 'age'), ('age', 'ROOT', 'age')]
patient	patient populatic	0.5	3878	44	patient	[('patient', 'ROOT', 'patient')]	[('patient', 'compound', 'population'), ('population', 'ROOT', 'population')]
tumor	tumor cell	0.5	15008	689	tumor	[('tumor', 'ROOT', 'tumor')]	[('tumor', 'compound', 'cell'), ('cell', 'ROOT', 'cell')]
tumor	tumor growth	0.5	15008	787	tumor	[('tumor', 'ROOT', 'tumor')]	[('tumor', 'compound', 'growth'), ('growth', 'ROOT', 'growth')]
tumor	solid tumor	0.5	15008	110	tumor	[('tumor', 'ROOT', 'tumor')]	[('solid', 'amod', 'tumor'), ('tumor', 'ROOT', 'tumor')]
tumor	tumor progressic	0.5	15008	490	tumor	[('tumor', 'ROOT', 'tumor')]	[('tumor', 'compound', 'progression'), ('progression', 'ROOT', 'progression')]
tumor	primary tumor	0.5	15008	273	tumor	[('tumor', 'ROOT', 'tumor')]	[('primary', 'amod', 'tumor'), ('tumor', 'ROOT', 'tumor')]
tumor	tumor suppresso	0.5	15008	785	tumor	[('tumor', 'ROOT', 'tumor')]	[('tumor', 'compound', 'suppressor'), ('suppressor', 'ROOT', 'suppressor')]
tumor	tumor microenvi	0.5	15008	454	tumor	[('tumor', 'ROOT', 'tumor')]	[('tumor', 'compound', 'microenvironment'), ('microenvironment', 'ROOT', 'microenvironment')]
tumor	tumor type	0.5	15008	78	tumor	[('tumor', 'ROOT', 'tumor')]	[('tumor', 'compound', 'type'), ('type', 'ROOT', 'type')]
tumor	tumor necrosis	0.5	15008	250	tumor	[('tumor', 'ROOT', 'tumor')]	[('tumor', 'compound', 'necrosis'), ('necrosis', 'ROOT', 'necrosis')]
tumor	tumor cancer	0.5	15008	17	tumor	[('tumor', 'ROOT', 'tumor')]	[('tumor', 'compound', 'cancer'), ('cancer', 'ROOT', 'cancer')]
tumor	brain tumor	0.5	15008	87	tumor	[('tumor', 'ROOT', 'tumor')]	[('brain', 'compound', 'tumor'), ('tumor', 'ROOT', 'tumor')]
tumor	human tumor	0.5	15008	121	tumor	[('tumor', 'ROOT', 'tumor')]	[('human', 'amod', 'tumor'), ('tumor', 'ROOT', 'tumor')]
tumor	breast tumor	0.5	15008	98	tumor	[('tumor', 'ROOT', 'tumor')]	[('breast', 'compound', 'tumor'), ('tumor', 'ROOT', 'tumor')]
tumor	tumor developm	0.5	15008	191	tumor	[('tumor', 'ROOT', 'tumor')]	[('tumor', 'compound', 'development'), ('development', 'ROOT', 'development')]
tumor	tumor tissue	0.5	15008	96	tumor	[('tumor', 'ROOT', 'tumor')]	[('tumor', 'compound', 'tissue'), ('tissue', 'ROOT', 'tissue')]
tumor	tumor size	0.5	15008	133	tumor	[('tumor', 'ROOT', 'tumor')]	[('tumor', 'compound', 'size'), ('size', 'ROOT', 'size')]
tumor	tumor formation	0.5	15008	150	tumor	[('tumor', 'ROOT', 'tumor')]	[('tumor', 'compound', 'formation'), ('formation', 'ROOT', 'formation')]
tumor	malignant tumor	0.5	15008	41	tumor	[('tumor', 'ROOT', 'tumor')]	[('malignant', 'amod', 'tumor'), ('tumor', 'ROOT', 'tumor')]

