Revealing the Relationship between Task Types and Query Methods through a Comparative Analysis of Latent Space with KNN and Direct Queries Methods

Tingting Zhang, 7278888

Faculty of Science,
at Utrecht University

Supervisors:
Dr. M. Behrisch
Prof. dr. ir. A.C. Telea

A thesis submitted for the degree of Master of Science (MSc.) in Human-Computer Interaction

October 9, 2024

Abstract

Research based on knowledge graphs has been a hot topic in various fields. Users can either perform direct queries using the rich semantic information provided by the knowledge graph or search through the latent space to complete their tasks. However, existing research lacks a comprehensive comparison between these two query methods. This thesis aims to clarify the relationships between these query methods and different task types in the paper domain. Therefore, we redefine a graph task classification for the paper domain, which enhances the understanding of user intent in task categorization. We present a Node2Vec or GraphSAGE with KNN model for queries in the latent space. We evaluate the recall of the direct query's link prediction algorithm and the Node2Vec or GraphSAGE with KNN models on the Cora and Movielens datasets. The experimental results show that Node2Vec or GraphSAGE with KNN outperforms the link prediction algorithm. Furthermore, the diversity of query results is assessed in both the paper and movie domains. The results reveal that direct querying performs better for Adjacency, Accessibility by Links, Common Connection, Nodes Attribute, and Hybrid tasks, while the latent space with KNN method is better for the Explore task. These findings fill a gap in current research and enhance the effectiveness of user queries. Additionally, we design a novel dynamically updated paper recommendation system, improving the explainability of recommendation results.

Contents

1 Introduction

In today's era of information explosion, recommendation systems are widely applied in people's daily lives [7]. The traditional recommendation systems are mainly divided into three categories: content-based recommendation systems [2], collaborative filtering-based recommendation systems [6], and hybrid recommendation systems [45]. They have achieved good performance, but cannot reflect the rich semantic information in the recommended items [23]. Since Google introduced the knowledge graph concept in 2012 and used it to enhance the capabilities of its search engine [14], the recommendation approach combining knowledge graphs with recommender systems has become a hot research topic. A knowledge graph is a graph where entities act as nodes and relationships as edges [35]. When applying knowledge graphs to recommendation systems, entities typically represent users or recommended items, while edges represent the relationships between entities [33]. Users can directly query the knowledge graph's rich semantic relations to retrieve desired information [35]. Alternatively, they can use the latent space to complete their tasks. For instance, when users seek paper recommendations within a knowledge graph of academic papers, they can use existing topological link prediction algorithms to find users similar to themselves in the graph, based on the principle that similar users like similar items and receive paper recommendations accordingly. Alternatively, they can transform the problem into a latent space, searching within a latent space created by "papers", "users", "friends "and "read", to identify the closest users to themselves and receive recommendations in this manner.

From a logical standpoint, although exploring the latent space is slower, the recommendations generated through this method tend to offer better explainability. However, if a user wants to query the author of the paper "Simulating the Geometric Growth of the Marine Sponge Crella Incrustans," they can simply use the direct querying approach through the knowledge graph's `"written_by_authors"` relationship to retrieve the result. They can also search within the latent space of papers, authors, and the `"written_by_authors"` relationship to find the author closest to this paper. For this task, direct querying not only provides faster results but also offers higher accuracy and trustworthiness. Based on these examples, we can find the performance of different query methods varies depending on the task type.

This thesis focuses on whether users should use latent space to perform problem space transformations for different task types in the paper recommendation domain. Currently, there is no research indicating how users should choose between the direct query method and latent space with the KNN method for different tasks. This study aims to fill this gap. From a commercial perspective, it also provides direction for improving existing paper recommendation platforms, enhancing the explainability of the recommendations and increasing user satisfaction. Since existing task classification methods are not specifically tailored to the domain of academic papers, this thesis redefines the task classification method for academic paper graphs based on established task classification frameworks. Additionally, the classification method is tested in the Movie domain to assess its generalizability. Regarding the link prediction algorithms used for direct querying, this thesis selects the Adamic-Adar algorithm for the paper dataset, as it is better suited for handling heterogeneous and imbalanced data.

For the algorithms operating in the latent space, unlike previous research that focuses on improving algorithm accuracy [20] [25] [38], this thesis does not prioritize precision enhancement. This is because it may lead users into an "information bubble" and increase model training costs. With the growing maturity of Neo4j's algorithm library, we directly use Neo4j's node embedding combined with KNN to explore the latent problem space. The node embedding algorithms provided by Neo4j are broadly classified into inductive embeddings (FastPR, GraphSAGE [13], and HashGNN) and transductive embeddings (Node2Vec [12]). This thesis chooses the Node2Vec algorithm because it is the only transductive algorithm available. In contrast, GraphSAGE is selected among the inductive algorithms because the academic paper knowledge graph is a heterogeneous graph. GraphSAGE is the only algorithm capable of distinguishing different node types and assigning weights to different relationship types. Additionally, the performance of the Node2Vec or GraphSAGE combined with KNN will be compared to the link prediction algorithms on the benchmark datasets Cora and MovieLens to test whether the proposed latent space design outperforms direct link prediction algorithms in terms of recall.

Finally, the current 2D paper representation in virtual conference platforms based on the MiniConf is often simplistic and static, like Figure 1 shown. To address the limitations, this thesis proposes a novel dynamic updated 2D visualization that combines with recommendation algorithms. The 2D representation of papers and users can dynamically change based on user interest and present recommendations. This design can enhance the explainability of recommendation results.

1.1 Research Question and Contribution

1.1.1 Research Question

The main objective of this thesis is to explore the relationship between query methods and task types in the domain of paper recommendation. Additionally, this study aims to quantitatively evaluate the accuracy of latent space exploration methods compared to direct link prediction algorithms, to validate the effectiveness of the Node2Vec or GraphSAGE combined with KNN models. Since our ultimate goal is to recommend papers, we will focus on the

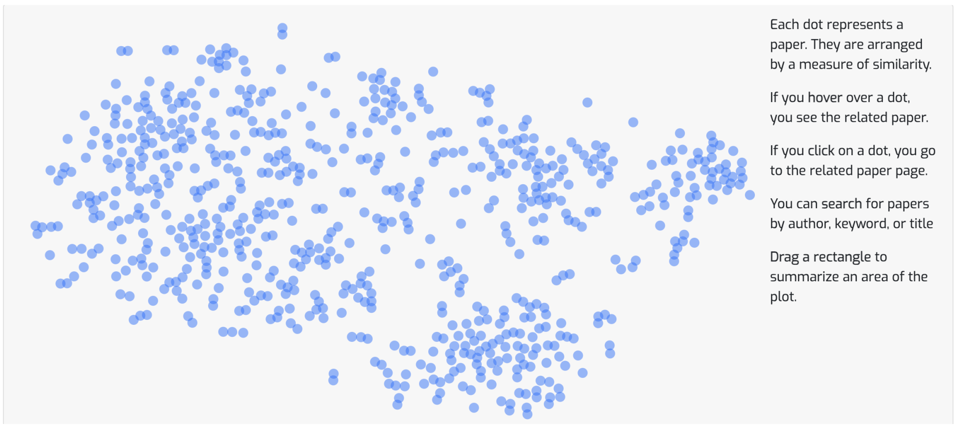

Each dot represents a paper. They are arranged by a measure of similarity.
If you hover over a dot, you see the related paper.
If you click on a dot, you go to the related paper page.
You can search for papers by author, keyword, or title
Drag a rectangle to summarize an area of the plot.

Figure 1: Paper's 2D visualization of Miniconf comes from `https://iclr.cc/virtual_2020/paper_vis.html`.We can find that the 2D visualizations of the papers are static and are not combined with recommendation functions. It can not give users personalised explanations for recommendations.

proportion of truly needed papers among the recommended ones, aiming to minimize the rate of false positives. We will choose Recall as the evaluated metric on the Cora and Movielens-1M datasets. Therefore, we propose two main research questions:

1. **Does using GraphSAGE or Node2Vec with KNN outperform the direct use of Neo4j's link prediction algorithm for recommendations in the Cora or Movielens-1M datasets in terms of Recall?**

2. **Which query methods can be used for which tasks in the paper domain?**

Based on Research Question 1, we can break it down into two sub-questions:

1. **Sub-research Question 1_1:**Does using GraphSAGE or Node2Vec with KNN outperform the direct use of Neo4j's link prediction algorithm for recommendations in the Cora dataset in terms of Recall?

2. **Sub-research Question 1_2:**Does using GraphSAGE or Node2Vec with KNN outperform the direct use of Neo4j's link prediction algorithm for recommendations in the Movielens-1M dataset in terms of Recall?

For these two sub-questions, we propose the following hypotheses:

- **For the Sub-research Question 1_1:**

- $H0_1$: Using GraphSAGE and Node2Vec with KNN based on Neo4j does not outperform the direct use of Neo4j's link prediction algorithm for recommendations in the Cora dataset in terms of Recall with at least a 1% margin.
- $H1_1$: Using GraphSAGE and Node2Vec with KNN based on Neo4j outperforms the direct use of Neo4j's link prediction algorithm for recommendations in the Cora dataset in terms of Recall with at least a 1% margin.

- **For the Sub-research Question 1_2:**
  - $H0_2$: Using GraphSAGE and Node2Vec with KNN based on Neo4j does not outperform the direct use of Neo4j's link prediction algorithm for recommendations in the Movielens-1M dataset in terms of Recall with at least a 1% margin.
  - $H1_2$: Using GraphSAGE and Node2Vec with KNN based on Neo4j outperforms the direct use of Neo4j's link prediction algorithm for recommendations in the Movielens-1M dataset in terms of Recall with at least a 1% margin.

In this thesis, we will use the number of query results to represent the diversity of the results, serving as an evaluation metric. Additionally, we have reclassified the tasks into six categories, with a detailed explanation provided in Section 3. Therefore, for Research Question 2, we propose the following hypotheses:

1. **For the Adjacency task:**
   - $H0_3$ There is no difference in diversity between those obtained via direct querying and those obtained via latent space with KNN.
   - $H1_3$Direct querying results are more diverse.
   - $H2_3$Latent space with KNN results are more diverse.
2. **For the Accessibility by Links task:**
   - $H0_4$ There is no difference in diversity between results obtained via direct querying and those obtained via latent space with KNN.
   - $H1_4$Direct querying results are more diverse.
   - $H2_4$Latent space with KNN results are more diverse.
3. **For the Common Connection task:**
   - $H0_5$ There is no difference in diversity between results obtained via direct querying and those obtained via latent space with KNN.
   - $H1_5$Direct querying results are more diverse.

- $H2_5$Latent space with KNN results are more diverse.

4. **For the Nodes Attribute task:**
   - $H0_6$ There is no difference in diversity between results obtained via direct querying and those obtained via latent space with KNN.
   - $H1_6$Direct querying results are more diverse.
   - $H2_6$Latent space with KNN results are more diverse.

5. **For the Explore task:**
   - $H0_7$ There is no difference in diversity between results obtained via direct querying and those obtained via latent space with KNN.
   - $H1_7$Direct querying results are more diverse.
   - $H2_7$Latent space with KNN results are more diverse.

6. **For the Hybrid task:**
   - $H0_8$ There is no difference in diversity between results obtained via direct querying and those obtained via latent space with KNN.
   - $H1_8$Direct querying results are more diverse.
   - $H2_8$Latent space with KNN results are more diverse.

1.1.2 Contribution

The main contributions of this project are as follows:

1. Explored the performance of direct querying versus latent space with KNN querying across different task categories. This not only fills a gap in the existing research but also provides actionable insights for improving the design of current paper recommendation systems. This also improves the efficiency of user queries

2. Redefined a task classification method specifically for the paper recommendation field. It increases the relevance of the classification method to the paper domain and helps strengthen the system's understanding of user intent and needs.

3. Proposed a model design combining Node2Vec or GraphSAGE with KNN for the paper recommendation. This addresses gaps identified in the existing research. Experiments prove that our proposed model outperforms link prediction algorithms.

4. Designed a new dynamic, user-interest-based 2D embedding visualization. This approach not only supports personalized recommendations but also improves the explainability of the recommendation results.

1.2 Structure

In Section 2, this thesis will review the literature related to this project. It begins with a summary of the research gap that this project aims to fill. Then, this section will be divided into four subsections: first, a discussion of the most widely used knowledge graph-based recommendation systems; next, a focus on key knowledge graph embedding algorithms; then, an overview of the Neo4j platform, which is central to this project; and finally, a review of existing task classification methods.

Section 3 will describe the process of integrating existing task classification methods to design new tasks for this study. Section 4 will provide an in-depth explanation of the technological pipelines used in this project, including the algorithms and specific configurations. There are two scientific pipelines based on the two research questions. The subsequent Section 5 will cover the datasets, parameter tuning, and analysis of the experimental results, with an explanation of the purpose of each experiment and the decisions made throughout.

In Section 6, this thesis describes the visualization design in detail. The Section 7 will discuss the key findings of the project, outline current limitations, and suggest directions for future work. Finally, the Section 8 will briefly restate the objectives, significance, and outcomes of the project.

2 Related Work

2.1 Knowledge Graph Embedding

Knowledge graph embedding techniques involve embedding KG entities and relations into low-dimensional continuous vector spaces. Research on graph embedding algorithms tailored to knowledge graphs has been evolving. Bordes [3] proposed TransE, a method that interprets relationships as translations operating on low-dimensional embeddings of entities, showing good performance in link prediction tasks, albeit primarily applicable to one-to-one relationships. Building upon TransE, Wang, Z [35] proposed TransH, which models relationships as hyperplanes and performs translation operations. To address the limitation of TransE and TransH modelling within a single semantic space, Lin, Y [22] created a method that embeds relationships and entities in separate spaces, projecting entities into the relationship space and constructing translations to learn embeddings. Ji, G [17]further improved representation models with TranD, using two vectors to represent entities or relationships, one capturing their meaning and the other establishing a dynamic mapping matrix. Qiu, J. [27] introduced DeepWalk, a graph embedding algorithm based on random walks. It generates random walk sequences of fixed lengths by randomly traversing neighbouring nodes in the network, then maps the generated sequences of fixed-length nodes into low-dimensional embedding vectors using the Skip-Gram model. Similarly, Grover, A. [12]presented a graph embedding method based on random walks, Node2Vec, as an improvement over DeepWalk. Node2Vec adjusts parameters for breadth-first and depth-first walk strategies to effectively explore different neighbourhoods, capturing both global and local graph structures. Hamilton, W. [13]introduced GraphSAGE, a framework for inductive representation learning on large graphs. Unlike DeepWalk and Node2Vec, GraphSAGE utilizes node feature attributes to generate node embeddings and achieves this by sampling and aggregating features of nodes' local neighbourhoods. Wu, T. [37] proposed a context-aware dynamic knowledge graph embedding method (DKFG) utilizing two attention graph convolutional networks, gate policies, and translation, providing two different representations for each entity and relationship. Zhang, Q. [42]presented an algorithm that captures relationship structure-context and edge structure-context information through quadratic interaction to model complex relationships, better capturing implicit structural information between entities. Khan, N. [19]introduced the Hashing-based Semantic Correlation Attribute Graph Embedding Enhancement (H-SAGE) algorithm.

In this subsection, we describe a brief history of the development of embedding methods for knowledge graphs, using time as a clue. From this, we can find that the Neo4j platform's embedding methods, Node2Vec and GraphSAGE, which are used in this thesis, are relatively early algorithms, and the platform has not yet been extended with the latest algorithms. This may be an area for future development on the knowledge graph platform.

2.2 Knowledge Graph Recommendation

Khan, N. [18]conducted a categorical study of benchmark datasets for knowledge graph-based recommendation systems from various application domains. These benchmark datasets encompass diverse fields such as e-commerce (e.g., Amazon, Taobao datasets), entertainment (e.g., Movielens dataset), and gastronomy (e.g., Dianping dataset). The expansion of such varied benchmark datasets underscores the current research fervour surrounding knowledge graph-based recommendation systems, with experts and scholars exploring its applications across multiple domains. Knowledge graph-based recommendation systems can be divided into four categories: ontology-based recommendation, linked open data (LOD)-based recommendation, embedding-based recommendation, and path-based recommendation [7].

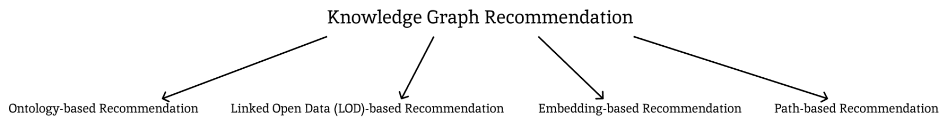

Knowledge Graph Recommendation
Ontology-based Recommendation
Linked Open Data (LOD)-based Recommendation
Embedding-based Recommendation
Path-based Recommendation

Figure 2: Recommendation systems based on knowledge graphs can be divided into four types.

Ontology-based recommendation methods involve utilizing ontologies to represent conceptual hierarchical structures [23]. Herein, an ontology models knowledge about items, contextual knowledge about users, and domain knowledge [40]. In contrast to traditional recommendation systems, ontology-based recommendation systems rely more on user, item, and domain knowledge rather than user ratings. Therefore, their advantages lie in mitigating issues such as cold start and rating sparsity [7] [29]. Zhang, Z [43]addressed the lack of consideration for rich semantic relationships between knowledge in existing traditional collaborative filtering algorithms in e-learning systems by proposing an ontology-based collaborative filtering recommendation algorithm. Similarly, aiming at the e-learning domain, Tarus, JK [29]conducted a literature review study on ontology-based e-learning systems, finding that adopting ontology-based knowledge graph recommendations improved the accuracy and quality of recommendations compared to traditional systems. Additionally, ontology-based knowledge recommendations have been widely applied in the e-learning domain.

Recommendation based on linked open data involves leveraging linked open data to acquire rich semantic information and subsequently computing the similarity between recommended items based on this information [23].

While this approach partially addresses the issue of data sparsity, it tends to overly rely on external data sources.

Embedding-based knowledge graph recommendation involves combining low-dimensional vectors obtained from graph embedding techniques with recommendation techniques [19]. Wang, H. [32] proposed the Knowledge Graph Convolutional Networks (KGCN) model for recommendation. It samples and aggregates neighboring information as entity representations, and then integrates GCN to provide recommendations. Yang, Z. [39]developed HAGERec, a hierarchical attention graph convolutional network for interpretable recommendation, which merges knowledge graphs to explore users' latent interests from the higher-order connectivity structure of heterogeneous knowledge graphs. After embedding users and items, it employs GCN models for bidirectional entity propagation strategies and then utilizes aggregated representations of users and items with predictive-level attention over interactions preserved in entity and neighbour network structures to make recommendations. Zhang, F [41]introduced the Collaborative Knowledge Base Embedding (CKE) recommendation model. It utilizes TranR to obtain structural embedding expressions of nodes and then employs stacked denoising autoencoders and stacked convolutional autoencoders to obtain embedding expressions of text and visual information. Finally, these embedding expressions are integrated with CF models to provide recommendations. Cao, Y. [5] proposed the Translation-based User Preference model (TUP) to better reveal users' latent preferences through integration with knowledge graphs. It includes TransH and a user preference component based on attention mechanisms. Wang, H. [31] introduced RippleNet, a model that integrates paths and embeddings, which can automatically iteratively expand users' interests through relationships between entities, and then infer user preferences. It can also represent entities through multi-hop aggregation and merging domain information to capture users' long-distance interests.

Path-based knowledge graph recommendation utilizes various connection patterns between entities in the knowledge graph to provide additional information for recommendations [31]. Zhao, H [44]proposed a recommendation model based on meta-graphs of heterogeneous information networks (HINs). The model employs matrix factorization and decomposition methods to address information fusion issues. Geng, S [11]introduced the Path Language Modeling Recommendation (PLM-Rec) framework. This model obtains training paths through random walks, then calculates similarity using Sentence-BERT after sequence enhancement operations, and finally explains the recommended paths through Transformer and Nucleus Sampling. Wang, X [34] created a new model named Knowledge-aware Path Recurrent Network (KPRN). It projects relationships between entities and their next nodes, encodes entity and relationship sequence combinations through LSTM, and finally provides recommendations through pooling. Xian, Y [38]proposed a method called Policy-Guided Path Reasoning (PGPR). Based on deterministic Markov decision processes, this method trains an RL agent using reinforcement learning to navigate to potential recommended items. Moreover, the obtained paths can serve as explanations

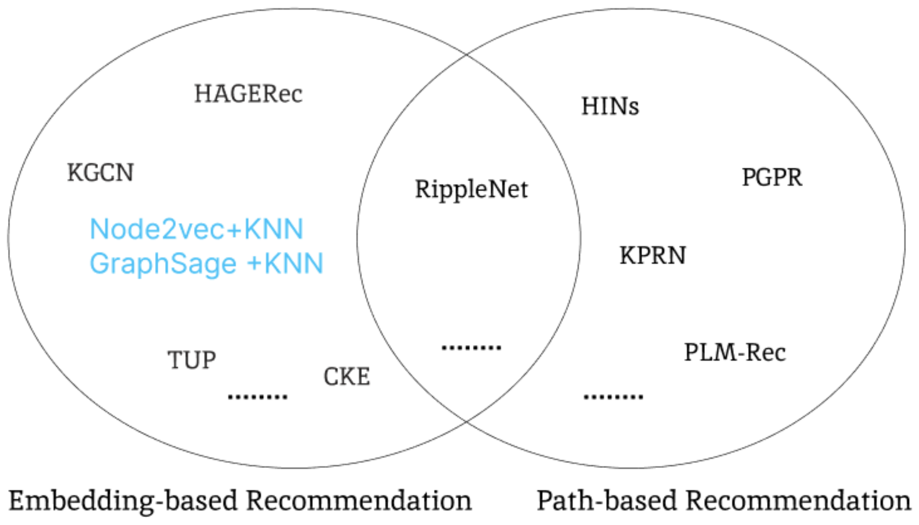


Figure 3: This figure is a summary of the existing embedding-based recommendation algorithms and path-based recommendation algorithms based on the literature review section. We can find that the Ripplnet algorithm combines the methods of embedded recommendation and path recommendation, in the intersection domain. The algorithms shown in blue are the ones used in this paper.

In the second subsection, this thesis briefly introduces four classifications of knowledge graph-based recommendation systems, with a detailed focus on the two main types: embedding-based and path-based recommendation systems. From the literature review, we can find that there is no research on using node2ve or GraphSAGE combined with KNN for recommendation in the field of paper recommendation. This project is designed to fill this gap.

2.3 Research Platform for Study: Neo4j

Neo4j is an open-source graph database software, widely regarded as one of the most popular databases [36]. It can be used to construct knowledge graphs, and when combined with the Cypher query language, it facilitates semantic queries effectively. For instance, Liu, P [24]constructed a knowledge graph of typhoon disasters using Neo4j, enabling a comprehensive understanding of historical typhoon disaster situations through Cypher query language. Dharmawan, I [10]utilized Neo4j's pattern-matching mechanism for book recommendations. Additionally, Neo4j's graph database includes various algorithms to aid research and analysis. Izdihar, A. H [16] employed

Netflix's movie dataset and combined Neo4j's graph database providing Fast Random Projection (FastRP) and the KNN algorithm for movie recommendations. In the sales domain, Dermawan, F [9] analyzed user purchasing patterns using Neo4j's graph database and Jaccard similarity to recommend products. In the financial sector, Patil, A [26] implemented a fraud detection system for banking transactions using Neo4j's graph database and its provided machine learning algorithms. However, there is currently no research on using Neo4j's graph database for paper recommendations.

We can find that studies of Neo4j are in many fields, but there is no research on the paper recommendation domain. Hence, our study will focus on the paper recommendation, which will fill the gap.

2.4 Research for the Task taxonomy

In "Task Taxonomy for Graph Visualization", Bongshin Lee [21] defined a list of graph visualization tasks based on a set of low-level visual analysis tasks. They categorized graph tasks into four types: Topology-Based Tasks, Attribute-Based Tasks, Browsing Tasks, and Overview Tasks.

Table 1 below briefly describes these four categories:

From Table 1, we can observe that Bongshin Lee categorizes graph tasks into nine subcategories. These tasks not only involve finding nodes and paths but also include counting nodes and paths, such as determining how many nodes are connected to a specific node. Such counting tasks might not be frequently used in our domain of paper recommendations. Additionally, if we consider using Cypher language for querying, there is some overlap among these tasks. For instance, in the Attribute-Based Tasks category, the "On the Links" task mentions finding nodes based on specific relationship types, which is almost identical to the Adjacency and Accessibility tasks under Topology-Based Tasks.

Moreover, the table does not fully capture some of the key points from their research. Bongshin Lee notes that the first three task types can be decomposed into a series of lower-level tasks. However, they do not provide a more detailed description of their low-level task portfolio in the Overview Task. Moreover, the Overview Task is the only one that is related to complex algorithms or computations. The other three types of tasks are more about the information contained in the knowledge graph itself, and only the Attribute-Based Tasks contain a computation about the highest or lowest feature value.

Matthew Brehmer and Tamara Munzner [4] also published a paper about the classification of visualisation tasks called "A Multi-Level Typology of Abstract Visualization Tasks". They proposed a multi-level typology of visualisation tasks to clearly distinguish the ends and means of tasks based on Bongshin Lee [21] and Amar 's [1] work on low-level tasks and interactions and high-level tasks. The following Figure 4 illustrates the structure of their

Table 1: Graph Task Taxonomy: Categories, Tasks, and Descriptions [21]

Category	Task	General Descriptions
Topology-Based Tasks	**Adjacency** (Direct Connection)	Find adjacent nodes or find nodes with maximum adjacencies.
	Accessibility (Direct/Indirect)	Find the set of nodes accessible from a node or find the set of nodes accessible from a node with the distance less than or equal to n
	Common Connection	Given nodes, find a set of nodes that are connected to all of them.
	Connectivity	Find the shortest path between two nodes, find bridges, find articulation points or identify clusters.
Attribute-Based Tasks	**On the Nodes**	Find nodes with specific attribute values.
	On the Links	Given a node, find the nodes connected only by certain types of links. The find node is connected by a link having the largest/smallest value.
Browsing Tasks	**Follow Path**	Follow a given path.
	Revisit	Return to a previously visited node.
Overview Task	**Overview**	A compound exploratory task to estimate values, and identify clusters, or patterns.

multi-level visualization task classification method:

Based on the Figure 4, we can clearly find that WHY refers to the purpose of the task, HOW refers to the means of the task, and WHAT refers to the input and output of the task. Their research divides the purpose of tasks into three levels: high-level goals such as "consume" and "produce," mid-level goals such as "search," and low-level goals such as "query." The methods of tasks are categorized into three types: "encode," "manipulate," and "introduce." By breaking down tasks through a clear process involving task inputs, outputs, purposes, and methods, their approach greatly simplifies task descriptions.

However, it is important to note that their research is mainly based on visual task classification, whereas our research is concerned with the classification of graph-based tasks. There are differences between these two domains. Therefore, our subsequent focus will be on the theoretical aspects

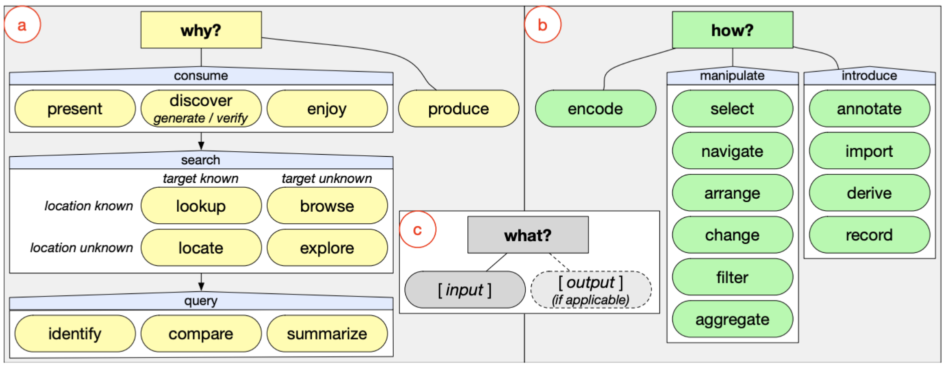


Figure 4: Multi-Level Typology of Abstract Visualization Tasks [4]

of classification related to the purpose of tasks ("why") rather than on the interactive techniques discussed in their study.

In conclusion, we can summarise the task taxonomy method by Matthew Brehmer and Tamara Munzner [4] is an abstract approach that does not target any specific domain and emphasizes the relationship between task targets and methods. On the other hand, Bongshin Lee's graph task classification method [21] provides a rather coarse description of tasks requiring more algorithmic integration, such as Overview Tasks. Moreover, there is currently no direct and clear graph task classification specifically for the domain of paper querying and recommendation. Therefore, this paper aims to redefine graph task classification tailored to the paper recommendation field, integrating insights from these studies, and ensuring the classification can accommodate tasks that require more complex algorithms and computations.

3 Tasks in the Paper Recommendation Domain

Table 2: Redefined Graph Tasks in Paper Recommendation Domain. In the Section 4's Figure 6,these tasks will be as the task pipeline's input. In the Section 5, we will compare the diversity of the direct query and latent Space with KNN's results based on the redefined task classification.

Task Type	**Task Description**	**Example**
Adjacency Task	Find the set of nodes adjacent to the given nodes.	`Give me the author and keywords of articles X and Y.`
Accessibility by Links Task	Given nodes, find the nodes connected only by certain types of links.	`Give me articles my friends of friends like and belong to short paper type.`
Common Connection Task	Given nodes, find a set of nodes that are connected to all of them.	`Give me articles of short article type containing keyword X.`
Nodes Attribute Task	Find the nodes having a specific attribute value.	`Give me the uid of X article.`
Explore Task	Given nodes, find the nodes connected by uncertain types of links.	`Give me some recommendations paper.`
Hybrid Task	Given nodes, find the nodes connected by uncertain types and certain types of links.	`Give me some recommendations for a short paper.`

Existing task classifications [4] [21] have not been specifically designed for the field of academic papers recommendations, nor do they clearly articulate the integration of complex algorithms. This oversight leads to a failure in adequately expressing users' goals and intentions in the realm of paper recommendation. Consequently, this study aims to redefine and reorganize these established task classifications [4] [21]. Given that our entire project is based on knowledge graphs, it is essential to consider the specificity of graph topology. Therefore, the method proposed in this paper categorizes tasks primarily based on graph tasks. The following Table 2 presents our approach to task classification:

First, we have categorized all tasks into six categories. The first category is the **Adjacency Task**, which refers to **finding the set of nodes adjacent to a node.** This definition is directly retained from Bongshin Lee [21]. In the context of our paper recommendation domain, tasks related to this category could include: `"Give me the author and keywords of this article X "` or `"Provide articles containing the keyword Y ."`

The second category is the **Accessibility by Links Task**, which describes the task of **finding nodes connected only by certain types of links.** Unlike the first category, which focuses on direct adjacency, this category emphasizes connections that are not necessarily direct but are made through specified types of links. This is very similar to Bongshin Lee's Accessibility class [21], which highlights indirect connections to known nodes that are already present in the knowledge graph. However, in our definition, we emphasize that the connections must be specified explicitly. This distinction is crucial to differentiate from the Explore tasks, which involve calculating and creating new implicit connections. For example, a task in this category could be: `"Give me articles that my friends of friends like and that belong to the short paper type."`

The third type is the **Common Connection Task**, defined as: Given nodes, find a set of nodes that are connected to all of them. To better illustrate the distinctions between these tasks, we have visualized the three categories mentioned: Adjacency, Accessibility by Links, and Common Connection—as shown in the Figure 5 below:

Based on the Figure 5, we can observe that the **Adjacency task** emphasizes direct connections between the given nodes and the target nodes, and the nodes and relationships of the whole process are known to the user. In contrast, the **Accessibility by Links task** allows for indirect connections between the given nodes and the target nodes. From the given two blue nodes to the red star-shaped target, a path may pass through unknown green nodes. However, the types of relationships along the path are entirely specified. The **Common Connection task**, on the other hand, also involves multiple given nodes, but each of them must be directly connected to the red star-shaped target node. Questions such as `"Give me articles of short article type containing keyword X"` are the Common Connection task.

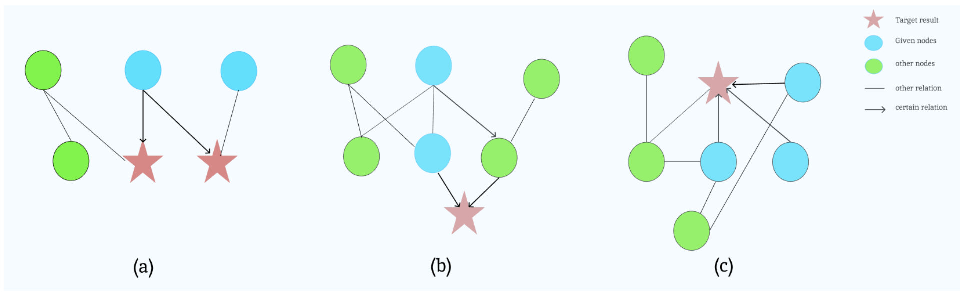


Figure 5: The visual representations of the first three tasks: (a) is the visualization for the Adjacency task, (b) represents the Accessibility by Links task, and (c) illustrates the Common Connection task. In these diagrams, the red stars represent the results obtained by the user.They are the reconmenndations The blue circles indicate the nodes that are already given in the task, while the green circles represent other nodes that are not related to the task. The arrows denote specific types of relationships identified in the task, with the direction of the arrow, indicating the direction in which the task operates within the knowledge graph. Straight lines represent other relationships that are not relevant to the task.

In the previous three task categories, our focus was primarily on tasks derived from the overall topological structure. However, information such as an article's DOI is likely stored as feature information within the nodes of the knowledge graph. Therefore, **the Nodes Attribute task** category emphasizes the intrinsic attributes of the nodes themselves. The task description for this category is **"Find the nodes having a specific attribute value."** In the context of academic papers, this type of task could include queries like `"Give me the DOI of X article."`

The second-to-last category is the **Explore tasks**. The name is derived from the "search" component of the "why" part in Matthew Brehmer and Tamara Munzner's typology [4], which further divides tasks into four parts: lookup, locate, browse, and explore—based on the known and unknown aspects of target and location. Our focus here is specifically on the "explore" part, which deals with unknown targets and unknown locations, emphasizing exploratory discovery.

