Personalized Context-based Job Recommendation System based on Knowledge Graph

Shivam Naveen Kumar Mishra - 1186614

Rutvik Patel - 1191668

Smit Mirani-1188571

Nikhil Ailani - 119445

Ziyang Ren - 1145033

Department of Computer Science

Lakehead University

Dec 5, 2023

**Abstract**

In the era of digital job hunting, our project revolutionizes the process using knowledge graphs and graph embedding. Knowledge graphs, representing complex job market relationships, offer transparency and accuracy, surpassing traditional methods. Integrating advanced machine learning models ensures adaptability to dynamic job trends. The project not only optimizes recommendations but also prioritizes transparency, adaptability, and user-centricity. Beyond individual job seekers, the impact extends to talent acquisition and workforce planning. Key research questions address knowledge graph construction, optimal graph embedding for job postings, and predictive capabilities of machine learning models. This document provides background, reviews related work, introduces the dataset, details methodology, explores applications, presents preliminary findings, and outlines the future trajectory.

# Introduction

The quest for the perfect job in today’s digital era is fraught with challenges. Despite numerous online platforms and advanced recruitment technologies, the journey from job seeking to job finding remains arduous for many. This is not only a problem for people, but it also reflects a systemic inefficiency that affects organizations all around the world. In response to this, our project emerges as an innovative approach to transform the job hunting landscape. Our methodology, which is based on knowledge graphs and graph embedding principles, aims to bring a new level of speed, customization, and clarity to the process of connecting job seekers with suitable prospects.

Knowledge graphs are critical components of our system, providing a complex and integrated representation of the employment market. These graphs, which intricately map out relationships between various entities like skills, roles, and qualifications, provide a more comprehensive approach to job matching compared to traditional keyword-based systems. In this complex web, each node and edge carry a meaning that resonates with real-world professional scenarios, thus enabling a more accurate and integrated job matching process.[1] And one intriguing aspect of knowledge graphs is that they are explainable, which means there is potential to enhance user understanding of job recommendations. This feature of knowledge graphs may help users understand why specific professions are suggested to them. For example, our system could illuminate how a user's specific set of skills and experiences matches the requirements of a recommended job. This kind of transparency may encourage trust in the system while also providing users with useful insights for career planning and skill development.[2]

Incorporating machine learning into this framework is a strategic decision. Advanced machine learning models like Graph Neural Networks (GNNs) or Graph Convolutional Networks (GCNs) are especially well-suited for interpreting the complexity of graph-structured data, allowing our system to capture and learn from the job market's dynamic trends. It is critical for these models to be able to adapt and change in response to the ever-changing landscape of employment prospects, ensuring that our recommendations stay relevant and accurate over time.[3] Our focus is on not only optimizing the recommendation process but also embedding our commitment to transparency, adaptability, and user-centricity. This project is not just about deploying advanced technology; it’s about creating a solution that understands and responds to the unique needs and challenges of job seekers and employers alike.

The potential of our technology extends beyond individual job seekers to the larger realms of talent acquisition and human resource management. Employers will profit from a more efficient and exact matching process, which will save time and money currently spent on finding the perfect candidate. Furthermore, the data gained from our technology could be used to inform workforce development initiatives, assisting organizations in identifying new skill trends and talent needs.

This project represents a significant leap forward in the realm of job matching systems. By using the power of knowledge graphs and machine learning, we are not just simplifying but also changing the job search process. Our project shows the potential of technology to create more meaningful connections in the job market, empowering both job seekers and employers with the tools and insights to make smarter, more informed decisions.

The primary questions we aim to address in this research include:

How can we effectively construct a knowledge graph from job posting data that accurately represents the current job market landscape?

What are the optimal graph embedding techniques suited for this type of data, considering the specific characteristics of job postings?

How can machine learning models, using these embedded graph representations, predict and analyze trends in the job market?

This introduction sets the stage for a detailed exploration of these questions. The document is structured to first provide necessary background information on knowledge graphs and graph embedding techniques (Section 2), followed by a review of related prior work in this domain (Section 3). And we introduce the dataset we use, including the analyzing and preprocessing (Section 4). We then delve into the specifics of our methodology, including the construction of the knowledge graph from job posting data and the application of graph embedding techniques, and evaluations as well (Section 5). Subsequent sections discussed the expected result (Section 6), and outlined the future trajectory of our work, including expected milestones and contributions (Sections 7 to 10). This comprehensive approach aims not only to contribute to the academic field but also to offer practical insights for industry professionals and policymakers in the labor market.

# Background

The knowledge graph (KG) is a graphical depiction of real-world knowledge, where entities are depicted as nodes, and the connections between them are represented by edges, indicating semantic relationships.

A KG relies on "SPO" triples, representing subject entities, predicates (relations), and object entities. Domain KGs define specific domain entities and relationships, involving ontology creation, rule design, relation extraction, and semantic data storage.

KGs have transformed conventional information retrieval [4] and elucidate semantic and attribute connections among concepts, aiding in reasoning about them. To utilize machine learning algorithms or knowledge graphs the embedding techniques are used. The primary goal of knowledge graph embedding is to convert symbolic data, often represented as triples (subject, predicate, object), into distributed and continuous representations that can be used in various machine learning tasks.[5]

KGE models embed entities (E) and relations (R) into low-dimensional real or complex vector spaces while preserving the knowledge graph structure and semantic information [6]. These models are valuable for various prediction and graph analysis tasks [7] and fall into three categories: translational distance-based (e.g., TransE, TransH, TransR), semantic matching-based (e.g., RESCAL, DistMult, ComplEx), and neural network-based models.[8] Translational models measure similarity by distance-based scoring functions, where relations shift entities to calculate embeddings. Semantic matching models employ similarity-based scoring functions. Translational models translate head entities to tail entities through relations, using scoring functions to assess triple correctness in the embedding space.

Two primary recommendation techniques, collaborative filtering (CF) and content-based filtering (CBF), have traditionally been employed[9]. CF suggests items for a user by predicting the ranking score for items based on user profiles, shared preferences, and historical interactions. Extensive research has been dedicated to enhancing CF-based recommender systems by converting user and item side information into feature vectors for rating prediction. Matrix factorization (MF) is a prominent CF technique that maps users and items into a common latent space using latent feature vectors, modeling interactions as the inner product of these vectors. Nonetheless, CF methods often encounter data sparsity and cold-start issues. To address these challenges, numerous studies have aimed to enhance MF and tackle the cold-start problem by incorporating content (side) features to represent users and items within the latent space. Consequently, the integration of context information is crucial for mitigating these challenges and improving recommender systems.