Fig.20: Dependency Parsing

1. NER (Name Entity Recognition): Entities are valued as per there category:

Row_Index	Entity	Label
2	tumor	CANCER
13	tumour	CANCER
81	p53	GENE_OR_GENE_PRODUCT
134	microrna	GENE_OR_GENE_PRODUCT
290	rna	GENE_OR_GENE_PRODUCT
378	egfr	GENE_OR_GENE_PRODUCT
430	pancreatic	CANCER
436	ros	GENE_OR_GENE_PRODUCT
449	histone	GENE_OR_GENE_PRODUCT
491	mrna	GENE_OR_GENE_PRODUCT
501	cyclin	GENE_OR_GENE_PRODUCT
516	vegf	GENE_OR_GENE_PRODUCT
573	crc	GENE_OR_GENE_PRODUCT
581	akt	GENE_OR_GENE_PRODUCT
611	ras	GENE_OR_GENE_PRODUCT
621	insulin	GENE_OR_GENE_PRODUCT
634	pten	GENE_OR_GENE_PRODUCT
672	myc	GENE_OR_GENE_PRODUCT
799	pi3k	GENE_OR_GENE_PRODUCT
820	caspase	GENE_OR_GENE_PRODUCT
881	lin	GENE_OR_GENE_PRODUCT
899	xenograft	CANCER
947	collagen	GENE_OR_GENE_PRODUCT
950	cxcr4	GENE_OR_GENE_PRODUCT
952	tp53	GENE_OR_GENE_PRODUCT
965	vimentin	GENE_OR_GENE_PRODUCT
1026	cin	GENE_OR_GENE_PRODUCT
1097	antitumor	CANCER
1105	wnt	GENE_OR_GENE_PRODUCT
1142	her2	GENE_OR_GENE_PRODUCT

Fig. 21: NER (1)

Name/Gene ID	Description	Aliases
ovarian cancer cancer re: 25	Biological_structure [Biomedical Entity]	o
nat clin pract oncol: 25	Diagnostic_procedure [Related Disease Grouping]	nat
receptor: 5765	Diagnostic_procedure [Related Disease Grouping]	receptor
cell: 5747	Diagnostic_procedure [Related Disease Grouping]	cell
activity: 5309	Diagnostic_procedure [Related Disease Grouping]	activity
human: 5445	Diagnostic_procedure [Related Disease Grouping]	human
gene: 7142	Diagnostic_procedure [Related Disease Grouping]	gene
disease: 6577	Diagnostic_procedure [Related Disease Grouping]	disease
protein: 6892	Diagnostic_procedure [Related Disease Grouping]	protein :
tumour: 6290	Sign_symptom [Related Disease Grouping]	tu
publish online ahead print: 25	Diagnostic_procedure [Related Disease Grouping]	print
basic fibroblast growth factor: 25	Diagnostic_procedure [Related Disease Grouping]	basic fibroblast growth factor :
mammary gland biol neoplasia: 25	Biological_structure [Biomedical Entity]	ma
injury cause death vol: 25	Disease_disorder [Disease]	injury cause death
nonsmall cell lung cancer: 25	Biological_structure [Biomedical Entity]	lung
kill cancer cell: 88	Diagnostic_procedure [Related Disease Grouping]	kill
lymphocytic leukemia chronic lymphocytic: 25	Disease_disorder [Disease]	lymphocytic leukemia
although: 4383	Diagnostic_procedure [Related Disease Grouping]	although
model: 4384	Diagnostic_procedure [Related Disease Grouping]	model
tissue: 4410	Diagnostic_procedure [Related Disease Grouping]	tissue
agent: 4335	Diagnostic_procedure [Related Disease Grouping]	agent
cell carcinoma clin cancer: 25	Disease_disorder [Disease]	cell carcinoma
use: 14307	Diagnostic_procedure [Related Disease Grouping]	use
drug: 8399	Diagnostic_procedure [Related Disease Grouping]	drug
study: 7194	Diagnostic_procedure [Related Disease Grouping]	study
cancer treatment pattern stage: 25	Diagnostic_procedure [Related Disease Grouping]	treatment pattern

