Integrating Graph-Based Features for Job Application Success Prediction in Online Recruitment Networks

- Using Node2Vec and Logistic Regression Approach -

Junior Natra Situmorang - 13524055
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: juniornatra72@gmail.com , 13524055@std.stei.itb.ac.id

***Abstract*— The utilization of online job recruitment applications has become increasingly prevalent. The relationships among job seekers, their skills, and the compatibility between job requirements and individual capabilities can be represented through a directed graph. This graph connects job seekers, the skills they possess, the skills required by job postings, as well as various supporting aspects that influence the likelihood of securing employment. By applying the Node2Vec approach, the relationships modeled within the graph can be analyzed to discover patterns that are capable of predicting the outcomes of job applications submitted by job seekers. To perform the prediction, logistic regression is used as a classification model due to its simplicity, interpretability, and effectiveness in handling binary outcome prediction problems such as job acceptance or rejection.**

Keywords—Graph, Node2Vec, Regression, Prediction, Recruitment

I. INTRODUCTION

. In this modern era, the use of online recruitment platforms such as LinkedIn, Glints, Jobstreet, and other similar applications has become widely adopted. The use of these online applications offers advantages to users, including both job seekers and companies. However, these job search applications only provide the available job listings without identifying the aspects or qualifications possessed by the users.

The use of such applications represents a new innovation that transforms the job application process. Companies are supported by the large number of applicants available. These job seekers can be used as a dataset for selecting candidates.

However, the use of these applications does not only bring benefits. A large number of applicants requires a selection process to determine candidates who are capable and have the skill sets that match the requirements. There is often a mismatch between the skills of the applicants and the skills required. A simple recommendation system only provides job suggestions based on the desired jobs, not based on deeper relationships and other relevant aspects.

One approach that can be used is a graph-based model to determine the relationships between job seekers and related aspects. Graphs help identify jobs that are aligned with the abilities of the users. This graph-based representation is able to show more complex relationships between workers and jobs.

Fig. 1.1. Graph representation of relationships among job seekers, skills, and job postings.

Source : Linkdin : The Power of Graph Theory: From Social Networks to Computer Networks (accessed Jun 15, 2025).

One of the approaches used to observe these relationships is **Node2Vec**. This algorithm converts the nodes in the graph into vector representations in an embedding space. These representations can be used to predict the results of job applications or provide job recommendations that are suitable to the abilities of job seekers.

By using the Node2Vec and Logistic Regression method approach, the process carried out by job seekers will become more appropriate based on their abilities, experiences, and the required skills.

II. THEORETICAL FRAMEWORK

2.1. Graph Definition

Graphs are commonly used to represent discrete objects and the relationships between those objects. By definition, a graph G is defined as G = (V, E), where V is a non-empty set of vertices (v_1, v_2, v_3, ..., v_n), and E is a set of edges (e_1, e_2, e_3, ..., e_n).

Based on the presence or absence of loops or multiple edges, graphs are classified into two types: simple graphs and non-simple graphs. A simple graph is one that does not contain any loops or multiple edges. On the other hand, a non-simple graph consists of two categories: multigraphs and pseudographs. A multigraph is a graph that contains multiple edges but no loops, whereas a pseudograph contains at least one loop.

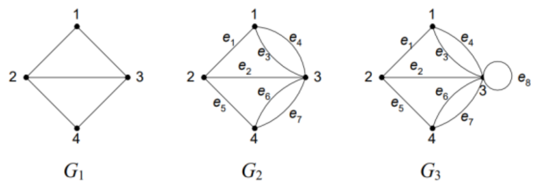


Fig. 2.1. (a) Simple Graph (G1), (b) multi-graph (G2), and (c) Pseudo-Graph (G3)

Source : https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf (accessed Jun 18, 2025).

Based on the orientation of the edges, graphs are classified into two types: directed graphs and undirected graphs. A directed graph is a graph in which each edge has a specified direction, while an undirected graph is a graph in which the edges have no direction.