When it comes to modeling, there are two types of learning models: one is independent learning and another is joint learning. The two-stage learning method also known as independent learning method involves a simple approach where we first learn embeddings (representations) for entities in a Knowledge Graph (KG) without considering user interactions. These embeddings are then treated as additional features and used in a separate recommendation system. The advantage is that we can learn KG embeddings without needing interaction data, making it computationally efficient, and since KGs are often stable, we don't need frequent updates. However, embeddings optimized for KGs may not be the best fit for recommendation tasks due to the different nature of these applications. Another method is to simultaneously train both the graph embedding module and the recommendation module in an end-to-end fashion. By doing this, the recommendation module actively influences the learning process of features in the graph embedding module. This integrated training approach ensures that the recommendation system and the graph embeddings are optimized together, allowing them to mutually inform and enhance each other during the training process.

While referring to different research papers we found some datasets which are not adequate for job recommendation. Which leads to one of the major issues faced in this field. This is called “sparity”. In the context of knowledge graphs, "sparsity" refers to the property of having a large proportion of missing or unspecified information within the graph.3

The paper**[10]** proposes a technique to reduce the sparsity from knowledge graph by constructing a personal knowledge graph of each employee and the fusing it with a semantic vector of staff entities(TransHR). Amalgamation of collaborative filtering with semantic similarity leads to better precision of the knowledge graph.

The workflow of the algorithm is as follows. The initial step involves constructing a comprehensive knowledge graph encompassing company staff information. Subsequently, this knowledge map is seamlessly integrated as auxiliary information into a collaborative filtering algorithm. The fusion of these two components gives rise to an enhanced collaborative filtering algorithm, specifically tailored to facilitate accurate and efficient personnel-to-post matching during workforce changes. Experimental results demonstrate the algorithm's capability to overcome challenges associated with cold starts and data sparsity, typical of traditional methods. The proposed algorithm exhibits improved accuracy and recommendation quality, showcasing its effectiveness in optimizing personnel recommendations in corporate environments.

We can infer that by integrating an auxiliary information component in our job recommendation model, we can greatly reduce the sparsity in the model. However, one major limitation in the paper was it did not consider multi-step relationships that entities can have which may restrict the scalability of the model.

A knowledge graph is particularly well-suited for building job recommendation systems due to its ability to represent complex relationships and dependencies between various entities in the job domain.

The paper **[11]** job-seekers have a system that recommends jobs based on their profile and explains why each job has been recommended. The proposed system uses a knowledge graph structure to model job-postings and user profiles in one structure. The system mines graph relations between job-seekers and job-postings through natural language processing of textual content. Then, a human-readable explanation is generated for each recommendation based on the graph structure and a customized named entity classifier. This way, job-seekers can better evaluate the suitability of a recommended job.

# Related Prior Work

Graphs, as a versatile representation model, can be categorized into homogeneous and heterogeneous types based on the nature of entities and relationships they encapsulate. In homogeneous graphs, nodes and edges are of the same type, providing simplicity and uniformity, making them suitable for scenarios like social network analysis. Conversely, heterogeneous graphs embrace diverse node and edge types, allowing for rich representations and comprehensive modeling of intricate relationships.

Aggarwal et al. [12] introduced a homogeneous graph-based approach, where users serve as nodes and predictability measures between users form the edges. Unlike traditional nearest neighbor search methods, this approach navigates paths from the starting user to connected nodes in order to locate a user who has rated the item in question. However, akin to other user-based collaborative filtering approaches, this method encounters scalability issues as the user population grows significantly.

In the next paper, Wang et al.[13] proposed IHGCN, an improved semi-supervised heterogeneous graph convolutional network model for job recommendation. This paper focuses on the people who are looking for entry level jobs with minimal experience.The model used in this paper IHGCN (Inductive Heterogeneous Graph Convolutional Network) is novel and has 3 different layers.

In the initial layer, N nodes representing early job seekers' resumes are encoded into an N × D-dimensional matrix X. The graph structure is represented by an N × N-dimensional adjacency matrix B. Subsequent hidden layers apply a non-linear function f(H(l),B), where H(l) denotes the hidden layer's output, W(l) is the weight matrix, and RELU(⋅) is the rectified linear unit activation function. The first hidden layer utilizes X as input ( H(0)=X), while the final hidden layer produces graph-level outputs H(L)=Z). To enforce self-loops in the graph, the identity matrix is added to B( =B+I). Normalization involves dividing each row of B by the sum of its rows, resulting in D−1B. Symmetric normalization is then performed to obtain =D−1/2​BD−1/2​. The IHGCN model's graph convolutional layer update is defined by

H(l+1)=σ(D−1/2​D−1/2​H(l))W(l))+H(l).

Here, σ represents the activation function, typically the rectified linear unit (ReLU).

In summary, the IHGCN model integrates self-loops, normalization, and graph convolutional layers to efficiently process information within the resume graph. These design elements enable the model to capture non-linear relationships and generate graph-level outputs.The paper demonstrate that IHGCN outperforms the baselines by around 10%. The study presents several positive aspects, including the development of a specialized labeling method for early jobseekers' resumes, demonstrating a commitment to tailored data preparation. Framing the job recommendation as a multi-class node classification problem and leveraging the IHGCN model to capture essential aspects of education and skills showcases a thoughtful and task-specific modeling approach. However, the the paper lacks specificity regarding the evaluation metrics used, and the mention of future exploration lacks detailed plans, impacting the clarity of the research direction.

Jia et al. [14], in their paper focuses on A.I (Artificial Intelligence) jobs in the market and implements Knowledge graph using Long Short-term Memory Network (LSTM) algorithm. The paper focused on differentiating two type of jobs which are General and specific based on their role. The paper also does a comparative analysis of previous paper and comes to a conclusion that the dataset for job recommendation is limited. Previous papers referred uses static based approach to find the relationship between job and skills. The following paper proposes an algorithm where it composes of 4 steps.The model is called N-gram model. N-gram model where N is the sequence of (entity-relationship-entity) is use to build probabilistic model which can predict, identify common patterns and link prediction based on observed behavior.

Working of the algorithm - This paper undertook the construction of an artificial intelligence job-skills knowledge graph, employing a diverse set of methodologies. Skill Named Entity Recognition (NER) was executed using LSTM models, while job entity linking harnessed the power of bi-gram-related algorithms. The outcomes of this information extraction efforts were systematically organized and stored as RDF triples, providing a structured representation for relationships within the knowledge graph. Noteworthy insights emerged from our knowledge graph analysis, which focused on delineating job and skill perspectives. This analysis revealed a categorization of jobs into fundamental algorithm roles and more specialized positions within specific research fields. This granular categorization underscored the nuanced skill requirements associated with diverse positions, emphasizing the necessity for job seekers to tailor their skill sets according to the specific demands of their targeted roles. This structural flow of the paper and the defined model is useful for building a recommendation model which we intend to develop. The highlight of the paper is the methodology employed in constructing the job-skills knowledge graph which holds promises for adaptation to diverse industries, presenting a novel and impactful approach for understanding the intricate relationships between jobs and skills. However, there is one drawback that we observe. This paper builds a knowledge graph using two major nodes that are jobs and skills, however in the real world it can get more complex than that. We plan to integrate additional entities and relationships, such as salary, company size, and geographic location. This expansion will afford a more comprehensive and nuanced understanding of the job-skills landscape. Additionally, we plan to extend our study to integrate job seekers' resumes, thereby laying the foundation for the development of an intelligent resume screening and recommendation system based on the knowledge graph