Fig. 22: NER (2)

1. Knowledge Graph: Visualization of entities and their connection with each other

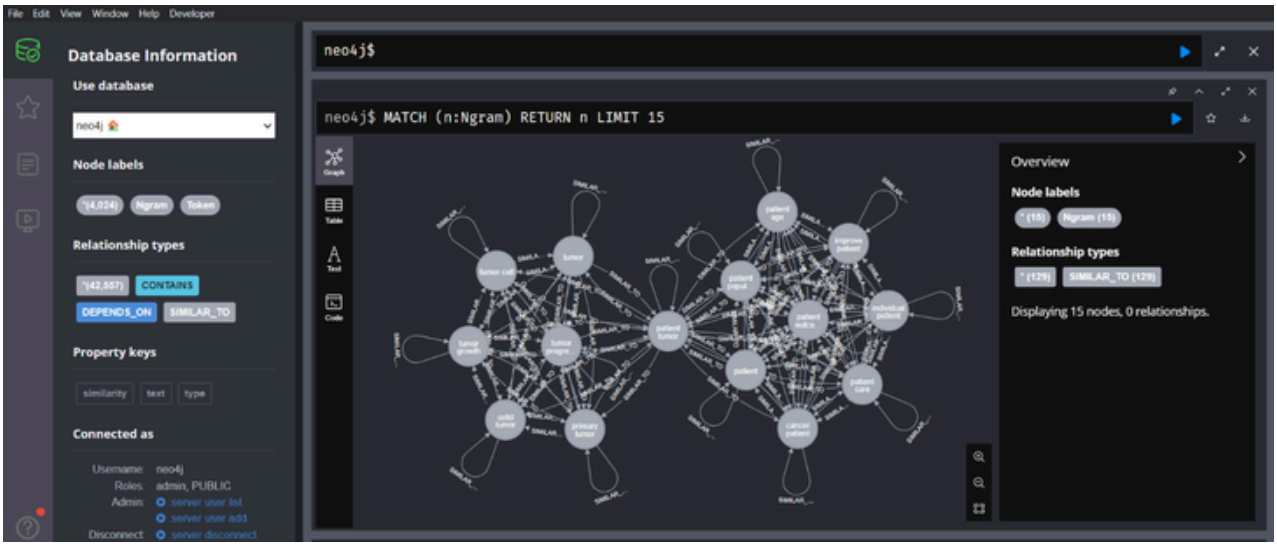


Fig. 23: Knowledge Graph

9. Nodes and path identification: Random keywords and their relation were discovered in knowledge graph

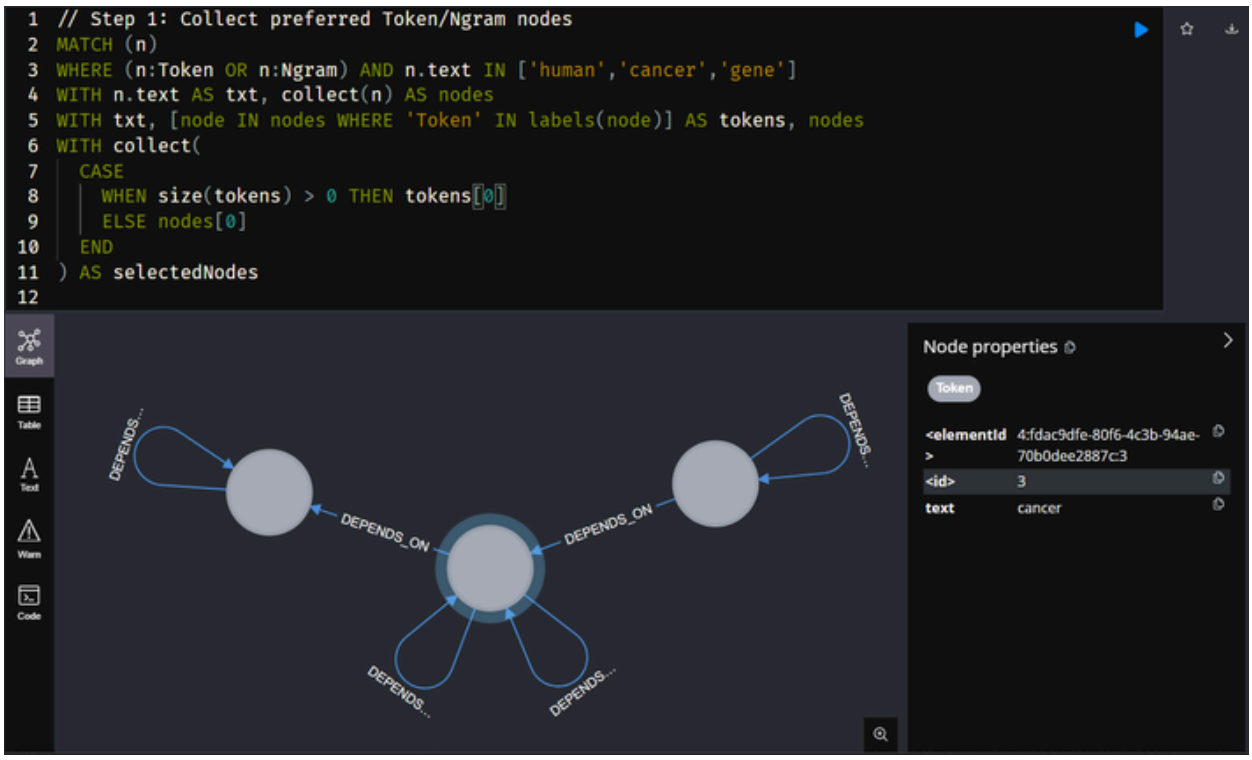


Fig. 24: Nodes and Path Identification in Neo4j

10. UI to get data from the Graph Database: Getting the data from the Graph Database about the term and result is the term related to that.

Fig. 25 : UI to get data from the Graph Database

The findings suggest how ScholarGraph dramatically decreases the workload involved in preparing an initial data gathering and review. Automated key term extraction and clustering into a knowledge graph facilitated the easy identifying the research topics and literature gaps for users.

In comparison to conventional approaches based on manually browsing PDFs, ScholarGraph improves the productivity of literature review by a lot according to the average time spent by testers. The system also promoted exploration by enabling users to visually trace the relationship between ideas and lists or keyword searches.

CHAPTER 6: CONCLUSION

6.1 Conclusion

In the current world, where there is new research papers published daily, it is simple for researchers to get lost or confused. They spend much time looking for the correct papers, reading them attentively, and attempting to grasp the central concepts. ScholarGraph was created to address this issue. It is a smart tool that assists researchers in saving time and grasping their subjects more effectively by automatically discovering, analysing, and visualizing research papers.

How ScholarGraph Fixes the Problem

- ScholarGraph functions as a helping assistent. It looks up research papers automatically for the user, gathers them, and reads them. But it does not end there. It proceeds further by:
- Extracting Key Information: It reads out the text of each paper, sanitizes it, and finds the most relevant words, phrases, and terms. For instance, if it is "Cancer," it will identify words like "Cancer", "Patient", "Treatment", and so on.
- Discovering Interconnections: ScholarGraph is more than gathering words. It comprehends how words are interlinked. For instance, it will notice that "Cancer" is linked with "Patient", and "Treatment". The Project employs innovative methods to look for these links.
- Constructing a Knowledge Graph: All the significant concepts and how they relate to each other are represented in a visual, interactive graph. It is a map of knowledge. Users can click on any concept to view how it relates to other ideas. This makes it much simpler to comprehend a difficult subject at a glance.