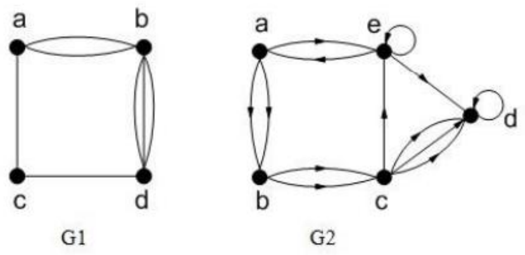


Fig. 2.2. Undirecred Graph (G1)and (b) Directed Graph or Digraph(G2).

Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf (accessed Jun 18, 2025).

There is also a type of graph known as a weighted graph, in which each edge carries a specific value or weight that represents additional information such as distance, cost, time, or the strength of connection between nodes. These weights provide deeper context to the relationships within the graph, allowing for more accurate and meaningful analysis—such as shortest path search, network modeling, or graph representation learning—compared to unweighted graphs.

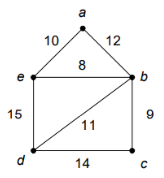


Fig. 2.3. Weighted Graph.

Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf (accessed Jun 18, 2025).

2.2. *Terminologies of Graph*

Adjacent, Two nodes are considered adjacent if they are directly connected by an edge. In the diagram below, it can be seen that node 1 is adjacent to node 2 and node 3, but not adjacent to node 4.

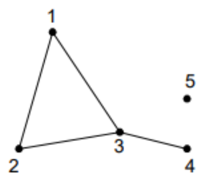


Fig. 2.4. Graph with adjacent nodes

Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf (accessed Jun 18, 2025).

Incidency, An edge and a node that are directly connected are said to be **incident**. For any edge denoted as *e = (vj, vk)*, both nodes *vj* and *vk* are incident to the edge *e*. Referring to figure G1, edge (2,3) is incident to nodes 2 and 3, and edge (2,4) is incident to nodes 2 and 4. In contrast, edge (1,2) is not incident to node 4.

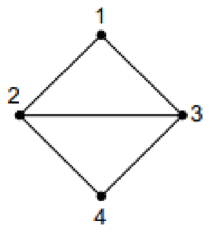


Fig. 2.5. Graph with incident.

Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf (accessed Jun 18, 2025).

Empty graph, A graph that contains only vertices without any connecting edges is known as a null graph or an empty graph.

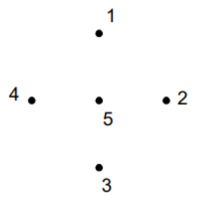


Fig. 2.6. Empty Graph.

Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf (accessed Jun 18, 2025).

Degree, The degree of a node, denoted as *d(v)*, represents the number of edges incident to that node. In the figure below, for graph G1, node 1 and node 4 have the same degree since both are connected to the same number of edges. Therefore, *d(1) = d(4) = 2*.

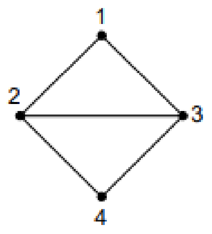


Fig. 2.7. Graph illustration.

Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf (accessed Jun 18, 2025).

The **Handshake Lemma** is a theorem which states that the sum of the degrees of all vertices in a graph is equal to twice the number of edges. Based on this principle, it follows that the total degree of a graph is always an even number.

Path, finite or infinite sequence of edges that connects a sequence of vertices. It begins at a starting vertex v0v_0v0 and ends at a final vertex vnv_nvn, with the length of the path defined by the number of edges it contains.

A path of length n from a starting node v_0 to a destination node v_n in a graph G is a sequence that alternates between nodes and edges in the form $(v_0, e_1, v_1, e_2, v_2, ..., v_{n-1}, e_n, v_n)$, such that $e_1 = (v_0, v_1)$, $e_2 = (v_1, v_2)$, ..., $e_n = (v_{n-1}, v_n)$ are all edges of the graph G.