Fig.1

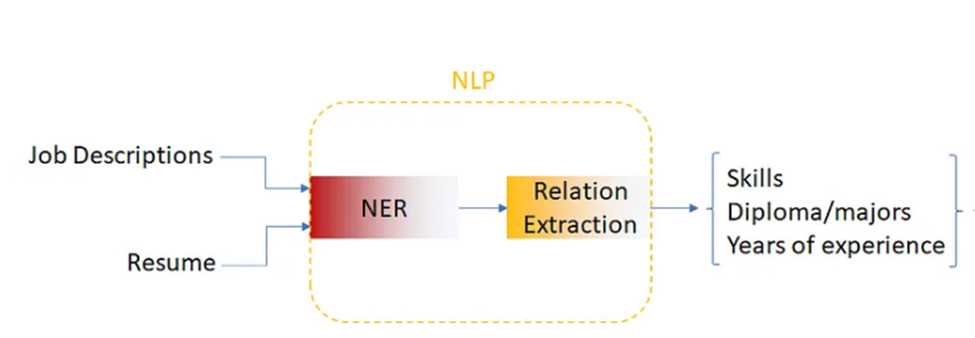
Elsafty et al. [15] conducted a study focusing on leveraging dense vector representations to improve the effectiveness of a large-scale job recommendation system. Their investigation involved the ranking of German-based job postings by measuring their similarities, leading to the generation of essential document embeddings. In a separate work, Wang et al. [19] proposed an explainable approach for document recommendations that relies on knowledge graphs. Their method incorporates higher-order relations among connected users and items within the knowledge graph to enhance the recommendation process. Named Knowledge Graph Attention Network (KGAT), their approach employs attention mechanisms to discern the significance of a node's neighbors in the graph, contributing to more refined and context-aware recommendations.

# Dataset

In the domain of job recommendation systems, the importance of comprehensive and well-curated datasets is paramount. These datasets, which include detailed job postings and extensive job seeker profiles, form the cornerstone for building multi-dimensional knowledge graphs. Through rigorous extraction and standardization of critical entities like skills, experience, and educational backgrounds, these datasets facilitate the visualization of intricate job market relationships and power the intelligent matching of candidates with job opportunities. The depth and precision of these datasets critically enhance the knowledge graph's effectiveness, directly impacting the system's capability to provide informed recommendations.

The Job Posting Dataset is a robust collection of diverse professional opportunities, rich in details such as Job Titles, Required Qualifications, Preferred Skills, and Salary Ranges. It spans various sectors, offering a comprehensive view of industry-specific roles and requirements. Key entities like Job ID, Job Description, and Employment Type undergo meticulous parsing and normalization, laying a structured foundation for in-depth data analyses. Employing advanced data mining and NLP techniques, the dataset is refined for accuracy, feeding into algorithms for market trend analysis, demand prediction, and skill gap identification.

The Job Seekers Dataset, as exemplified by the extensive 70k Job Applicants Data from Kaggle, is a critical component in creating a holistic knowledge graph. This dataset assembles a diverse range of candidate profiles, each a complex mix of Skills, Professional Experiences, Educational Backgrounds, Certifications, and Employability Scores. Essential entities like Candidate ID, Professional Achievements, Academic Qualifications, and Geographical Preferences are systematically extracted and standardized using advanced NLP and machine learning methodologies. This structured transformation is key to assimilating varied candidate data into the knowledge graph, allowing for a nuanced representation of the job market. The granularity of this dataset lays a fertile ground for sophisticated predictive analytics, significantly boosting the job recommendation system's precision in aligning candidate profiles with relevant jobs.



## 4.1 Data Transformation and Normalization Using Machine Learning and NLP for Job Posting Dataset

In our quest to make the job posting dataset more structured and analysis-ready, we've accomplished a series of data transformation and normalization tasks. Through the application of machine learning algorithms and Natural Language Processing (NLP) techniques, we've achieved the following:

### Job Titles Standardization:

* **Objective**: Standardize job titles to their base forms, eliminating variations.
* **Approach**: Utilized text classification models employing algorithms like Support Vector Machines (SVM) or Random Forest to categorize job titles into standardized categories.
* **Explanation**: Job titles often come in various forms, including seniority levels, abbreviations, and variations. For instance, we might encounter titles like "Senior Java Developer," "Java Developer," and "Java Dev." To make sense of these titles, we built a machine learning model that understands the relationships between different titles and classifies them into standardized categories. As a result, "Senior Java Developer" would be standardized as "Java Developer."

### Custom Tokenization:

* **Objective**: Develop custom tokenizers tailored to the nuances of job descriptions.
* **Approach**: Built custom tokenization functions using NLP libraries like spaCy, handling domain-specific terms, abbreviations, and special characters.
* **Explanation**: Job descriptions can contain domain-specific jargon, abbreviations, and unique terms that standard tokenizers might not handle effectively. To address this, we crafted custom tokenization functions using NLP libraries like spaCy. These custom tokenizers are fine-tuned to recognize domain-specific terms and ensure that important details in job descriptions are accurately captured.

### Text Featurization:

* **Objective**: Convert text data into numerical vectors suitable for machine learning models.
* **Approach**: Employed TF-IDF vectorization, a widely-used text featurization technique, using libraries like scikit-learn.
* **Explanation**: Machine learning models require numerical input, but job descriptions are textual. To bridge this gap, we employed the TF-IDF (Term Frequency-Inverse Document Frequency) technique. It assigns numerical values to words based on their importance in a document relative to a corpus. This way, job descriptions are transformed into numerical vectors that can be used for model training and analysis.

### Relevant Column Extraction from Job Descriptions:

* **Objective**: Extract key information from job descriptions to create a structured dataset.
* **Approach**: Utilize NLP techniques to identify and extract relevant information from unstructured job descriptions.
* **Explanation**: Job descriptions often contain essential information, but it's typically unstructured text. To make this data usable for analysis and knowledge graph creation, we need to extract specific details from the job descriptions. This includes information like required skills, experience criteria, and major requirements.

The extraction of relevant columns from job descriptions involves natural language processing techniques to identify and capture specific details, enriching the dataset with valuable information for job matching and recommendation systems.

## 4.2 Data Transformation and Normalization for the Job Seekers Dataset

### Candidate ID Extraction

* **Objective**: To establish unique identifiers for each job seeker profile.
* **Methodology**: A deterministic hashing algorithm is implemented to generate distinct Candidate IDs. For instance, a SHA-256 hash function is applied to a combination of personal data points (like name and date of birth) to ensure uniqueness.
* **Outcome**: This results in a reliable indexing system that enhances the efficiency of data retrieval and tracking across the dataset.