Key Achievements of ScholarGraph :

- Automated Retrieval of Research Papers: No longer must researchers search for papers by hand. ScholarGraph does it for us.
- Intelligent Text Analysis: It doesn't simply read the papers that understands them. Through Natural Language Processing (NLP), it recognizes key words, names, and even broad themes of the papers.
- Easy Understanding with Knowledge Graph: Rather than reading each paper separately, researchers can observe the key points and how they relate to one another at a glance using the interactive graph.
- Interactive and User-Friendly: The graph is not a photograph—it is completely interactive. Users can click, search, and navigate the graph to find out more.

How ScholarGraph Makes Research Simpler

Prior to ScholarGraph, a researcher would need to:

- Spend hours searching for papers online.
- Download and read each one manually.
- Take notes in order to recall the key points.
- Attempt to work out how the various ideas in the papers relate to one another.
- With ScholarGraph, all this is quicker and easier:
- The system automatically discovers and downloads the papers.
- It reads the papers, extracts the key ideas, and displays them in a tidy, linked graph.
- Researchers can navigate the graph to get a quick grasp of their subject without getting lost.

6.2 Limitations

ScholarGraph is a great tool for searching research papers and learning about complicated subjects, but it is not flawless. Like any system, ScholarGraph has some limitations. Knowing these limitations is significant because it informs us where ScholarGraph can be enhanced in the future. The following are the primary limitations of ScholarGraph:

- Limited Language Support

ScholarGraph works best with English-language research papers at the moment.

Why it is a problem: If a user wants to explore research papers in other languages (such as Chinese, Spanish, or French), ScholarGraph is unable to process them.

- Limited Data Sources

ScholarGraph harvests research papers primarily from Google Scholar. Although Google Scholar is widely used, there are numerous other relevant research databases (such as PubMed for medical literature, IEEE Xplore for engineering, and ScienceDirect for scientific publications) that ScholarGraph has no access to.

- Dependency on PDF Quality

ScholarGraph reads and processes research papers in PDF format. But the quality of these PDFs can be different. Some PDFs can be poorly scanned or include images of text rather than actual text. In those situations, ScholarGraph cannot read or process them correctly.

- Static Knowledge Graph

The knowledge graph ScholarGraph builds is on a set of fixed research papers. Once the graph is built, it does not update automatically when new papers are published. Users who wish to maintain the latest research need to update by hand new papers and recreate the graph.

- Computational Constraints

ScholarGraph employs sophisticated text processing methods (such as NLP, LSA, LDA), which are computationally expensive and time-intensive, particularly with thousands of research papers. A user attempting to process thousands of papers simultaneously can make the system slow or even crash.

6.3 Major Achievements

ScholarGraph, in spite of its limitations, has recorded a number of key accomplishments that make it a worthwhile resource for researchers.:

1. Automated Research Paper Retrieval

ScholarGraph is able to automatically find, gather, and download research papers from Google Scholar using a user's topic. It saves users hours of time which would otherwise be wasted manually browsing and downloading papers.

2. Advanced Text Processing with NLP

ScholarGraph employs Natural Language Processing (NLP) to clean, analyze, and comprehend the text of every research paper. It is able to extract significant words, phrases, and even distinguish the names of diseases, approaches, and research institutions.

4. Dynamic Knowledge Graph Generation

ScholarGraph converts the primary concepts and their inter-connections into an interactive Knowledge Graph. This visual map makes it easy for users to see how different concepts are connected without reading every paper.

5. Interactive User-Friendly Visualization

The Knowledge Graph is not just a static picture that is fully interactive. Users can click on any concept to learn more, zoom in and out, and view the relationships between ideas.

6. Scalable and Extensible System

ScholarGraph is modularly designed, meaning that it can be readily updated or expanded in the future.It can be enhanced to cover more languages, gather papers from more places, and even recognize more complicated relations between concepts.