If the graph contains multiple edges (parallel edges), then each edge e_i must be explicitly written in the path. However, if the graph is simple (i.e., contains no multiple edges), the edges do not need to be listed explicitly.

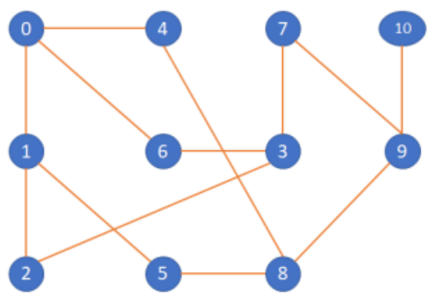


Fig. 2.8. Unweighted Graph.

Source: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/20-Graf-Bagian1-2024.pdf (accessed Jun 18, 2025).

Consider the following simple graph G: the path 0, 6, 3, 7, 9, 10 is a valid path from node 0 to node 10, passing through the edges (0, 6), (6, 3), (3, 7), (7, 9), and (9, 10). The length of the path is defined as the number of edges in the path. Therefore, the path 0, 6, 3, 7, 9, 10 has a length of 5.

Cycle, A cycle is a path in a graph where the starting and ending vertices are the same. The length of a cycle is determined by the number of edges it contains.

2.3. Graph Embedding

Graph embedding is a technique that transforms graph nodes into low-dimensional vector representations, capturing essential graph characteristics such as semantic relationships and structural patterns. These embeddings enable machine learning models to process graph data efficiently by converting complex relational information into compact, dense vectors. Unlike traditional graph representations, graph embeddings preserve critical topological and attribute-based features while reducing computational overhead, leading to more efficient processing in terms of both time and resources.

Have you ever considered how social media platforms accurately recommend potential connections or how navigation systems predict traffic congestion in advance? These capabilities stem from the analysis of graph networks of interconnected entities such as users, locations, or objects. To interpret these intricate relationships effectively, a key tool is employed: graph embeddings, which serve as a bridge between complex network structures and machine-readable data.

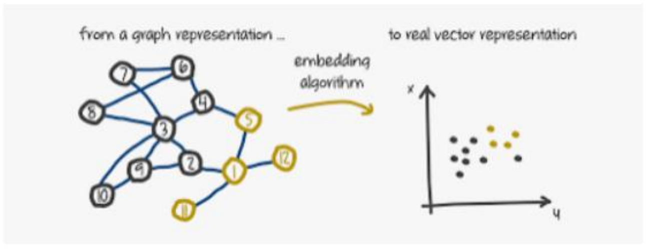

from a graph representation ...
embedding algorithm
to real vector representation
x
y

Fig. 2.9. Graph embedding illustration.

Source: https://www.nebula-graph.io/posts/graph-embeddings (accessed Jun 18, 2025).

Think of them as a magic trick that transforms intricate networks of vertices and edges into compact numerical representations. These "embeddings" capture the essence of each node (vertex) and its relationship to others, distilling the complex network into a format readily understood by machine learning algorithms.

2.3. Node2Vec Algorithm

Node2Vec is a graph embedding algorithm that extends the DeepWalk approach by introducing a biased random walk strategy to capture both local structure (homophily) and global structure (structural equivalence) in a graph. This algorithm is designed to produce vector representations (embeddings) of nodes in such a way that the topological and semantic relationships among them are preserved in the embedding space.

At the core of Node2Vec lies a flexible random walk process controlled by two hyperparameters: the return parameter (p) and the in-out parameter (q). The return parameter regulates the likelihood of revisiting a node in the walk, while the in-out parameter controls the walk's tendency to explore further nodes (depth-first search-like behavior) or stay close to the starting node (breadth-first search-like behavior). By tuning these parameters, Node2Vec can emphasize homophily, where similar nodes (e.g., friends in a social network) are embedded closely, or structural equivalence, where nodes with similar structural roles (e.g., hubs or bridges in a network) are represented by similar vectors.