### Name Retrieval

* **Objective**: To accurately capture the full names of job seekers.
* **Technique**: Regex-based parsing, in conjunction with NLP models, is employed to extract names from unstructured data. This includes handling various name formats and cultural variations.
* **Outcome**: High accuracy in extracting names, preserving the integrity and authenticity of candidate identities.

### Skills Unveiling

* **Objective**: To extract and catalog job seekers' skills.
* **Approach**: Named Entity Recognition (NER) models from libraries like spaCy are used to identify and classify skill-related entities from resume texts.
* **Outcome**: Comprehensive skill sets are compiled for each candidate, providing a robust base for skill-based matching in the recommendation system.

### Experience Calculation

* **Objective**: To compute the total years of professional experience.
* **Method**: Temporal analysis algorithms parse work history data, translating qualitative experience narratives into a unified numerical format.
* **Outcome**: Experience is standardized across profiles, offering a quantifiable metric for experience-related matching criteria.

### Major Discovery

* **Objective**: To identify educational backgrounds and fields of specialization.
* **Technique**: Text classification algorithms discern and categorize academic majors from educational histories.
* **Outcome**: Profiles are enhanced with accurately classified educational qualifications, adding depth to the knowledge graph.

### Location Revelation

* **Objective**: To determine current and preferred locations of job seekers.
* **Strategy**: Geocoding and location parsing algorithms extract and standardize geographical data from unstructured location references.
* **Outcome**: Insightful geographic preferences are obtained, crucial for location-based job recommendations.

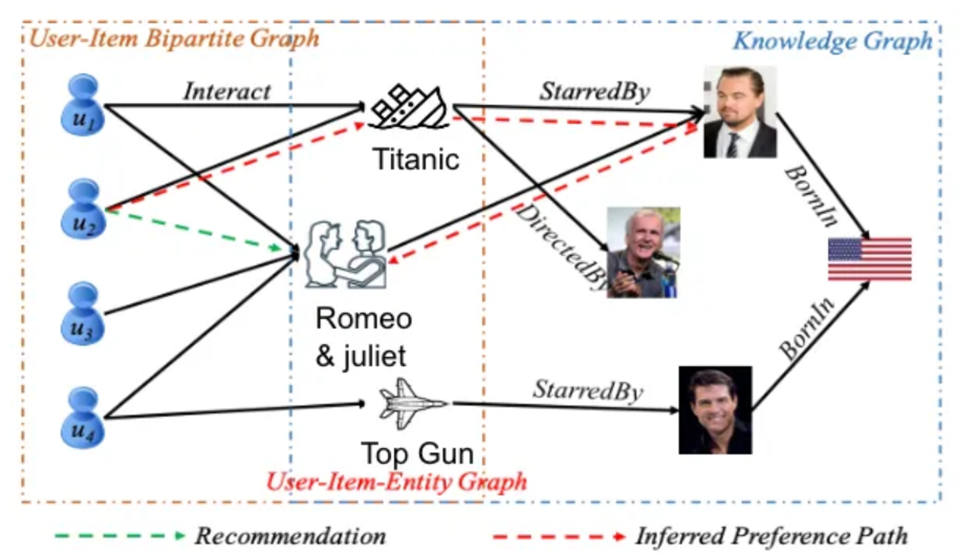
This transformation is crucial for integrating diverse candidate information into the knowledge graph, enabling a multifaceted view of the job market. The dataset's depth and granularity provide a rich foundation for predictive modeling and analytics, enhancing the job recommendation system's ability to align candidate profiles with potential job opportunities accurately.

# Methodology

The methodology for creating an advanced job recommendation system revolves around constructing a comprehensive knowledge graph, visualizing it, and applying cutting-edge graph embedding techniques. This process involves meticulously extracting and structuring data from two key datasets: job postings and job seekers. The creation of the knowledge graph is pivotal, as it captures the complex relationships between various job market entities. This is followed by a visualization phase to make the complex relationships comprehensible. Finally, graph embedding techniques are employed to transform the graphical data into a format suitable for machine learning models, enhancing the recommendation system's predictive power.

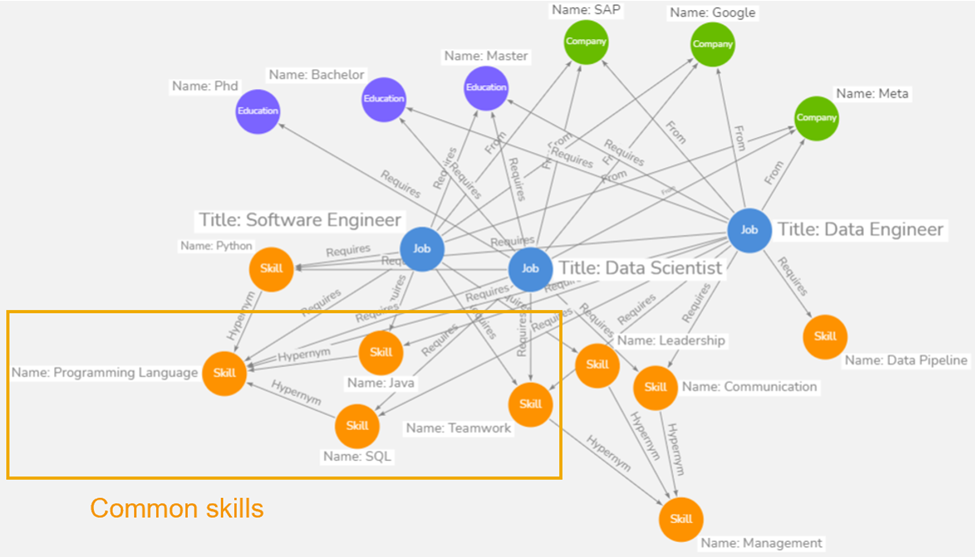
## 5.1 Knowledge Graph

In recent times, knowledge graphs have emerged as a powerful tool in the realm of recommender systems, addressing challenges like the sparsity of user-item interactions and the cold start problem, common in Collaborative Filtering (CF) methods. These graphs achieve this by encapsulating detailed information about both users and items within a unified data structure.



The accompanying illustration showcases a movie knowledge graph and illustrates how recommendations are generated for a user, u₂, using this graph. The recommendation process is intuitive: user u₂, having watched "Titanic," is recommended "Romeo and Juliet." This suggestion is based on the connection that both movies feature Leonardo DiCaprio as an actor. The figure, adapted from a study available on arxiv.org, highlights these connections through red dashed lines.

Knowledge graphs offer a significant advantage over previous recommender systems by not only storing lists of properties but also harnessing the semantic relationships between various entities. Their flexible structure allows for easy integration with other knowledge graphs, enhancing the depth of information. For instance, the movie graph could be expanded to include detailed attributes of each film. A key strength of knowledge graph-based recommender systems lies in their ability to utilize the structure of the graph itself to deliver more refined and accurate recommendations.



### Algorithm for Creating the Knowledge Graph:

· The knowledge graph construction starts by initializing a graph **G** using a graph database like Neo4j. This graph serves as a structural representation of relationships between various entities in the job market.