However, our definition of "explore" differs from Matthew Brehmer's study. We place more emphasis on exploring implicit relationships through algorithms like KNN. Thus, while we retain the term "explore" as described in "Task Taxonomy for Graph Visualization," it specifically refers to tasks where the description is **"Given nodes, find the nodes connected by uncertain types of links."** The key distinction from the Accessibility by Links task is the need for advanced computation and algorithms to uncover hidden connections within the knowledge graph. An example of this task could be: `"Give me some recommendations papers."` In this example, the given node is the User "me", but there is no direct relationship between the user and the recommendations paper. At the same time, we also don't know the explicit relationship path from the given node to the recommendations paper. In our thesis, we use algorithms to create the implicit relationships between users in this task. This is because we choose to give recommendations based on user-to-user similarity.

The final category is the **Hybrid Tasks**, defined as **"Given nodes, which find the nodes connected by uncertain types and certain types of links."** From this definition, it is evident that Hybrid Tasks are very similar to Explore Tasks. The key difference is that Explore Tasks do not include known connections to filter, whereas Hybrid Tasks do include known connections. For the Hybrid Tasks, we need to explore not only the unknown relationships between the given nodes and the target result nodes but also satisfy a portion of the known relationships. The Hybrid Tasks can be viewed as an extension of Explore Tasks with an additional filtering operation. A common example of a Hybrid Task is: `Give me some recommendations for a short paper.`

To tackle all Table 2 tasks, we have two approaches:

1. **Directly Query**: Use the Cypher language to make queries based on known relationships and node or link predictions in the knowledge graph.

2. **Latent Space with KNN** : Transform the problem into latent space combined with KNN for querying.

Table 3: Redefined Graph Tasks in the Movie Domain which will be test in the Section 5 to confirm the general applicability of the relationships between task types and query methods

Task Type	**Example**
Adjacency	`Give me the actors of movie X and Y.`
Accessibility by links	`Give me movies my friends of friends like and belong to genres X.`
Common connection	`Give me films directed by X in the genre of Y.`
Nodes attribute	`Give me the ID of Movie X.`
Explore task	`Give me some recommendations movie.`
Hybrid task	`Give me some recommendations for genres X of movie.`

A detailed introduction to these two query methods will be presented in Section 4, followed by a comparative experimental analysis based on task classification in Section 5.To verify the general applicability of this classification method and the relationships between task types and query methods, this project also conducted experiments in the domain of movie recommendations. The Table 3 shows the tasks performed in the movie domain. The detailed experimental procedures will be described in the Section 5.

4 Pipeline Data science

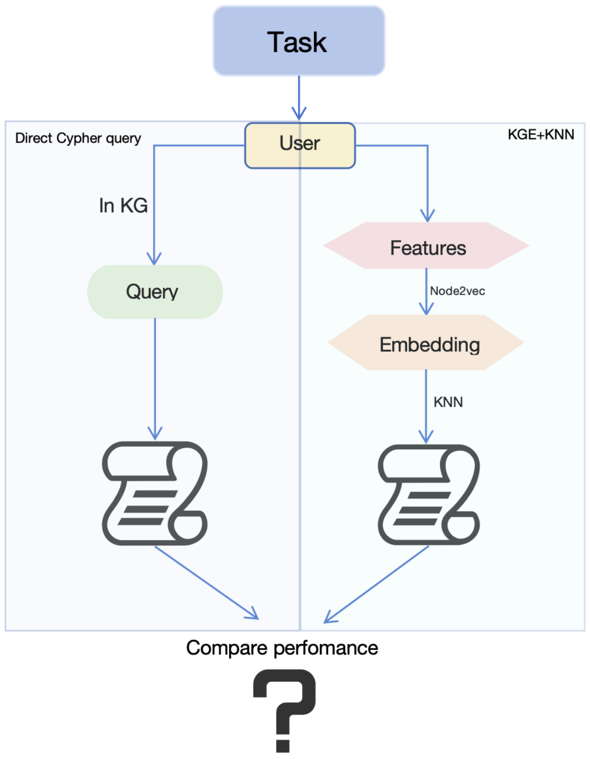


Figure 6: Task Pipeline: The light blue module on the left side of the diagram is the flow for direct querying using cypher. The light green module on the right side of the figure is the flow of the indirect method. The user selects the projected feature space based on interest. Then use the Node2Vec algorithm combined with KNN to give recommendations.

The Task Pipeline primarily addresses Research Question 2: **Which query methods can be used for which tasks in the paper domain?** The Figure 6 illustrates a specific workflow of the Task Pipeline.

For different tasks, such as those mentioned in the Table 2, users have two options. The first option, illustrated by the blue module in the Figure 7, involves directly querying the knowledge graph to obtain results. The second option, represented by the green module in the Figure 7, requires users to first confirm the feature space in which the problem will be projected through feature selection. Subsequently, the Node2Vec algorithm is employed to obtain node embeddings. Finally, these embeddings are combined with KNN to produce the query results. In executing this pipeline experiment, we did not conduct real user studies with the task pipeline. Instead, we created virtual users in the paper dataset for experimentation. Therefore, we cannot assess the two methods based on the accuracy of the recommendations. Instead,

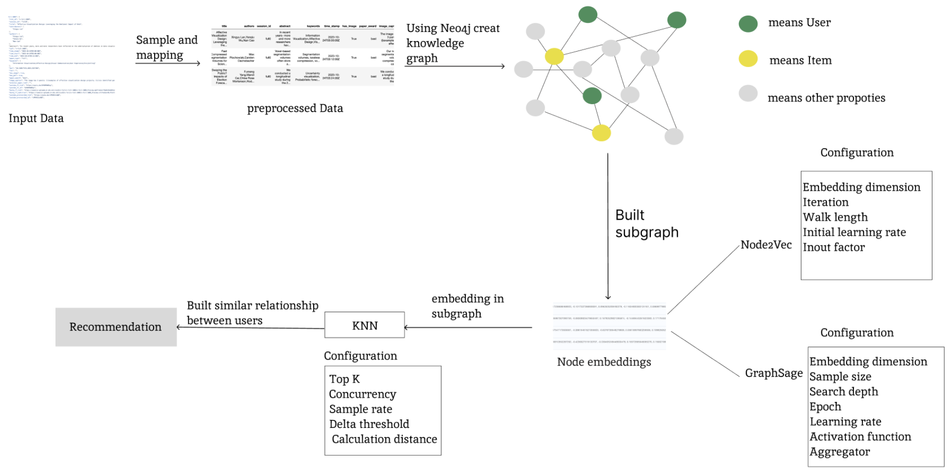


Figure 7: Latent Space with KNN Pipeline for Explore Task: We first process the input data through operations such as sampling and mapping, and then construct the knowledge graph. Next, we obtain node embeddings using Node2Vec or GraphSAGE, and then compute the similarity between pairs of nodes using the KNN algorithm. Finally, we provide Top-K recommendations based on these similarities.

we will evaluate these query methods based on the diversity of the results.

The following subsections will describe the operation of these two query methods and the algorithms they use.

4.1 Measurement Methods of Direct Query Method's Link Prediction Algorithms

For querying unknown relationships in a knowledge graph likes the **Explore Tasks** and **Hybrid Tasks** in the Table 2, the direct querying method needs to use link prediction algorithms. The Neo4j Graph Data Science (GDS) library allows for direct calculation of topological link prediction. Unlike link prediction, which may utilize node attributes, topological link prediction relies solely on the graph's topology to compute the proximity between nodes. It increases the execution speed, but the accuracy of the results is lower than the link prediction algorithm's accuracy. Since most nodes in our constructed paper knowledge graph do not contain attribute information, the impact of using topological link prediction algorithms on accuracy is not significant. Neo4j offers six measurement methods for this purpose: **Adamic-Adar, Common Neighbors, Preferential Attachment, Resource Allocation, Same Community, and Total Neighbors.** The Table 4 below briefly describes these six measurement methods and their formulas.

The table shows that the first three methods calculate closeness based on

Table 4: Different measurement methods and their formulas.These formulations are from`https://neo4j.com/docs/graph-data-science/current/algorithms/linkprediction/`

Method	**Description**	**Formula**
Adamic-Adar(finally choose)	The similarity between two nodes is equal to the weighted sum of their common neighbors. The weight is the inverse of the logarithm of the neighbor node's degree.	$\mathrm{A}(x,y) = \sum_{u\in\mathcal{N}(x)\cap\mathcal{N}(y)} \frac{1}{\log\lvert\mathcal{N}(u)\rvert}$
Common Neighbors	The more common neighbors two nodes share, the closer they are considered to be. The number of common neighbors represents the closeness between nodes.	$\mathrm{CN}(x,y) = \lvert\mathcal{N}(x)\cap\mathcal{N}(y)\rvert$
Resource Allocation	If two nodes share many common neighbors, they are more likely to form a connection. The fewer connections the common neighbors have, the closer the two nodes are considered to be.	$\mathrm{RA}(x,y) = \sum_{u\in\mathcal{N}(x)\cap\mathcal{N}(y)} \frac{1}{\lvert\mathcal{N}(u)\rvert}$
Preferential Attachment	The more connections a node has, the more likely it is to connect to a new node. The closeness between nodes is represented by the product of the number of neighbors of the two nodes.	$\mathrm{PA}(x,y) = \lvert\mathcal{N}(x)\rvert \times \lvert\mathcal{N}(y)\rvert$
Total Neighbors	The closeness between nodes is calculated using the union of the neighbors of the two nodes.	$\mathrm{TN}(x,y) = \lvert\mathcal{N}(x)\rvert + \lvert\mathcal{N}(y)\rvert$
Same Community	The closeness is calculated based on the principle that nodes in the same community are more likely to be connected.	**No specific formula**

the common neighbors between nodes. These methods yield more accurate results but take longer to execute. On the other hand, the**Same Community** method relies entirely on community detection to determine closeness. Although it is highly accurate,Neo4j does not provide a clear formula for this method, we do not plan to use it in our project.

The **Preferential Attachment** and **Total Neighbors** methods calculate closeness based on the number of neighbors of the nodes themselves, following the principle that nodes with more connections are more likely to form new connections. These methods are heavily influenced by high-degree nodes but have faster runtime. In this project, we aim to calculate the proximity between users. Relying solely on their number of connections seems inadequate. This is because we will create virtual users in the paper dataset, where the number of books read by each user will be identical. In such cases, all nodes will have the same number of neighbors, and direct queries based on **Preferential Attachment** and **Total Neighbors** will not differentiate between users.

Therefore, we are primarily considering the **Adamic-Adar, Common Neighbors, and Resource Allocation** methods. The **Common Neighbors** algorithm directly uses the number of common neighbors to represent similarity. However, it does not account for the varying significance of different neighbors in real scenarios, which makes it unsuitable for our needs.

Our paper's knowledge graph has an uneven degree distribution. For example, category nodes have much higher degrees compared to other nodes, while keywords nodes have lower degrees. Compared to **Resource Allocation, Adamic-Adar** includes an additional logarithmic operation. This operation helps to balance the influence of high-degree and low-degree common neighbors more effectively. Thus, despite being more complex than **Resource Allocation**, **Adamic-Adar**is preferred for our project.

4.2 Latent Space to Learn Semantic Relations and KNN to Retrieve Query-related Recommendations

The pipeline about "Latent Space with KNN" shown in Figure 6, primarily focuses on the paper dataset and does not include the GraphSAGE component. Additionally, there is no detailed explanation of the parameters used in this process. The Figure 7 provides a more detailed illustration of the entire **Latent Space with KNN** workflow for **Explore Tasks** as applied in our thesis. This process will also be employed for the Cora and Movielens datasets. In this design, we will directly utilize the Node2Vec and GraphSAGE algorithms available in the Neo4j Graph Data Science (GDS) library to obtain node embeddings. We will then combine these embeddings with the KNN algorithm to provide recommendations.

Using the paper data that we will be working with as an example, we first need to preprocess the dataset. This includes converting the paper dataset from JSON files to CSV files, removing special characters, and dividing keywords as required by the project. For the subsequent experiments, such as those using the MovieLens and Cora datasets, additional steps for data sampling and comparisons are needed. For instance, the MovieLens dataset is combined with data from DPMedia to construct a knowledge graph. Therefore, we need to match the movie IDs from the MovieLens ratings with the movie IDs from DPMedia.

Detailed procedures for each dataset will be described in the Section 5 of this paper.

After preprocessing the data, we will use Neo4j to build the knowledge graph. It is important to note that these operations are performed on a subgraph within Neo4j. This subgraph construction will help in selecting user-relevant features for the recommendation system. Specifically, in Neo4j, algorithm operations are conducted within the subgraph, so we need to carefully select node types and relationship types to construct the subgraph. For example, if user interest is focused on session times, we would add the session ID node type and the corresponding presented_in relationship to the subgraph.

Then, we will use Node2Vec or GraphSAGE to obtain node embeddings. Parameters such as embedding dimensions, iterations, initial learning rate, and sample size will be set accordingly. The embeddings will be written to the knowledge graph, and the KNN algorithm will be employed to compute the similarity between node pairs, specifically user node pairs. Recommendations will then be made based on the principle that similar users tend to like similar items.

Next, this thesis will detail the three algorithms used in the pipeline of latent space with KNN.

4.2.1 Using Node2Vec to Get Embedding

Node2Vec generates sequences of nodes by using random walks on a graph. In this thesis, we use the `"gds.Node2Vec.write"` function provided by the Neo4j platform to generate node embeddings and write the embeddings into the knowledge graph. The Table 5 below shows the parameters that will be adjusted in the subsequent experiments.

4.2.2 Using GraphSAGE to Get Embedding

Unlike Node2Vec, which is based on random walks, GraphSAGE is an inductive framework that efficiently generates embeddings for new nodes by leveraging node attribute information. It samples nodes and generates node representations by aggregating neighborhood information. GraphSAGE can seamlessly integrate node attribute information. The following Figure 8 illustrates the workflow of the GraphSAGE algorithm. Note that this image is from the article Inductive Representation Learning on Large Graphs [13].

From the Figure 8, we can observe that GraphSAGE first samples neighbors, then aggregates the features of nodes layer by layer, and finally performs downstream tasks. This paper utilizes the GraphSAGE algorithm from Neo4j's Graph Data Science library. First, the model is trained using the "gds.beta.graphSage.train" function before executing the node embeddings. Then, the results are written back into the knowledge graph using the "gds.beta.graphSage.write" function. In this design, we focus on tuning the parameters of the "gds.beta.graphSage.train" function, as its configuration is more complex and has a greater impact on the results. The Table 6 also

Table 5: Part Configuration of the Node2Vec Algorithm.In the Section 5.2,we will use these configurations to do tuning parameter experiments.

Configuration	Description
Embedding Dimension	The size of the node embeddings, with a default value of 128. If the embedding dimension is too high, it significantly increases training time and noise. However, if it is too low, the resulting embeddings may not fully capture the node information.
Iteration	The number of training iterations, with a default value of 100. If the number of iterations is too low, the model may not be adequately optimized. However, too many iterations may lead to overfitting.
Walk Length	The number of steps taken in each random walk. A higher value allows the embedding to capture more global structural information.
Initial Learning Rate	The initial learning rate used for training the neural network. The learning rate decreases after each training iteration. If the initial learning rate is too high, the model may miss the optimal solution. However, if it is too low, the training duration will be very long, and the model may not learn adequately.
InOut Factor	Controls whether the random walk tends towards depth-first or breadth-first exploration. A higher value indicates a stronger tendency to capture local information.

presents the configuration of the `"gds.beta.graphSage.train"`function.

From the Table 6 we know that the search depth is related to the sampling step, while the choice of aggregator is related to the second aggregation step of the GraphSAGE algorithm. The table also shows the details of these configurations, setting the stage for subsequent tuning.

4.2.3 Using KNN to Find Node Similarity

The KNN algorithm calculates the distance between node pairs based on node attributes. This algorithm uses various traditional similarity measures such as Pearson correlation, Spearman rank correlation, cosine similarity,

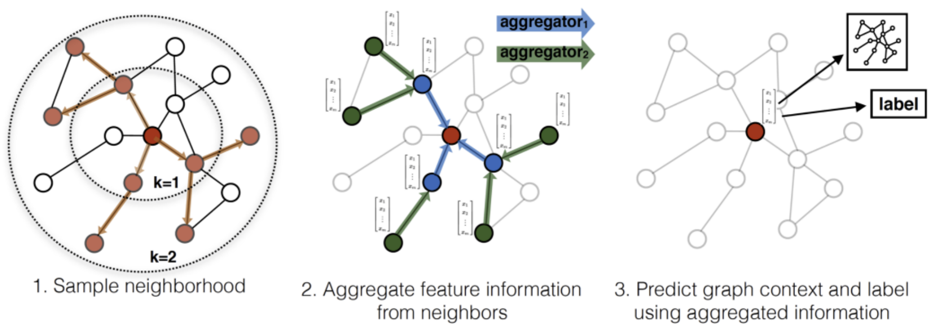

aggregator1
aggregator2
k=1
k=2
label
1. Sample neighborhood
2. Aggregate feature information from neighbors
3. Predict graph context and label using aggregated information

Figure 8: A visual description of the GraphSAGE sample and aggregate [13].The GraphSAGE algorithm completes the embedded representation of nodes in the first two steps and then performs downstream tasks such as link prediction or node classification.

Table 6: Configuration of the GraphSAGE Algorithm. In the Section 5.2, we will use these configurations to do tuning parameter experiments.

Configuration	**Description**
Embedding Dimension	The size of node embeddings and their hidden layers. If the embedding dimension is too high, it will significantly increase training time and memory usage. If the dimension is too low, the embedding may not fully capture the information.
Sample Size	The size of each sample represents one hidden layer. More layers allow node embeddings to consider further neighbors.
Search Depth	The depth of random walks in each batch. If this value is too high, it may become difficult to distinguish between similar nodes.
Learning Rate	The extent to which the weights are updated after each iteration.
Epoch	Represents the maximum number of training cycles. It helps to limit training time and prevent overfitting.
Activation Function	Can be chosen from `Sigmoid` and `Leaky ReLU`.
Aggregator	Can be chosen from `MEAN` and `POOL`. `MEAN` is simpler and faster, while `POOL` is more complex but better at encoding domain-specific information.

Jaccard similarity, etc. When using the KNN algorithm in Neo4j, we applied the node embeddings obtained from Node2Vec or GraphSAGE as node properties and then used the `"gds.knn.write "`function to create "similar" relationships between users within the knowledge graph. The `"gds.knn.write"` algorithm has many configuration options, but we mainly focused on the calculation distance parameter. The following table is about the relationship between the method of measuring distance and the type of embedding.

Table 7: Property type and Metric Configurations.In the Section 5.2,we will use these metric to do the tuning parameters experiments.

Property Type	**Metric**
List of Float	`COSINE, EUCLIDEAN, PEARSON`
List of Integer	`JACCARD, OVERLAP`

By observing the node embeddings produced by Node2Vec or GraphSAGE, we found that the embedding values are lists of floating-point numbers. Therefore, the available distance measurement methods were limited to `COSINE`, `EUCLIDEAN`, and `PEARSON`.

`EUCLIDEAN` measures the geometric distance between two points and is a standard metric with intuitive results, but it is generally used for two- or three-dimensional spatial data [15]. `COSINE` measures the cosine of the angle between vectors and is often used for text data. However, it only considers the direction between vectors, which may introduce some errors [15]. `PEARSON` measures the linear correlation between two vectors and is typically used for continuous data with linear relationships, though it may fail to detect non-linear patterns accurately [28].

Because the node embeddings we obtained are high-dimensional, non-linear continuous data, we could not determine theoretically which distance measurement method would perform better. Therefore, we used all three measurement methods simultaneously to evaluate their performance in our subsequent experiments.

5 Experiment

5.1 Experiment Data

In this thesis, we utilize three datasets. The Paper dataset is used to build the actual paper recommendation system and to explore the relationship between task classification and query methods. The Cora dataset is smaller and simpler compared to both the Paper and Movielens datasets, and it allows for a rapid evaluation of the performance of our Node2Vec or GraphSAGE combined with KNN design. The Movielens dataset is very large and complex, resembling the real-world application scenario of our model design. Experiments conducted on this dataset provide a better assessment of our algorithm's performance in complex, real-life settings.

5.1.1 Experiment Data 1:Paper data

In this design, we primarily utilized the dataset of papers presented at the 2023 conference, collected from the vis-virtual website conference website. The dataset named paper_list.json is located at sitedata/2023. The website address is: `https://github.com/ieee-vgtc/vis-virtual-website/blob/master/sitedata/2023/paper_list.json`.

This paper dataset comprises 388 papers and encompasses 28 attributes such as authors, abstracts, and keywords, among others. Based on this dataset, we ultimately created 7 types of nodes: authors, paper,time_stamp, time_end,paper_type,paper_award, keywords,session_id).In the paper node, it contains other properties such as uid,has_image and so on. And we create 7 types of relationships. The Table 8 shows the 7 relationship types we established.

Table 8: 7 Relationships in the Knowledge Graph

Source Node	**Target Node**	**Relationship Type**
paper	keywords	contains_keywords
paper	authors	written_by_authors
paper	session_id	presented_in_session
paper	time_stamp	published_at_time
paper	time_end	finished_at_time
paper	paper_award	has_award
paper	paper_type	belongs_to_type

However, an important issue with this dataset is that it does not contain user information. Therefore, to address Research Question 2, we need to artificially create user data within this dataset. Additionally, we will introduce a new node type, "User," and create a "read" relationship between users and papers within the knowledge graph. The **??** below presents the

Table 9: Detailed information about nodes and relationships.

Node Name	# Nodes	Relationship Name	# Relationships
authors	1331	written_by_authors	1748
keywords	970	contains_keywords	1234
paper_award	2	has_award	22
paper_type	4	belongs_to_type	388
session_id	64	presented_in_session	388
time_end	103	finished_at_time	388
time_stamp	99	published_at_time	388
User	4	read	20
paper	388		

assumed information for the virtual users we have created. Among others, Dan and Matt, Matt and Jeff, and Matt and Annie are all friends with each other.

Table 10: Virtual users and the papers read.Each user read five articles, with Dan and Annie reading only one different article. Theoretically Dan and Annie should be very similar. This ensures that when we use Dan as a user to do the **Explore Task** experiment in the Section 5.3, there should theoretically be no zero recommendations.

User Name	Articles Read
Dan	1.Gridded Glyphmaps for Supporting Spatial COVID-19 Modelling 2.TimePool: Visually Answer "Which and When" Questions On Univariate Time Series 3. ExoplanetExplorer: Contextual Visualization of Exoplanet Systems 4. WUDA: Visualizing and Transforming Rotations in Real-Time with Quaternions and Smart Devices 5.Design of an Ecological Visual Analytics Interface for Operators of Time-Constant Processes
Annie	1.Gridded Glyphmaps for Supporting Spatial COVID-19 Modelling 2.TimePool: Visually Answer "Which and When" Questions On Univariate Time Series 3.ExoplanetExplorer: Contextual Visualization of Exoplanet Systems 4.WUDA: Visualizing and Transforming Rotations in Real-Time with Quaternions and Smart Devices

	5.Augmented Scale Models: Presenting Multivariate Data Around Physical Scale Models in Augmented Reality
Matt	1.CLEVER: A Framework for Connecting Lived Experiences with Visualisation of Electronic Records 2.Explain-and-Test: An Interactive Machine Learning Framework for Exploring Text Embeddings 3.What Is the Difference Between a Mountain and a Molehill? Quantifying Semantic Labeling of Visual Features in Line Charts 4.Do You Trust What You See? Toward A Multidimensional Measure of Trust in Visualization 5.Augmented Reality for Scholarly Publication of 3D Visualizations in Astronomy: An Empirical Evaluation
Jeff	1.Simulating the Geometric Growth of the Marine Sponge *Crella Incrustans* 2.Automatic Scatterplot Design Optimization for Clustering Identification 3.Too Many Cooks: Exploring How Graphical Perception Studies Influence Visualization Recommendations in Draco 4.Planar Symmetry Detection and Quantification using the Extended Persistent Homology Transform 5.Towards Adaptive Refinement for Multivariate Functional Approximation of Scientific Data

Dan is the user we will use as the subject of our study. This means that if a user interacts with our recommendation system, they will automatically take on the role of Dan. In designing this user, we hypothesized some extreme scenarios based on the types of articles read. For example, there might be users who have only read short papers or users who have all but one paper in common. In our virtual user design, both Dan and Matt have only read short papers. Annie and Dan have four same papers, with the only difference being one paper, which in Annie's case is of the full type. Dan and Annie should be very similar. Then there should theoretically be no zero recommendation when experimenting with the **Explore Task** in the Section 5.3. Jeff's papers and their types are entirely different from Dan's. The types of papers Jeff has read, in the order listed in the table, are short, full, full, workshop, and associated. The reason I chose to design based on paper types is that, in the subsequent web design, we need to visualize the latent space of nodes.

In this thesis, we opted to use PCA to reduce the dimensionality of high-dimensional node embeddings. By applying the functions: `pd.DataFrame`, `PCA()` and `pca.fit_transform()`, the high-dimensional node embeddings are reduced to two dimensions. The choice of two dimensions is based on the fact that users find 2D visualizations easier to understand. The horizontal axis represents the direction of the largest variance in the node embeddings, while the vertical axis represents the second-largest variance direction. After performing PCA dimensionality reduction on the high-dimensional embeddings of paper data (excluding users), we observed that the paper nodes formed four clusters. Since the paper types also fall into four categories, we initially attempted to assign different colors to the nodes in the PCA plot based on the paper types. It turned out that these four clusters were indeed distinguished by paper types. The following Figure 9 shows the visualization results after PCA dimensionality reduction. We can see that paper type is a significant feature, so I chose to design users based on this feature.This will amplify the distance between similar and dissimilar users even more. Test whether the link prediction algorithm or the Node2Vec combined with KNN algorithm can successfully give recommendations when executing the **Explore Task** and **Hybird Task**.

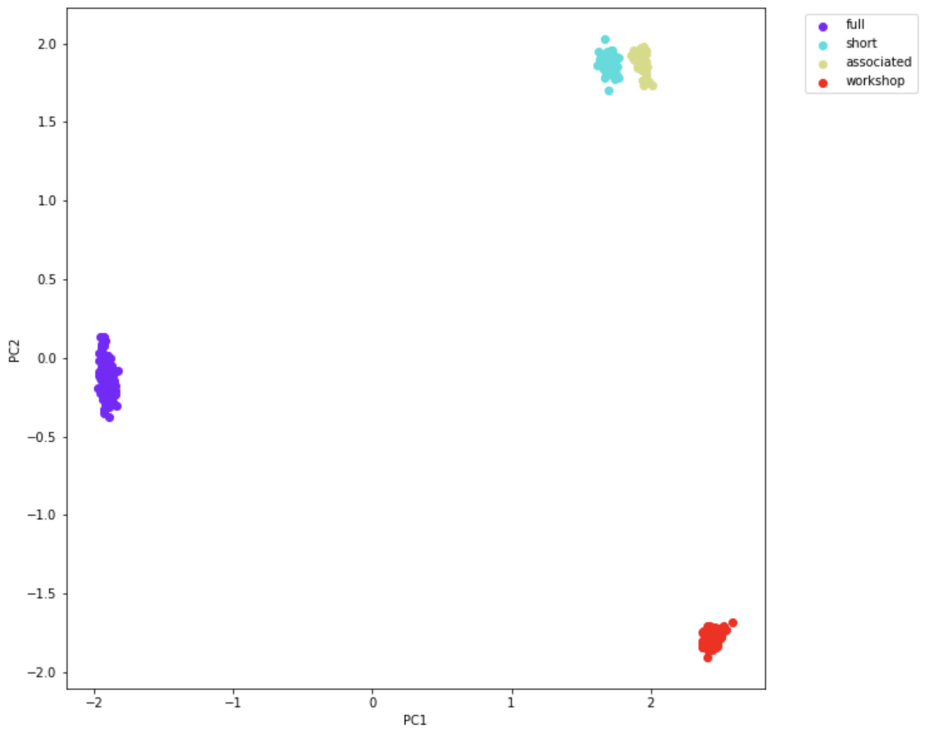
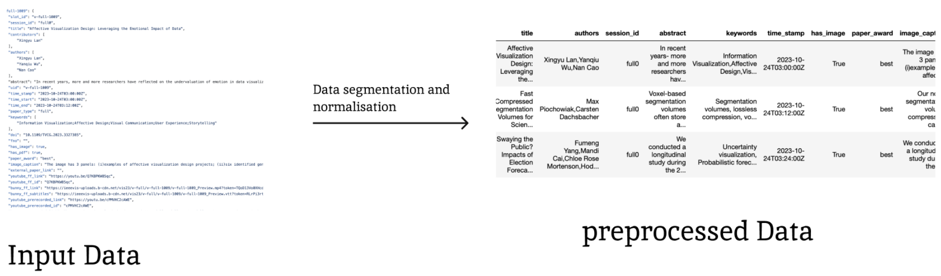


Figure 9: PCA plot of paper data in 255 dimensions. Where the paper nodes are divided into four clusters according to article type. We can find the paper_type is the significant feature. Therefore, this thesis design users based on paper_type

After confirming the necessary node types and relationship types for the paper knowledge graph, we needed to preprocess our raw data. The image below shows our preprocessing process. The left side of the image represents the content of the JSON file obtained directly from the website. We first traverse through the papers and clean the data by removing spaces and hyphen symbols. Then, we convert all the data into comma-separated string types. Since the dataset is relatively small, we did not do a data sampling step.



```
full-1009": {
 "slot_id": "v-full-1009",
 "session_id": "full0",
 "title": "Affective Visualization Design: Leveraging the Emotional Impact of Data",
 "contributors": [
     "Xingyu Lan"
 ],
 "authors": [
     "Xingyu Lan",
     "Yanqiu Wu",
     "Nan Cao"
 ],
 "abstract": "In recent years, more and more researchers have reflected on the undervaluation of emotion in data visualiz
 "uid": "v-full-1009",
 "time_stamp": "2023-10-24T03:00:00Z",
 "time_start": "2023-10-24T03:00:00Z",
 "time_end": "2023-10-24T03:12:00Z",
 "paper_type": "full",
 "keywords": [
     "Information Visualization;Affective Design;Visual Communication;User Experience;Storytelling"
 ],
 "doi": "10.1109/TVCG.2023.3327385",
 "fno": "",
 "has_image": true,
 "has_pdf": true,
 "paper_award": "best",
 "image_caption": "The image has 3 panels: (i)examples of affective visualization design projects; (ii)six identified gen
 "external_paper_link": "",
 "youtube_ff_link": "https://youtu.be/Q7KBPKW85qc",
 "youtube_ff_id": "Q7KBPKW85qc",
 "bunny_ff_link": "https://ieeevis-uploads.b-cdn.net/vis23/v-full/v-full-1009/v-full-1009_Preview.mp4?token=TQoD13VoBXHcd
 "bunny_ff_subtitles": "https://ieeevis-uploads.b-cdn.net/vis23/v-full/v-full-1009/v-full-1009_Preview.vtt?token=RLrPi3rt
 "youtube_prerecorded_link": "https://youtu.be/cPMVHC2cAWE",
 "youtube_prerecorded_id": "cPMVHC2cAWE",
```

title	authors	session_id	abstract	keywords	time_stamp	has_image	paper_award	image_capt
Affective Visualization Design: Leveraging the...	Xingyu Lan,Yanqiu Wu,Nan Cao	full0	In recent years- more and more researchers hav...	Information Visualization,Affective Design,Vis...	2023-10-24T03:00:00Z	True	best	The image 3 pan (i)example affec
Fast Compressed egmentation Volumes for Scien...	Max Piochowiak,Carsten Dachsbacher	full0	Voxel-based segmentation volumes often store a...	Segmentation volumes, lossless compression, vo...	2023-10-24T03:12:00Z	True	best	Our n segmenta volu compress ca
Swaying the Public? Impacts of Election Foreca...	Fumeng Yang,Mandi Cai,Chloe Rose Mortenson,Hod...	full0	We conducted a longitudinal study during the 2...	Uncertainty visualization, Probabilistic forec...	2023-10-24T03:24:00Z	True	best	We conduc a longitud study du the

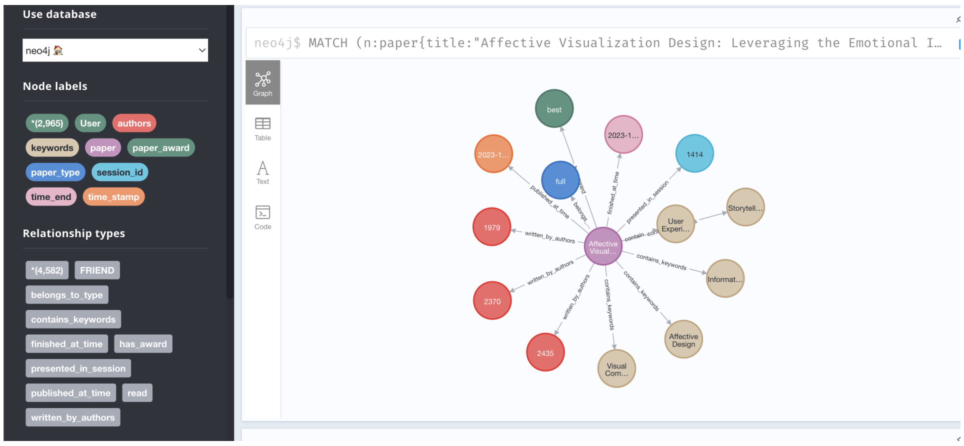
Figure 10: Paper data preprocessing process. The content of the input json file is used to build the knowledge graph after data cleaning, segmentation and normalisation operations.For example, the information in the json file for the paper **"Affective Visualisation Design: Leveraging the Emotional Impact of Data "** shown on the left side of the figure. Then through data segmentation and normalisation, the data of the article is stored in a form that becomes the first row on the right side of the figure.

The preprocessed data will be directly used to build the paper knowledge graph. The paper knowledge graph established through Neo4j is shown in the Figure 11 below.

In the end, we established a total of 2,965 nodes and 4,576 relationships. Among these, the authors node type is the most frequent with 1,331 instances, and the written_by_authors relationship type is also the most common, totalling 1,748. However, we observed that the number of has_award relationships is less than 388. The award node type is divided into best and honourable. Since not all papers receive awards, the count of has_award relationships being 22 is expected. Detailed information on the 2,956 nodes and 4,576 relationships is shown in the Table 9.

5.1.2 Experiment Data 2:Cora

The Cora dataset consists of 2708 scientific publications and 5429 cited links. Each paper entity can be represented by a 1,433-dimensional feature vector. In this dataset, we take the 1,433-dimensional feature vector of a paper as an attribute of each paper node. In the knowledge graph, it contains only one node type i.e. paper node. As well, there is only one relation type. We randomly select 80% of the 5,429 cite relations as our training set to build the knowledge graph. The remaining 20% is used as a test set to go for the link prediction task. Although the task of our overall project is a recommendation task, which seems to be slightly different from a link prediction task.



```
neo4j$ MATCH (n:paper{title:"Affective Visualization Design: Leveraging the Emotional I…
```

Figure 11: Knowledge graph of paper data. The top left side shows node type information and the bottom left side shows relationship type information. The right side is a complete display of the paper **"Affective Visualisation Design: Leveraging the Emotional Impact of Data "** , which is also shown in the Figure 10.