The Node2Vec process involves two main steps. First, it generates sequences of nodes through biased random walks to simulate possible paths within the graph. Then, it applies the Skip-gram model from Word2Vec to learn embeddings by maximizing the probability of neighboring nodes co-occurring in the generated walks.

2.4. Logistic Regression

Logistic regression is a statistical analysis technique used to model the relationship between one or more independent variables and a categorical dependent variable, typically binary (such as yes/no or 1/0). The goal is to predict the probability of a certain event occurring based on historical data.

Mathematically, logistic regression uses the sigmoid function (also known as the logistic function) to map predicted values to a probability range between 0 and 1. This probability can then be used to classify an entity into one of two categories.

The sigmoid function is defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Where:

- $z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$
- β_i are the model coefficient.
- x_i are the features values

This function ensures that the output σ(z) lies between 0 and 1, making it suitable for probabilistic interpretation.

Example: Logistic regression can be used to predict whether a website visitor will click the checkout button in an online store. The model considers features such as the time spent on the website and the number of items in the cart. Based on historical patterns, if visitors spend more than five minutes and add more than three items, the likelihood of clicking checkout increases. Using this relationship, the logistic regression model can predict the behavior of new users.

In the context of this research, logistic regression is applied as a classification algorithm to determine whether a job applicant is a suitable match for a given position, using vector representations (embeddings) of nodes derived from a graph structure via Node2Vec.

2.5. Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a multivariate statistical analysis technique. At its core, PCA is a projection-based method used in multivariate data analysis to reduce the dimensionality of large-scale datasets by summarizing them into a smaller set of variables or summary indices. This technique is commonly used to condense complex datasets while retaining as much relevant information as possible.

The PCA process consists of five main stages: (1) standardizing the variables so they are on the same scale, (2) calculating the covariance matrix to understand the relationships between variables, (3) extracting eigenvalues and eigenvectors from the covariance matrix to determine the direction of the principal components, (4) selecting the most significant components based on the magnitude of the eigenvalues, and (5) transforming the data into the new axes defined by the principal components.

In this study, PCA is applied after the embedding process using the Node2Vec algorithm to reduce the high-dimensional vector space (e.g., 64 dimensions) into two dimensions. This reduction allows the resulting embeddings to be visualized in a 2D scatter plot, enabling the examination of patterns and the distribution of nodes (entities) across the reduced space.

III. Method

3.1 Data Preparation

To perform job acceptance prediction, data is required that will later be represented as nodes and edges in a graph. The nodes can represent entities such as applicants, jobs, skills, companies, schools, or cities. Meanwhile, the edges capture the relationships between these entities—for example, an applicant possesses a certain skill, applies to a particular job, or graduated from a specific school. By constructing a graph from this data, a graph embedding can be performed using the Node2Vec algorithm, which transforms the graph structure into vector representations suitable for machine learning models.