· For each unique **location**, **experience**, and **major** in the job postings and job seekers datasets, nodes are added to **G**. These nodes are categorized by their types, such as 'Location', 'Experience', and 'Major'.

· In the job postings dataset, each job title is linked to its required skills, experience level, academic major, and location. This is achieved by adding edges between the job title node and nodes representing these attributes.

· Similarly, in the job seekers dataset, nodes representing individual candidates are connected to their respective skills, experiences, majors, and locations through edges.

· The complete graph **G** thus represents a rich network of interlinked job market attributes, serving as the foundation for the recommendation system.

### Storing and Querying the Knowledge Graph:

· The knowledge graph is stored in a graph database like Neo4j, which excels in handling connected data and complex relationships.

· **Example of Storing Data:**

· To store a new job posting in Neo4j, a series of Cypher queries are executed. For instance:

CREATE (j:Job {title: 'Data Scientist'}) MERGE (s:Skill {name: 'Machine Learning'}) CREATE (j)-[:REQUIRES]->(s)

· This creates a 'Job' node and a 'Skill' node, and establishes a relationship indicating that the job requires the skill.

· **Example of Querying Data:**

· To find candidates with specific skills, a Cypher query like the following can be used:

MATCH (c:Candidate)-[:HAS\_SKILL]->(s:Skill {name: 'Machine Learning'}) RETURN c.name

· This query retrieves the names of all candidates who have the skill 'Machine Learning'.

### Visualization of the Knowledge Graph:

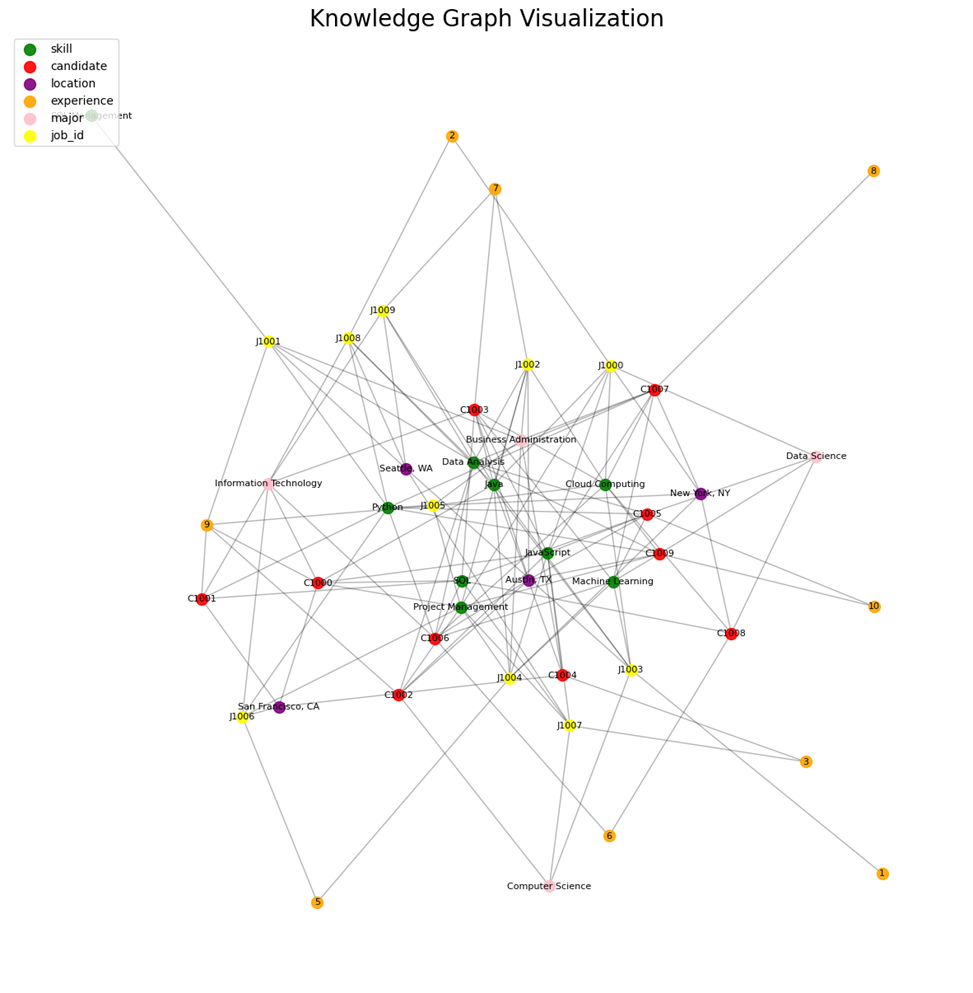
· Visualization is achieved using tools like Gephi or Neo4j's own visualization capabilities. The graph **G** is displayed using different colors for various node types, aiding in better understanding and analysis.

· The layout for the graph is computed using algorithms like the spring layout, which positions nodes based on their connections, creating an intuitive visual representation of the relationships.

### Main Execution:

· The complete methodology involves initializing the graph creation process by calling a function like **create\_knowledge\_graph()**, which follows the steps outlined above.

· Once the graph is created, it is visualized through a function like **visualize\_graph(graph)**, offering insights into the intricate web of connections in the job market.



### Output after querying the graph:

*Skills of C1001: Python, SQL*

*Experience of C1001: 9*

*Jobs that require the skills and experience of C1001:*

*Job Title: Machine Learning Engineer, Job ID: J1001*

*Job Title: Data Analyst, Job ID: J100*

The methodology encapsulates the intricacies of graph theory and database management, employing them to create a dynamic and insightful representation of the job market. This forms the foundation for applying advanced techniques like graph embedding, which will further enhance the job recommendation system's capabilities.

## 5.2 Graph Attention Network

Graph attention Network excel at capturing user-item interactions, identifying important patterns, and providing context-aware recommendations. The ability to adaptively weigh the contributions of different neighbors makes GATs particularly suitable for recommendation scenarios where the relationships between users and items are complex and dynamic.

**Graph Representation:**

Nodes: Users (job seekers) and items (job postings).

Edges: Interactions or relationships between users and items, representing actions like job applications, views, or other relevant engagements.

**Node Features:**

Each user and job posting node is associated with feature vectors. User features might include skills, experience, while job posting features could include required skills, qualifications, and job category.

**Initial Embeddings:**

Node feature vectors are initially transformed using a learnable weight matrix W to create initial embeddings for users and job postings.



**Attention Mechanism:**

The attention mechanism calculates attention scores based on the embeddings of the interacting nodes. For a user u i and a job posting j i the attention score eij is computed using a shared attention vector a :



**Attention Coefficients:**

The attention scores are normalized using the softmax function to obtain attention coefficients:



**Aggregation and Message Passing:**

The attention coefficients are used to aggregate information from neighboring nodes. The aggregated message for a user or job posting is a weighted sum of the embeddings of its neighbors:



**Multi-Head Attention:**

Multiple attention heads are often employed in parallel, each with its own attention parameters. The outputs from different heads are concatenated or averaged to obtain the final representation.