But since we are making recommendations based on user-to-user proximity, i.e., we want to predict links between users. Whereas the Cora dataset only contains paper nodes, it seems reasonable to use our recommendation design for paper-to-paper cite relationship prediction. The Cora dataset has a smaller data volume and a simple knowledge graph structure compared to the Movielesn dataset we introduced later. It greatly reduces the experiment time.

Since Neo4j's database contains the Cora dataset, we called it directly from within Neo4j. We randomly selected 4343 cited relations as the training set and 1086 cited relations as the test set in the Cora_cites file. For the file Cora_content, raw[0] is used as the ID of the paper node, and then raw[2……1434] is used as the feature of each node. The visualisation of the knowledge graph based on the Cora dataset in Neo4j is shown below:

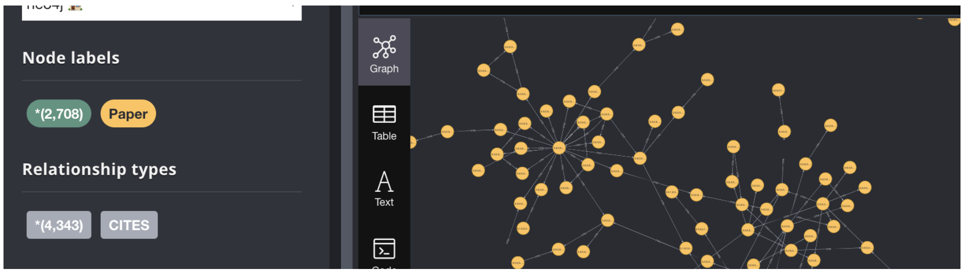


Figure 12: The visualization of Cora Knowledge Graph in Neo4j

5.1.3 Experiment Data 3:Movielens

We utilized the MovieLens-1M dataset, combined with DBpedia, to construct the MovieLens Knowledge Graph. In this knowledge graph, we created nine node types: country, director, film, genre, language, user, rating, star, person_or_entity_appearing_in_film and writer. Additionally, we established 12 relationship types. The following Table 11 provides a description of each relationship type:

Table 11: 12 relationship types in the MovieLens Knowledge Graph.

Source Node	**Target Node**	**Relationship Type**
film	director	film_director
film	genre	film_genre
film	language	film_language
film	writer	film_writer
writer	film	writer_film
user	film	likes
director	film	film_director_film
actor	film	actor_film
film	star	film_star
film	rating	film_rating
person_or_entity _appear- ing_in_film	film	personorentityappearinginfilm_film
film	country	film_film_country

Since the interactions between users and movies in MovieLens are numerous, We then randomly selected 85,768 interactions. These interactions were then split into a training set (train.txt, containing 67,814 interactions) and a test set (test.txt, containing 16,954 interactions).

Finally, we use the traing set to create the knowledge graph. The graph contains a total of 26,108 nodes and 131,147 relationships. Among these are 5,802 user nodes, 2,445 film nodes, 6274 actor nodes and 23 genres, among others. The interaction information between users and films, represented by the likes relationship, comprises 67,814 entries.The detaild information is shown in the Table 12.

Since the knowledge graph constructed from the MovieLens dataset will also be used to compare the performance of different query methods across different tasks in the Section 5.3, we randomly select a user as the target user, which the user ID is 116.

Table 12: Node and Relationship Information

Node Name	**# Node**	**Relationship Name**	**# Relationship**
director	1567	film_director	2581
genre	23	film_genre	5274
language	73	film_language	3135
writer	2441	film_writer	3476
film	2445	writer_film	3472
user	5802	likes	67814
actor	6247	actor_film	18254
country	11	director_film	2581
star	1251	film_country	2256
rating	5	personorentity appearingin-film_film	18244
personorentity appearinginfilm	6243	film_rating	1899
		film_star	2164

5.2 Experiments Comparing the Recall of Link Prediction with Latent Space with KNN on benchmark sets

In this subsection, we will utilize the Latent Space with KNN pipeline mentioned in the Section 4.2and the link prediction algorithm discussed in the Table 4 to conduct the precision comparison based on the Cora and MovieLens datasets described in Section 5.1.2 and Section 5.1.3. In this thesis, we will use recall as the evaluation metric.

5.2.1 Experiments Comparing the Recall of Link Prediction with Latent Space with KNN on Cora

As mentioned in the Section 5.1.2, we will use the sampled training set to construct the knowledge graph. We then apply Node2Vec or GraphSAGE for graph embedding. Afterwards, we perform K-nearest neighbor (KNN) analysis on each node pair. If a node pair in the top-K contains a pair from the test set, it is considered a positive match. We calculate recall by dividing the number of correctly selected node pairs by the total number of node pairs in the test set. As a baseline, we directly use the link prediction algorithms to create node pairs, then use the same approach to calculate recall for comparison. In this thesis, we fix the value of K to be 10.This value is more moderate and can be used to a certain extent without missing the similarity relationship. the Table 5 and Table 6 in the Section 4.2 illustrates that the Node2Vec and GraphSAGE algorithms have multiple adjustable parameters. The Figure 13 presents the specific operational workflow on the Cora dataset, incorporating the parameter settings shown in the Table 5 and Table 6 .

Experiments of Using Node2Vec with KNN on the Cora First, we will introduce the experiments using the Node2Vec algorithm combined with KNN.As shown in the Table 5, there are numerous parameters to consider for Node2Vec. Initially, we fixed the parameters walklength=80 and iteration=10 to save training time , and used the default values for other parameters. We then tuned the embedding dimensions. The experiments began with an embedding dimension of 60 because this dimension is higher than the number of feature types, placing it in the high-dimensional category while still ensuring shorter runtime. As the dimension was gradually increased to 65, there were no significant changes in the results. To test further, the dimension was increased directly to 85, but the results remained similarly unchanged. Consequently, an ultra-high dimension of 1225 was applied to observe any further effects. The embedding dimensions varied from 60, 61, 62, 63, 64, 65, 85, 105,125, up to 1285, with the KNN distance method set to Pearson correlation. A total of 67 experiments were conducted. Detailed experimental results are shown in the Appendix B. The results of matching each paper-paper pair obtained from the experiments with the test set pairs

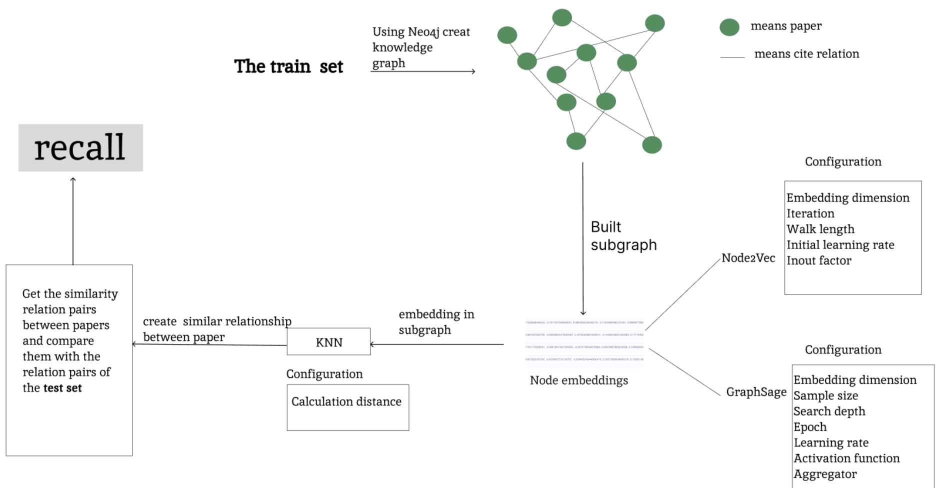


Figure 13: Flowchart of running Node2Vec or GraphSAGE combined with KNN on the Cora dataset.For the Cora dataset, this thesis will conduct parameter tuning experiments for the Node2Vec algorithm using five parameters outlined in Table 5. Due to the complexity of the GraphSAGE algorithm, we will perform parameter tuning based on seven parameters listed in the Table 6. After obtaining the node embeddings, KNN will be used to create the similarity relationships between papers. In this step, we will primarily focus on tuning the `"Calculation Distance"` parameter, as described in Section 4.2.3.

are summarized in the Figure 17 below.

From the figure, we can observe that the number of successful matches consistently hovers around 630, while our test set contains a total of 1086 paper pairs. This indicates that our recall value is approximately 0.58. To rule out the influence of different distance measurement methods used in the KNN algorithm, we conducted 67 additional experiments within the same range of dimensions, using `EUCLIDEAN` and `COSINE` distance metrics, respectively. The area plots of the dimensions and the number of successful matches obtained are shown below:

Then, by comparing the results of these 201 experiments, we can observe that, overall, the method using `EUCLIDEAN` distance measurement performed the best. Its maximum value reached 662, but even with this value, the corresponding recall was only around 0.61. Therefore, we can confirm that, under the parameter settings of walklength=80 and iteration=10, the embedding dimension does not significantly impact the results. The Appendix B shows the results of the 201 experiments. Consequently, we proceeded to adjust the walklength parameter.

First, we reduced the walklength from its original value to 10, becasue we wanted to test walklength from a small value. To shorten the exper-

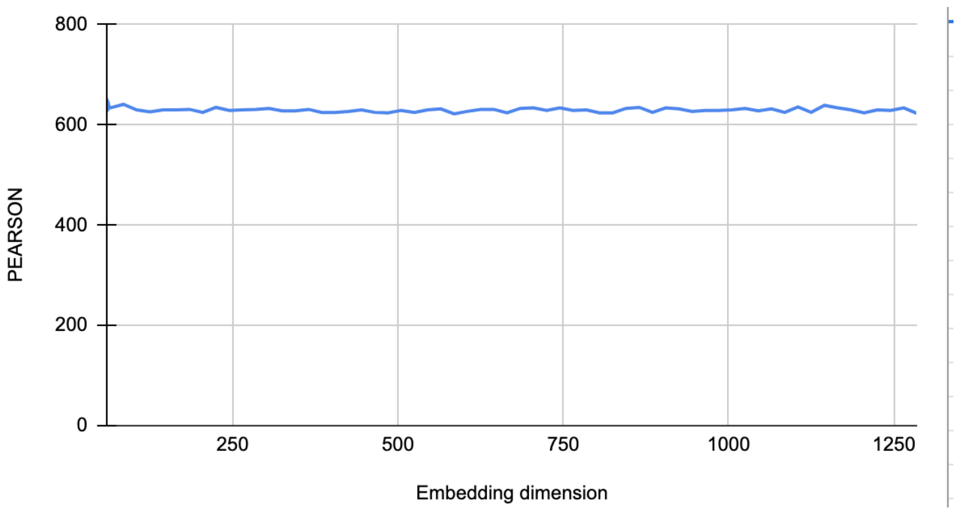


Figure 14: Dimension setting line chart based on Pearson measure method. We can find that changes in the embedding parameters have almost no effect on the results.

iment time, we selected lower embedding dimensions, ranging from 60 to 64, and conducted experiments with all three measurement methods.The Section C.1 shows the results of the experiment The results of these experiments are shown in Figure 16a. We observed that the number of successful matches significantly decreased compared to the previous experiments. Therefore, we increased the walklength to 40, and the results are shown in Figure 16b and Section C.2. We noticed an overall improvement compared to the walklength=10 scenario, but it still fell short of the results obtained with walklength=80.

Thus, we considered that the walklength should be adjusted in the direction of increasing from 80. We then set the walklength to 100, and the results are shown in Figure 16c and Section C.3. The number of successful matches improved across all dimensions. Finally, we increased the walklength directly to 200, and the experimental results are shown in Figure 16d and Section C.4. It is important to note that, to reduce experiment time, the embedding dimensions were chosen to range from 10 to 55, with a step size of 5.

The walklength is increased to 200 and we can see that the results of the experiment have improved. However, the highest value is still around 667, which is not a significant improvement compared to walklength=100. Therefore, we concluded that a walklength of 100 is a suitable value. With walklength=100, we conducted another set of experiments focusing on the embedding dimensions ranging from 5 to 1000 and performed 21 experiments under each of the three distance measures. The final results are shown in

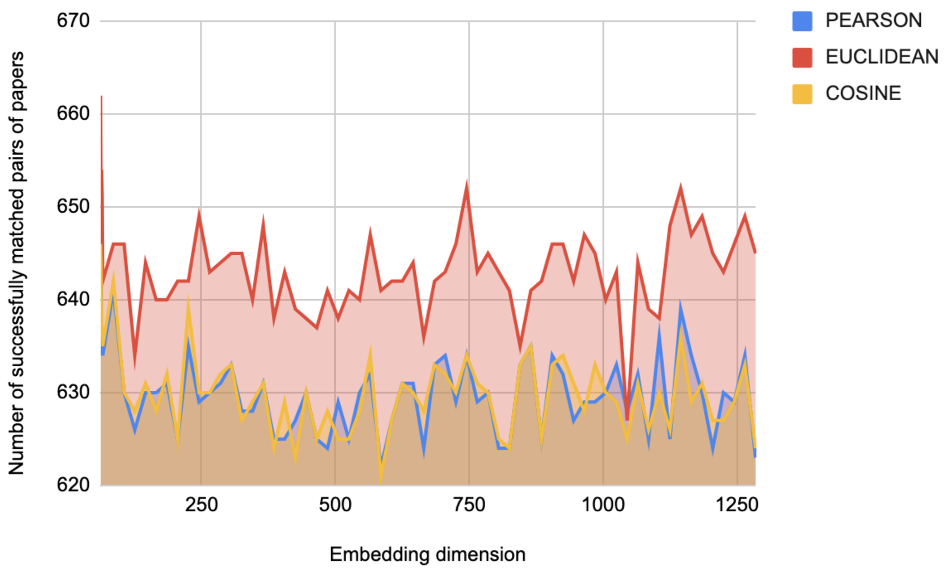
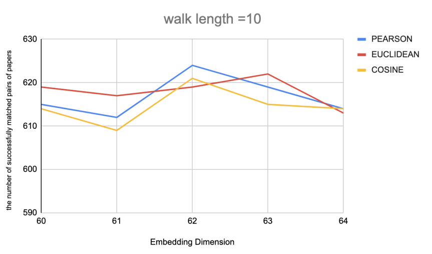


Figure 15: Dimension setting area chart based on three measurement methods. We find that the use of the `EUCLIDEAN` outperforms the other two methods in all dimensions, and the highest value can exceed 660.

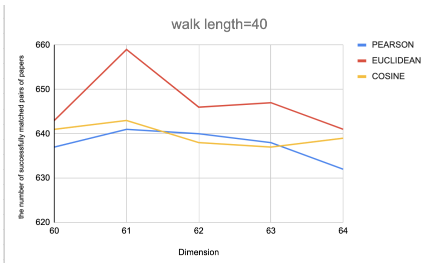
the figure below:

After conducting these 120 tuning experiments on the walklength parameter, we observed that the walklength value should not be too small. For instance, when the walklength was set to 10, there was a significant decrease in the number of successfully matched paper pairs. However, increasing the walklength excessively did not lead to a significant improvement in results and only added to the experiment's runtime. Therefore, we chose to set the walklength value to 100 for subsequent experiments. Next, we focused on tuning the iteration parameter. Initially, with the walklength fixed at 100, we set the iteration to 30 to observe whether increasing the number of iterations would significantly improve the results. The experimental results are shown in Figure 18. We found that when the embedding dimension exceeded 10, there was no significant increase in the number of successful matches. Therefore, we can assume that the limit value of the embedding dimension is around 10. Additionally, when comparing these results to those obtained with iteration = 10 and walklength = 100 in the Figure 16c, the number of successful matches remained within the 640-670 range. To rule out the influence of walklength, we also conducted experiments with iteration = 30 and walklength = 40. The results of these experiments are shown in Figure 19.

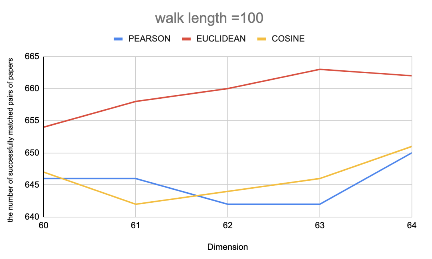
When the dimension value exceeded 260, the results began to deteriorate as the value increased. Overall, the maximum number of successful matches did not exceed 650, which is consistent with the results observed



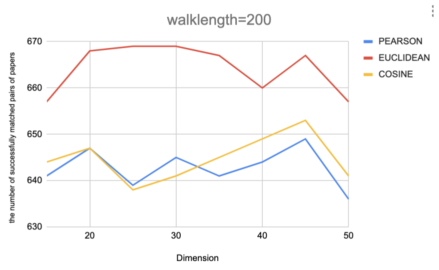
(a) Line chart of walklengh =10. We can find there is not much difference in the results of the three measures. Moreover, there is an overall decrease in the number of successful matches compared to the previous results



(b) Line chart of Walklengh = 40.The number of successful matches increased, with the highest value being close to 660. and the measurement method using `EUCLIDEAN` outperformed the other two methods.



(c) Line chart of Walklengh = 100.The number of successful matches increased for all three methods, with the highest value exceeding 660.



(d) Line chart of Walklengh = 200.The number of successful matches increased for all three methods, with the highest value near 670. and the method using `EUCLIDEAN` outperformed the other two methods.

Figure 16: Line charts of the walklength tuning.There is a slow increase in the number of successful matches as walklength increases. However, the overall change is not significant. The number of matches is still around 660.

when iteration was set to 10. After conducting these 45 experiments, we can preliminarily conclude that the iteration value does not have a significant impact on the experimental results.

Next, we will tune the initial learning rate parameter. It is important to note that in the previous tuning experiments, we did not specifically adjust the initial learning rate parameter, so the default value of 0.01 was used. To quickly determine the appropriate direction for tuning the initial learning rate, we conducted experiments with an initial learning rate of 0.001 and 0.01 under the condition of iteration set to 10. Similar to the previous approach, we conducted experiments within the embedding dimension range of 15-55 and used three different distance measurement methods. The experimental results are shown in the figure below:

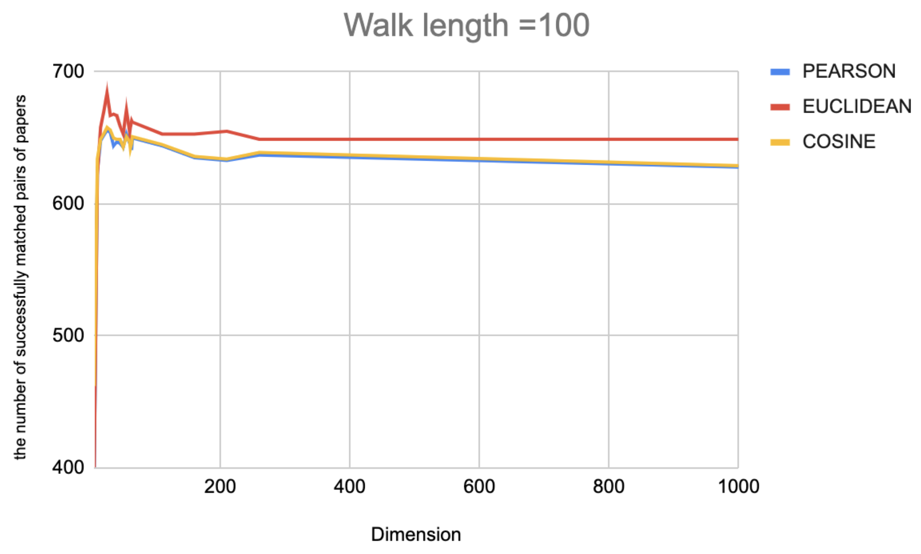


Figure 17: Line chart of the walklength =100 and embedding dimension from 5 to 1000. The number of successful matches did not change significantly as the dimensions changed. Although a spike is seen in the graph. But this could be due to the randomness of the Node2Vec algorithm. Detailed experimental results are in the Section C.3

From the two figures, we can clearly observe that the results obtained with an initial learning rate of 0.001 are lower across all dimensions compared to those obtained with an initial learning rate of 0.01. Therefore, we can hypothesize that increasing the initial learning rate may lead to correctly matching more paper pairs. To further test this hypothesis, we set the iteration value to 30, as we previously assumed that iteration does not significantly impact the experimental results. By using iteration = 30, we can further validate our earlier assumptions. Subsequently, with the Walklength set to 100, we conducted experiments with initial learning rates of 0.1, 0.15, 0.25, and 0.5. The experimental results are shown in the Figure 22 below:

Comparing Figure 22a with Figure 22b, the number of successfully predicted papers with an `initial learning rate of 0.15` is higher than that with an initial learning rate of 0.1 in all dimensions. Therefore, we further increase the `initial learning rate to 0.25` and take a very extreme value of `initial learning rate = 0.5`. The results are shown in Figure 22cand Figure 22d respectively. At initial learning rate=0.25, as the embedding dimension increases, the number of correct predictions falls off a cliff at dimension 45. Also at the `initial learning rate=0.5`, all the values are for 347. For these two cases, it may be that there is an overlearning condition because the initial learning rate is too high. To further explore this overlearn-

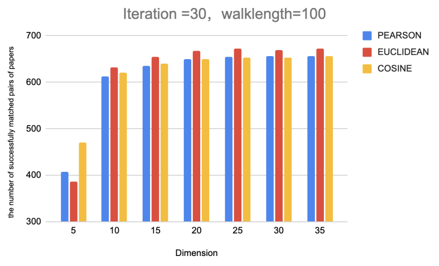


Figure 18: Bar chart of walklength =100.Matching results are worst when the dimension is equal to 5, while the difference in matching results is not significant as soon as the embedding dimension exceeds 10. The limit value of the embedding dimension is around 10.Detailed experimental results are in the Section D.1.

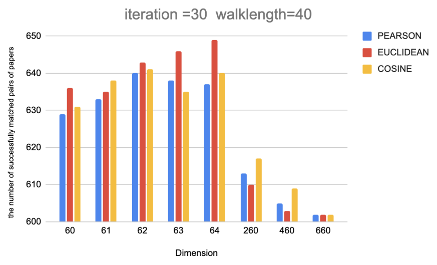


Figure 19: Bar chart of walklength=40.The highest value of matches is close to 650, but lower than the highest value for iteration=10 in the Figure 16b. At dimensions starting from 260, the matches start to decrease with increasing dimensions. Finally, at the dimension equal to 660, the three methods get the same match value. Detailed experimental results are in the Section D.2.

ing boundary, we chose 0.34 which is between 0.25 and 0.5 to experiment again. The results of the experiment are shown below.

From Figure 23a, it is evident that when the `initial learning rate is set to 0.34` and the `dimension is at least 25`, the number of successful predictions consistently remains at 347. This likely indicates a case of overfitting. Based on the 105 previous experiments, we observed that `the initial learning rate of around 0.1` yields better performance, when `iteration is equal to 10`. Consequently, we conducted 15 additional experiments with the `initial learning rate set to 0.1` and increased the number of `iterations to 50`. The results are shown in the Figure 23b However, the results showed no significant improvement with the increased iterations. This further confirms our earlier conclusion that, for the Cora dataset, the iteration parameter has no significant impact on the results when running the Node2Vec algorithm in combination with KNN.

Finally, we conducted experiments focusing on the `inOutFactor` parameter, which controls the distance of a node's random walk from the initial node. The default value is 1.0, and a higher value implies that the nodes tend to walk farther away. Based on previous experiments, we initially set `walklength=100`, `iteration=30`, and `initial learning rate=0.1`. We then increased the `inOutFactor to 3 and 6`, aiming to capture global information from the knowledge graph. After conducting 30 experiments, Figure 24a and Figure 24b show the results for `inOutFactor set to 3 and 6`,

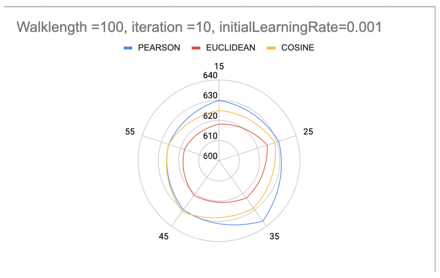


Figure 20: Radar chart of initial learning rate=0.001.The `Pearson`measurements gave better results than the other two methods. Moreover, the result interval is 610-635, which is the same as the result interval obtained in the Figure 16a at an initial learning rate of 0.01. Detailed experimental results are in the Section E.1.

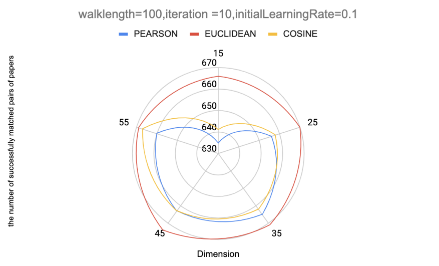


Figure 21: Radar chart of initial learning rate=0.1.The `EUCLIDEAN` measure gives better results than the other two methods. Moreover, the maximum value of the results is around 665. This result is much larger than the one presented at an initial learning rate of 0.001 as depicted in theFigure 20. Detailed experimental results are in the Section E.2.

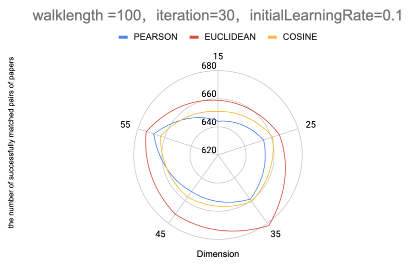
respectively. The results with `inOutFactor=3` generally outperformed those with `inOutFactor=6`. When compared with the results in Figure 22a, we observed that increasing the `inOutFactor from 1 to 3` did not result in any significant improvement overall.

To further validate the effect of the `inOutFactor`, we conducted 5 additional experiments with `iteration=50` and `inOutFactor=3`. As shown in Figure 24c, there is an overall improvement compared to Figure 23b, but the maximum value still did not exceed 680. Therefore, we conclude that adjusting the `inOutFactor` does not significantly enhance the recall value of the experiment.

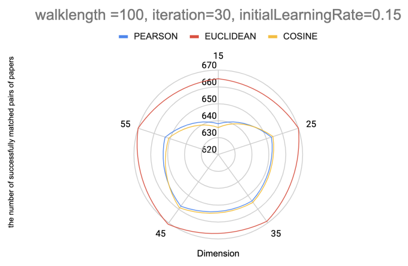
We conducted 537 parameter-tuning experiments using the Node2Vec algorithm combined with KNN on the Cora dataset. Through these experiments, we observed that regardless of how we adjusted the parameters `embedding, iteration, initial learning rate, walklength, and inOutFactor`, the highest experimental results remained around 680. Consequently, our recall value stayed around 0.63. This value is not particularly impressive. To further investigate the reasons behind this, we analyzed the loss per iteration values obtained from the previous Node2Vec experiments. The following Figure 25 illustrate the loss per iteration for two different setups: one with `Embedding=325, iteration=10` and other parameters set to their default values in the Figure 25a. Another with `Embedding=35, walklength=100, iteration=30` and other parameters set to their default values in the Figure 25b.

We visualized the loss per iteration values for the results obtained from 179 different parameter settings using the Node2Vec algorithm. The loss per iteration graph for each experiment is provided in the appendix of this paper. Here, we only present two of these loss per iteration graphs. However, we observed that the loss did not converge in any of the Node2Vec experiments. This could explain why parameter adjustments did not result in significant improvements. In the previous experiments, to save time, I set the number of iterations relatively low. To make the experiments more convincing, I increased the number of `iterations to 10000`. The Figure 26 illustrates the results.

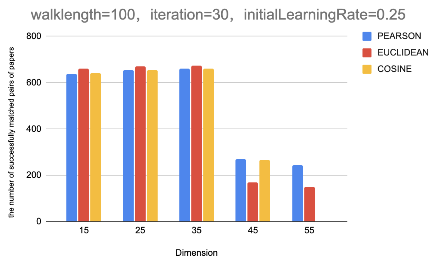
In this case, we set the `embedding dimension to 30`. This choice was made partly because this value did not perform poorly in previous results, but primarily to reduce runtime. With this setting, the code took approximately 16 hours to run. Higher embedding dimensions would significantly increase the time cost. As shown in the figure above, the loss did not converge. Therefore, we can confirm that the primary reason for our failure to optimize the parameters was the lack of convergence in the Node2Vec algorithm on our dataset. Finally, we took the best result from these tuning attempts, with a maximum successful match count of 680, resulting in a **recall of 0.626 for the Node2Vec algorithm combined with KNN on the Cora dataset**.



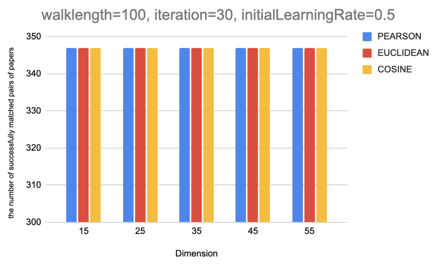
(a) Radar chart of initial learning rate=0.1 and iteration =30.Compared to the Figure 21, iteration increases. However, the overall resultant interval did not change either. Detailed experimental results are in the Section E.3.



(b) Radar chart of initial learning rate=0.15.The number of successful matches has risen, with a maximum value of nearly 670.

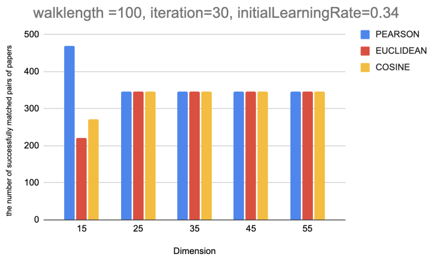


(c) Bar chart of initial learning rate=0.25.The number of successful matches decreases. And, there is a significant decrease in the number of matches at dimension 45 . Detailed experimental results are in the Section E.4.

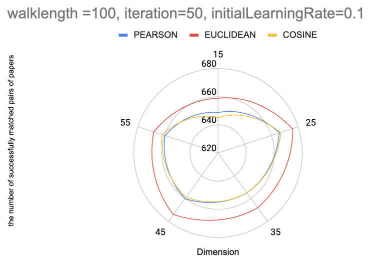


(d) Bar chart of initial learning rate=0.5.The number of successful matches has remained at the value of 347. An overlearning problem may have occurred. Detailed experimental results are in the Section E.5.

Figure 22: charts of the initial learning rate tuning based on iteration =30. The initial learning rate of 0.15 gives better values, but it is not better thanFigure 21.

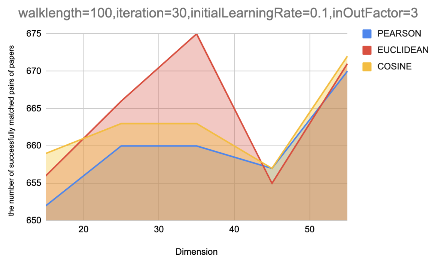


(a) Bar chart of `initial learning rate=0.34`.The results obtained remained stable at 347 as the dimensionality increased. This indicates that the experiment appeared to be over-learning. Detailed experimental results are in the Section E.6.

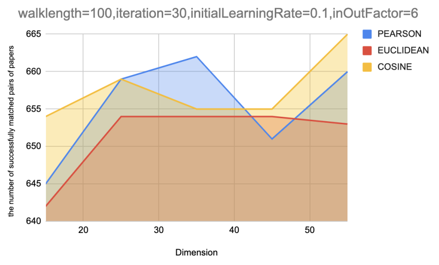


(b) Radar chart of `initial learning rate=0.1` and `iteration = 50`.Compared to the Figure 21, the number of iterations is increased. However, the results do not show a significant improvement. Detailed experimental results are in the Section E.7.

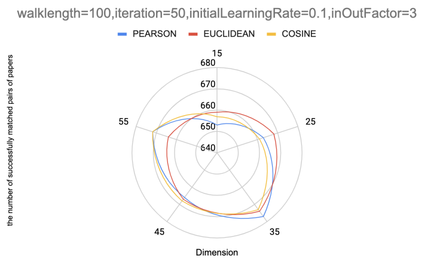
Figure 23: Checked experiments about initial learning rate and iteration



(a) Area chart of `inOutFactor=3`.The highest value for a successful match is 675. this is not a significant improvement over the results in the **??**. Detailed experimental results are in the Section F.1.

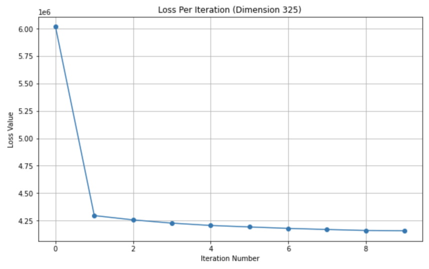


(b) Area chart of `inOutFactor=6`.The highest value for a successful match is 665, lower than the results in the **??**. Detailed experimental results are in the Section F.2.

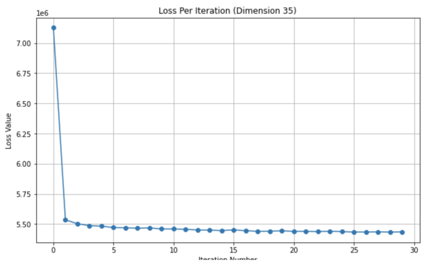


(c) Radar chart of `iteration=50`, `inOutfactor=3`.The maximum value of the obtained matching results exceeds 670. however, the results are not significantly improved compared to the Figure 24a. Detailed experimental results are in the Section F.3.

Figure 24: Charts of the `inOutFactor`tuning



(a) line chart for `Embedding=325,` `iteration=10` and other parameters are default values. We can find that the loss did not converge.



(b) line chart for `Embedding=35,` `walklength=100,iteration=30` and other parameters are default values. Like Figure 25a, we can find that the loss also did not converge.

Figure 25: Parts of lossPerIteration charts. We can find that the loss did not converge, regardless of the parameter settings.

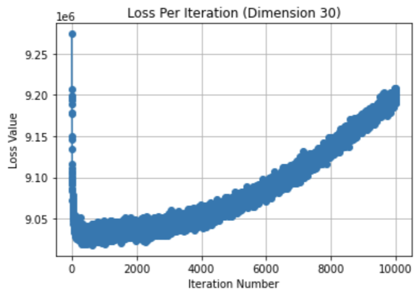


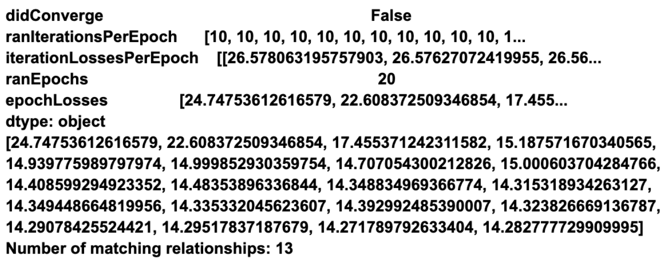
Figure 26: Loss chart of `itration=10000, embedding =30` and other parameters are default values. We can find the loss still did not converge.