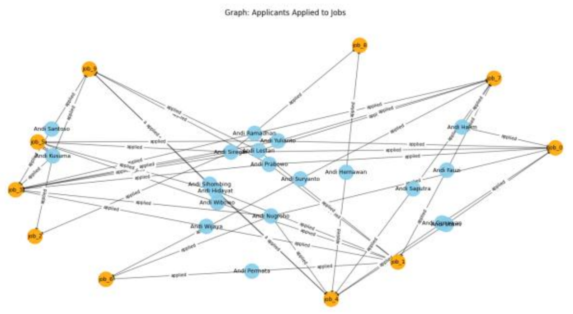


Fig. 3.1. Sub-Digraph Applicant To Jobs

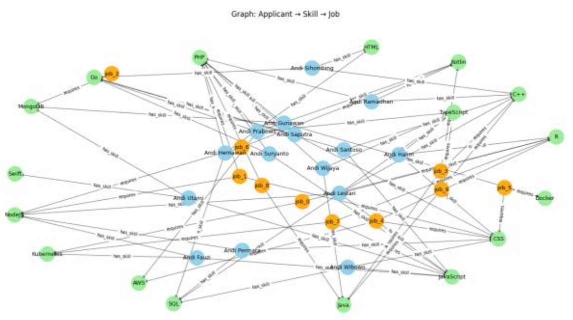


Fig. 3.2. Sub-Digraph Applicant to Jobs to Skill

During the data preparation stage, the dataset is not sourced from real-world data but is generated using a custom data generator to simulate the relationships between relevant entities in the context of job acceptance prediction. The generator is designed to create nodes representing entities such as applicants, jobs, skills, schools, and more, along with edges representing relationships such as “applied to,” “has skill,” or “graduated from.” Although synthetic, this data is structured to resemble real-world patterns and distributions, ensuring its usefulness for modeling and experimentation. The data generator produces three output files: one containing a list of nodes and their types, another containing pairs of connected nodes along with the relation types, and the last one recording the application status of each applicant (accepted = 1 or rejected = 0).

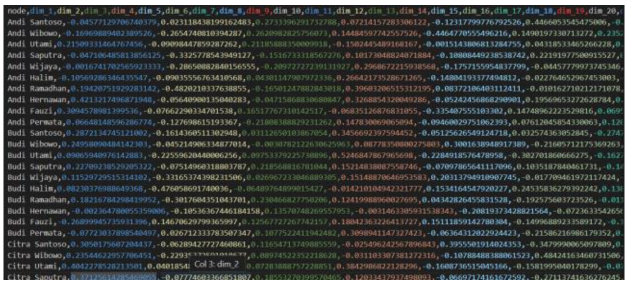
The generated data is stored in three separate files: edges.csv, nodes.csv, and labels.csv. The edges.csv file contains the relationships between the nodes, representing interactions such as applicants applying to jobs or possessing certain skills. The nodes.csv file lists all node names along with their corresponding types (e.g., applicant, job, skill, company, etc.). Lastly, the labels.csv file provides information on whether an applicant was accepted or not for a job they applied to, serving as the ground truth for prediction tasks.

The result of this data consists of 1806 applicants, 20 skills, 10 jobs, 10 companies, 10 schools, 10 Job Role, 6 cities, 5 experience levels, 5 certifications, 5 GPA levels, and 5 languages stored in nodes.csv. This data will later be interconnected to form a graph (as shown in Fig 3.1 and Fig 3.2).

3.2 Graph Embedding Process

After the graph data is represented, the next step is to perform graph embedding using the Node2Vec algorithm. In this process, each node—whether it is an applicant, job, skill, or any other entity—will be mapped into a fixed-dimensional numerical vector. This step allows the system to capture relationships or associations between nodes based on the graph structure that has been constructed.