**Final Output:**

The final output for each user or job posting is obtained by passing the aggregated representations through a non-linear activation function:



**Training:**

The GAT is trained using backpropagation and optimization techniques to minimize a job recommendation-specific loss function. The model learns to adjust attention parameters and node representations to improve the accuracy and relevance of job recommendations for users.

## 5.3 Knowledge Graph Evaluation

When evaluating the effectiveness and reliability of a knowledge graph, several key dimensions are considered. These dimensions are critical in determining how well the knowledge graph performs in various aspects, from how accurately it reflects the real world to how up-to-date its information is. Each of these dimensions plays a vital role in the overall utility and reliability of the knowledge graph.

**Accuracy:** This refers to how well the data in the knowledge graph reflects reality. It's about the correctness of the information contained in the graph. For a knowledge graph to be valuable, the data needs to be accurate, representing real-world entities and their relationships correctly.

**Consistency:** Consistency is about ensuring that the data in the knowledge graph does not have contradictions. It's important that different parts of the graph do not provide conflicting information about the same entities or relationships.

**Integrity:** This dimension refers to the correctness of the relationships and connections within the graph. Integrity ensures that the graph maintains valid relationships between entities, adhering to predefined rules and structures. This is crucial for the reliability of the insights derived from the graph.

**Freshness:** Freshness is about how up-to-date the data in the knowledge graph is. In a rapidly changing world, information can quickly become outdated. A knowledge graph needs to be regularly updated to maintain its relevance and usefulness.

### Prediction Evaluation

In the model evaluation section of our analysis, we will delve into a comprehensive assessment of our predictive model's performance. Central to this evaluation are key metrics that offer insights into the accuracy and effectiveness of our model. These metrics include the confusion matrix, precision and recall, and the F1 score. By exploring these metrics, we can gain a holistic view of our model's performance and identify areas for improvement.

**Confusion Matrix**

The confusion matrix provides a detailed breakdown of the model's predictions, distinguishing between true positives, true negatives, false positives, and false negatives.



Figure 1: Confusion Matrix

In our study, the calculation of TP, FP, TN, and FN might change because our prediction result might be more than one value. we have to calculate those values for each job seeker and their prediction result, then combine them. Here is the explanation:

**True Positives (TP):** Count each instance where a job ID is correctly predicted as suitable for the job seeker.

**False Positives (FP):** Count each instance where a job ID is incorrectly predicted as suitable for the job seeker.

**False Negatives (FN):** Count each instance where a job ID is suitable for the job seeker but is not predicted by the model.

**True Negatives (TN):** In our case, true negatives refer to job ids that are not predicted, and they are not suitable for the job seeker. It would be a massive calculation and not very useful in this case, so we choose to not calculate it in our study.

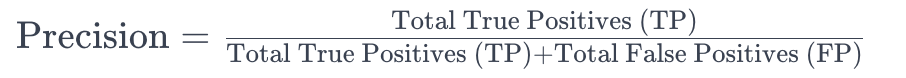
**Example:** Suppose for Job Seeker A, the model predicts job IDs {101, 102, 104} as suitable, but the actual suitable jobs are {101, 103, 104}. For this seeker, TP = 2 (for jobs 101 and 104), FP = 1 (for job 102), and FN = 1 (for job 103).

**Aggregation Across All Job Seekers:** We would perform a similar calculation for each job seeker in our test dataset and then add up all the TPs, FPs, and FNs. After calculating TP, FP, and FN for each job seeker, sum these counts across all job seekers in your testing dataset to get the total TP, FP, and FN.

**Precision and Recall**

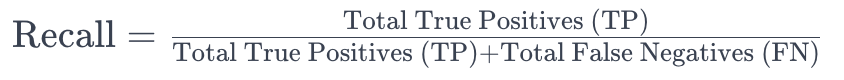
Precision and recall are critical in understanding the model's ability to correctly identify positive instances and its efficiency in classifying actual positive cases.

**Precision:** Precision for the whole system is then calculated as the total TP divided by the total of TP and FP (i.e., all the instances where the model predicted a job ID as suitable).



* **Cost of Low Precision (False Positives):** If precision is low, it means the model is making a lot of false positive errors, which means recommending jobs that are not suitable.

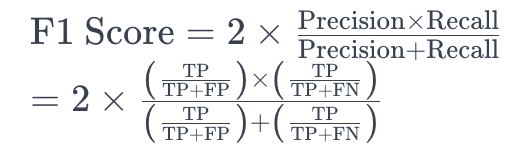
**Recall:** Recall is calculated as the total TP divided by the total of TP and FN (i.e., all the instances where the job ID was suitable).



* **Cost of Low Recall (False Negatives):** If the recall is low, it means the model is missing out on recommending suitable jobs - missing jobs that suit the user.

In our application, neither false positives nor false negatives yield favorable outcomes for the users. False positives, which occur when the model recommends unsuitable job positions, can lead to misunderstandings and diminish user trust in the system. On the other hand, false negatives, where the model fails to identify and recommend genuinely suitable job opportunities, result in users potentially missing out on important career prospects. Given these implications, it is imperative to maintain an equilibrium between precision and recall to optimize the model's performance. This necessitates the calculation of the F1 score, a metric that harmonizes the balance between precision and recall.

**F1 Score:** F1 score serves as a critical measure, ensuring that the model neither inundates the user with irrelevant job suggestions nor overlooks significant employment opportunities, thus providing a more holistic evaluation of the system's efficacy.



* **The value of F1 score:** The closer the F1 score is to 1, the better the model's performance in terms of both precision and recall. An F1 score near 1 indicates that the model has a high precision rate (most of its positive predictions are correct) and a high recall rate (it successfully identifies most of the relevant instances). This balance makes the F1 score a valuable metric for evaluating models, especially in situations where both false positives and false negatives carry significant costs.

### Proxy Metrics

**Probabilistic Modeling of Click-Through Rate (CTR):**

* **Methodology**: Utilize Bayesian Hierarchical Models to account for user-specific and job-specific random effects in CTR [16]. This approach allows for the pooling of information across different users and jobs, providing robust estimates even in the face of sparse data.
* **Implementation**: Employ Markov Chain Monte Carlo (MCMC) methods for parameter estimation, specifically Gibbs sampling or Hamiltonian Monte Carlo, to infer the posterior distributions of model parameters [17].

**Survival Analysis for Dwell Time:**

* **Methodology**: Implement Advanced Survival Analysis models, like the Cox Proportional Hazards Model with time-varying covariates, to analyze the dwell time on job recommendations [18]. This would incorporate both user-specific attributes and job-specific features as covariates in the model.
* **Implementation:** Use partial likelihood estimation for model fitting, and Schoenfeld residuals for assessing the proportional hazards assumption [19].