Experiments of Using GraphSAGE with KNN on the Cora Next, we will discuss the experiments involving GraphSAGE combined with KNN on the Cora dataset. As mentioned in the Table 6, the parameters that can be adjusted for the GraphSAGE algorithm include`embedding dimension, sample size, search depth, epoch, learning rate, activation` function, `and aggregator`. Based on our previous experience with Node2Vec, we found that confirming data convergence is crucial before performing detailed parameter adjustments. In Neo4j, the GraphSAGE algorithm's training mode returns a Boolean value did converge to directly indicate whether the training has converged. However, to more closely observe changes in loss, we visualized Iteration Losses per Epoch to detect any significant fluctuations in the model. Since didConverge directly indicates whether the data has converged, we did not visualize epochLosses for convergence detection.

Initially, we set the parameters to `learningRate=0.005, epoch =20, batchSize=30, aggregator=POOL, searchDepth=5 and embedding` Dimension=255. These initial values were chosen based on parameter settings used in other GraphSAGE experiments. We then conducted tuning experiments for the epoch parameter. The following Figure 27 shows the results of the GraphSAGE algorithm with these parameter settings, including the successful match values for the training set and our visualization of Iteration Losses per Epoch.

From the Figure 27, we can immediately observe that the training did not converge, and the number of successfully predicted papers was only 13. Consequently, we conducted further experiments with `epoch values of 3 and 40`, keeping all other parameters constant. It was found that an epoch of 3 resulted in a higher number of successful matches. Therefore, we tentatively assumed `epoch=3` as the optimal setting and proceeded with tuning the `Aggregator parameter`. Changing the `Aggregator to MEAN` led to a significant improvement in the results. Therefore, we assumed `MEAN` to be the optimal Aggregator. Next, we conducted experiments on the `Activation Function`. We found that switching the `Activation Function to ReLU` resulted in a substantial improvement, with the number of successful matches increasing from around one hundred to over four hundred. Therefore, we continued to use `ReLU as the Activation Function` for the `Search Depth` experiments. The experiments with `Search Depths of 10 and 3` revealed that a `search depth of 5` yielded the best results. The outcomes of the seven experiments mentioned above are summarized in the Table 13.

Although we found that a `Search Depth of 5` yielded better results than a depth of 3, the difference between the two was not substantial. Considering the issue of runtime, we decided to use the `Search Depth` of 3 as the basis for further parameter tuning. We then conducted experiments on the `Sample Size` parameter, starting with the initial value of [10, 10]. We tested four different configurations: `[3, 3], [5, 5], [8, 8], and [15, 15].` The results, as shown in the Figure 28, indicate that the configura-



```
didConverge                                          False
ranIterationsPerEpoch      [10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 1...
iterationLossesPerEpoch    [[26.578063195757903, 26.57627072419955, 26.56...
ranEpochs                                             20
epochLosses              [24.74753612616579, 22.608372509346854, 17.455...
dtype: object
[24.74753612616579, 22.608372509346854, 17.455371242311582, 15.187571670340565,
14.939775989797974, 14.999852930359754, 14.707054300212826, 15.000603704284766,
14.408599294923352, 14.48353896336844, 14.348834969366774, 14.315318934263127,
14.349448664819956, 14.335332045623607, 14.392992485390007, 14.323826669136787,
14.29078425524421, 14.29517837187679, 14.271789792633404, 14.282777729909995]
Number of matching relationships: 13
```

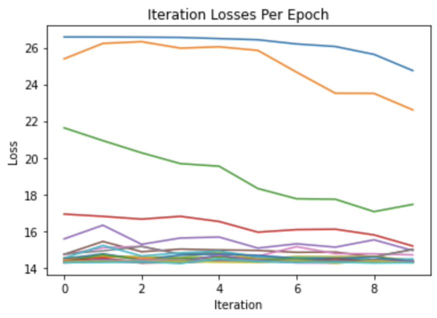


Figure 27: Plot of experimental results with `epoch of 20`. The top part is information about the results returned using the GraphSAGE train model. The number of matched relations in the middle intermediate part is the number of relations predicted by the algorithm that appear in the test set. The bottom half is a visualisation of the iteration Losses per Epoch data. We can find the training did not converge and the number of successfully predicted papers was only 13.

Table 13: Experimental Results for GraphSAGE in the Cora. We can find the highest matched number is 436. Detailed experimental results are in the Section G.1.

Epoch	didConverge	Aggregator	Activation Function	Search Depth	Number Matched
20	FALSE	POOL	Sigmoid	5	13
40	FALSE	POOL	Sigmoid	5	7
3	FALSE	POOL	Sigmoid	5	26
3	FALSE	MEAN	Sigmoid	5	129
3	FALSE	MEAN	ReLu	5	436
3	FALSE	MEAN	ReLu	10	405
3	FALSE	MEAN	ReLu	3	414

tion `[8, 8]` produced the highest number of successful matches, reaching 433. Based on this, we proceeded to tune the learning rate with the `Sample Size set to [8, 8]`. Drawing on previous experience, we decided to avoid a high learning rate to prevent overfitting. The `initial learning rate` `was 0.005`, so we reduced it to 0.001 for this experiment. Given the inherent randomness of the GraphSAGE algorithm, we ran the experiment twice, with results of 588 and 613, respectively. Detailed experimental results are in the Section G.3.

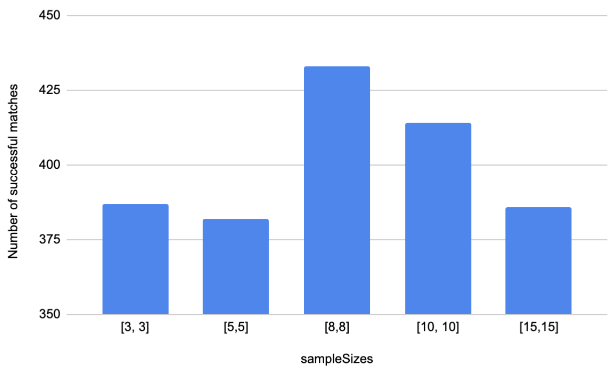


Figure 28: Bar graph of different `Sample Size`.We can observe that `Sample Size =[8,8]` have the highest matched number,433. Detailed experimental results are in the Section G.2.

At this point, we have conducted 13 experiments using GraphSAGE combined with KNN on the Cora dataset. However, none of these experiments showed convergence, and overall, the performance was inferior to that of Node2Vec. From the earlier experiments, it is evident that the results from the Node2Vec algorithm consistently exceeded 610. The reason we did not perform as detailed parameter tuning for the GraphSAGE algorithm as we did for Node2Vec is that we already recognized that tuning is of limited value when the training does not converge. Finally, we conducted experiments to assess the impact of the embedding dimension on convergence, testing dimensions of 255 and 25. The results confirmed that the training still did not converge. At this stage, we decided to halt further tuning of GraphSAGE on the Cora dataset, as parameter tuning is ineffective when the data does not converge, leading only to endless experimentation. Additionally, the algorithm itself is inherently stochastic. We therefore used the highest observed value, 613, to calculate the recall for GraphSAGE combined with KNN, which resulted in a final recall of 0.56.

Experiments of Using Link prediction on the Cora In the previous section on algorithm descriptions, we analyzed several topological link prediction algorithms provided by Neo4j and ultimately decided to use Adamic-Adar. We employed the `gds.alpha.linkprediction.adamicAdar` algorithm to compute scores for paper node pairs regarding the CITES relationship. Comparing the results with the test set, we found that the number of correct matches was only 61, which is significantly lower than the results obtained using Node2Vec or GraphSAGE combined with KNN. Although we initially selected `Adamic-Adar` based on theoretical analysis, we also conducted link prediction experiments using the other five algorithms. The results of these experiments are summarized in the Table 14 below.

Table 14: Topological Link Prediction Results. We can find that using `Total Neighbors` can get the highest number,566.

Topological Link Prediction	**Number of Matching Relationships**
`Adamic Adar`	61
`Common Neighbors`	110
`Preferential Attachment`	254
`Resource Allocation`	110
`Same Community`	0
`Total Neighbors`	**566**

As shown in the table, the `Total Neighbors` method achieved the highest number of successful matches, with a total of **566**. Therefore, we used 566 as the successful match result for Neo4j's link prediction, yielding a recall of 0.521. At this point, the experiments for the three algorithms on the Cora dataset have been completed, and their final recall values are presented in the Table 15. We find the Node2Vec+KNN gets the highest recall And the recalls of latent space with KNN methods are higher than link predictions'.

Table 15: Recall Values for Different Algorithms. Using Node2Vec+KNN gets the highest recall.

Algorithm	**Recall**
`Node2Vec with KNN`	0.626
`GraphSAGE with KNN`	0.56
`Topological Link Prediction`	0.521

5.2.2 Experiments Comparing the Recall of Link Prediction with Latent Space with KNN on Movielens

For the second sub-problem, we will conduct experiments on the Movielens dataset. The overall experimental process will be similar to that used on the Cora dataset. The key difference is that in this dataset, we need to establish user-to-user similarity and generate user-movie recommendation pairs based on the principle: `(u)-[:similar]->(other)-[:likes]->(film) WHERE NOT (u)-[:likes]->(film)`. Finally, these pairs will be compared with the test set, which had 16,954 user-film pairs.The complete process is illustrated in the Figure 29.

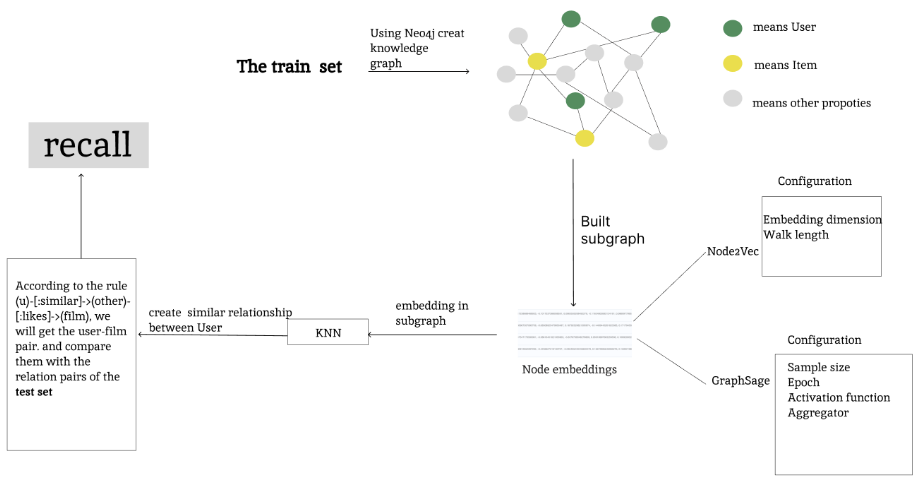
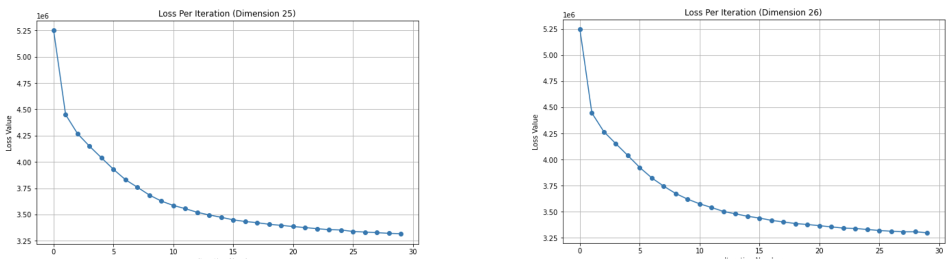


Figure 29: Flowchart of running Node2Vec or GraphSAGE combined with KNN on the Movie dataset. Unlike the pipeline on the Cora dataset illustrated in the Figure 13, this thesis performs tuning experiments on only two parameters of the Node2Vec algorithm. Also, we only tune four parameters of GraphSAGE. This is because the Movielens dataset is very large and fewer parameter experiments can save time and cost.

Experiments of Using Node2Vec with KNN on the Movielens As shown in the pipeline above, we adjust the parameters for Node2Vec on the dataset movie, mainly starting with `embedding dimension and walklength`. Firstly, we set `walklength=10` and `embedding dimension to 25 and 26`, the loss diagram is shown below: From the two loss graphs, we can notice that the loss does not converge. The obtained embeddings combined with the KNN algorithm finally get the values of 5169 and 5021 for successful matches. The subsequent choices of adjusting the walklength to 100 and 50 are again experimented with. Figure 31shows the final results of these six experiments.



(a) Loss Chart of `Walklength=10,embedding dimension=25`.We can find the loss did not converge.

(b) Loss Chart of `Walklength=10,embedding dimension=26`.We can find the loss also did not converge.

Figure 30: Parts of lossPerIteration charts. And, the 2 loss charts both showed that the loss also did not converge

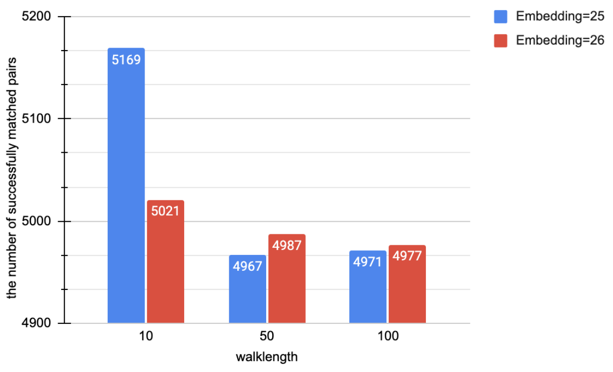


Figure 31: Histogram of the number of successful matches at different walk-lengths for embedding dimensions 25 and 26, each respectively.When the `walklength = 10`, we get the highest number,5169.

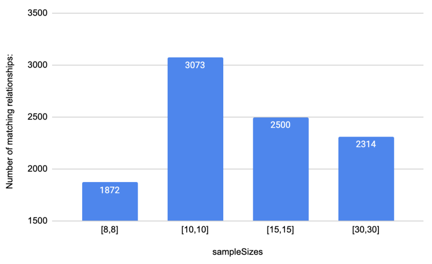
As the Figure 31 shows, the highest number of successful matches occurs when the `walklength is set to 10` and the `embedding dimension is 25`. This indicates that increasing the walk length does not lead to a higher number of matches. Therefore, in the next step, we chose to decrease the `walklength to 3` and conducted experiments with `embedding dimensions of 25, 125, and 155`. The table below summarizes the results of these three experiments.

So far, we have performed 9 tuning experiments, but find that none of the data converge. In this situation, the tuning parameter does not improve the experiment greatly. Therefore, we chose to take the maximum value of the experimental results, **5169**, to represent the final Node2Vec result on the Movielens dataset. **The final recall is 0.263.** Detailed results of the nine experiments are in Appendix H.

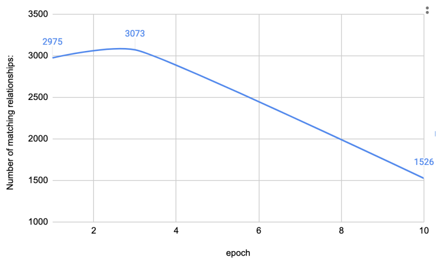
Table 16: Embedding Dimension and Convergence Results based on walk length=3

Embedding	Number of Matching Relationships	Convergence
25	4414	Not Converge
125	4553	Not Converge
155	4524	Not Converge

Experiments of Using GraphSAGE with KNN on the Movielens
We initially set the experimental parameters as follows: embeddingDimension=255, `epoch=3, aggregator="MEAN", searchDepth=5,batchSize=10, sampleSizes=[8,8], and activationFunction="ReLu" .` As shown in the Figure 29, we adjusted four parameters: epoch, aggregator, activationFunction, and sampleSizes.First, we experimented with four different values for the sample size. The results are shown in Figure 32a, where the `sample size of [10,10]` yielded the best results. Next, we conducted experiments with different epoch values, as shown in Figure 32b. The highest number of successful matches was achieved with an epoch value of 3.



(a) Bar chart of successfully matched user-film pairs for different sample sizes. When the `sample size =10`, we get the highest number,3073.



(b) Line graph of successfully matched user-film pairs at different epochs.When the `epoch = 3`, we also get the highest number,3073.

Subsequently, we adjusted the aggregator parameter, setting it to `POOL`. Although this resulted in a significant increase in the number of successful matches, the training did not converge. To address this, we increased the number of `epochs to 10` and conducted another experiment. Despite the lack of convergence, the number of matches improved.

Following this, we experimented with a `sample size of [30,30]` and also explored the effects of using `Sigmoid as the activation function` in combination with different epoch settings. The results of these tuning experiments are summarized in the Table 17.

After the aforementioned parameter tuning experiments, we found that the overall best results should surpass those obtained with Node2Vec. Consequently, we decided not to continue with further parameter tuning. Instead, we selected the highest value obtained from the experiments, which

Table 17: Parameter Settings and Results. The Highest Matched number is 5664. Detailed results of these experiments are in Appendix J

Parameter Settings				**Results**	
Aggregator	Activation Function	Sample Sizes	Epoch	Did Converge	Matched Number
MEAN	ReLu	[8,8]	3	TRUE	1872
MEAN	ReLu	[10,10]	3	TRUE	3073
MEAN	ReLu	[15,15]	3	TRUE	2500
MEAN	ReLu	[30,30]	3	TRUE	2314
MEAN	ReLu	[10,10]	10	TRUE	1526
MEAN	ReLu	[10,10]	1	TRUE	2975
POOL	ReLu	[10,10]	3	FALSE	5150
POOL	ReLu	[10,10]	10	FALSE	5237
POOL	ReLu	[30,30]	10	FALSE	**5664**
POOL	Sigmoid	[30,30]	10	FALSE	5584
POOL	Sigmoid	[30,30]	3	FALSE	5494
POOL	Sigmoid	[30,30]	10	TRUE	5567

was **5664**, as the number of successfully matched user-movie pairs using the GraphSAGE combined with KNN on the Movielens dataset. This resulted in a **recall value of 0.288**.

Experiments of Using Link Prediction on the Movielens Similar to the operation on the Cora dataset, but we need to create the proximity relationship between users and users first, and then again based on `(u)-[:similar]->(other)-[:likes]->(film), WHERE NOT (u)-[:likes]-`>(film) to get the user-film pairs. Then `Adamic Adar` is utilised for link prediction between users. At this time, algorithms that perform better on the Cora dataset such as `Toal Neighbours` are not used. Because the Cora dataset has only one node type, it can be justified by looking only at the number of neighbours. However, Movielens has more node types and is very unevenly distributed across node types. It would not be reasonable to use a method that relies on the total number of neighbours or the total number of common neighbours again. Subsequently, it was found that using `gds.alpha.linkprediction.adamicAdar` in Neo4j, the procedure does not work. Further analysis revealed that the time complexity of the whole procedure is $\mathbf{O(n\hat{2})}$. We have 5802 users, and about 17 million node pairs need to be processed. Neo4j can't perform such a large number of calculations directly, and the program keeps breaking. The experiment also tried to call `apoc.periodic.iterate` for batch processing, but it could not run successfully either. Therefore, **we believe that Neo4j's topological link prediction cannot be effectively applied to Movielens data.**

5.3 Experiments Comparing Direct Query and Latent Space with KNN for Different Tasks

We will investigate the relationship between task classification and query methods. Simultaneously, we will develop a paper recommendation system in the Section 6. Due to the significantly longer training time for the GraphSAGE algorithm, which substantially reduces real-time responsiveness to dynamic changes on the interface and negatively affects user experience, and considering that previous experimental results indicate that Node2Vec and GraphSAGE perform similarly, we will use Node2Vec for node embeddings in the subsequent experiments. Additionally, we will continue to use the `Adamic-Adar` method for link prediction in the direct querying approach.

5.3.1 Experiments Comparing Direct Query and Latent Space with KNN for Different Tasks in the Paper domain

The Table 18 below shows the tasks that we carried out inside this paper domain. The parameters of the Node2Vec algorithm are `embedding=25, walklength=10, and iteration=30.` Because this dataset does not have real user data, it is impossible to perform tuning experiments to find the optimal parameters. Compared to the Cora dataset, the paper dataset is more similar to the movie dataset. Therefore, we directly chose the parameter settings that achieved the best results in the Movielens dataset.

For Task 1, one method is to use the known relationships: written_by _authors and `contains_keywords` to directly query.In another way, we first create a subgraph containing only paper, authors, keywords and the corresponding relationships. Then, use Node2Vec to get their embeddings. Finally, construct the similar relationship between them by KNN. We use similar relationships instead of `written_by_authors` and `contains_keywords` for querying. Experiments found that **the results obtained by these two methods are exactly the same.**

For Task 2, using `Cypher` for direct querying follows a similar approach to Task 1. However, for the latent space with KNN, the main adjustment is in the subgraph construction. This step determines how we project the problem into a specific node type space. In this case, the `similar` relationship replaces the `FRIEND, belongs_to_type, and read` relationships. Ultimately, the direct querying method successfully returned five papers, whereas the other method did not find any. **Therefore, for Task 2, direct querying is the optimal approach.**

For Task 3, using the embedding method combined with KNN requires projecting the problem onto a subgraph that only contains papers, paper types, and keywords. The similar relationship created by KNN replaces the `contains_keywords and belongs_to_type` relationships. On the other hand, for direct querying, the query is as simple as: `MATCH (article:paper)-`

Table 18: Experimental Tasks in the Paper Domain

Task Type	**Tasks**
Adjacency	`T1: Find the author and keywords of articles "Simulating the Geometric Growth of the Marine Sponge Crella Incrustans" and "Do You Trust What You See? Toward A Multidimensional Measure of Trust in Visualization"`
Accessibility by Links	`T2: Find articles my friends like and belong to the short paper type.`
Common Connection	`T3: Give me articles of short article type containing the keyword Human-centered computing.`
Nodes Attribute	`T4:Find the uid of the article "Simulating the Geometric Growth of the Marine Sponge Crella Incrustans".`
Explore Task	`T5:Recommend some papers.`
Hybrid Task	`T6:Recommend some papers in session_id as short6.`

[:contains_keywords]->(keywords:keywords name: 'Human-centered computing'), (article)-[:belongs_to_type]->(type:paper_type name: 'short') RETURN article.title. The embedding method did not successfully retrieve any articles, while the direct query method found two. Therefore, for Task 3, **the direct querying approach proves to be more effective.**

For Task 4, querying the attributes of the nodes themselves is required, which can only be done using the direct querying method. By inputting the following `Cypher query:MATCH(article:paper`title:"Simulating the Geometric Growth of the Marine Sponge Crella Incrustans") RETURN article. uid the result **v-short-1067** is obtained.

For Task 5, we utilized the `Adamic Adar` algorithm for direct querying to predict links between `Dan` and other user nodes. This method identified two users similar to `Dan` and recommended six articles.

In contrast, using the embedding space method required treating the entire knowledge graph as a subgraph to obtain user node embeddings. Then, we established similarity between users through KNN. This method identified three users similar to `Dan` and recommended 11 articles.

Comparing the results from both methods, the KNN-based approach included the recommendations found by direct querying and provided additional, more diverse results. Therefore, for Task 5, **the embedding-based**

method combined with KNN is more effective in generating diverse recommendations.

In Task 6, the direct querying method follows a similar procedure to Task 5. This approach successfully recommended the article "Simulating the Geometric Growth of the Marine Sponge Crella Incrustans."

On the other hand, the latent space with KNN approach required an additional step of using the `similar` relationship instead of the presented_in _session relationship. However, no recommendations were generated by this method. Therefore, **using direct querying is the most effective way to solve Task 6.**

Summarizing the results of the experiments for the various tasks, the comparison between methods is presented in the Table 22 below. The ✓symbol indicates the best method for each task.

Task number	**Cypher Direct**	**Latent Space with KNN**
T1	✓	✓
T2	✓	
T3	✓	
T4	✓	
T5		✓
T6	✓	

Table 19: Comparison of methods for different task types on paper domain

5.3.2 Experiments Comparing Direct Query and Latent Space with KNN for Different Tasks in the Movie domain

To validate the generalizability of our conclusions, we conducted similar experiments in the movie domain. The Table 20 presents the tasks we tested in this domain. We will sequentially introduce the two methods used to perform each task and the results obtained. In this experiment, we employed Node2Vec for node embeddings. The algorithm's parameters were set according to the optimal values obtained in our previous experiments on the Movielens dataset, specifically: `embedding=25, walklength=10,` iteration=30, with all other parameters set to their default values.

For Task 7, the direct query method involves inputting the following `Cypher query:MATCH (a:actorid:"4230")-[:actor_film]->(m:film)` RETURN `DISTINCT m.id AS movieTitle.`

For the KNN method, we first run the Node2Vec algorithm on a subgraph containing only actors and films to obtain their embeddings. We then use KNN to establish similarity relationships between the nodes. By executing the `query MATCH (a:actorid:"4230")-[:similar]->(m:film) RETURN` m.id, we find the films most similar to the actor with ID 4230. The

Task Type	**Tasks**
Adjacency	`T7: Which movies have the actor with ID 4230 starred in?`
Accessibility by links	`T8: Find out what other movies the actor with the film ID 1740 has acted in .`
Common connection	`T9: Find films where the actor ID is 4230 and the genre ID is 3066.`
Nodes attribute	`T10: Find all IDs of genres.`
Explore task	`T11: Recommend me some films.`
Hybrid task	`T12: Recommend me some films in the genre 2624.`

Table 20: Task types and corresponding tasks.

following Table 21 shows all results of task 7.

Table 21: Task 7's results.We can find the direct query methods finds more results.

Task methods	**film ID**
Direct Query	**"1946", "1604", "1660", "1578"**, "2280", "689", "1208", "1155", "1209", "765"
Node2Vec+ KNN	**"1660", "1604", "1578", "1946"**

Using the direct query method, we retrieved 10 movie IDs, while the node embedding combined with the KNN method yielded only 4 movie IDs. However, all the movies identified through node embeddings were also included in the results from the direct query. The direct query method provides a more comprehensive set of movie IDs for this task. Therefore, **for Task 7, the direct query method is more effective.**

For Task 8, the direct query method involves inputting the following `Cypher query: MATCH` (a:actor)-[:actor_film]->(m:filmid:"1740") WITH a `MATCH (a)-[:actor_film]->(m1:film) WHERE m1.id` <> 1740 RETURN `DISTINCT m1.id AS movieTitle.`

Using the embedding combined with the KNN method, we first run the Node2Vec algorithm on a subgraph containing only `actors` and `films` to obtain their embeddings. We then use KNN to establish similarity relationships. The process involves first identifying the actors who have a similar relationship with the `film ID 1740` and then finding films that have a similar relationship with these actors.

The direct query returned 71 results, while the embedding and KNN method only returned 13 results, with 7 results overlapping between the two methods. Since the direct query method yields more results and directly utilizes the known knowledge from the knowledge graph, it is more reliable. Therefore, **for Task 8, the direct query method is the better approach.**

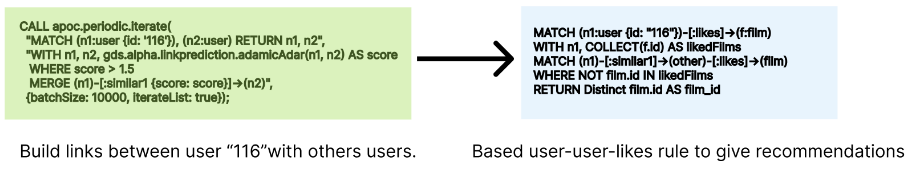
For Task 9, the direct query method involves inputting the following `Cypher query:` `MATCH`(a:actorid:"4230")-[:actor_film]->(m:film),(m)-[:film _genre]->`(g:genreid:"3066") RETURN DISTINCT m.id AS movieTitle.`

In contrast, the embedding combined with KNN method involves running the Node2Vec algorithm on a subgraph containing films, actors, and genres. After obtaining the embeddings for these nodes, KNN is used to establish similarity relationships. By executing the `query:` `MATCH` (m:film)<-[:similar]-(g:genreid:"3066"),(a:actorid:"4230")-[:similar]->(m:film) RETURN `DISTINCT m.id AS movieTitle`, we aim to find film nodes in the latent space that are closest to `actor 4230` and `genre 3066`.

The direct query successfully identified 7 movie nodes, while the embedding-based KNN approach did not identify any movie nodes. Therefore, **for Task 9, the direct query method is more effective.**

For Task 10, it is about the internal attributes of the node. Only direct queries can be used. Running `Match(g:genre) Return g.id` got 23 out of 23 genres.

For Task 11, the direct query method utilizes Neo4j's `Adamic Adar` algorithm for link prediction between users. Given the large size of the dataset, we employed the APOC library to batch-process the link prediction tasks. The specific operations are illustrated in the figure below. Using this method, we identified 3 users similar to user 116, and with a score threshold set at 1.5, we were able to recommend 222 movies to the user.



```
CALL apoc.periodic.iterate(
  "MATCH (n1:user {id: '116'}), (n2:user) RETURN n1, n2",
  "WITH n1, n2, gds.alpha.linkprediction.adamicAdar(n1, n2) AS score
   WHERE score > 1.5
   MERGE (n1)-[:similar1 {score: score}]→(n2)",
  {batchSize: 10000, iterateList: true});
```

```
MATCH (n1:user {id: "116"})-[:likes]→(f:film)
WITH n1, COLLECT(f.id) AS likedFilms
MATCH (n1)-[:similar1]→(other)-[:likes]→(film)
WHERE NOT film.id IN likedFilms
RETURN Distinct film.id AS film_id
```

Build links between user “116”with others users.
Based user-user-likes rule to give recommendations

Figure 33: Flowchart of the direct query method for task 11

In contrast, the alternative method involved running Node2Vec on the entire graph and using KNN to establish similarity relationships between users. This approach identified 10 users similar to user 116 and allowed us to recommend 50 movies. Notably, the users identified by the two methods were entirely different, with only 22 overlapping movies between the two sets.

From the perspective of movie diversity, the direct query method yielded a broader range of recommendations. However, from the perspective of user diversity, the second method resulted in greater user variety. Since our recommendation is based on user similarity, user diversity should be prioritized. Additionally, while the first method identified only three similar users, it managed to recommend 222 movies that `User 116` had not seen. This suggests that the similarity of the users identified by the first method may not be high. This issue likely stems from the low threshold setting used in the `Adamic Adar algorithm`.

Finally, for Task 12, the direct query method is similar to the procedure used in Task 4, with the addition of a filter to select `genres with ID 2624`. This approach allows us to recommend 23 movies to `User 116`.

For the embedding combined with KNN method, after following the same steps as before, we need to include genre nodes and the film_genre relationship in the subgraph used to create similarity relationships. Filtering based on the similarity between movies and `genre 2624`, returned no results. This outcome highlights a potential issue with KNN: when the subgraph includes multiple types of nodes such as movies, users, and genres, the algorithm tends to create more similar relationships within the same node type. Consequently, with a limited value of K, there are fewer cross-node-type relationships established. Summarising all the task experiments above in the movie domain, the results of the comparison of the different methods for different tasks are shown in the Table 22 below. To summarize the results obtained

Table 22: Comparison of methods for different task types on movie domain

Task number	**Cypher Direct**	**Latent Space with KNN**
T7	✓	
T8	✓	
T9	✓	
T10	✓	
T11		✓
T12	✓	

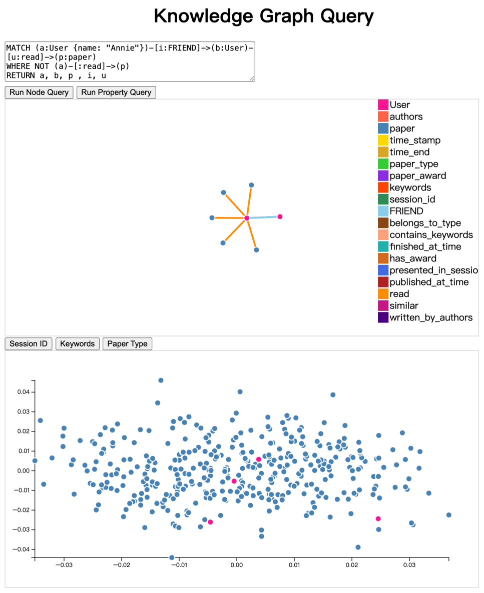
from both domains, discrepancies were observed only in the first type of task. In the paper domain, both methods were suitable, while in the movie domain, the direct query approach proved to be more effective. Therefore, this paper concludes that the direct query method, which performs well in both domains, is the best approach for the first type of task. The Table 23 below shows the answer to the preferred query methods for different tasks.

Table 23: Comparison of methods for different task types

Task Type	**Cypher Direct**	**Latent Space with KNN**
Adjacency	✓	
Accessibility by Links	✓	
Common Connection	✓	
Nodes Attribute	✓	
Explore Task		✓
Hybrid Task	✓	

6 Visualization Design

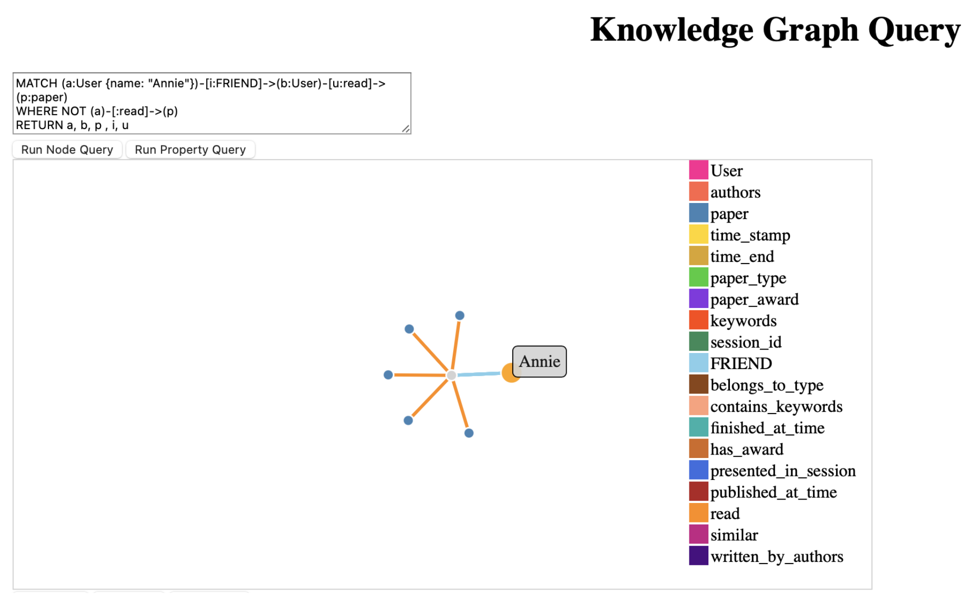
The entire browser backend uses node.js as the server runtime environment and the express framework to manage routes, handle HTTP requests, static file serving, etc. In the front end, the entire web page uses `d3.js` to create 2D visualisations of the knowledge graph, PCA, related interactions, etc. The following Figure 34 show the thesis recommendation web page we designed. The whole page is divided into three modules: a user input box, a query result visualisation box, and a PCA interaction box between the paper and the user.



```
MATCH (a:User {name: "Annie"})-[i:FRIEND]->(b:User)-
[u:read]->(p:paper)
WHERE NOT (a)-[:read]->(p)
RETURN a, b, p , i, u
```

Figure 34: The whole website page. The top part is the user input box and has two buttons. Below that is a section for visualising the results of nodes query. On the right, the corresponding legend is displayed. Users can select the features they are interested in by using the three buttons under the visualisation module. At the bottom is the 2D visualisation obtained by performing Node2Vec and PCA dimensionality reduction using user, paper and the features of interest to the user.