The embedding process is carried out by adjusting parameters such as vector dimensions, walk length, and the number of random walks. These parameters are tuned to effectively capture the connection patterns present in the graph. Once the embeddings are generated, the resulting vectors are saved and can then be used as input features for a job acceptance prediction model.



```
node,dim_1,dim_2,dim_3,dim_4,dim_5,dim_6,dim_7,dim_8,dim_9,dim_10,dim_11,dim_12,dim_13,dim_14,dim_15,dim_16,dim_17,dim_18,dim_19,dim_20,
Andi Santoso,-0.04577129706740379,0.023118438199162483,0.2733396291732788,0.07214157283306122,-0.12317799776792526,0.4466053545475006,-0.
Andi Wibowo,-0.16969889402389526,-0.2654740810394287,0.2620982825756073,0.14484597742557526,-0.4464770555496216,0.1490197330713272,0.235
Andi Utami,0.21509331464767456,-0.09098447859287262,0.21185888350009918,-0.1502445489168167,-0.0015143806813284755,0.04318533465266228,0.
Andi Saputra,-0.047106485813856125,-0.3325778543949127,-0.1516733318567276,0.10173048824071884,-0.10808449238538742,0.22191977500915527,
Andi Wijaya,-0.0016741702565923333,-0.28650882840156555,-0.20972727239131927,0.2968672215938568,-0.1757155954837799,-0.04457779973745346,
Andi Halim,-0.10569286346435547,-0.09035556763410568,0.04301147907972336,0.26642173528671265,-0.14804193377494812,-0.022764652967453003,
Andi Ramadhan,0.19420751929283142,-0.4820210337638855,-0.16501247882843018,0.39603206515312195,0.08372106403112411,-0.010162710212171078,
Andi Hernawan,0.4213217496871948,-0.05640900135040283,-0.04715868830688847,0.3268854320049286,-0.05242456868290901,0.19569653272628784,0.
Andi Fauzi,0.3094578981399536,-0.07662290334701538,0.16517767310142517,-0.06835126876831055,-0.335407555103302,0.14748962223529816,0.069
Andi Permata,0.06648148596286774,-0.127698615193367,-0.21808388829231262,0.147830069065094,-0.09460029751062393,0.07612045854330063,0.12
Budi Santoso,0.2872134745121002,-0.1614360511302948,0.03112650103867054,0.3456692397594452,-0.05125626549124718,0.032574363052845,-0.274
Budi Wibowo,0.24958090484142303,-0.045214906334877014,-0.0030782122630625963,0.08778350800275803,0.3001638948917389,-0.21605712175369263,
Budi Utami,0.09065940976142883,-0.22559620440006256,0.09753379225730896,0.5246847867965698,-0.2284918576478958,-0.302701860666275,-0.162
Budi Saputra,0.22709238529205322,-0.07514960318803787,0.218568816781044,0.15214838087558746,-0.07097865641117096,0.1035187840461731,-0.14
Budi Wijaya,0.11529729515314102,-0.33165374398231506,0.026967233046889305,0.15148870646953583,0.20313794910907745,-0.017709461972117424,
Budi Halim,0.08230376988649368,-0.476058691740036,-0.06489764899015427,-0.014210104942321777,0.1534164547920227,0.24535836279392242,0.13
Budi Ramadhan,0.18216784298419952,-0.3017604351043701,0.230466827750206,0.12419988960027695,0.04342826455831528,-0.192575603723526,-0.01
Budi Hernawan,-0.002364780055359006,-0.10536367446184158,0.13570748269557953,-0.0031463305931538343,-0.20819373428821564,-0.07236335426
Budi Fauzi,-0.26899945735931396,0.14670629799365997,0.12567727267742157,0.18042363226413727,0.15111859142780304,-0.14996889233589172,-0.
Budi Permata,-0.07723037898540497,0.026712333783507347,0.1077522411942482,0.3098941147327423,-0.06364312022924423,-0.21586216986179352,0.
Citra Santoso,0.3050175607204437,-0.06289427727460861,0.11654713749885559,-0.025496242567896843,0.3955501914024353,-0.3479990065097809,0.
Citra Wibowo,0.23544622957706451,-0.2293[illegible],0.08974522352218628,-0.031103307381272316,-0.10788488388061523,0.48424163460731506,
Citra Utami,0.4042278528213501,0.0401854 Col 3: dim_2 0.07283888757228851,0.3842986822128296,-0.1608736515045166,-0.1581995040178299,-0.0
Citra Saputra,[illegible],-0.07774603366851807,0.18553270399570465,0.12033437937498093,-0.06697174161672592,-0.27113741636276245,
```

Fig. 3.3. Embedding Result.

The embedding result is then saved into a file called node_embedding.csv. The result will later be used to make a prediction.

3.3 PCA Visualization

After the embedding process is carried out using the Node2Vec algorithm, Principal Component Analysis (PCA) is applied to visualize the embedding results. Initially, each node is represented as a 64-dimensional vector; after applying PCA, these vectors are reduced to two dimensions.