### Qualitative Evaluation

**NLP-Driven Feedback Analysis:**

* **Methodology**: Implement advanced NLP techniques like Latent Dirichlet Allocation (LDA) for topic modeling and BERT-based sentiment analysis to extract and categorize themes from user feedback [20].
* **Implementation**: Use Expectation-Maximization (EM) algorithms for inferring the latent topics in LDA, and transformer models pre-trained on large corpora for sentiment analysis [21].

**Expert Review with Decision Support Systems:**

* **Methodology**: Develop a Decision Support System (DSS) that assists experts in evaluating recommendations against high-dimensional criteria, integrating rule-based systems with machine learning classifiers [22].
* **Implementation**: Utilize Multi-Criteria Decision Making (MCDM) methods, like Analytic Hierarchy Process (AHP) or Technique for Order Preference by Similarity to Ideal Solution (TOPSIS), for systematic evaluation [23].

### Heuristic Evaluation

**Graph Centrality and Matching Algorithms:**

* **Methodology**: Employ graph centrality measures (Katz centrality, PageRank) to evaluate the importance of user skills and job requirements within the graph. Use bipartite graph matching algorithms (e.g., Hungarian Algorithm) to assess the alignment between users and jobs [24].
* **Implementation**: Leverage spectral graph theory for centrality analysis, and combinatorial optimization techniques for optimal job-user matching [25].

**Case-Based Reasoning with Algorithmic Heuristics:**

* **Methodology**: Use case-based reasoning to compare new recommendation scenarios with historical precedents. Develop algorithmic heuristics based on graph similarity measures (e.g., Graph Edit Distance) to evaluate the quality of matches [26].
* **Implementation**: Implement dynamic programming or heuristic search strategies to compute Graph Edit Distances, and similarity scoring algorithms to compare new cases with historical data [27].

## 5.4 Application

This proposal paper not only introduces a novel job recommendation system using a knowledge graph but also explores the versatile applications of the dataset within the recruitment domain. The dataset's rich and detailed information provides a foundation for several innovative applications:

**Predictive Modeling:** The dataset is instrumental in developing predictive models to assess the employability of applicants based on their profile attributes. Such models can enable organizations to efficiently prioritize candidates, ensuring a more streamlined recruitment process.

**Feature Analysis for Insightful Recruitment:** A thorough analysis of the dataset can offer recruiters deep insights into the key attributes and qualifications that are most significant for employability. This understanding is crucial in crafting effective job descriptions and developing robust candidate evaluation criteria, leading to more informed hiring decisions.

**Talent Pool Segmentation:** Utilizing the dataset to segment the talent pool allows for more targeted recruitment strategies. Candidates can be grouped based on various factors like skills, experience, or employability scores, enabling recruiters to tailor their approach for specific positions or skill requirements.

**Bias Detection and Fair Hiring Practices:** The dataset can serve as a valuable tool for identifying potential biases in the hiring process. Analyzing patterns and trends within the data can reveal biases related to gender, age, or other factors. This knowledge is essential for organizations to implement corrective measures, promoting fair and inclusive hiring practices.

**Enhancing Job-Candidate Matching:** Beyond these applications, the primary use case of the dataset is in enhancing the job-candidate matching process. The knowledge graph, constructed from the dataset, facilitates a nuanced understanding of both job requirements and candidate profiles, leading to highly accurate and personalized job recommendations.

# Expected Results

The main contributions of this dissertation, focusing on the development of a job recommendation system using a knowledge graph, are outlined as follows:

1. **Classification of Candidate and Job Characteristics (100% Completed):** This project has successfully classified various characteristics present in both candidate resumes and job postings. This classification is pivotal in understanding the intricate details that go into forming a comprehensive knowledge graph, which is the foundation of the job recommendation system.
2. **Algorithm to Construct Knowledge Graphs from Resumes and Job Postings (100% Completed):** An algorithm has been developed and refined to effectively construct knowledge graphs from the dataset of resumes and job postings. This algorithm is crucial for transforming unstructured data into a structured format that can be analyzed and processed for recommendations.
3. **Implementation of the Knowledge Graph Construction Algorithm (80% Completed):** The implementation of the algorithm for constructing knowledge graphs is currently 80% complete. This involves the practical application of the algorithm to the datasets to create a dynamic, interconnected knowledge graph that accurately represents the relationships and attributes of job seekers and jobs.
4. **Extending the Concept of Skill Matching to Time-Varying Job Market Trends (To be done):** The next step involves extending the concept of skill matching to adapt to time-varying job market trends. This will allow the system to not only match candidates based on current job requirements but also to predict and adapt to future market changes.
5. **Computing Match Scores for Candidate-Job Pairings (50% Completed):** A methodology has been developed to compute match scores for each candidate-job pairing. This involves a complex analysis of the knowledge graph to evaluate the suitability of a candidate for a particular job role, based on various factors like skills, experience, education, and job requirements.
6. **Developing an Algorithm for Dynamic Job-Candidate Recommendation using Graph Embedding model (To be done):** The final contribution, still to be developed, is an algorithm that performs dynamic job-candidate recommendations. This algorithm will leverage the knowledge graph to continuously update and provide relevant job recommendations to candidates and suitable candidate suggestions to employers, based on real-time data.

# Milestone

In this section we consider a few data-sets that we will use to demonstrate capabilities. All data-sets consist of scalar values sampled on a regular grid in space and time. We create a simplicial complex using the sample points as vertices and extend the scalar value to all space and time using barycentric interpolation - i.e the scalar value at any point in the convex hull of the set of points is got by interpolating the scalar value at the vertices of the simplex that contains the point. We consider three classes of data-sets based on their size.