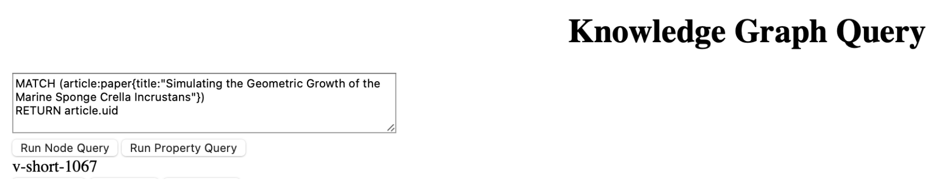
At the top is the user input box, where users can enter `Cypher queries`. Below the input box are two buttons. For node-related queries, corresponding to tasks 1, 2, 3 and 6 described in the Table 2, users should click the "Run Node Query" button. For queries related to node properties, such as task 4 described in the Table 2, users need to click the "Run Property Query" button. The results of a node query are visualized below, with different node and relationship types represented by different colors. Hovering over a node displays its corresponding name. The results of a property query are displayed directly as text. The Figure 35 illustrates an example of executing a node query. The user enters the following query in the input box: `MATCH` (a:User name: "Annie")-[i:FRIEND]->(b:User)-[u:read]->(p:paper) WHERE NOT (a)-[:read]->(p) `RETURN a, b, p, i, u`. This query searches for papers that Annie's friends have read but she has not. This query belongs to our "Accessibility by Links" task. The results of the query are visualized as shown below Figure 35. By hovering over the nodes and relationships, we can identify that one user node represents Annie, and the other user node represents her friend, `Matt`. The paper nodes correspond to the five articles that Matt has read.


Knowledge Graph Query

```
MATCH (a:User {name: "Annie"})-[i:FRIEND]->(b:User)-[u:read]->
(p:paper)
WHERE NOT (a)-[:read]->(p)
RETURN a, b, p , i, u
```

Figure 35: Example diagram for executing a node query

The Figure 36illustrates an example of performing a node property query. To retrieve the UID of a specific paper, the user enters the following query in the input box:`MATCH` (article:paper title: "Simulating the Geometric Growth of the Marine Sponge Crella Incrustans") `RETURN article.uid`. For this type of task, the query results are not visualized. Instead, the results are displayed directly as text below the input box.

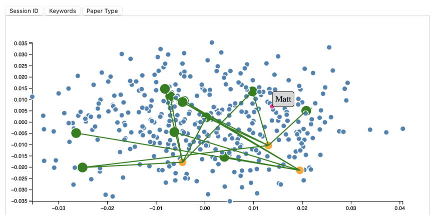

Knowledge Graph Query

```
MATCH (article:paper{title:"Simulating the Geometric Growth of the
Marine Sponge Crella Incrustans"})
RETURN article.uid
```

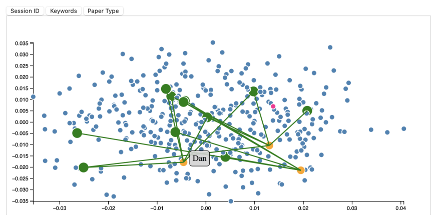
```
v-short-1067
```

Figure 36: Example diagram for executing a node property query

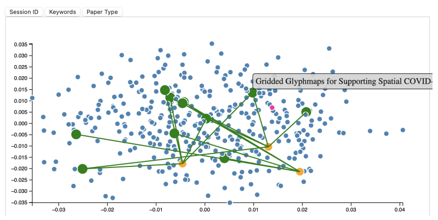
Task 5 primarily involves operations on the PCA plot. Users can select features of interest using `session_id`, `paper_type`, and `keywords`. Based on these selected features, a subgraph is generated in the background, incorporating both papers and users. Node2Vec is then applied to this subgraph to obtain high-dimensional embeddings for the user and paper nodes. Next, KNN is used to construct similar relationships between users. These embeddings are subsequently reduced in dimensionality using PCA and visualized. By clicking on a user node, the following `query` is executed in the background: `MATCH (n:User name:  'name')`-[r:similar]->(friend:User)-[:read]->(paper:paper) `RETURN friend.name AS user`, collect(paper.title) AS papers, `r.score AS scoreValue`. This operation retrieves similar users to the target user and the books that these similar users have read. This approach combines Node2Vec with KNN for recommendations as part of Task 5. The recommendation results are visualized by highlighting the similar users and the recommended paper nodes. Connections between similar users and the books they have read will appear, and hovering over these connections will display the recommendation scores for the recommended articles. These recommendation scores are the scores of similar users obtained through KNN. The Figure 37 below shows the recommendation results for the `User Matt`, focusing on the `session_id` feature.



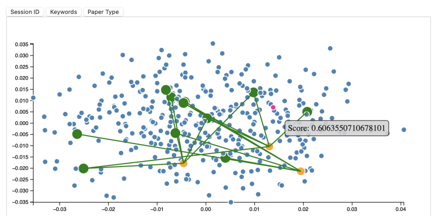
(a) Show the user who uses this recommender system



(b) Show similar users obtained by Node2Vec and KNN



(c) Present the recommended papers



(d) Present the recommended score of recommended papers

Figure 37: The result graph of Matt's paper recommendation on the feature session_id.

7 Discussion and Future Work

7.1 Answers to the Research Question

In SectionSection 4 we first introduce the various algorithms, followed by an explanation of the Node2Vec or GraphSAGE combined with KNN pipeline. This helps to clarify the overall design process. The Cora and Movielens datasets mentioned in Section Section 5 are then applied to the latent space with KNN design described in SectionSection 4, followed by parameter tuning experiments. Based on the experimental results in SectionSection 5, we observe that the recall obtained from Node2Vec or GraphSAGE combined with KNN is higher than that from the direct use of link prediction in the Cora dataset. Moreover, the link prediction algorithm cannot run on large datasets like Movielens.

Therefore, we can answer Research Question 1:**Does using GraphSAGE or Node2Vec with KNN outperform the direct use of Neo4j's link prediction algorithm for recommendations in the Cora or the Movielens-1M datasets in terms of Recall?**and confirm hypotheses:

1. $H1_1$**:Using GraphSAGE and Node2Vec with KNN based on Neo4j outperform the direct use of Neo4j's link prediction algorithm for recommendations in the Cora dataset in terms of Recall with at least a 1% margin.**

2. $H1_2$**: Using GraphSAGE and Node2Vec with KNN based on Neo4j outperforms the direct use of Neo4j's link prediction algorithm for recommendations in the Movielens-1M dataset in terms of Recall with at least a 1% margin.**

In SectionSection 3, we redefined a taxonomy for graph tasks, which provides a more effective way to distinguish and generalize query tasks in the paper domain. In Section Section 4, we introduced the task pipeline, which includes two query methods: one involves directly querying the knowledge graph using Cypher, and the other uses a Feature-Based Embedding combined with KNN. In sectionSection 5, we applied the task taxonomy from section Section 3 to create six types of tasks in both the paper and movie domains. These tasks were then applied to the task pipeline described in sectionSection 4, to determine the optimal method for each task.

Moreover, by comparing the experimental results, we observe that the findings in the paper domain and the movie domain are consistent, which demonstrates the generalizability of our research results. The experimental outcomes also address Research Question 2: **Which query methods can be used for which tasks in the paper domain?** and validate the hypotheses presented in the table below.

Table 24: the Accepted Hypotheses regard to Research Question 2

Task Type	Hypothesis
Adjacency Task	$H1_3$: Direct querying results are more diverse.
Accessibility by Links Task	$H1_4$: Direct querying results are more diverse.
Common Connection Task	$H1_5$: Direct querying results are more diverse.
Nodes Attribute Task	$H1_6$: Direct querying results are more diverse.
Explore Task	$H2_7$: latent space with KNN results are more diverse.
Hybrid Task	$H1_8$: Direct querying results are more diverse.

7.2 Findings in the Experiments

- The limitations of the Node2Vec, GraphSAGE, and KNN algorithms contribute to the lower accuracy of the model's experiments compared to state-of-the-art algorithms.

We observe that the Node2Vec or GraphSAGE combined with KNN model performs better than the link prediction algorithm in the Section 5.2. **However, the accuracy achieved is not as high compared to state-of-the-art algorithms.** For example, in the Cora dataset, the MTGAE model achieves an accuracy of 0.946 for link prediction tasks [30]. In the Movielens dataset, the GHRS model's precision is 0.792 [8].**This could be due to the Node2Vec algorithm not capturing node attributes fully, leading to embeddings that do not completely represent node information. Alternatively, although the GraphSAGE algorithm can capture node information, it might not perform well in non-connected graphs with many outliers.** Additionally, the KNN algorithm can not learn node relationships effectively when finding nearest neighbors.

- On large and complex datasets, embeddings obtained through the GraphSAGE algorithm are better at capturing semantic information. However, on smaller and simpler datasets, the Node2Vec algorithm tends to perform better.

We also notice that although GraphSAGE performs worse than Node2Vec on the Cora dataset, it performs better on the Movielens dataset. This could be because the Cora dataset is a homogeneous graph, while Movielens is a heterogeneous graph. The Node2Vec algorithm cannot distinguish between different nodes and relationships in a heterogeneous graph, whereas

GraphSAGE can. Thus, the embeddings obtained by GraphSAGE can better represent node information.

- The recommendation results of the Node2Vec algorithm combined with KNN on the Cora and Movielens datasets do not show significant improvement with changes in parameters such as `embedding dimension, walk length, initial learning rate, and iterations.`

In the Section 5.2, we conducted multiple parameter tuning experiments on the Node2Vec model combined with KNN. We found that, excluding cases where parameters were set beyond their practical limits (e.g., `initial learning rate = 0.5 or embedding dimension = 5`), the experimental results did not show significant improvement with parameter adjustments. Further analysis of the **loss** function from Node2Vec revealed that it **does not converge**, which indicates that the model may not be adequately trained. As a result, the generated **embeddings fail to effectively capture semantic information**. This could explain the lack of noticeable improvements in the tuning experiments conducted in the Section 5.2.

- For tasks involving the exploration of known relationships, direct querying methods are more efficient. However, when the exploration involves only unknown relationships, the latent space with KNN approach proves to be more effective.

It is observed that direct querying performs better in most tasks in the Section 5.3. In particular, direct querying is more accurate for querying known relationships. However, for the explore task, the latent space with KNN approach demonstrates greater diversity. This is because the explore task deals exclusively with link prediction for unknown relationships, similar to the study of comparing the recall of link prediction with latent space with KNN of this thesis in theSection 5.2. In theSection 5.2, we find that the Node2Vec or GraphSAGE combined with KNN model outperforms the direct use of link prediction algorithms.

Although the Hybrid Task also involves link prediction for unknown relationships, it includes additional filtering based on known relationships. This reduces the effectiveness of the latent space with KNN approach in the hybrid task compared to the Explore Task.

7.3 Limitation and Future Work

In the Section 5.2, the Node2Vec algorithm did not converge with the Cora dataset. An experiment was conducted with 10,000 iterations, but the data still did not converge. Due to the long experiment time, only one experiment was performed, and the influence of other parameters cannot be ruled out.

Future work could involve increasing the number of iterations or conducting more experiments with various parameter settings.

The results obtained were not from the optimal parameters in the parameter tuning experiments for the GraphSAGE algorithm. Although this does not affect the comparison with the direct use of Neo4j's topological link prediction algorithm (baseline), more parameter-tuning experiments should be conducted in the future to determine the best parameters.

In the Section 5.3, the number of tasks in each category is still insufficient, which may lead to the results being somewhat coincidental. We need to set a larger number of tasks in the future. Additionally, in the paper domain, we directly used the same parameter settings and lower embedding dimensions as in the movie domain. This may affect the accuracy of the results. Future work should include more parameter-tuning experiments. Considering training time and other issues, we only selected the Node2Vec algorithm for embeddings in the recommendation system, but GraphSAGE might perform better on larger graphs.

For the recommendation part of the system, we primarily rely on the principle that similar users like similar items. However, the design combining Node2Vec or GraphSAGE with KNN can also facilitate content-based recommendations, such as directly recommending items adjacent to the user. Future work could incorporate this functionality into the recommendation system.

8 Conclusion

This thesis clarifies the appropriate query methods that users should adopt for different task categories within the domain of paper recommendation. For most tasks, direct query methods prove more effective. However, the latent space with KNN approach is more suitable for the Explore Task. This allows users to perform queries more efficiently. Future research can build on this work to explore how humans can better interact with knowledge graph-based systems. Additionally, this thesis redefines task classification methods based on graph structures for paper recommendation, enhancing the relevance and accuracy of task classification in this domain. This also offers a direction for future studies on task classification methods for specialized fields.

Moreover, this thesis proposes a model design using Node2Vec or GraphSAGE with KNN for the paper recommendation. To validate the model's accuracy, it is compared against link prediction algorithms used in direct query methods on the Cora and Movielens datasets. Experimental results demonstrate that the proposed model performs better, offering a promising research direction for knowledge graph-based paper recommendation systems.

Additionally, the thesis introduces a novel dynamically updated paper recommendation system. Users can adjust features according to their personal needs, enhancing system personalization. The system presents recommendation results through dynamic 2D visualization, which improves interpretability and increases user trust in the system. This work provides a foundation and direction for future in-depth research in knowledge graph-based recommendation systems.

References

[1] Robert Amar, James Eagan, and John Stasko. Low-level components of analytic activity in information visualization. In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, pages 111–117. IEEE, 2005.

[2] Chandra Bhagavatula, Sergey Feldman, Russell Power, and Waleed Ammar. Content-based citation recommendation. *arXiv preprint arXiv:1802.08301*, 2018.

[3] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[4] Matthew Brehmer and Tamara Munzner. A multi-level typology of abstract visualization tasks. *IEEE Transactions on Visualization and Computer Graphics*, 19:2376–85, 12 2013.

[5] Yixin Cao, Xiang Wang, Xiangnan He, Zikun Hu, and Tat-Seng Chua. Unifying knowledge graph learning and recommendation: Towards a better understanding of user preferences. In *The world wide web conference*, pages 151–161, 2019.

[6] Rui Chen, Qingyi Hua, Yan-Shuo Chang, Bo Wang, Lei Zhang, and Xiangjie Kong. A survey of collaborative filtering-based recommender systems: From traditional methods to hybrid methods based on social networks. *IEEE access*, 6:64301–64320, 2018.

[7] Janneth Chicaiza and Priscila Valdiviezo-Diaz. A comprehensive survey of knowledge graph-based recommender systems: Technologies, development, and contributions. *Information*, 12(6):232, 2021.

[8] Zahra Zamanzadeh Darban and Mohammad Hadi Valipour. Ghrs: Graph-based hybrid recommendation system with application to movie recommendation. *Expert Systems with Applications*, 200:116850, 2022.

[9] Fitrio Dermawan, Chang Hong Kwang, Muhammad Dimas Adijanto, Nur Aini Rakhmawati, and Naufal Rafiawan Basara. Product recommendations through neo4j by analyzing patterns in customer purchases. In *2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems, ICETSIS 2024*, 2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems, ICETSIS 2024, pages 624–627, United States, 2024. Institute of Electrical and Electronics Engineers Inc.

[10] I. Dharmawan and Riyanarto Sarno. Book recommendation using neo4j graph database in bibtex book metadata. pages 47–52, 10 2017.

[11] Shijie Geng, Zuohui Fu, Juntao Tan, Yingqiang Ge, Gerard De Melo, and Yongfeng Zhang. Path language modeling over knowledge graphsfor explainable recommendation. In *Proceedings of the ACM Web Conference 2022*, pages 946–955, 2022.

[12] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[13] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[14] Sihang Hu, Zhiying Tu, Zhongjie Wang, and Xiaofei Xu. A poi-sensitive knowledge graph based service recommendation method. In *2019 IEEE International Conference on Services Computing (SCC)*, pages 197–201. IEEE, 2019.

[15] Anna Huang et al. Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, volume 4, pages 9–56, 2008.

[16] Awliya Hanun Izdihar, Nazriyah Deny Tsaniyah, Faraz Nurdini, Belva Rizki Mufidah, and Nur Aini Rakhmawati. Building a movie recommendation system using neo4j graph database: A case study of netflix movie dataset. In *2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems, ICETSIS 2024*, 2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems, ICETSIS 2024, pages 614–618, United States, 2024. Institute of Electrical and Electronics Engineers Inc.

[17] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, pages 687–696, 2015.

[18] Nasrullah Khan, Zongmin Ma, Aman Ullah, and Kemal Polat. Categorization of knowledge graph based recommendation methods and benchmark datasets from the perspectives of application scenarios: A

comprehensive survey. *Expert Systems with Applications*, 206:117737, 2022.

[19] Nasrullah Khan, Zongmin Ma, Li Yan, and Aman Ullah. Hashing-based semantic relevance attributed knowledge graph embedding enhancement for deep probabilistic recommendation. *Applied Intelligence*, 53(2):2295–2320, 2023.

[20] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[21] Bongshin Lee, Catherine Plaisant, Cynthia Sims Parr, Jean-Daniel Fekete, and Nathalie Henry. Task taxonomy for graph visualization. In *Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*, BELIV '06, page 1–5, New York, NY, USA, 2006. Association for Computing Machinery.

[22] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

[23] Chan Liu, Lun Li, Xiaolu Yao, and Lin Tang. A survey of recommendation algorithms based on knowledge graph embedding. In *2019 IEEE International Conference on Computer Science and Educational Informatization (CSEI)*, pages 168–171. IEEE, 2019.

[24] Pengcheng Liu, Yinliang Huang, Ping Wang, Qifan Zhao, Juan Nie, Yuyang Tang, Lei Sun, Hailei Wang, Xuelian Wu, and Wenbo Li. Construction of typhoon disaster knowledge graph based on graph database neo4j. pages 3612–3616, 08 2020.

[25] Weizhi Ma, Min Zhang, Yue Cao, Woojeong Jin, Chenyang Wang, Yiqun Liu, Shaoping Ma, and Xiang Ren. Jointly learning explainable rules for recommendation with knowledge graph. In *The world wide web conference*, pages 1210–1221, 2019.

[26] Ayushi Patil, Shreya Mahajan, Jinal Menpara, Shivali Wagle, Preksha Pareek, and Ketan Kotecha. Enhancing fraud detection in banking by integration of graph databases with machine learning. *MethodsX*, 12:102683, 04 2024.

[27] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 459–467, 2018.

[28] Leily Sheugh and Sasan H Alizadeh. A note on pearson correlation coefficient as a metric of similarity in recommender system. In *2015 AI & Robotics (IRANOPEN)*, pages 1–6. IEEE, 2015.

[29] John K Tarus, Zhendong Niu, and Ghulam Mustafa. Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning. *Artificial intelligence review*, 50:21–48, 2018.

[30] Phi Vu Tran. Multi-task graph autoencoders. *arXiv preprint arXiv:1811.02798*, 2018.

[31] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. Exploring high-order user preference on the knowledge graph for recommender systems. *ACM Transactions on Information Systems (TOIS)*, 37(3):1–26, 2019.

[32] Hongwei Wang, Miao Zhao, Xing Xie, Wenjie Li, and Minyi Guo. Knowledge graph convolutional networks for recommender systems. In *The world wide web conference*, pages 3307–3313, 2019.

[33] Xiang Wang, Xiangnan He, and Tat-Seng Chua. Learning and reasoning on graph for recommendation. In *Proceedings of the 13th international conference on web search and data mining*, pages 890–893, 2020.

[34] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. Explainable reasoning over knowledge graphs for recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 5329–5336, 2019.

[35] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28, 2014.

[36] Jim Webber. A programmatic introduction to neo4j. In *Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity*, pages 217–218, 2012.

[37] Tianxing Wu, Arijit Khan, Melvin Yong, Guilin Qi, and Meng Wang. Efficiently embedding dynamic knowledge graphs. *Knowledge-based systems*, 250:109124, 2022.

[38] Yikun Xian, Zuohui Fu, Shan Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. Reinforcement knowledge graph reasoning for explainable recommendation. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, pages 285–294, 2019.

[39] Zuoxi Yang and Shoubin Dong. Hagerec: Hierarchical attention graph convolutional network incorporating knowledge graph for explainable recommendation. *Knowledge-Based Systems*, 204:106194, 2020.

[40] Zhiwen Yu, Yuichi Nakamura, Seiie Jang, Shoji Kajita, and Kenji Mase. Ontology-based semantic recommendation for context-aware e-learning. In *Ubiquitous Intelligence and Computing: 4th International Conference, UIC 2007, Hong Kong, China, July 11-13, 2007. Proceedings 4*, pages 898–907. Springer, 2007.

[41] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 353–362, 2016.

[42] Qianjin Zhang, Ronggui Wang, Juan Yang, and Lixia Xue. Structural context-based knowledge graph embedding for link prediction. *Neurocomputing*, 470:109–120, 2022.

[43] Zijian Zhang, Lin Gong, and Jian Xie. Ontology-based collaborative filtering recommendation algorithm. In *Advances in Brain Inspired Cognitive Systems: 6th International Conference, BICS 2013, Beijing, China, June 9-11, 2013. Proceedings 6*, pages 172–181. Springer, 2013.

[44] Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. Meta-graph based recommendation fusion over heterogeneous information networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 635–644, 2017.

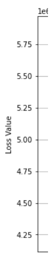
[45] Bo Zhu, Remigio Hurtado, Jesus Bobadilla, and Fernando Ortega. An efficient recommender system method based on the numerical relevances and the non-numerical structures of the ratings. *IEEE Access*, 6:49935–49954, 2018.

A Results of Tuning Experiments of Node2Vec with KNN on Cora Dataset

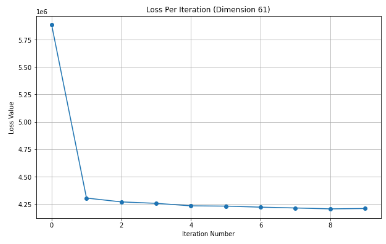
B Tuning Experiments on Embedding Dimension and Measure Methods of Node2Vec with KNN on Cora Dataset

In these experiments, we set `Walklength = 80 and iteration =10`.

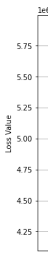
Embedding dimension: 60
PEARSON Mean similarity: 0.9252205678243933
EUCLIDEAN Mean similarity: 0.3678838515035159
COSINE Mean similarity: 0.9249058524826252
Number of matching relationships--PEARSON: 638
Number of matching relationships--EUCLIDEAN: 650
Number of matching relationships--COSINE: 643
[5883856.555550681, 4304585.631791878, 4269671.14669953, 4256433.688099816, 4233671.721277088, 4231290.1593799265, 4221563.16703808, 4214189.057552348, 4206021.744658194, 4209275.253458172]



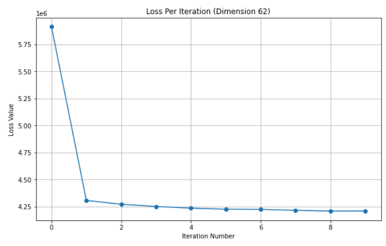
Embedd
PEARS
EUCLID
COSINE
Number
Number
Number
[591066
4242002
4201266
4182939



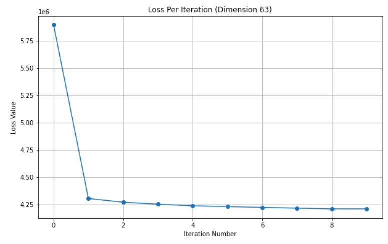
Embedding dimension: 61
PEARSON Mean similarity: 0.9254141507507956
EUCLIDEAN Mean similarity: 0.36678182976636786
COSINE Mean similarity: 0.9244149432048487
Number of matching relationships--PEARSON: 647
Number of matching relationships--EUCLIDEAN: 662
Number of matching relationships--COSINE: 646
[5911277.89638523, 4306073.039836621, 4270554.353383733, 4250197.125361699, 4234298.206486743, 4225000.442108905, 4222888.812994194, 4214311.943005538, 4207761.932237508, 4208465.706513351]



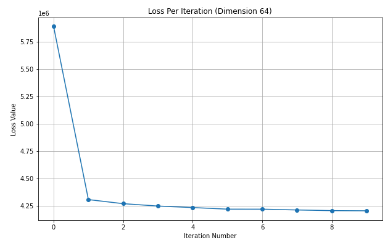
Embedd
PEARS
EUCLID
COSINE
Number
Number
Number
[593292
4234975
4193794
4172130



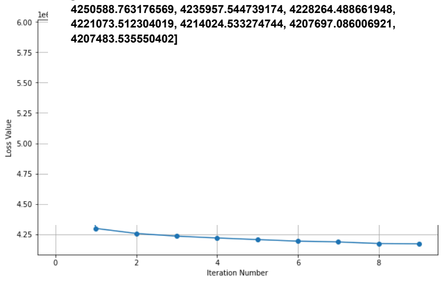
Embedding dimension: 62
PEARSON Mean similarity: 0.9246995176487166
EUCLIDEAN Mean similarity: 0.36616077486486165
COSINE Mean similarity: 0.9239937073723076
Number of matching relationships--PEARSON: 644
Number of matching relationships--EUCLIDEAN: 652
Number of matching relationships--COSINE: 644
[5894907.631453699, 4302629.742173649, 4268434.187552092, 4250588.763176569, 4235957.544739174, 4228264.488661948, 4221073.512304019, 4214024.533274744, 4207697.086006921, 4207483.535550402]



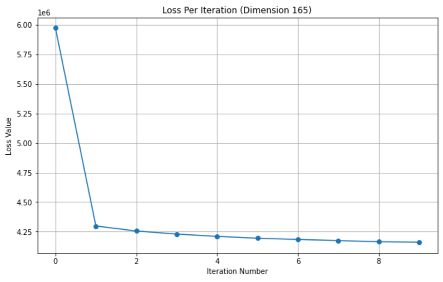
Embedding dimension: 63
PEARSON Mean similarity: 0.9236495861802883
EUCLIDEAN Mean similarity: 0.3652761243823008
COSINE Mean similarity: 0.9232815549503929
Number of matching relationships--PEARSON: 634
Number of matching relationships--EUCLIDEAN: 654
Number of matching relationships--COSINE: 637
[5887093.32450492, 4306697.9816962285, 4269489.602571646, 4247639.218362877, 4234001.51766868, 4219424.564842084, 4219054.853386553, 4211900.896945172, 4205902.008921519, 4204355.65270042]



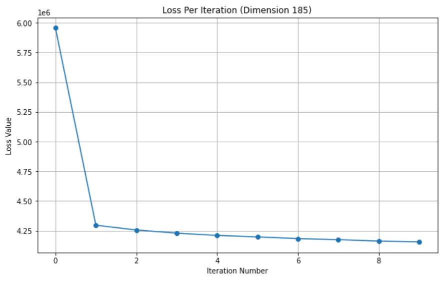
Embedding dimension: 64
PEARSON Mean similarity: 0.9236137266173144
EUCLIDEAN Mean similarity: 0.3646706986110354
COSINE Mean similarity: 0.9233487060186254
Number of matching relationships--PEARSON: 636
Number of matching relationships--EUCLIDEAN: 651
Number of matching relationships--COSINE: 637
[5888778.37290165, 4308209.0288021285, 4267090.186353746, 4245792.86452519, 4232446.588068151, 4226858.425849916, 4217517.799135599, 4211809.790758935, 4206653.945101671, 4201570.898561442]



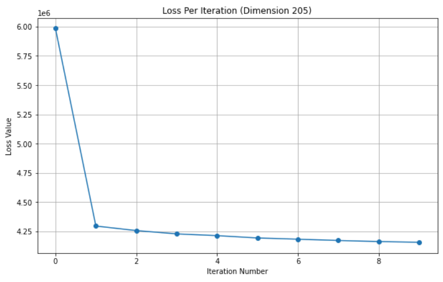
4184327.485902884, 4174974.1574592018, 4165049.5119686564,
4161340.8161650733]



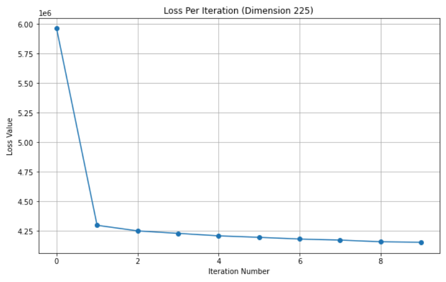
```
Embedding dimension: 165
PEARSON Mean similarity: 0.9070471657960066
EUCLIDEAN Mean similarity: 0.34311238560754054
COSINE Mean similarity: 0.9070042718636585
Number of matching relationships--PEARSON: 630
Number of matching relationships--EUCLIDEAN: 640
Number of matching relationships--COSINE: 628
[5955030.185620224, 4296279.567670797, 4256038.95718903,
4229495.948211201, 4210304.911086054, 4198024.287517422,
4183600.3191866176, 4174907.9240555903, 4162396.074199055,
4156981.9153903383]
```



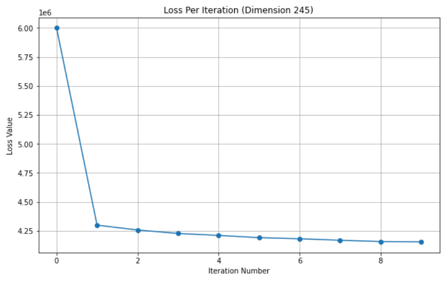
```
Embedding dimension: 185
PEARSON Mean similarity: 0.9057997383782571
EUCLIDEAN Mean similarity: 0.34240795653891176
COSINE Mean similarity: 0.9061511666595143
Number of matching relationships--PEARSON: 631
Number of matching relationships--EUCLIDEAN: 640
Number of matching relationships--COSINE: 632
[5984140.605220685, 4294399.077855757, 4255473.79753522,
4227268.126113902, 4212201.88579614, 4192274.8048163485,
4181728.6108782613, 4170777.525987099, 4161531.1912871157,
4155447.1866888134]
```



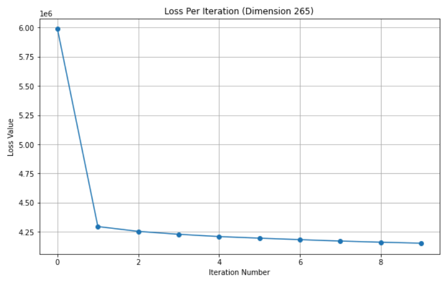
```
Embedding dimension: 205
PEARSON Mean similarity: 0.9058250365264306
EUCLIDEAN Mean similarity: 0.34251087088704635
COSINE Mean similarity: 0.9059358720765333
Number of matching relationships--PEARSON: 625
Number of matching relationships--EUCLIDEAN: 642
Number of matching relationships--COSINE: 625
[5961463.942540921, 4298633.218386855, 4251445.065313152,
4230452.710666289, 4209591.993794804, 4196889.697111268,
4182475.240243969, 4173625.626637568, 4159616.1096623703,
4155115.056957082]
```



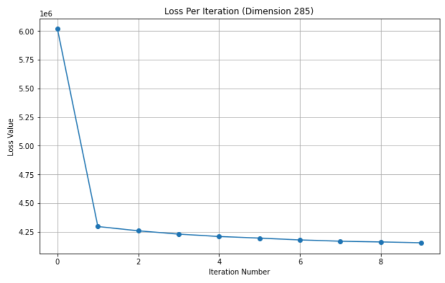
```
Embedding dimension: 225
PEARSON Mean similarity: 0.9056146199101007
EUCLIDEAN Mean similarity: 0.3421945668393787
COSINE Mean similarity: 0.9054946859971107
Number of matching relationships--PEARSON: 635
Number of matching relationships--EUCLIDEAN: 642
Number of matching relationships--COSINE: 639
[5999006.218450369, 4298662.065201526, 4257065.182225969,
4226939.262234945, 4210289.554250923, 4191032.874678534,
4181248.018702538, 4168520.8346796203, 4157338.651109587,
4155382.5791045553]
```



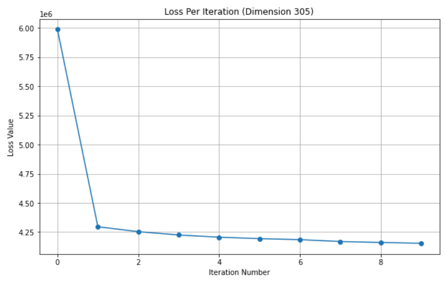
```
Embedding dimension: 245
PEARSON Mean similarity: 0.9051498297574425
EUCLIDEAN Mean similarity: 0.34206365862836513
COSINE Mean similarity: 0.9054610217163447
Number of matching relationships--PEARSON: 629
Number of matching relationships--EUCLIDEAN: 649
Number of matching relationships--COSINE: 630
[5987472.332110549, 4295046.427238903, 4253738.091163274,
4228727.1713688215, 4209173.391315706, 4195525.399290327,
4182718.1002478125, 4170689.676991642, 4160886.668879131,
4152301.1089664507]
```



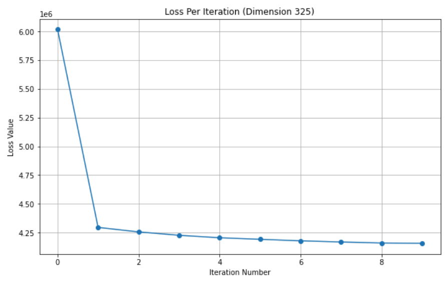
```
Embedding dimension: 265
PEARSON Mean similarity: 0.9051923340497376
EUCLIDEAN Mean similarity: 0.3421243066801806
COSINE Mean similarity: 0.9048104741196513
Number of matching relationships--PEARSON: 630
Number of matching relationships--EUCLIDEAN: 643
Number of matching relationships--COSINE: 630
[6016393.243775175, 4295650.688343349, 4258539.491035027,
4229741.902000501, 4208788.481124451, 4194937.535816038,
4178934.174215717, 4167859.9497326566, 4161235.69634547,
4154283.3476964375]
```