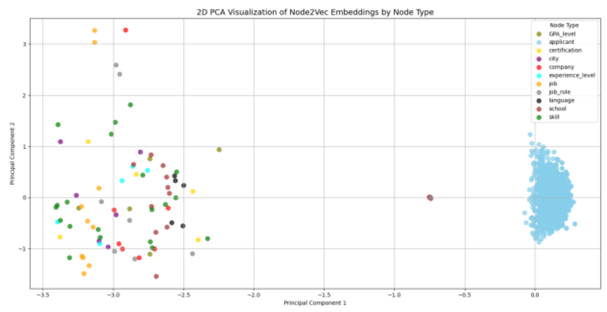


Fig. 3.4. PCA Digraph Visualization.

This visualization helps us observe the distribution patterns and relationships between nodes based on their proximity in the reduced vector space. For instance, applicants who are strongly associated with specific jobs tend to appear closer to those jobs in the plot.

3.4 Prediction Using Logistic Regression

In this section, the logistic regression process is used to predict applicant-job pairs and determine whether an applicant will be accepted for a specific job or not. Logistic regression is chosen in this research because the dataset is not too large, a binary output (1 or 0) is required, and there are not many outliers.

The resulting embedding vectors are trained on the model using labels.csv, which contains labels indicating whether an applicant is accepted (1) or not accepted (0) for a particular job.

Table result of logistic regression

TABLE I. LOGISTIC REGRESSION RESULTS

Class	Precision	Recall	F1-Score	Support
0	0.5147	0.4167	0.4605	336
1	0.5654	0.6589	0.6086	387
Accuracy			0.5463	723
Macro AVG	0.5401	0.5378	0.5346	723
Weighted AVG	0.5418	0.5463	0.5398	723

The evaluation table shows that the model has an accuracy of 54.36%, which is not considered good, as optimal results are typically expected to be above 70%. The model performs relatively better on class 1 (accepted) compared to class 0, as indicated by the higher recall and F1-score for class 1. However, both class performances are still far from optimal.

By using the logistic regression algorithm, predictions can be made based on the applicant and job pairs that have already been embedded.

IV. RESULT DAN DISCUSSION

4.1 Result

At this stage, testing is conducted on the prediction model that has been created using node2vec embedding and the Logistic Regression algorithm. The testing is carried out by comparing the existing applicant data with the prediction results, with the data determined according to user preferences. Several trials have been conducted on various applicants.

TABLE II. TEST RESULTS

Applicant	Job	Probability	Predicted Class	Reality Class
Puti Wicaksono	Job_0	0.6082	1 (Accepted)	0 (Rejected
Puti Lestari	Job_6	0.8562	1 (Accepted)	1 (Accepted)
Yani Haryanto	Job_5	0.6989	1 (Accepted)	1 (Accepted)
Eka Siregar	Job_1	0.6430	1 (Accepted)	1 (Accepted)
Eka Kusuma	Job_3	0.4614	0 (Rejected)	0 (Rejected)
Andi Halim	Job_0	0.6872	1 (Accepted)	1 (Accepted)
Nina Utami	job_2	0.5026	1 (Accepted)	0 (Rejected)

It can be seen from Table II, out of a total of 7 trials that have been conducted, five predictions matched the values in the data, while there were two incorrect predictions. This indicates that the model has an accuracy rate of 71.4%. These prediction errors fall into the false positive category, which shows that the model is more inclined to predict that someone will be accepted, especially if the probability value is slightly above the classification threshold (0.5).

4.2 Discussion

From the model results, it's clear that some false positive predictions still happen. This could be due to a few things — like missing or incomplete data from applicants, jobs, or other related variables, an imbalance in the dataset, or maybe because logistic regression isn't the best fit for this kind of classification problem.

Also, the quality of the embeddings generated by Node2Vec plays a big role in how well the model performs. If the vector representations of the nodes don't properly reflect the relationships (edges) between entities (nodes), then the model might have a hard time telling which applicant-job pairs are actually a good match.