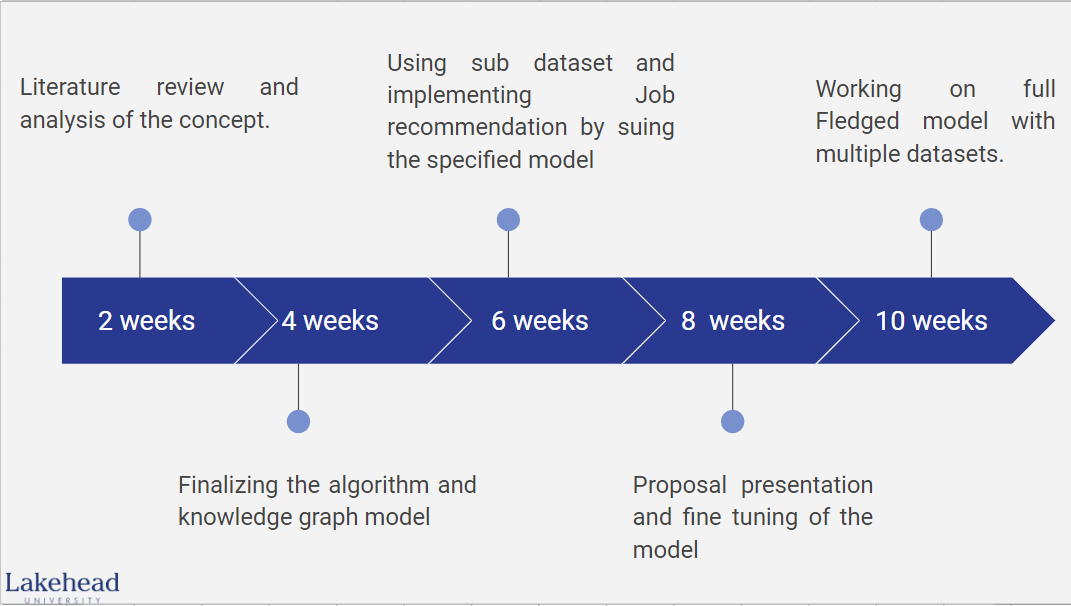
* Small: Grids of size 644 and smaller. These include toy data-sets (mainly used to test the algorithm) and sub-sampled versions of larger data-sets.
* Medium: Grids of size 644 to 1284. We plan to use these data-sets to stress test our algorithm on desktop computers. Available data-sets: J**ET**D**ATA** - a simulation of a supersonic Jet stream (courtesy of the Advanced Visualization Data Center’s data repository), C**OMBUSTION** - simulation of fuel combustion (courtesy of Suresh Menon and Chris Stone at Georgia Tech).
* Large: Grids of size 1284 and more. We expect to run our algorithm on high performance computer such as those available in Lawrence Livermore National Labs. A data-set in this class is the 10244 sized PPM data-set (courtesy of Valerio Pascucci), which is a simulation of the Richtmyer-Meshkov instability that occurs when a shock wave passes through an interface of two fluids of differing density. The large size of this data-set will certainly require special attention to memory usage, disk access patterns and cache behavior making visualization a challenging task.

# Contributions of Co-Authors

The research on this project involved collaborative efforts from Rutvik and Shivam. Smit played a crucial role in discovering and providing the dataset, with additional support from Nikhil in dataset preprocessing. Ziyang actively contributed to both documentation and dataset preprocessing, researching various methods to convert the dataset into a knowledge graph. Rutvik focused on researching algorithmic methods, while Shivam delved into the exploration of graph embedding techniques. Smit took charge of creating the operational knowledge graph, and Nikhil was instrumental in planning the project's applications. The overall workload for the proposal and presentation was evenly distributed among the five of us. While this whole research was guided by dr. shabah mohammed.

# Planned Milestones

The Following chart shows the planned milestones of our project. We have completed the literature review and build a prototype based on our chosen algorithm. The next goal is to build a full fledged model by using multiple dataset.[Fig. 2]

****

**Fig. 2**

# References

1. Fensel, Dieter, Umutcan Şimşek, Kevin Angele, Elwin Huaman, Elias Kärle, Oleksandra Panasiuk, Ioan Toma et al. "Introduction: what is a knowledge graph?." Knowledge graphs: Methodology, tools and selected use cases (2020): 1-10.
2. Xie, Lijie, Zhaoming Hu, Xingjuan Cai, Wensheng Zhang, and Jinjun Chen. "Explainable recommendation based on knowledge graph and multi-objective optimization." Complex & Intelligent Systems 7 (2021): 1241-1252.
3. Makarov, Ilya, Dmitrii Kiselev, Nikita Nikitinsky, and Lovro Subelj. "Survey on graph embeddings and their applications to machine learning problems on graphs." PeerJ Computer Science 7 (2021): e357.
4. Shah, Lipi, Hetal Gaudani, and Prem Balani. "Survey on recommendation system." International Journal of Computer Applications 137, no. 7 (2016): 43-49.
5. Hao, Xuejie, Zheng Ji, Xiuhong Li, Lizeyan Yin, Lu Liu, Meiying Sun, Qiang Liu, and Rongjin Yang. "Construction and application of a knowledge graph." Remote Sensing 13, no. 13 (2021): 2511.
6. Wang, Meihong, Linling Qiu, and Xiaoli Wang. "A survey on knowledge graph embeddings for link prediction." Symmetry 13, no. 3 (2021): 485.
7. Rendle, Steffen, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. "Fast context-aware recommendations with factorization machines." In Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, pp. 635-644. 2011.
8. Bruna, Joan, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. "Spectral networks and locally connected networks on graphs." arXiv preprint arXiv:1312.6203 (2013).
9. Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." Advances in neural information processing systems 30 (2017).
10. He, Xiangnan, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. "Neural collaborative filtering." In Proceedings of the 26th international conference on world wide web, pp. 173-182. 2017.
11. Wang, Hongxing, Zhenhao Gu, and Mengyu Gu. "Job Recommendation Algorithm Based on Knowledge Graph." In 2022 IEEE 8th International Conference on Computer and Communications (ICCC), pp. 2023-2027. IEEE, 2022.
12. C. C. Aggarwal, J. L. Wolf, K.-L. Wu, and P. S. Yu, “Horting hatches an egg: A new graph-theoretic approach to collaborative filtering,” in Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 201–212, ACM, 1999
13. Wang, Hao, Wenchuan Yang, Jichao Li, Junwei Ou, Yanjie Song, and Yingwu Chen. "An improved heterogeneous graph convolutional network for job recommendation." Engineering Applications of Artificial Intelligence 126 (2023): 107147.
14. Jia, Shanshan, Xiaoan Liu, Ping Zhao, Chang Liu, Lianying Sun, and Tao Peng. "Representation of job-skill in artificial intelligence with knowledge graph analysis." In 2018 IEEE symposium on product compliance engineering-asia (ISPCE-CN), pp. 1-6. IEEE, 2018.
15. Elsafty, Ahmed, Martin Riedl, and Chris Biemann. ”Document-based recommender system for job postings using dense representations.” In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers), pp. 216-224. 2018
16. Gelman, A., et al. "Bayesian Data Analysis." CRC Press, 2013.
17. Hoff, P. D. "A First Course in Bayesian Statistical Methods." Springer, 2009.
18. Kleinbaum, D. G., & Klein, M. "Survival Analysis: A Self-Learning Text." Springer, 2012.
19. Therneau, T. M., & Grambsch, P. M. "Modeling Survival Data: Extending the Cox Model." Springer, 2000.
20. Blei, D. M., et al. "Latent Dirichlet Allocation." Journal of Machine Learning Research, 2003.
21. Devlin, J., et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805, 2018.
22. Saaty, T. L. "The Analytic Hierarchy Process." McGraw-Hill, 1988.
23. Hwang, C. L., & Yoon, K. "Multiple Attribute Decision Making: Methods and Applications." Springer-Verlag, 1981.
24. Newman, M. E. J. "Networks: An Introduction." Oxford University Press, 2010.
25. Kuhn, H. W. "The Hungarian Method for the Assignment Problem." Naval Research Logistics Quarterly, 1955.
26. Bunke, H., & Riesen, K. "Recent Advances in Graph-Based Pattern Recognition with Applications in Document Analysis." Pattern Recognition, 2011.
27. Aamodt, A., & Plaza, E. "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches." AI Communications, 1994.