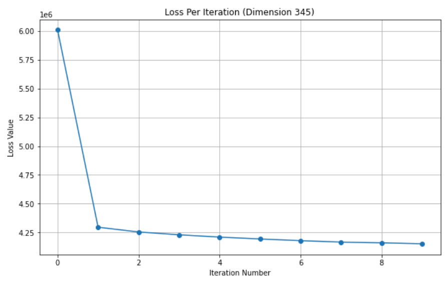
```
Embedding dimension: 285
PEARSON Mean similarity: 0.9050572957386822
EUCLIDEAN Mean similarity: 0.34232179178268907
COSINE Mean similarity: 0.9051394542880037
Number of matching relationships--PEARSON: 631
Number of matching relationships--EUCLIDEAN: 644
Number of matching relationships--COSINE: 632
[5987836.095276792, 4295804.116588896, 4253499.302333847,
4225010.446652183, 4206096.968709783, 4193762.5268792207,
4184950.1549664903, 4169100.9645216265, 4160787.758461342,
4153936.964306149]
```



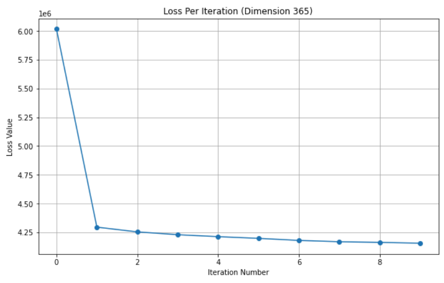
```
Embedding dimension: 305
PEARSON Mean similarity: 0.9045491358025916
EUCLIDEAN Mean similarity: 0.34180243469587634
COSINE Mean similarity: 0.904786703259061
Number of matching relationships--PEARSON: 633
Number of matching relationships--EUCLIDEAN: 645
Number of matching relationships--COSINE: 633
[6015802.918606155, 4295907.696272539, 4256883.425223401,
4227303.652949014, 4206071.096347203, 4192436.87910027,
4179365.624728164, 4169275.5458386955, 4160317.4801997948,
4158027.364447135]
```



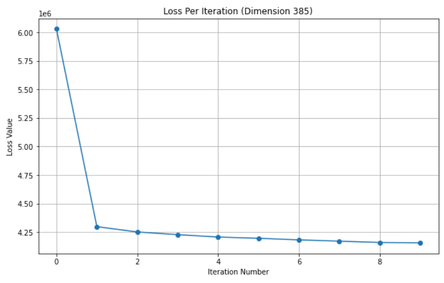
```
Embedding dimension: 325
PEARSON Mean similarity: 0.9044513037497043
EUCLIDEAN Mean similarity: 0.3418868140948928
COSINE Mean similarity: 0.9044244943728651
Number of matching relationships--PEARSON: 628
Number of matching relationships--EUCLIDEAN: 645
Number of matching relationships--COSINE: 627
[6007327.683502792, 4296571.345811992, 4255359.8157536965,
4230609.443583673, 4210810.921030984, 4194474.325033717,
4179851.737953049, 4167425.5885738265, 4161127.1167471022,
4152850.8414397445]
```



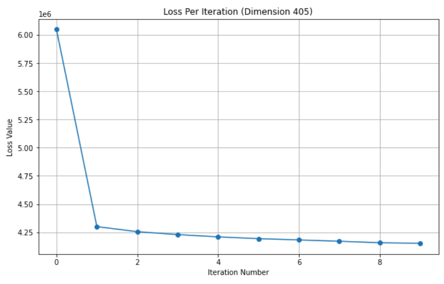
```
Embedding dimension: 345
PEARSON Mean similarity: 0.9048186790995971
EUCLIDEAN Mean similarity: 0.3422074590511125
COSINE Mean similarity: 0.9046576368932006
Number of matching relationships--PEARSON: 628
Number of matching relationships--EUCLIDEAN: 640
Number of matching relationships--COSINE: 629
[6015783.18077245, 4295092.860910067, 4254006.216405518,
4229511.203192889, 4212548.671594664, 4197684.638543637,
4179819.9993094946, 4168431.747832978, 4162693.848730135,
4155654.86525385]
```



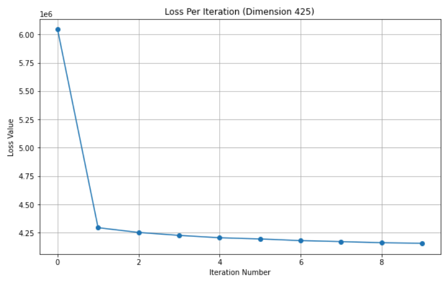
```
Embedding dimension: 365
PEARSON Mean similarity: 0.9047107997990077
EUCLIDEAN Mean similarity: 0.34261946100604057
COSINE Mean similarity: 0.9045122859396998
Number of matching relationships--PEARSON: 631
Number of matching relationships--EUCLIDEAN: 648
Number of matching relationships--COSINE: 631
[6029332.189293582, 4298173.598707879, 4251659.8328238325,
4228041.252775366, 4207325.001507165, 4196278.539890526,
4182363.7115600305, 4171101.0006345804, 4159560.0337802973,
4157007.232331264]
```



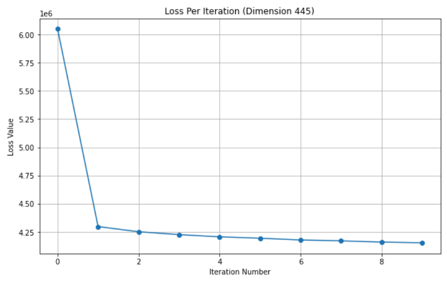
```
Embedding dimension: 385
PEARSON Mean similarity: 0.9046011311969165
EUCLIDEAN Mean similarity: 0.34223541318930056
COSINE Mean similarity: 0.9044488768810007
Number of matching relationships--PEARSON: 625
Number of matching relationships--EUCLIDEAN: 638
Number of matching relationships--COSINE: 624
[6047645.4298871625, 4300124.168977604, 4255095.997720819,
4229385.648858702, 4209058.158524428, 4193266.6566540413,
4181956.663321192, 4170302.3604599875, 4156986.9157061763,
4152326.547283041]
```



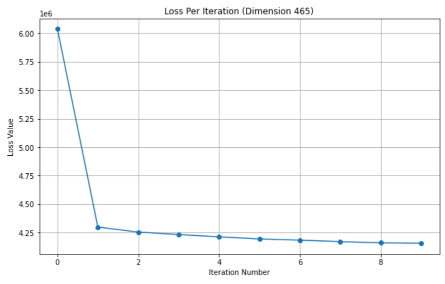
```
Embedding dimension: 405
PEARSON Mean similarity: 0.9045425113582188
EUCLIDEAN Mean similarity: 0.342423070304679
COSINE Mean similarity: 0.9044034504291755
Number of matching relationships--PEARSON: 625
Number of matching relationships--EUCLIDEAN: 643
Number of matching relationships--COSINE: 629
[6043238.909249231, 4294939.3275324935, 4252995.204146886,
4227135.580148362, 4206252.926559636, 4195594.694943711,
4181076.9013233664, 4172197.075564964, 4162302.895810322,
4157475.9144466016]
```



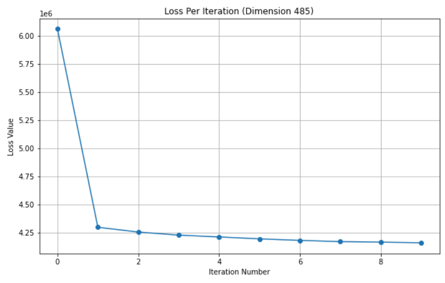
```
Embedding dimension: 425
PEARSON Mean similarity: 0.904744487745547
EUCLIDEAN Mean similarity: 0.3422614439199383
COSINE Mean similarity: 0.904778236546932
Number of matching relationships--PEARSON: 627
Number of matching relationships--EUCLIDEAN: 639
Number of matching relationships--COSINE: 623
[6047907.750447888, 4298119.904351442, 4252341.859147504,
4226492.077240803, 4207336.92481194, 4194959.176095729,
4179406.8563869623, 4171798.390873719, 4160889.7642436167,
4154639.7596410587]
```



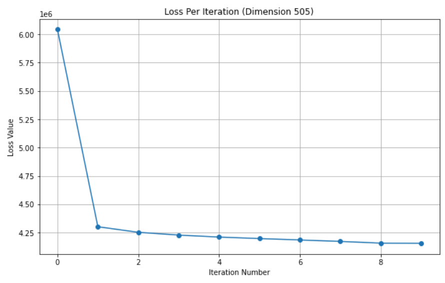
```
Embedding dimension: 445
PEARSON Mean similarity: 0.9041145440218544
EUCLIDEAN Mean similarity: 0.3426036479025859
COSINE Mean similarity: 0.9044285003684648
Number of matching relationships--PEARSON: 630
Number of matching relationships--EUCLIDEAN: 638
Number of matching relationships--COSINE: 630
[6037067.9988161195, 4297875.9620246375, 4254178.059345987,
4232201.092474582, 4212048.235104669, 4193701.1271569594,
4183005.126427357, 4170166.3999308096, 4159349.1907550637,
4156267.3444333947]
```



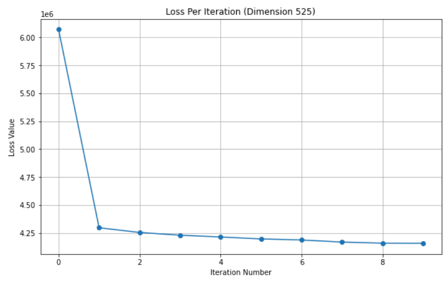
```
Embedding dimension: 465
PEARSON Mean similarity: 0.9044833663647425
EUCLIDEAN Mean similarity: 0.3425848455865767
COSINE Mean similarity: 0.9046223596772908
Number of matching relationships--PEARSON: 625
Number of matching relationships--EUCLIDEAN: 637
Number of matching relationships--COSINE: 625
[6060632.318188168, 4297360.62128766, 4254997.658575683,
4227465.656142491, 4211264.001789518, 4194419.390492938,
4180632.4474988915, 4169798.6328645735, 4165139.7407079125,
4159508.937732628]
```



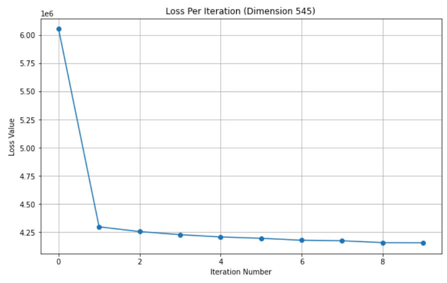
```
Embedding dimension: 485
PEARSON Mean similarity: 0.9042525349198627
EUCLIDEAN Mean similarity: 0.3425718623038761
COSINE Mean similarity: 0.9041571810115211
Number of matching relationships--PEARSON: 624
Number of matching relationships--EUCLIDEAN: 641
Number of matching relationships--COSINE: 628
[6042441.391733196, 4302110.808619763, 4252770.61083107,
4228256.840449234, 4210707.226084114, 4197568.292872887,
4185147.8808660237, 4172399.034996728, 4157194.0476633627,
4155976.5849667047]
```



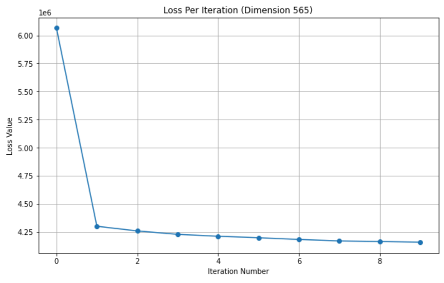
```
Embedding dimension: 505
PEARSON Mean similarity: 0.9044184286097833
EUCLIDEAN Mean similarity: 0.34275821836519454
COSINE Mean similarity: 0.9041570421159708
Number of matching relationships--PEARSON: 629
Number of matching relationships--EUCLIDEAN: 638
Number of matching relationships--COSINE: 625
[6070910.912483594, 4296064.04019345, 4254004.586844532,
4229339.634868467, 4213654.614212055, 4196016.709777263,
4186576.728855772, 4167976.791724059, 4158153.627095255,
4157187.612196693]
```



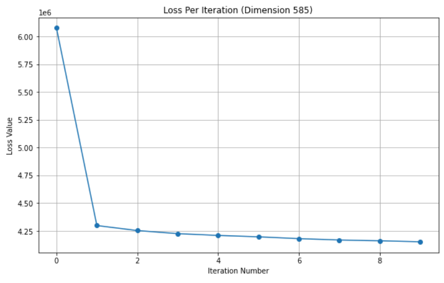
```
Embedding dimension: 525
PEARSON Mean similarity: 0.9040555658157327
EUCLIDEAN Mean similarity: 0.342356286760271
COSINE Mean similarity: 0.9039920373012256
Number of matching relationships--PEARSON: 625
Number of matching relationships--EUCLIDEAN: 641
Number of matching relationships--COSINE: 625
[6052861.041067078, 4297976.564014667, 4255722.171488363,
4228264.895385503, 4208127.066681387, 4196099.026857423,
4178716.861093529, 4174312.51421974, 4157727.547897656,
4156415.7205356625]
```



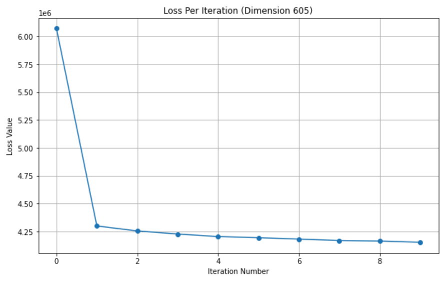
```
Embedding dimension: 545
PEARSON Mean similarity: 0.9041968166740113
EUCLIDEAN Mean similarity: 0.3425536529000084
COSINE Mean similarity: 0.9042192701398886
Number of matching relationships--PEARSON: 630
Number of matching relationships--EUCLIDEAN: 640
Number of matching relationships--COSINE: 628
[6065064.140121881, 4298956.1045243, 4257359.171889935,
4226863.392922757, 4209970.135840638, 4196687.966263355,
4181341.367586384, 4168329.8861162486, 4162931.9434934426,
4156991.5537313554]
```



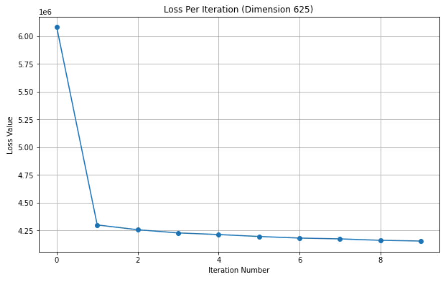
```
Embedding dimension: 565
PEARSON Mean similarity: 0.9044501965295969
EUCLIDEAN Mean similarity: 0.34294062829968497
COSINE Mean similarity: 0.9043921277653343
Number of matching relationships--PEARSON: 632
Number of matching relationships--EUCLIDEAN: 647
Number of matching relationships--COSINE: 634
[6076467.7290897295, 4298511.155189814, 4253589.740589268,
4225550.798996254, 4209581.7271644585, 4197053.386634201,
4180654.6082967496, 4168480.2554071415, 4161290.10006016,
4152538.4074417218]
```



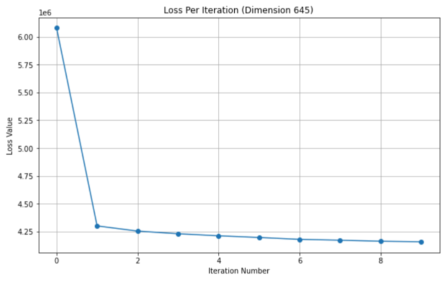
```
Embedding dimension: 585
PEARSON Mean similarity: 0.9041833240841481
EUCLIDEAN Mean similarity: 0.34254822111200195
COSINE Mean similarity: 0.9039086506673117
Number of matching relationships--PEARSON: 622
Number of matching relationships--EUCLIDEAN: 641
Number of matching relationships--COSINE: 621
[6070881.172609721, 4299740.024461461, 4255357.124643056,
4227602.883821108, 4204832.268372606, 4194521.62278747,
4182205.390247991, 4169110.094398275, 4164739.5954559,
4153563.0889399014]
```



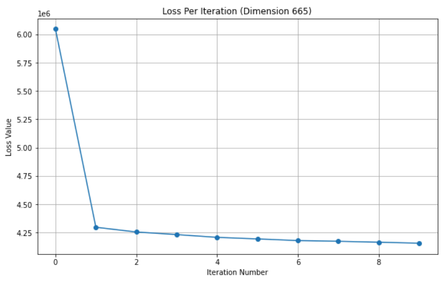
```
Embedding dimension: 605
PEARSON Mean similarity: 0.9043111412352615
EUCLIDEAN Mean similarity: 0.34272844435721417
COSINE Mean similarity: 0.9044914510436741
Number of matching relationships--PEARSON: 627
Number of matching relationships--EUCLIDEAN: 642
Number of matching relationships--COSINE: 627
[6079280.7185532795, 4297828.679747888, 4254603.873011452,
4225899.273108698, 4211245.745630354, 4193778.3134496706,
4179410.622319399, 4172160.812396765, 4159266.459933868,
4152776.252979377]
```



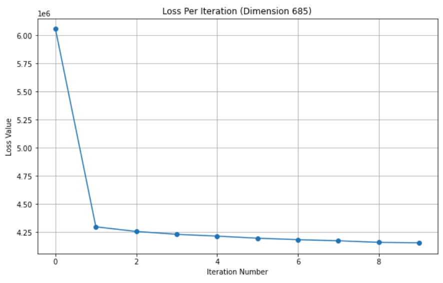
```
Embedding dimension: 625
PEARSON Mean similarity: 0.9045102056055336
EUCLIDEAN Mean similarity: 0.3430209597245277
COSINE Mean similarity: 0.9044384453462181
Number of matching relationships--PEARSON: 631
Number of matching relationships--EUCLIDEAN: 642
Number of matching relationships--COSINE: 631
[6079794.463033184, 4299492.771770361, 4252957.273211087,
4228866.643041185, 4210604.995095546, 4195389.5558354575,
4178447.907729198, 4170985.03285698, 4161627.865357158,
4156761.7302391664]
```



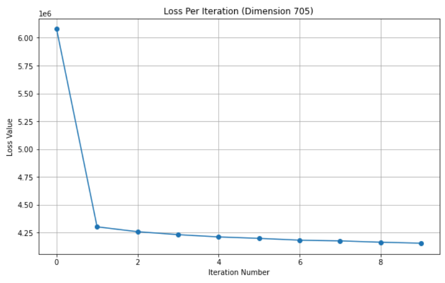
```
Embedding dimension: 645
PEARSON Mean similarity: 0.9040581014342991
EUCLIDEAN Mean similarity: 0.3431589482277852
COSINE Mean similarity: 0.9043010038322324
Number of matching relationships--PEARSON: 631
Number of matching relationships--EUCLIDEAN: 644
Number of matching relationships--COSINE: 630
[6046272.766888468, 4297268.5231854655, 4255537.899000927,
4232616.974221769, 4208232.13442828, 4194319.971546403,
4179831.917737171, 4173569.637753389, 4165330.3278582874,
4156576.2217713543]
```



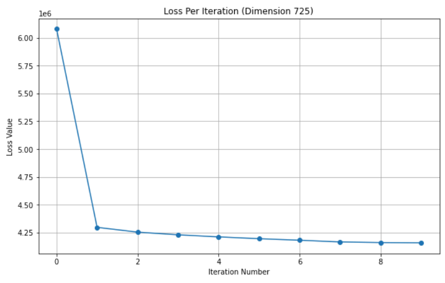
```
Embedding dimension: 665
PEARSON Mean similarity: 0.9041428130697816
EUCLIDEAN Mean similarity: 0.34297051302967607
COSINE Mean similarity: 0.9043545214429035
Number of matching relationships--PEARSON: 624
Number of matching relationships--EUCLIDEAN: 636
Number of matching relationships--COSINE: 628
[6057239.079887346, 4296040.988492977, 4254983.515166098,
4229415.727639898, 4213432.682560355, 4194898.666257698,
4181670.9540884653, 4172120.330009625, 4157860.003626209,
4154130.9572446723]
```



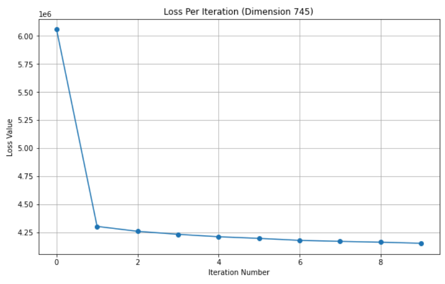
```
Embedding dimension: 685
PEARSON Mean similarity: 0.9041062119792448
EUCLIDEAN Mean similarity: 0.3431313163948904
COSINE Mean similarity: 0.9044932513328563
Number of matching relationships--PEARSON: 633
Number of matching relationships--EUCLIDEAN: 642
Number of matching relationships--COSINE: 633
[6077977.367698334, 4300691.266976029, 4256758.398630418,
4230331.863528816, 4209899.869081589, 4196857.298368417,
4180644.908184567, 4174138.695937064, 4162331.1062220866,
4153425.1726426734]
```



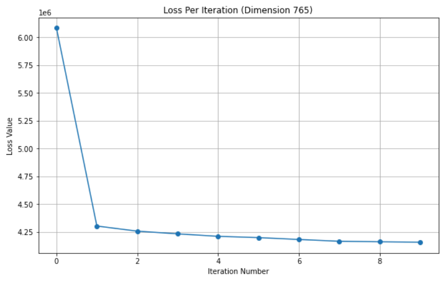
```
Embedding dimension: 705
PEARSON Mean similarity: 0.9043879200471909
EUCLIDEAN Mean similarity: 0.3432816075081311
COSINE Mean similarity: 0.9043051484416471
Number of matching relationships--PEARSON: 634
Number of matching relationships--EUCLIDEAN: 643
Number of matching relationships--COSINE: 632
[6078621.119741175, 4297664.32105765, 4254738.509356653,
4230514.851650077, 4211975.324559297, 4195776.056935199,
4181868.944147008, 4166621.3591715833, 4160972.7441216097,
4159130.2542161695]
```



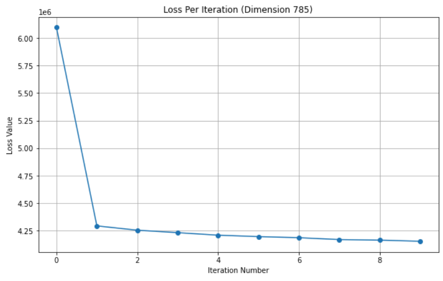
```
Embedding dimension: 725
PEARSON Mean similarity: 0.904324050351119
EUCLIDEAN Mean similarity: 0.34322744239908554
COSINE Mean similarity: 0.9040150032437887
Number of matching relationships--PEARSON: 629
Number of matching relationships--EUCLIDEAN: 646
Number of matching relationships--COSINE: 630
[6054934.978632934, 4302555.7856730735, 4258877.5871596,
4231990.205320817, 4210492.830086933, 4195926.146499623,
4178252.5954075246, 4169339.354357239, 4161961.811181345,
4152384.078722503]
```



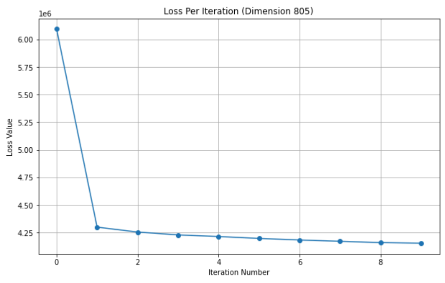
```
Embedding dimension: 745
PEARSON Mean similarity: 0.9042112102536718
EUCLIDEAN Mean similarity: 0.34301038753334845
COSINE Mean similarity: 0.9040678692004881
Number of matching relationships--PEARSON: 634
Number of matching relationships--EUCLIDEAN: 652
Number of matching relationships--COSINE: 634
[6081318.391685242, 4301941.141750962, 4256038.873635068,
4231620.814291155, 4209444.178278193, 4197839.387377739,
4181073.398081734, 4165013.967642512, 4160443.7513884916,
4156606.0611438826]
```



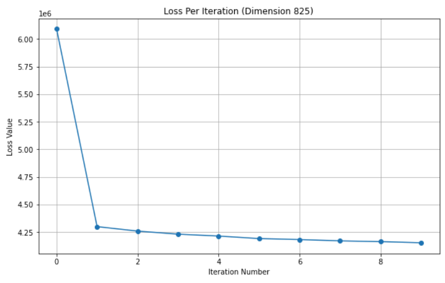
```
Embedding dimension: 765
PEARSON Mean similarity: 0.904333028976463
EUCLIDEAN Mean similarity: 0.3432002685270774
COSINE Mean similarity: 0.9044585533649213
Number of matching relationships--PEARSON: 629
Number of matching relationships--EUCLIDEAN: 643
Number of matching relationships--COSINE: 631
[6096053.400760389, 4293959.786409268, 4254803.282621654,
4232440.9711369425, 4209484.156973254, 4196547.726843374,
4186635.575559064, 4169706.148081259, 4164553.6574242255,
4154184.1090868763]
```



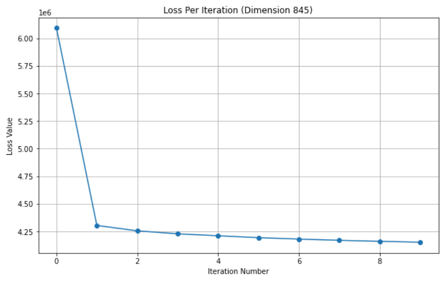
```
Embedding dimension: 785
PEARSON Mean similarity: 0.9043386600218284
EUCLIDEAN Mean similarity: 0.34323085949727317
COSINE Mean similarity: 0.9041159789002277
Number of matching relationships--PEARSON: 630
Number of matching relationships--EUCLIDEAN: 645
Number of matching relationships--COSINE: 630
[6094163.825740423, 4298405.853888585, 4253984.260823085,
4227823.090787411, 4213439.037123806, 4195972.284169775,
4181889.1487724697, 4169901.1402812107, 4158370.2868055664,
4152811.7999706194]
```



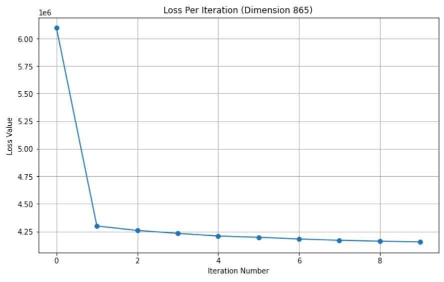
```
Embedding dimension: 805
PEARSON Mean similarity: 0.9042589917316747
EUCLIDEAN Mean similarity: 0.34330837976633183
COSINE Mean similarity: 0.9043312633407344
Number of matching relationships--PEARSON: 624
Number of matching relationships--EUCLIDEAN: 643
Number of matching relationships--COSINE: 625
[6091124.48210846, 4298668.971978722, 4258220.7689286005,
4230221.095826236, 4213254.204729862, 4190291.677041627,
4181388.6875821818, 4169465.802268459, 4162745.949437316,
4152191.5099494504]
```



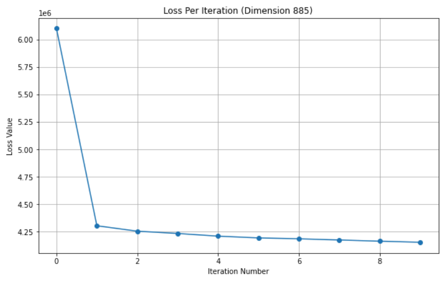
```
Embedding dimension: 825
PEARSON Mean similarity: 0.9040978712028379
EUCLIDEAN Mean similarity: 0.34337737239798205
COSINE Mean similarity: 0.904122399086438
Number of matching relationships--PEARSON: 624
Number of matching relationships--EUCLIDEAN: 641
Number of matching relationships--COSINE: 624
[6092737.836771482, 4304606.253793874, 4255863.229377883,
4228851.316592192, 4211353.441593712, 4193879.758830732,
4181257.442462145, 4170025.570560882, 4160899.469898176,
4152930.439842345]
```



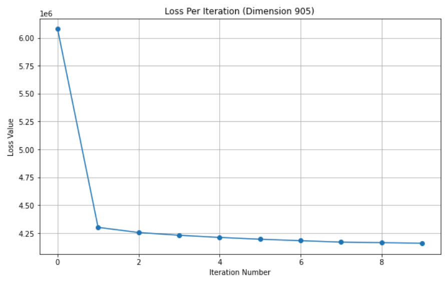
```
Embedding dimension: 845
PEARSON Mean similarity: 0.9043675773429025
EUCLIDEAN Mean similarity: 0.34370545108350026
COSINE Mean similarity: 0.9046038682562914
Number of matching relationships--PEARSON: 633
Number of matching relationships--EUCLIDEAN: 635
Number of matching relationships--COSINE: 633
[6095437.874170452, 4298583.032647645, 4258562.9045351455,
4231660.754347186, 4207944.550826855, 4196378.376998458,
4180763.9681514, 4168954.4768050415, 4160567.4969660253,
4155542.749071468]
```



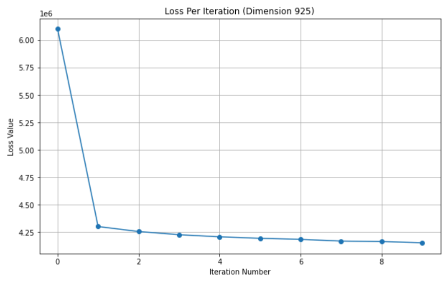
```
Embedding dimension: 865
PEARSON Mean similarity: 0.9042151883574673
EUCLIDEAN Mean similarity: 0.34365061706419003
COSINE Mean similarity: 0.9042677244020177
Number of matching relationships--PEARSON: 635
Number of matching relationships--EUCLIDEAN: 641
Number of matching relationships--COSINE: 635
[6100683.138999915, 4302714.288146027, 4253256.566211056,
4232336.046641136, 4207955.044579648, 4191921.2898756424,
4183791.4175198623, 4172981.140603363, 4161998.719066974,
4152101.780266618]
```



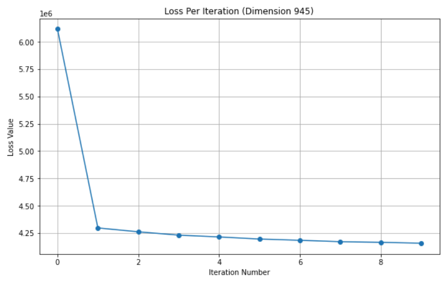
```
Embedding dimension: 885
PEARSON Mean similarity: 0.9042109169671243
EUCLIDEAN Mean similarity: 0.34357859824221454
COSINE Mean similarity: 0.9043222086468334
Number of matching relationships--PEARSON: 625
Number of matching relationships--EUCLIDEAN: 642
Number of matching relationships--COSINE: 625
[6078264.600293264, 4301424.031494208, 4255267.767125513,
4230637.399367019, 4211242.940501962, 4195097.446133877,
4181965.303743323, 4169266.609254282, 4164145.9040498696,
4159463.192364946]
```



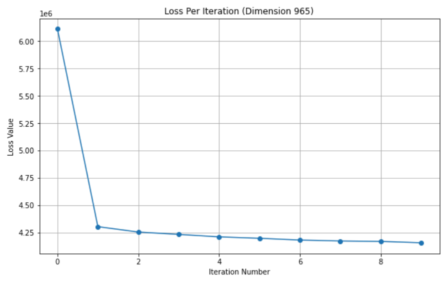
```
Embedding dimension: 905
PEARSON Mean similarity: 0.9040821447330176
EUCLIDEAN Mean similarity: 0.3438437154057107
COSINE Mean similarity: 0.9041288981585595
Number of matching relationships--PEARSON: 634
Number of matching relationships--EUCLIDEAN: 646
Number of matching relationships--COSINE: 633
[6100947.541513124, 4301930.728734484, 4256522.513514657,
4227293.952128068, 4208269.286179772, 4194735.149560582,
4184974.127759196, 4169532.129029412, 4165158.6981001287,
4153984.4001714103]
```



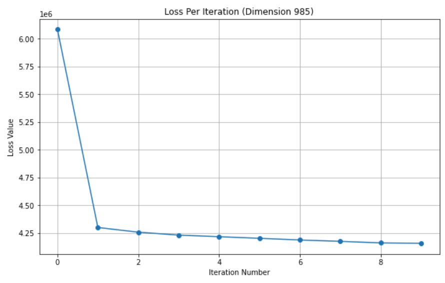
```
Embedding dimension: 925
PEARSON Mean similarity: 0.9042460648664872
EUCLIDEAN Mean similarity: 0.3436221059351235
COSINE Mean similarity: 0.9044986879807984
Number of matching relationships--PEARSON: 632
Number of matching relationships--EUCLIDEAN: 646
Number of matching relationships--COSINE: 634
[6116206.753376877, 4296648.092167848, 4261321.977466879,
4230612.319944756, 4214238.357831754, 4194974.026685093,
4183567.9690515082, 4170752.5804949235, 4164863.0058936607,
4156939.490371226]
```



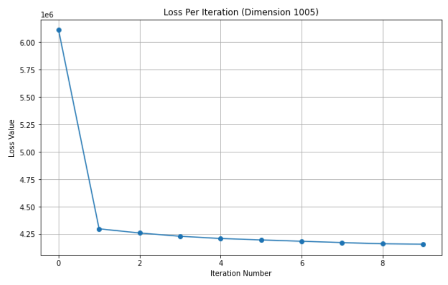
```
Embedding dimension: 945
PEARSON Mean similarity: 0.9043426905284075
EUCLIDEAN Mean similarity: 0.3438097137468076
COSINE Mean similarity: 0.9044855120615205
Number of matching relationships--PEARSON: 627
Number of matching relationships--EUCLIDEAN: 642
Number of matching relationships--COSINE: 631
[6111442.248440595, 4302150.452518694, 4252904.229430642,
4231117.436708504, 4208689.835913911, 4195652.302548449,
4179505.1606567404, 4170930.574605756, 4167105.8714216133,
4154641.853985739]
```



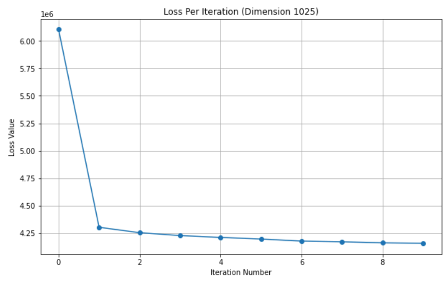
```
Embedding dimension: 965
PEARSON Mean similarity: 0.9044317113067762
EUCLIDEAN Mean similarity: 0.343874703693249
COSINE Mean similarity: 0.9043584178434341
Number of matching relationships--PEARSON: 629
Number of matching relationships--EUCLIDEAN: 647
Number of matching relationships--COSINE: 628
[6084608.597861104, 4298419.626675557, 4256228.053733421,
4229689.154631051, 4215309.222114534, 4200840.784709983,
4185636.508030059, 4173481.276002934, 4159241.4337971397,
4156010.188483618]
```