6.4 Future Scope

The most promising ideas for improving ScholarGraph and making it even more helpful:

1. Multi-Language Support

Expand ScholarGraph to comprehend research papers in languages other than English, e.g., Chinese, Spanish, French, or German. This would allow ScholarGraph to be of use to a worldwide audience. To do that we can include support for multilingual Natural Language Processing (NLP) and language translation.

2. Increased Support for More Data Sources

Integrate ScholarGraph with additional research databases such as PubMed (for biomedical papers), IEEE Xplore (for engineering papers), and ScienceDirect (for scientific papers).Due to that more users would have access to more research papers. For that we can use APIs for such databases just like we used them for Google Scholar.

3. Updating Graphs in Real-Time

Let the Knowledge Graph update automatically every time new research papers are included. This would keep the graph fresh and updated at all times with the newest research. To proceed we can include a background process which periodically looks for new documents on the user's subject and refreshes the graph.

4. Better Visualization Choices

Let users personalize the Knowledge Graph with various layouts, coloring, and filtering.Users would be able to concentrate on the areas of the graph most relevant to them.To proceed with that we could employ powerful graph visualization libraries such as D3.js or Cytoscape.js.

5. Automated Summarization

Include a feature that can provide fast summaries of each research paper. Users would be able to comprehend the key ideas of each paper without having to read the entire text.We can leverage advanced text summarization methods (such as transformer models) to create understandable, readable summaries.

References

[1] C. Zhang, D. Patel, and M. Brown, "Natural Language Processing in Literature Reviews: Techniques and Applications," *Springer Journal of Digital Libraries*, vol. 10, no. 2, pp. 145-167, 2021.

[2] Y. Nakamura, T. Saito, and L. Hernandez, "Concept Mapping and Knowledge Graphs for Research Paper Analysis," *IEEE Access*, vol. 11, pp. 98834-98850, 2023.

[3] Beltagy, I., Lo, K., & Cohan, A. (2018). Entity Recognition in Scientific Literature using BERT-based NER and SpaCy. In *Proceedings of the 2018 Conference on Neural Information Processing Systems (NeurIPS)*, pp. 3456-3465.

[4] Sinha, A., Shen, Z., Song, Y., Ma, H., Eide, D., Hsu, B., & Wang, K. (2020). An Overview of Microsoft Academic Graph (MAG). *Journal of Data Science and Information Systems*, 7(2), 56-72.

[5] Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., & Su, Z. (2019). AMiner: A Large-scale Academic Knowledge Graph. *Science China Information Sciences*, 62(2), 209-223.

[6] Liu, Z., Wang, X., & Li, H. (2023). Dynamic Knowledge Graphs for Research: Real-time Updates Using AI Models. *Journal of Artificial Intelligence and Research*, 45(3), 178-195.

[7] Lo, K., Wang, L., Neumann, M., Kinney, R., & Weld, D. S. (2022). Semantic Scholar Open Research Corpus: Large-Scale AI Training for Scholarly Papers. *Transactions on Machine Learning and Data Mining*, 10(4), 255-270.

[8] Peroni, S., & Shotton, D. (2021). Scigraph: Springer Nature's Scholarly Knowledge Graph. In *Proceedings of the 2021 International Semantic Web Conference (ISWC)*, pp. 512-527.

[9] Wang, L. L., Lo, K., Chandrasekhar, Y., Reas, R., Yang, J., Eide, D., & Kohlmeier, S. (2020). CORD-19: The COVID-19 Open Research Dataset Knowledge Graph. *Nature Digital Medicine*, 3(5), 102-115.

[10] Gao, J., Wu, Y., & Zhang, X. (2019). Constructing the ArXiv Knowledge Graph Using Machine Learning Models. *IEEE Transactions on Knowledge and Data Engineering*, 31(7), 1448-1461.

[11] Zhang, T., Liu, H., & Xu, W. (2024). Deep Learning for Scientific Knowledge Graphs: Automated AI-driven Knowledge Extraction. *Journal of Computational Intelligence and AI Research*, 19(1), 12-27.