To improve this, we can try a few things — like balancing the data using oversampling methods such as SMOTE, and making sure all the important attributes are included (like work experience match, job role details, or other factors related to hiring decisions).

V. Conclusion

In predicting job application outcomes, a graph-based representation approach using Node2Vec can be utilized to model the relationships between each node—such as the connections between applicants, jobs, skills, and other supporting attributes that contribute to enhancing the prediction.

By transforming the relational graph into numerical embeddings, the prediction of job acceptance becomes more structured and automated. This research can still be improved by expanding the dataset, refining the graph structure, and incorporating additional relevant features to better capture the complexity of real-world recruitment scenarios.

This research can be utilized to predict job application outcomes. In addition, it can also serve as a recommendation system for job applications in online job offering platforms.

VI. Appendix

To complement this research paper, several external resources are provided for readers to explore the implementation and processes involved in this study:

1. Source Code Repository
   All source code used in this study is publicly available on the following GitHub repository:
   https://github.com/Jerannn24/Analisis-Job-Application
2. Research Presentation Video
   A brief explanation of the background, methodology, and results of this research is also presented in a video format, which can be accessed via the following link:
   https://youtu.be/OTkdQPCt3IU

VII. Acknoledgment

First and foremost, I would like to express my gratitude to God Almighty, for it is by His grace that this paper, titled "Integrating Graph-Based Features for Job Application Success Prediction in Online Recruitment Networks – Using Node2Vec and Logistic Regression Approach", could be completed.

I would also like to thank Dr. Ir. Rinaldi, M.T., as the lecturer of IF1220 Discrete Mathematics – K01, who has guided and taught me throughout this semester.

Lastly, I sincerely thank my parents for their constant support, as well as my family. I am also grateful to Toleransi Kopi for providing a comfortable space to work on this paper.

References

[1] Malathy Saranya, "Graph theory is a fundamental area of discrete mathematics that studies the relationships and connections between different entities. These relationships are modeled as graphs, which consist of vertices (or nodes) and edges (or links) that connect pairs of vertices.," Linkedin.com, Dec. 12, 2024. https://www.linkedin.com/pulse/power-graph-theory-from-social-networks-computer-malathy-saranya-iwdbe (accessed Jun. 15, 2025).

[2] R. Munir, "IF1220 Matematika Diskrit - Semester II Tahun 2024/2025," *Itb.ac.id*, 2024. https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025-2/matdis24-25-2.htm (accessed Jun. 15, 2025).

[3] A. Adila, "Song Recommendation Using Weight-Directed Graph and Reinforcement Learning," *Rinaldi Munir Page*, Dec. 11, 2020. https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Makalah/Makalah-Matdis-2020%20(97).pdf (accessed Jun. 15, 2025).

[4] NebulaGraph, "What are graph embeddings ?," *NebulaGraph*, Feb. 21, 2024. https://www.nebula-graph.io/posts/graph-embeddings (accessed Jun. 18, 2025).

[5] "Apa itu Regresi Logistik? - Penjelasan tentang Model Regresi Logistik - AWS," *Amazon Web Services, Inc.* https://aws.amazon.com/id/what-is/logistic-regression (accessed Jun. 18, 2025).

[6] Tomaz Bratanic, "Complete guide to understanding Node2Vec algorithm - TDS Archive - Medium," *Medium*, Aug. 16, 2021. https://medium.com/data-science/complete-guide-to-understanding-node2vec-algorithm-4e9a35e5d147 (accessed Jun. 18, 2025).

[7] "Principal Component Analysis: Pengertian dan Cara Kerjanya," *Algoritma*, Mar. 29, 2022. https://algorit.ma/blog/principal-component-analysis-2022 (accessed Jun. 18, 2025).

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 1 Juni 2025

[illegible]

Junior Natra Situmorang - 13524055