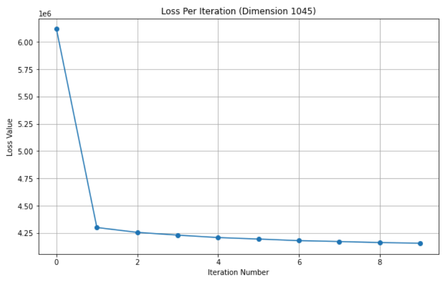
```
Embedding dimension: 985
PEARSON Mean similarity: 0.9042272463506926
EUCLIDEAN Mean similarity: 0.34368648902352134
COSINE Mean similarity: 0.9041986282326094
Number of matching relationships--PEARSON: 629
Number of matching relationships--EUCLIDEAN: 645
Number of matching relationships--COSINE: 633
[6112170.966420652, 4297835.252176315, 4259533.780030678,
4229740.80843689, 4209417.142964881, 4196278.340959745,
4184172.3569164504, 4171955.2516489127, 4161344.6305249343,
4157360.9703598847]
```



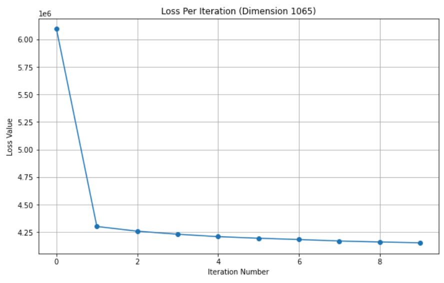
```
Embedding dimension: 1005
PEARSON Mean similarity: 0.9043696557049209
EUCLIDEAN Mean similarity: 0.34374307621176875
COSINE Mean similarity: 0.9045581099201693
Number of matching relationships--PEARSON: 630
Number of matching relationships--EUCLIDEAN: 640
Number of matching relationships--COSINE: 630
[6103924.723303133, 4302652.078287401, 4253493.875602235,
4227227.966322633, 4210227.853115536, 4195682.479106981,
4176885.8292162707, 4170747.837680503, 4160965.105507168,
4157071.089680046]
```



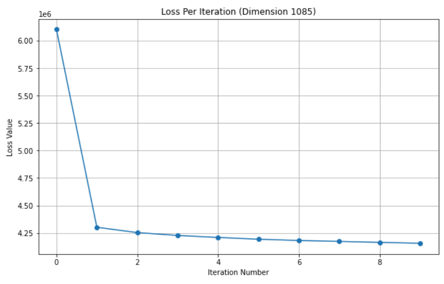
```
Embedding dimension: 1025
PEARSON Mean similarity: 0.9046057685615393
EUCLIDEAN Mean similarity: 0.344031141568498
COSINE Mean similarity: 0.9044124932901898
Number of matching relationships--PEARSON: 633
Number of matching relationships--EUCLIDEAN: 643
Number of matching relationships--COSINE: 629
[6118436.050311854, 4300082.967586911, 4255584.32080235,
4230337.084215631, 4208113.684951444, 4194436.533549336,
4179808.056638784, 4171704.1491096797, 4162156.5393342446,
4156331.3787497273]
```



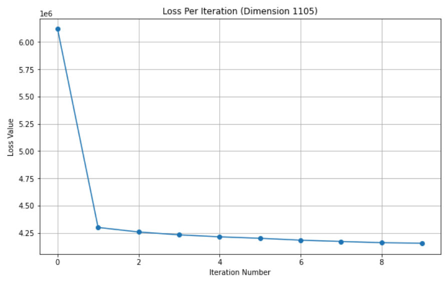
```
Embedding dimension: 1045
PEARSON Mean similarity: 0.9043185381980907
EUCLIDEAN Mean similarity: 0.343687291828533
COSINE Mean similarity: 0.9041285485249326
Number of matching relationships--PEARSON: 628
Number of matching relationships--EUCLIDEAN: 627
Number of matching relationships--COSINE: 625
[6094296.057230148, 4301795.175141901, 4259295.411659789,
4231431.924648377, 4209650.091660712, 4195210.428422944,
4183404.6683308403, 4170170.95509516, 4161138.6671289774,
4154020.8475964554]
```



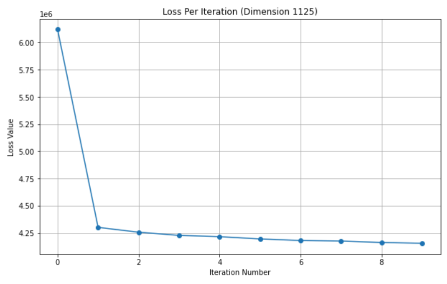
```
Embedding dimension: 1065
PEARSON Mean similarity: 0.9042991480411772
EUCLIDEAN Mean similarity: 0.34403521123629727
COSINE Mean similarity: 0.9042770211066242
Number of matching relationships--PEARSON: 632
Number of matching relationships--EUCLIDEAN: 644
Number of matching relationships--COSINE: 631
[6101347.366813464, 4301513.703144243, 4253182.08594501,
4227100.184120265, 4209047.111753337, 4192758.360563533,
4181400.5236672703, 4172880.751147477, 4164264.52563816,
4155991.4913080847]
```



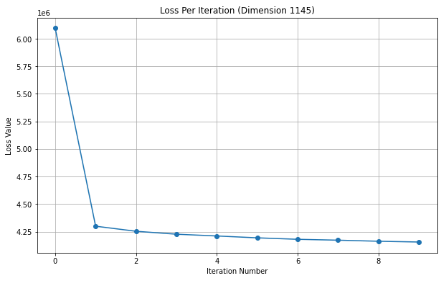
```
Embedding dimension: 1085
PEARSON Mean similarity: 0.9043153900867728
EUCLIDEAN Mean similarity: 0.3438014310783262
COSINE Mean similarity: 0.9043325226979558
Number of matching relationships--PEARSON: 625
Number of matching relationships--EUCLIDEAN: 639
Number of matching relationships--COSINE: 626
[6117386.459819261, 4299748.628731675, 4258155.488671518,
4232078.024619526, 4213842.098293813, 4200399.43609126,
4183067.202774581, 4171284.114222489, 4160040.1582389977,
4155379.683650423]
```



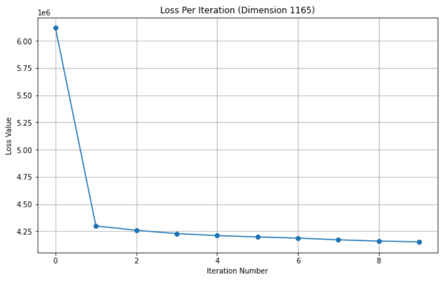
```
Embedding dimension: 1105
PEARSON Mean similarity: 0.9047110570234245
EUCLIDEAN Mean similarity: 0.34405207831186946
COSINE Mean similarity: 0.9044552962220049
Number of matching relationships--PEARSON: 636
Number of matching relationships--EUCLIDEAN: 638
Number of matching relationships--COSINE: 630
[6119260.871597962, 4301030.4325865265, 4256697.876863211,
4227315.322625108, 4215638.172128886, 4194709.8717632145,
4180432.839141152, 4175267.9276078385, 4162253.9356494015,
4154928.061221025]
```



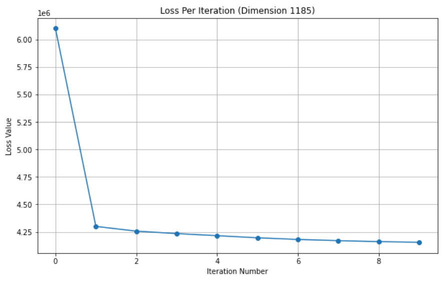
```
Embedding dimension: 1125
PEARSON Mean similarity: 0.9043129849116947
EUCLIDEAN Mean similarity: 0.3440293861456777
COSINE Mean similarity: 0.9042681154507478
Number of matching relationships--PEARSON: 625
Number of matching relationships--EUCLIDEAN: 648
Number of matching relationships--COSINE: 626
[6094799.087514361, 4299394.266384188, 4253509.502800984,
4226838.847493634, 4210806.837178803, 4193864.9105851175,
4180571.7603702545, 4172592.5274353735, 4162915.858607511,
4155901.883388365]
```



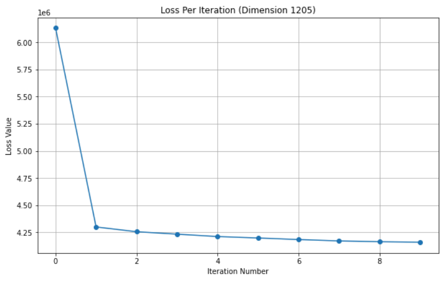
```
Embedding dimension: 1145
PEARSON Mean similarity: 0.9044267507916539
EUCLIDEAN Mean similarity: 0.3437765187667779
COSINE Mean similarity: 0.9043499638093979
Number of matching relationships--PEARSON: 639
Number of matching relationships--EUCLIDEAN: 652
Number of matching relationships--COSINE: 636
[6116559.029901898, 4299670.787803523, 4259083.923450363,
4228949.311391169, 4210564.000454803, 4198198.910720809,
4187397.140983826, 4171952.0884032506, 4159816.8927647234,
4153756.4208581736]
```



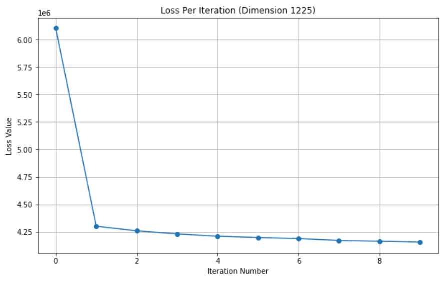
```
Embedding dimension: 1165
PEARSON Mean similarity: 0.9046056871400098
EUCLIDEAN Mean similarity: 0.34399135722015245
COSINE Mean similarity: 0.9045048002302206
Number of matching relationships--PEARSON: 634
Number of matching relationships--EUCLIDEAN: 647
Number of matching relationships--COSINE: 629
[6098939.0465518255, 4299766.359741903, 4256781.368566061,
4234441.170688384, 4215460.065482852, 4196089.967496756,
4181166.4454739895, 4170184.3292322424, 4161317.2012759773,
4155445.547845625]
```



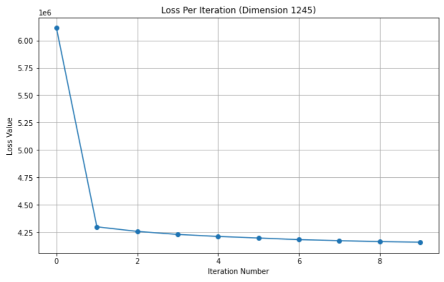
```
Embedding dimension: 1185
PEARSON Mean similarity: 0.9040702481741744
EUCLIDEAN Mean similarity: 0.34398068804156307
COSINE Mean similarity: 0.9043254147956431
Number of matching relationships--PEARSON: 630
Number of matching relationships--EUCLIDEAN: 649
Number of matching relationships--COSINE: 631
[6130185.96538312, 4299774.004832536, 4255872.698829565,
4233315.6562060835, 4211492.791601313, 4198136.9921304835,
4183860.305617726, 4171215.7971636197, 4163839.9466139367,
4159427.6959029087]
```



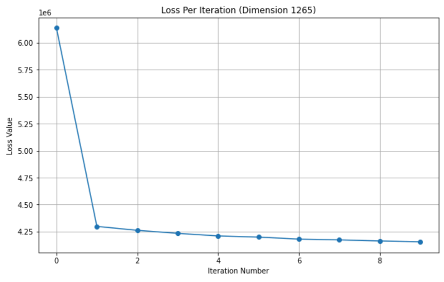
```
Embedding dimension: 1205
PEARSON Mean similarity: 0.9044700309832304
EUCLIDEAN Mean similarity: 0.3442642476040996
COSINE Mean similarity: 0.9045307103076748
Number of matching relationships--PEARSON: 624
Number of matching relationships--EUCLIDEAN: 645
Number of matching relationships--COSINE: 627
[6103892.55577773, 4300207.831692472, 4257944.9200094845,
4229814.776037049, 4208690.2598094735, 4196846.394709421,
4187429.6213457026, 4170204.6389363036, 4162704.7390050734,
4155647.5655970983]
```



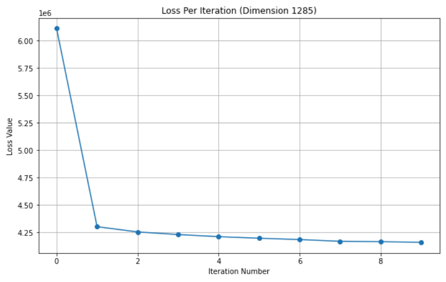
```
Embedding dimension: 1225
PEARSON Mean similarity: 0.9045496919482638
EUCLIDEAN Mean similarity: 0.34470626445077196
COSINE Mean similarity: 0.9045331721115676
Number of matching relationships--PEARSON: 630
Number of matching relationships--EUCLIDEAN: 643
Number of matching relationships--COSINE: 627
[6113262.915596962, 4298718.353436403, 4256779.0609127665,
4229223.699646336, 4210813.6741306335, 4196530.337983615,
4181583.6052221986, 4172335.1032256745, 4163788.170957185,
4158272.191006861]
```



```
Embedding dimension: 1245
PEARSON Mean similarity: 0.9043882832041167
EUCLIDEAN Mean similarity: 0.34427394233038716
COSINE Mean similarity: 0.9043040212183266
Number of matching relationships--PEARSON: 629
Number of matching relationships--EUCLIDEAN: 646
Number of matching relationships--COSINE: 629
[6135511.443470084, 4298306.568125915, 4262095.876658207,
4234038.63030334, 4209941.449013263, 4199405.529034329,
4180583.6379593294, 4173457.9141750485, 4163311.9992041048,
4156098.932177159]
```



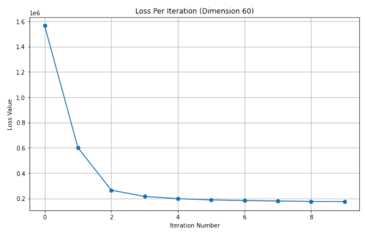
```
Embedding dimension: 1265
PEARSON Mean similarity: 0.9045846584100314
EUCLIDEAN Mean similarity: 0.34439144606428274
COSINE Mean similarity: 0.9045256111787376
Number of matching relationships--PEARSON: 634
Number of matching relationships--EUCLIDEAN: 649
Number of matching relationships--COSINE: 633
[6110895.459922119, 4301575.72561925, 4254008.429722363,
4229807.132430209, 4210703.487886657, 4196422.24874567,
4184250.4459362393, 4168279.7912669512, 4164593.7389339563,
4159456.7991768736]
```



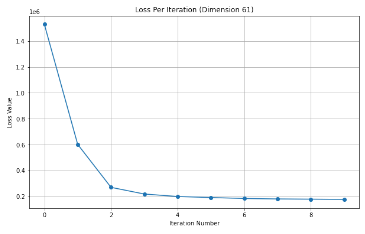
```
Embedding dimension: 1285
PEARSON Mean similarity: 0.9044869737047565
EUCLIDEAN Mean similarity: 0.3443318301500915
COSINE Mean similarity: 0.9045823586039916
Number of matching relationships--PEARSON: 623
Number of matching relationships--EUCLIDEAN: 645
Number of matching relationships--COSINE: 624
```

C Tuning Experiments on Walklength and Measure Methods of Node2Vec with KNN on Cora Dataset

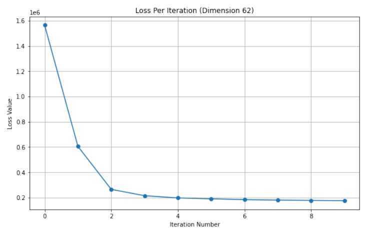
C.1 Results of Tuning Experiments on Walklength=10



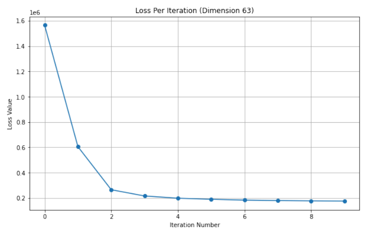
```
Embedding dimension: 60
PEARSON Mean similarity: 0.9452529169079824
EUCLIDEAN Mean similarity: 0.4603691767378431
COSINE Mean similarity: 0.9448522407159847
Number of matching relationships--PEARSON: 615
Number of matching relationships--EUCLIDEAN: 619
Number of matching relationships--COSINE: 614
[1529867.906140164, 601438.5789261346, 269112.11179238046,
217398.74431680932, 198237.20930534246, 190256.18599948054,
182932.37692469297, 179433.19931490673, 177613.18372376647,
175501.75016469488]
```



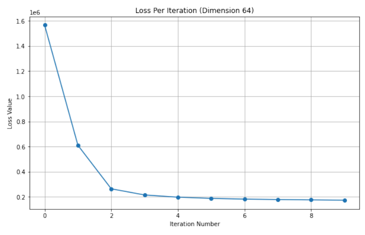
```
Embedding dimension: 61
PEARSON Mean similarity: 0.9457667405707445
EUCLIDEAN Mean similarity: 0.4618549830100033
COSINE Mean similarity: 0.9450182952768165
Number of matching relationships--PEARSON: 612
Number of matching relationships--EUCLIDEAN: 617
Number of matching relationships--COSINE: 609
[1564180.2988345902, 604664.0506536184, 264337.11240597867,
214294.70836551668, 196750.9475817614, 189455.21369143404,
183007.27604068822, 179512.878390573, 177456.07041646083,
175283.62467056932]
```



```
Embedding dimension: 62
PEARSON Mean similarity: 0.9445096893535935
EUCLIDEAN Mean similarity: 0.4598354767133073
COSINE Mean similarity: 0.9443692366516925
Number of matching relationships--PEARSON: 624
Number of matching relationships--EUCLIDEAN: 619
Number of matching relationships--COSINE: 621
[1564654.633842849, 604809.687445015, 264870.7138816951,
215548.4600159427, 197983.3980053087, 189230.57670665183,
182825.7164378639, 179356.4957662974, 177065.29882090475,
175432.2583844047]
```

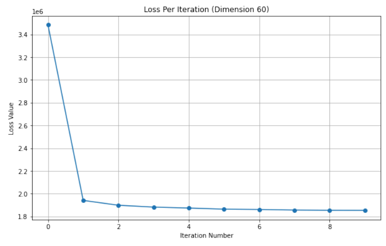


```
Embedding dimension: 63
PEARSON Mean similarity: 0.9446633309346006
EUCLIDEAN Mean similarity: 0.4595568285735002
COSINE Mean similarity: 0.9441292758530316
Number of matching relationships--PEARSON: 619
Number of matching relationships--EUCLIDEAN: 622
Number of matching relationships--COSINE: 615
[1565835.0851703547, 610335.8259257312, 264778.6894485299,
215831.007246039, 198234.61817383836, 188503.384887818,
182831.54979506732, 179645.91082443297, 176947.0292548761,
174234.8734583343]
```

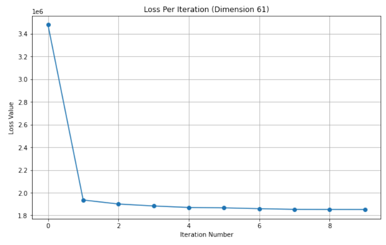


```
Embedding dimension: 64
PEARSON Mean similarity: 0.9440861497707169
EUCLIDEAN Mean similarity: 0.4592918413604556
COSINE Mean similarity: 0.9437486042828468
Number of matching relationships--PEARSON: 614
Number of matching relationships--EUCLIDEAN: 613
Number of matching relationships--COSINE: 614
[3482846.9108588696, 1941366.671237366, 1898773.7391055133,
1882578.668545418, 1874236.5781358744, 1864658.3478462538,
1861520.0190446575, 1856576.2183024813, 1854685.165771622,
1854105.256607025]
```

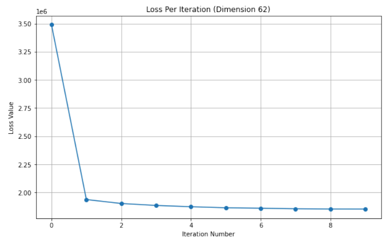
C.2 Results of Tuning Walklength Experiments on Walklength=40



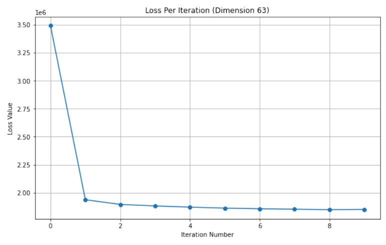
```
Embedding dimension: 60
PEARSON Mean similarity: 0.9276531382888952
EUCLIDEAN Mean similarity: 0.3813062773497453
COSINE Mean similarity: 0.927568788119926
Number of matching relationships--PEARSON: 637
Number of matching relationships--EUCLIDEAN: 643
Number of matching relationships--COSINE: 641
[3479278.339981241, 1935265.8201509216, 1900098.867934915,
1882715.974475006, 1869490.911208253, 1866334.0818388022,
1858874.7801095084, 1852882.679989125, 1852165.6295819054,
1851771.8956255594]
```



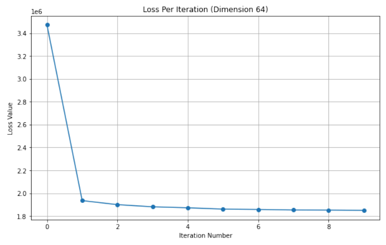
```
Embedding dimension: 61
PEARSON Mean similarity: 0.9273508401529827
EUCLIDEAN Mean similarity: 0.3808595356596097
COSINE Mean similarity: 0.9270133036806453
Number of matching relationships--PEARSON: 641
Number of matching relationships--EUCLIDEAN: 659
Number of matching relationships--COSINE: 643
[3489437.451798389, 1936816.247956731, 1901801.6626036947,
1884352.9779876722, 1873141.9984732063, 1863723.92961643,
1859816.7536442787, 1854943.3868086091, 1852845.874395678,
1852462.174152706]
```



```
Embedding dimension: 62
PEARSON Mean similarity: 0.9270636647279364
EUCLIDEAN Mean similarity: 0.37990931725044197
COSINE Mean similarity: 0.9267496599228379
Number of matching relationships--PEARSON: 640
Number of matching relationships--EUCLIDEAN: 646
Number of matching relationships--COSINE: 638
[3489653.0245607416, 1938293.2107963315, 1896816.4960180675,
1883074.9825699958, 1872567.9603032004, 1863575.694466059,
1858008.978296804, 1853881.946583526, 1850274.7107097842,
1852331.6470752964]
```

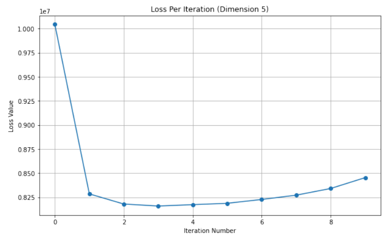


```
Embedding dimension: 63
PEARSON Mean similarity: 0.9269599018632221
EUCLIDEAN Mean similarity: 0.38012439081200466
COSINE Mean similarity: 0.9265898449558275
Number of matching relationships--PEARSON: 638
Number of matching relationships--EUCLIDEAN: 647
Number of matching relationships--COSINE: 637
[3472190.798177841, 1935051.355080307, 1899724.3876238079,
1880856.4058097443, 1872137.1575469046, 1860813.1500582458,
1857464.3969639568, 1852877.9162421878, 1852111.658308576,
1849840.2944054415]
```

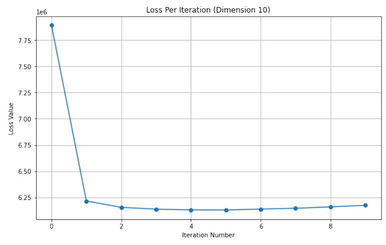


```
Embedding dimension: 64
PEARSON Mean similarity: 0.9258781624685538
EUCLIDEAN Mean similarity: 0.37971023400389814
COSINE Mean similarity: 0.9260587593653353
Number of matching relationships--PEARSON: 632
Number of matching relationships--EUCLIDEAN: 641
Number of matching relationships--COSINE: 639
```

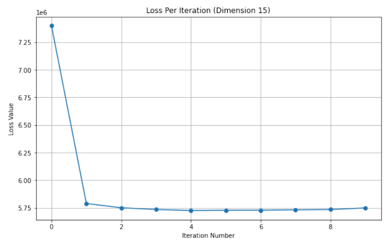
C.3 Results of Tuning Walklength Experiments on Walklength=100



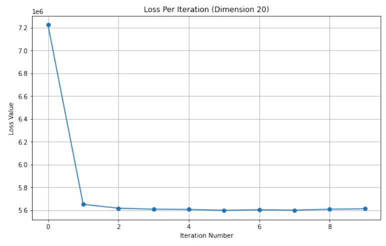
```
Embedding dimension: 5
PEARSON Mean similarity: 0.997028945855235
EUCLIDEAN Mean similarity: 0.5904801308844607
COSINE Mean similarity: 0.9949498899225998
Number of matching relationships--PEARSON: 393
Number of matching relationships--EUCLIDEAN: 398
Number of matching relationships--COSINE: 462
```



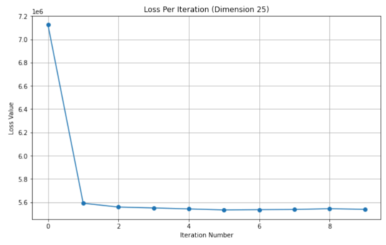
```
Embedding dimension: 10
PEARSON Mean similarity: 0.9785802762300613
EUCLIDEAN Mean similarity: 0.48678765775886207
COSINE Mean similarity: 0.9768414391724715
Number of matching relationships--PEARSON: 625
Number of matching relationships--EUCLIDEAN: 622
Number of matching relationships--COSINE: 634
```



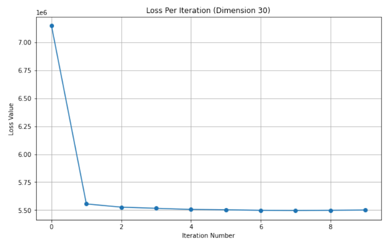
```
Embedding dimension: 15
PEARSON Mean similarity: 0.9672957734484088
EUCLIDEAN Mean similarity: 0.46143590083326513
COSINE Mean similarity: 0.9663185880314477
Number of matching relationships--PEARSON: 648
Number of matching relationships--EUCLIDEAN: 658
Number of matching relationships--COSINE: 649
```



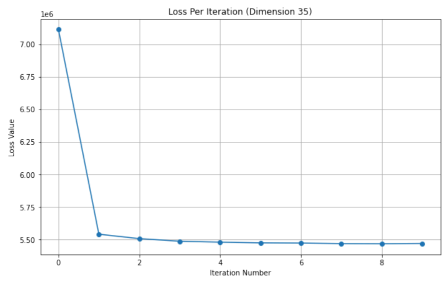
```
Embedding dimension: 20
PEARSON Mean similarity: 0.9596218838825535
EUCLIDEAN Mean similarity: 0.4417890581809964
COSINE Mean similarity: 0.9590493992477259
Number of matching relationships--PEARSON: 652
Number of matching relationships--EUCLIDEAN: 670
Number of matching relationships--COSINE: 653
```



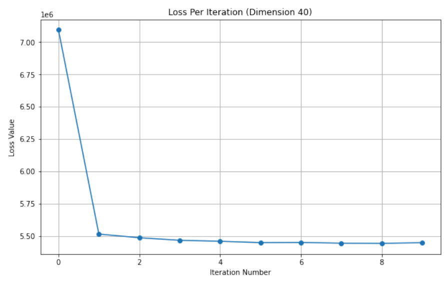
```
Embedding dimension: 25
PEARSON Mean similarity: 0.9526648728498857
EUCLIDEAN Mean similarity: 0.424972237160146
COSINE Mean similarity: 0.9520202107056205
Number of matching relationships--PEARSON: 656
Number of matching relationships--EUCLIDEAN: 684
Number of matching relationships--COSINE: 658
```



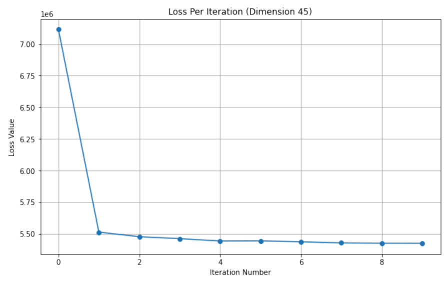
```
Embedding dimension: 30
PEARSON Mean similarity: 0.9477506527696438
EUCLIDEAN Mean similarity: 0.41074410696325486
COSINE Mean similarity: 0.9470599722474857
Number of matching relationships--PEARSON: 654
Number of matching relationships--EUCLIDEAN: 667
Number of matching relationships--COSINE: 656
```



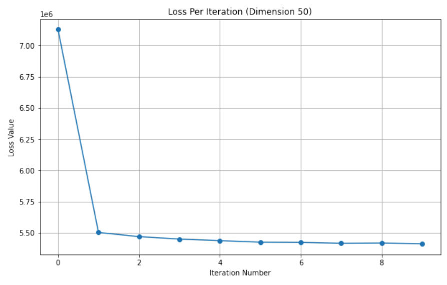
```
Embedding dimension: 35
PEARSON Mean similarity: 0.942622460572371
EUCLIDEAN Mean similarity: 0.3990250559994178
COSINE Mean similarity: 0.9424366608680264
Number of matching relationships--PEARSON: 644
Number of matching relationships--EUCLIDEAN: 668
Number of matching relationships--COSINE: 650
```



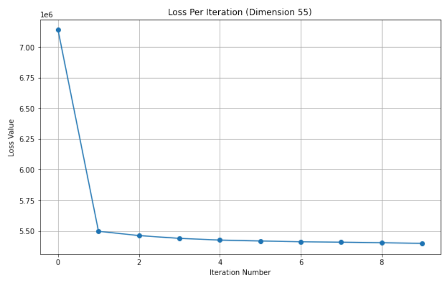
```
Embedding dimension: 40
PEARSON Mean similarity: 0.9385986691563661
EUCLIDEAN Mean similarity: 0.3891472953813291
COSINE Mean similarity: 0.9380013943422845
Number of matching relationships--PEARSON: 647
Number of matching relationships--EUCLIDEAN: 667
Number of matching relationships--COSINE: 649
```



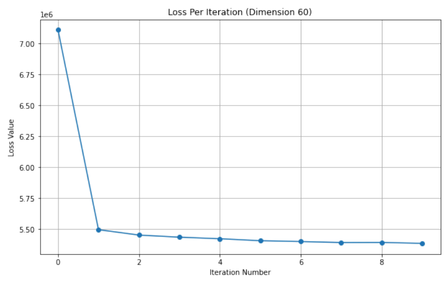
```
Embedding dimension: 45
PEARSON Mean similarity: 0.934621373659751
EUCLIDEAN Mean similarity: 0.38156330525610965
COSINE Mean similarity: 0.9341938221648314
Number of matching relationships--PEARSON: 646
Number of matching relationships--EUCLIDEAN: 659
Number of matching relationships--COSINE: 649
```



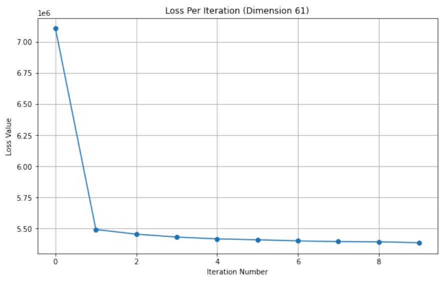
```
Embedding dimension: 50
PEARSON Mean similarity: 0.9321404728966942
EUCLIDEAN Mean similarity: 0.37533848542758597
COSINE Mean similarity: 0.9314320810788491
Number of matching relationships--PEARSON: 643
Number of matching relationships--EUCLIDEAN: 653
Number of matching relationships--COSINE: 643
```



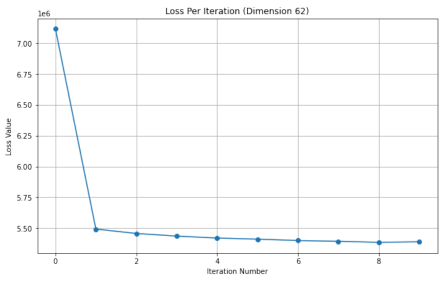
```
Embedding dimension: 55
PEARSON Mean similarity: 0.9289545270505649
EUCLIDEAN Mean similarity: 0.3700033700624632
COSINE Mean similarity: 0.9283567186296426
Number of matching relationships--PEARSON: 653
Number of matching relationships--EUCLIDEAN: 671
Number of matching relationships--COSINE: 651
```



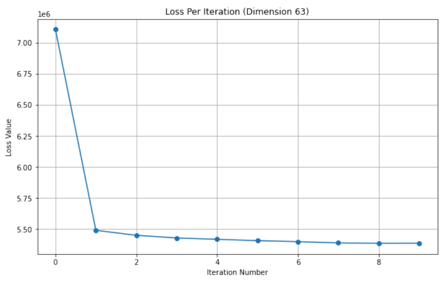
```
Embedding dimension: 60
PEARSON Mean similarity: 0.9261316017703081
EUCLIDEAN Mean similarity: 0.36521501076274293
COSINE Mean similarity: 0.9252629594225299
Number of matching relationships--PEARSON: 646
Number of matching relationships--EUCLIDEAN: 654
Number of matching relationships--COSINE: 647
```



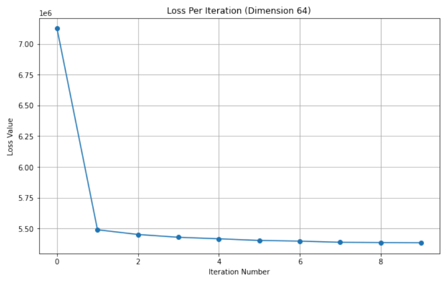
```
Embedding dimension: 61
PEARSON Mean similarity: 0.9247604265875091
EUCLIDEAN Mean similarity: 0.36459970284071774
COSINE Mean similarity: 0.925257393458041
Number of matching relationships--PEARSON: 646
Number of matching relationships--EUCLIDEAN: 658
Number of matching relationships--COSINE: 642
```



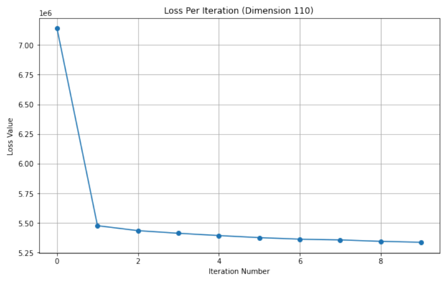
```
Embedding dimension: 62
PEARSON Mean similarity: 0.925017176561905
EUCLIDEAN Mean similarity: 0.36324660739307035
COSINE Mean similarity: 0.9243671569373442
Number of matching relationships--PEARSON: 642
Number of matching relationships--EUCLIDEAN: 660
Number of matching relationships--COSINE: 644
```



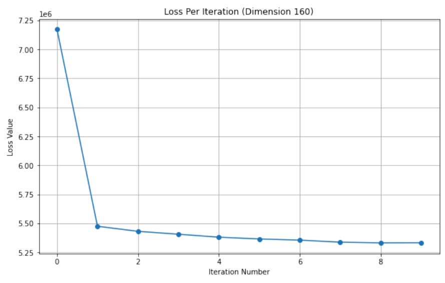
```
Embedding dimension: 63
PEARSON Mean similarity: 0.9243072399888821
EUCLIDEAN Mean similarity: 0.36304081870322274
COSINE Mean similarity: 0.9239985436421201
Number of matching relationships--PEARSON: 642
Number of matching relationships--EUCLIDEAN: 663
Number of matching relationships--COSINE: 646
```



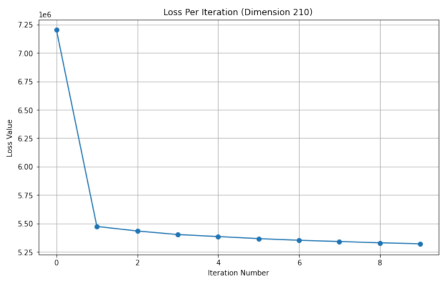
```
Embedding dimension: 64
PEARSON Mean similarity: 0.923887296758339
EUCLIDEAN Mean similarity: 0.36214588290655453
COSINE Mean similarity: 0.9237687968785161
Number of matching relationships--PEARSON: 650
Number of matching relationships--EUCLIDEAN: 662
Number of matching relationships--COSINE: 651
```



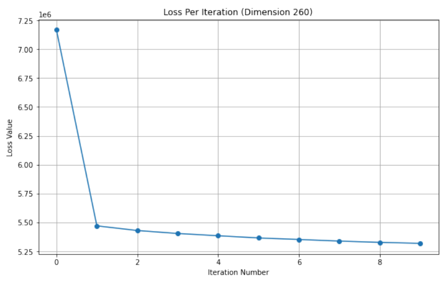
```
Embedding dimension: 110
PEARSON Mean similarity: 0.9101315333536844
EUCLIDEAN Mean similarity: 0.34242454385123544
COSINE Mean similarity: 0.9099470132915096
Number of matching relationships--PEARSON: 644
Number of matching relationships--EUCLIDEAN: 653
Number of matching relationships--COSINE: 645
```



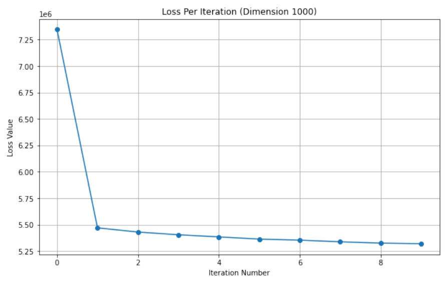
```
Embedding dimension: 160
PEARSON Mean similarity: 0.9042997819458188
EUCLIDEAN Mean similarity: 0.3363883922864274
COSINE Mean similarity: 0.9046242236386726
Number of matching relationships--PEARSON: 635
Number of matching relationships--EUCLIDEAN: 653
Number of matching relationships--COSINE: 636
```



```
Embedding dimension: 210
PEARSON Mean similarity: 0.9028081572742547
EUCLIDEAN Mean similarity: 0.33498783400351045
COSINE Mean similarity: 0.9029675462573459
Number of matching relationships--PEARSON: 633
Number of matching relationships--EUCLIDEAN: 655
Number of matching relationships--COSINE: 634
```

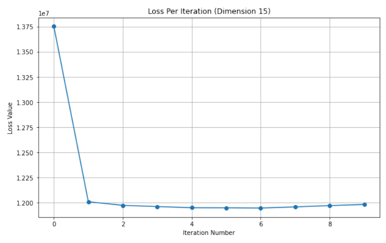


```
Embedding dimension: 260
PEARSON Mean similarity: 0.9019825018315054
EUCLIDEAN Mean similarity: 0.3343919844887986
COSINE Mean similarity: 0.9018959177825793
Number of matching relationships--PEARSON: 637
Number of matching relationships--EUCLIDEAN: 649
Number of matching relationships--COSINE: 639
```

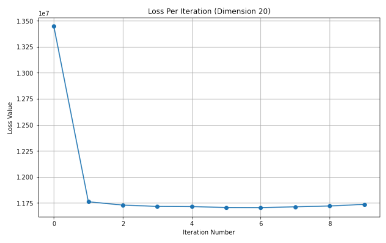


```
Embedding dimension: 1000
PEARSON Mean similarity: 0.9008322708715892
EUCLIDEAN Mean similarity: 0.33601611565627937
COSINE Mean similarity: 0.9007203940274621
Number of matching relationships--PEARSON: 628
Number of matching relationships--EUCLIDEAN: 649
Number of matching relationships--COSINE: 629
```

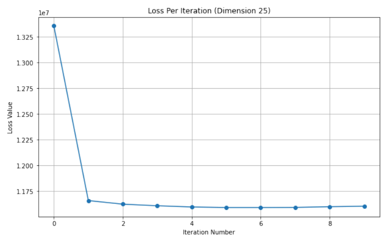
C.4 Results of Tuning Experiments on Walklength=200



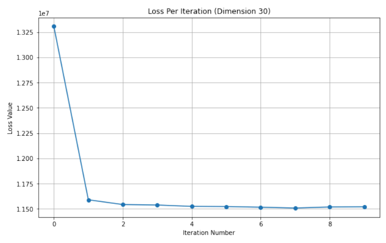
```
Embedding dimension: 15
PEARSON Mean similarity: 0.9688852507395441
EUCLIDEAN Mean similarity: 0.46437369188846833
COSINE Mean similarity: 0.9677861223544821
Number of matching relationships--PEARSON: 641
Number of matching relationships--EUCLIDEAN: 657
Number of matching relationships--COSINE: 644
```



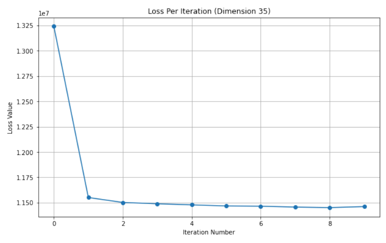
```
Embedding dimension: 20
PEARSON Mean similarity: 0.9610842649834547
EUCLIDEAN Mean similarity: 0.4458960576811076
COSINE Mean similarity: 0.9603070128087279
Number of matching relationships--PEARSON: 647
Number of matching relationships--EUCLIDEAN: 668
Number of matching relationships--COSINE: 647
```



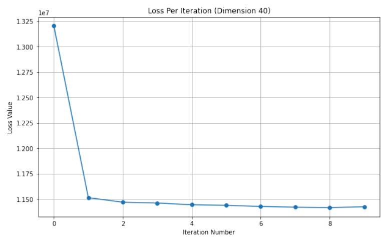
```
Embedding dimension: 25
PEARSON Mean similarity: 0.9551431684057329
EUCLIDEAN Mean similarity: 0.4274481588135442
COSINE Mean similarity: 0.9541716531953572
Number of matching relationships--PEARSON: 639
Number of matching relationships--EUCLIDEAN: 669
Number of matching relationships--COSINE: 638
```



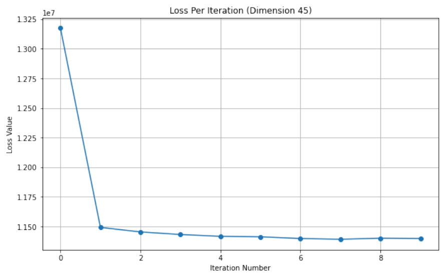
```
Embedding dimension: 30
PEARSON Mean similarity: 0.9490910770974096
EUCLIDEAN Mean similarity: 0.41202396184338075
COSINE Mean similarity: 0.9488883022719683
Number of matching relationships--PEARSON: 645
Number of matching relationships--EUCLIDEAN: 669
Number of matching relationships--COSINE: 641
```



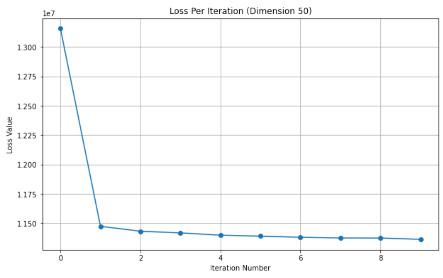
```
Embedding dimension: 35
PEARSON Mean similarity: 0.9456497305076971
EUCLIDEAN Mean similarity: 0.4004575877985539
COSINE Mean similarity: 0.9446514817073038
Number of matching relationships--PEARSON: 641
Number of matching relationships--EUCLIDEAN: 667
Number of matching relationships--COSINE: 645
```



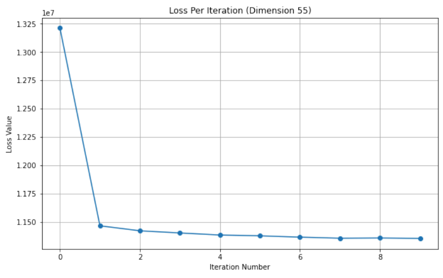
```
Embedding dimension: 40
PEARSON Mean similarity: 0.9413151852313314
EUCLIDEAN Mean similarity: 0.3904698829707226
COSINE Mean similarity: 0.9395868202784213
Number of matching relationships--PEARSON: 644
Number of matching relationships--EUCLIDEAN: 660
Number of matching relationships--COSINE: 649
```



```
Embedding dimension: 45
PEARSON Mean similarity: 0.9376949028567338
EUCLIDEAN Mean similarity: 0.3820394727997097
COSINE Mean similarity: 0.9369668149173348
Number of matching relationships--PEARSON: 649
Number of matching relationships--EUCLIDEAN: 667
Number of matching relationships--COSINE: 653
```



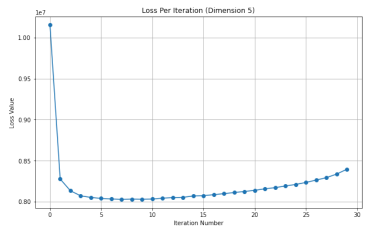
```
Embedding dimension: 50
PEARSON Mean similarity: 0.9339465075088569
EUCLIDEAN Mean similarity: 0.3757035097660307
COSINE Mean similarity: 0.9338815004469901
Number of matching relationships--PEARSON: 636
Number of matching relationships--EUCLIDEAN: 657
Number of matching relationships--COSINE: 641
```



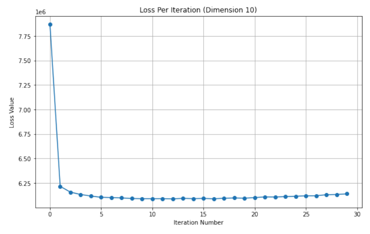
```
Embedding dimension: 55
PEARSON Mean similarity: 0.9311196835741863
EUCLIDEAN Mean similarity: 0.3692229492097345
COSINE Mean similarity: 0.9299755547212181
Number of matching relationships--PEARSON: 640
Number of matching relationships--EUCLIDEAN: 665
Number of matching relationships--COSINE: 642
```

D Tuning Experiments on Iteration and Measure Methods of Node2Vec with KNN on Cora Dataset

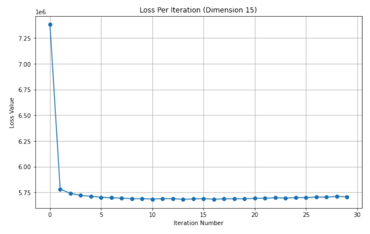
D.1 Results of Tuning Iteration Experiments on Walklength=100 and Iteration =30



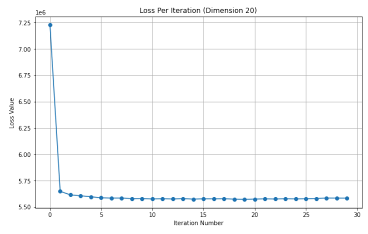
```
Embedding dimension: 5
PEARSON Mean similarity: 0.9969314783679506
EUCLIDEAN Mean similarity: 0.5725288901758687
COSINE Mean similarity: 0.9951096431895584
Number of matching relationships--PEARSON: 408
Number of matching relationships--EUCLIDEAN: 387
Number of matching relationships--COSINE: 471
```



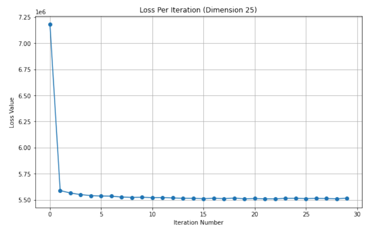
```
Embedding dimension: 10
PEARSON Mean similarity: 0.9794482195923212
EUCLIDEAN Mean similarity: 0.4761815027436971
COSINE Mean similarity: 0.9770408771238792
Number of matching relationships--PEARSON: 613
Number of matching relationships--EUCLIDEAN: 632
Number of matching relationships--COSINE: 621
```



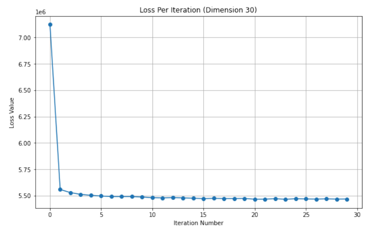
```
Embedding dimension: 15
PEARSON Mean similarity: 0.9678268945375609
EUCLIDEAN Mean similarity: 0.4507060318612982
COSINE Mean similarity: 0.9662910974184202
Number of matching relationships--PEARSON: 635
Number of matching relationships--EUCLIDEAN: 655
Number of matching relationships--COSINE: 640
```



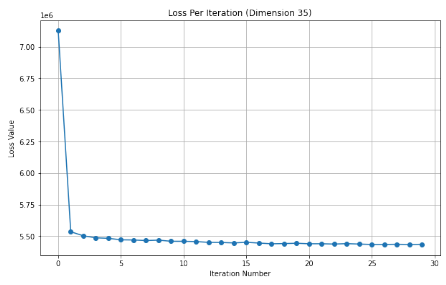
```
Embedding dimension: 20
PEARSON Mean similarity: 0.9591481370799122
EUCLIDEAN Mean similarity: 0.42605654513642216
COSINE Mean similarity: 0.957709426203718
Number of matching relationships--PEARSON: 649
Number of matching relationships--EUCLIDEAN: 668
Number of matching relationships--COSINE: 649
```



```
Embedding dimension: 25
PEARSON Mean similarity: 0.951249326032585
EUCLIDEAN Mean similarity: 0.4037039504621723
COSINE Mean similarity: 0.950229294296557
Number of matching relationships--PEARSON: 655
Number of matching relationships--EUCLIDEAN: 673
Number of matching relationships--COSINE: 653
```

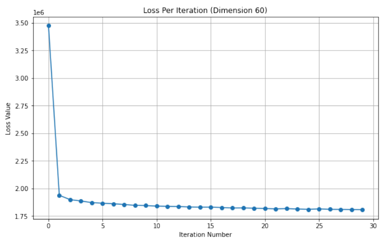


```
Embedding dimension: 30
PEARSON Mean similarity: 0.9456971369958874
EUCLIDEAN Mean similarity: 0.3862984567836609
COSINE Mean similarity: 0.9447575559292093
Number of matching relationships--PEARSON: 657
Number of matching relationships--EUCLIDEAN: 669
Number of matching relationships--COSINE: 653
```

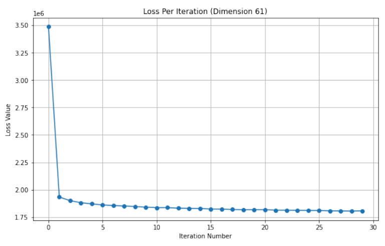


```
Embedding dimension: 35
PEARSON Mean similarity: 0.938786251400563
EUCLIDEAN Mean similarity: 0.37170303829311446
COSINE Mean similarity: 0.9382746838748544
Number of matching relationships--PEARSON: 657
Number of matching relationships--EUCLIDEAN: 672
Number of matching relationships--COSINE: 656
```

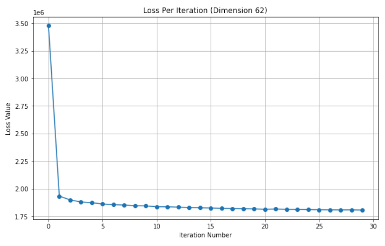
D.2 Results of Tuning Iteration Experiments on Walklength=40 and Iteration =30



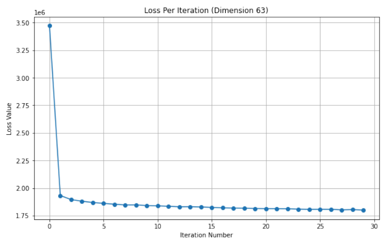
Embedding dimension: 60
PEARSON Mean similarity: 0.9069633945932769
EUCLIDEAN Mean similarity: 0.321096316186857
COSINE Mean similarity: 0.9062966010066924
Number of matching relationships--PEARSON: 629
Number of matching relationships--EUCLIDEAN: 636
Number of matching relationships--COSINE: 631



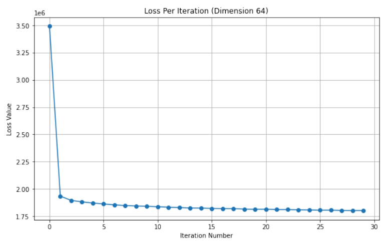
Embedding dimension: 61
PEARSON Mean similarity: 0.9057157009355778
EUCLIDEAN Mean similarity: 0.31958554569340175
COSINE Mean similarity: 0.9057143024010708
Number of matching relationships--PEARSON: 633
Number of matching relationships--EUCLIDEAN: 635
Number of matching relationships--COSINE: 638



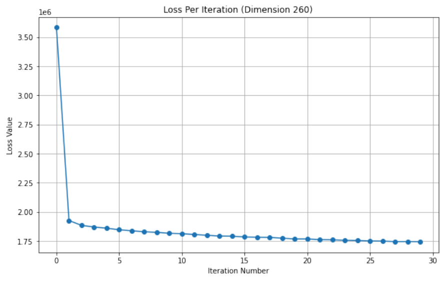
Embedding dimension: 62
PEARSON Mean similarity: 0.9052272796630859
EUCLIDEAN Mean similarity: 0.31880524260606863
COSINE Mean similarity: 0.9043874943450071
Number of matching relationships--PEARSON: 640
Number of matching relationships--EUCLIDEAN: 643
Number of matching relationships--COSINE: 641



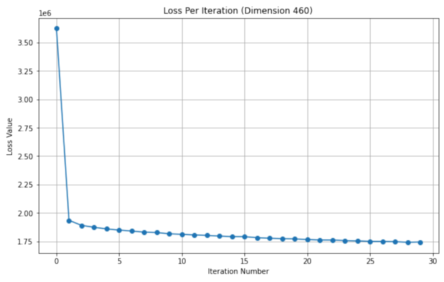
Embedding dimension: 63
PEARSON Mean similarity: 0.9046935189950062
EUCLIDEAN Mean similarity: 0.31799183173581097
COSINE Mean similarity: 0.9037476518481662
Number of matching relationships--PEARSON: 638
Number of matching relationships--EUCLIDEAN: 646
Number of matching relationships--COSINE: 635



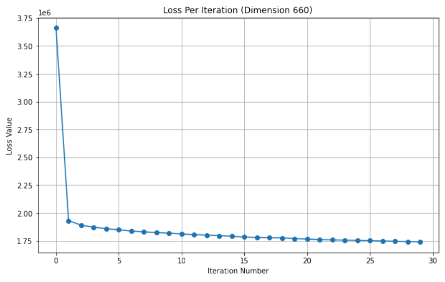
Embedding dimension: 64
PEARSON Mean similarity: 0.903533499702217
EUCLIDEAN Mean similarity: 0.31717951075893386
COSINE Mean similarity: 0.9034036647621955
Number of matching relationships--PEARSON: 637
Number of matching relationships--EUCLIDEAN: 649
Number of matching relationships--COSINE: 64



```
Embedding dimension: 260
PEARSON Mean similarity: 0.869560435251436
EUCLIDEAN Mean similarity: 0.286520964637993
COSINE Mean similarity: 0.8697774751851971
Number of matching relationships--PEARSON: 613
Number of matching relationships--EUCLIDEAN: 610
Number of matching relationships--COSINE: 617
```



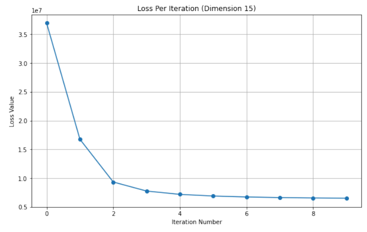
```
Embedding dimension: 460
PEARSON Mean similarity: 0.8683519664860194
EUCLIDEAN Mean similarity: 0.28661664803588055
COSINE Mean similarity: 0.8683760599336384
Number of matching relationships--PEARSON: 605
Number of matching relationships--EUCLIDEAN: 603
Number of matching relationships--COSINE: 609
```



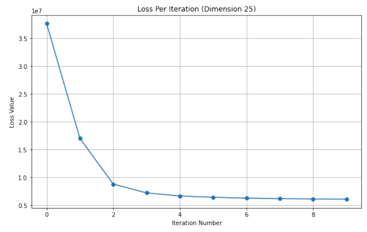
```
Embedding dimension: 660
PEARSON Mean similarity: 0.8682235596627217
EUCLIDEAN Mean similarity: 0.28695051877854316
COSINE Mean similarity: 0.8682482019308927
Number of matching relationships--PEARSON: 602
Number of matching relationships--EUCLIDEAN: 602
Number of matching relationships--COSINE: 602
```

E Tuning Experiments on Initial Learning Rate and Measure Methods of Node2Vec with KNN on Cora Dataset

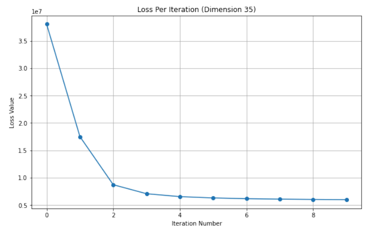
E.1 Results of Tuning Initial Learning Rate Experiments on Initial Learning Rate=0.001, iteration = 10



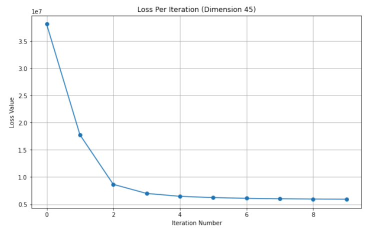
Embedding dimension: 15
PEARSON Mean similarity: 0.9701001041925112
EUCLIDEAN Mean similarity: 0.5279294964129675
COSINE Mean similarity: 0.968641717817794
Number of matching relationships--PEARSON: 630
Number of matching relationships--EUCLIDEAN: 618
Number of matching relationships--COSINE: 625



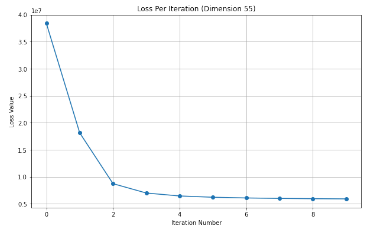
Embedding dimension: 25
PEARSON Mean similarity: 0.9622260250052109
EUCLIDEAN Mean similarity: 0.521219217618776
COSINE Mean similarity: 0.9619463575467402
Number of matching relationships--PEARSON: 631
Number of matching relationships--EUCLIDEAN: 625
Number of matching relationships--COSINE: 629



Embedding dimension: 35
PEARSON Mean similarity: 0.9586034709981133
EUCLIDEAN Mean similarity: 0.5205573848290492
COSINE Mean similarity: 0.9585333941077905
Number of matching relationships--PEARSON: 637
Number of matching relationships--EUCLIDEAN: 623
Number of matching relationships--COSINE: 629

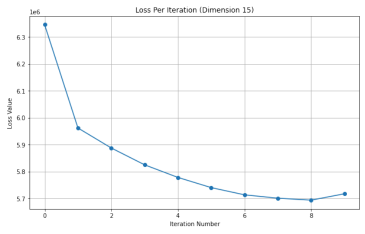


Embedding dimension: 45
PEARSON Mean similarity: 0.9571643260240202
EUCLIDEAN Mean similarity: 0.5209165594250272
COSINE Mean similarity: 0.9570376935702483
Number of matching relationships--PEARSON: 630
Number of matching relationships--EUCLIDEAN: 621
Number of matching relationships--COSINE: 631

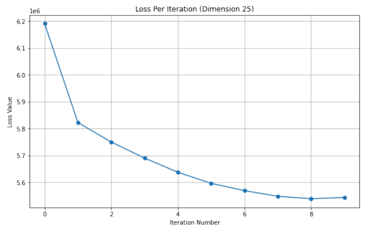


Embedding dimension: 55
PEARSON Mean similarity: 0.9557015045706948
EUCLIDEAN Mean similarity: 0.5221197300858829
COSINE Mean similarity: 0.9554677870284155
Number of matching relationships--PEARSON: 626
Number of matching relationships--EUCLIDEAN: 618
Number of matching relationships--COSINE: 626

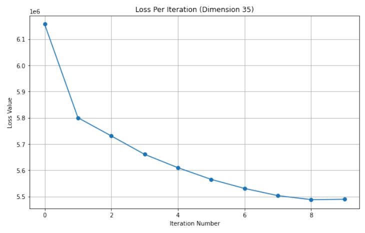
E.2 Results of Tuning Initial Learning Rate Experiments on Initial Learning Rate=0.1, iteration = 10



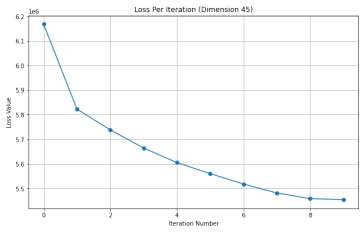
Embedding dimension: 15
PEARSON Mean similarity: 0.9673796903083385
EUCLIDEAN Mean similarity: 0.45660355813041925
COSINE Mean similarity: 0.9663991451967766
Number of matching relationships--PEARSON: 635
Number of matching relationships--EUCLIDEAN: 666
Number of matching relationships--COSINE: 641



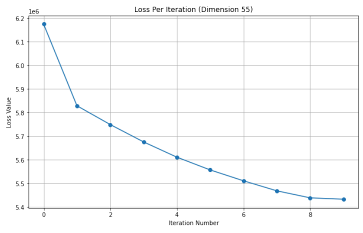
Embedding dimension: 25
PEARSON Mean similarity: 0.9514831900772703
EUCLIDEAN Mean similarity: 0.41733582572711625
COSINE Mean similarity: 0.9514463500751175
Number of matching relationships--PEARSON: 656
Number of matching relationships--EUCLIDEAN: 670
Number of matching relationships--COSINE: 658



Embedding dimension: 35
PEARSON Mean similarity: 0.9399562618757105
EUCLIDEAN Mean similarity: 0.3873517723168017
COSINE Mean similarity: 0.9395032468539397
Number of matching relationships--PEARSON: 665
Number of matching relationships--EUCLIDEAN: 671
Number of matching relationships--COSINE: 662

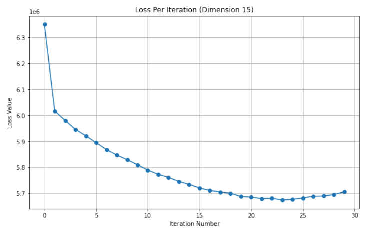


Embedding dimension: 45
PEARSON Mean similarity: 0.930304851560156
EUCLIDEAN Mean similarity: 0.3662030726451113
COSINE Mean similarity: 0.9300727925941547
Number of matching relationships--PEARSON: 663
Number of matching relationships--EUCLIDEAN: 674
Number of matching relationships--COSINE: 663

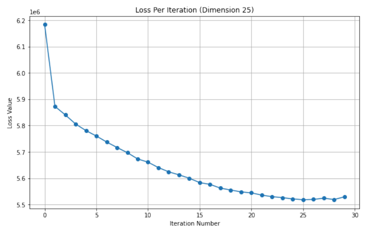


Embedding dimension: 55
PEARSON Mean similarity: 0.9225073877782554
EUCLIDEAN Mean similarity: 0.3504881573568595
COSINE Mean similarity: 0.9216686037477749
Number of matching relationships--PEARSON: 660
Number of matching relationships--EUCLIDEAN: 669
Number of matching relationships--COSINE: 667

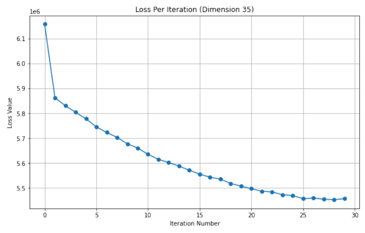
E.3 Results of Tuning Initial Learning Rate Experiments on Initial Learning Rate=0.1, iteration = 30



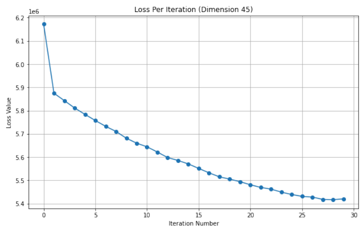
Embedding dimension: 15
PEARSON Mean similarity: 0.9680801977611186
EUCLIDEAN Mean similarity: 0.4559773393712685
COSINE Mean similarity: 0.9670393697268149
Number of matching relationships--PEARSON: 644
Number of matching relationships--EUCLIDEAN: 659
Number of matching relationships--COSINE: 651



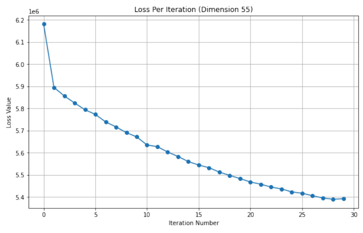
Embedding dimension: 25
PEARSON Mean similarity: 0.9501486810658848
EUCLIDEAN Mean similarity: 0.41090744010102626
COSINE Mean similarity: 0.9513527981463704
Number of matching relationships--PEARSON: 654
Number of matching relationships--EUCLIDEAN: 666
Number of matching relationships--COSINE: 661



Embedding dimension: 35
PEARSON Mean similarity: 0.9393122646269806
EUCLIDEAN Mean similarity: 0.3770785001391322
COSINE Mean similarity: 0.9388289314957454
Number of matching relationships--PEARSON: 659
Number of matching relationships--EUCLIDEAN: 682
Number of matching relationships--COSINE: 661

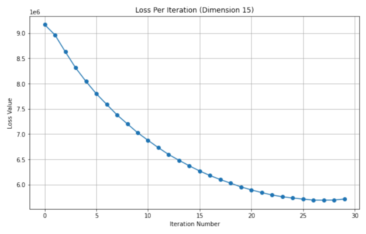


Embedding dimension: 45
PEARSON Mean similarity: 0.9281456540219013
EUCLIDEAN Mean similarity: 0.35430662811594843
COSINE Mean similarity: 0.9275366066125507
Number of matching relationships--PEARSON: 652
Number of matching relationships--EUCLIDEAN: 672
Number of matching relationships--COSINE: 657

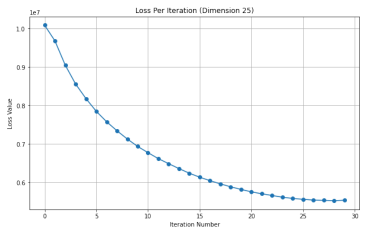


Embedding dimension: 55
PEARSON Mean similarity: 0.91994189201817O2
EUCLIDEAN Mean similarity: 0.3383824901355422
COSINE Mean similarity: 0.9190694651188139
Number of matching relationships--PEARSON: 668
Number of matching relationships--EUCLIDEAN: 674
Number of matching relationships--COSINE: 663

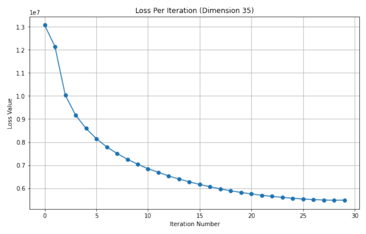
E.4 Results of Tuning Initial Learning Rate Experiments on Initial Learning Rate=0.25, iteration = 30



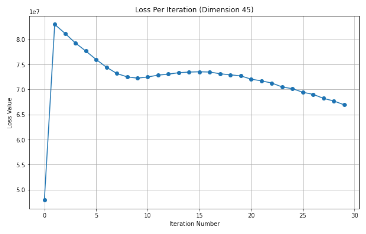
Embedding dimension: 15
PEARSON Mean similarity: 0.9678859671250404
EUCLIDEAN Mean similarity: 0.458852782383275
COSINE Mean similarity: 0.9676648383654731
Number of matching relationships--PEARSON: 638
Number of matching relationships--EUCLIDEAN: 660
Number of matching relationships--COSINE: 641



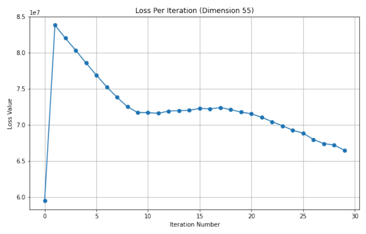
Embedding dimension: 25
PEARSON Mean similarity: 0.9538220778878014
EUCLIDEAN Mean similarity: 0.4178473634593068
COSINE Mean similarity: 0.9525108700841005
Number of matching relationships--PEARSON: 653
Number of matching relationships--EUCLIDEAN: 670
Number of matching relationships--COSINE: 654



Embedding dimension: 35
PEARSON Mean similarity: 0.9408940158883438
EUCLIDEAN Mean similarity: 0.3873504613315689
COSINE Mean similarity: 0.9399894345282099
Number of matching relationships--PEARSON: 661
Number of matching relationships--EUCLIDEAN: 675
Number of matching relationships--COSINE: 660

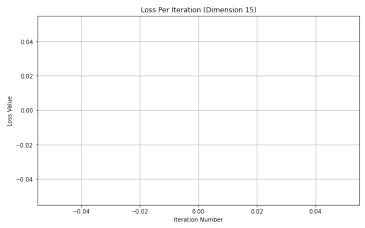


Embedding dimension: 45
PEARSON Mean similarity: 0.8784332368363306
EUCLIDEAN Mean similarity: 0.04647278426639877
COSINE Mean similarity: 0.8771420074530507
Number of matching relationships--PEARSON: 271
Number of matching relationships--EUCLIDEAN: 168
Number of matching relationships--COSINE: 265

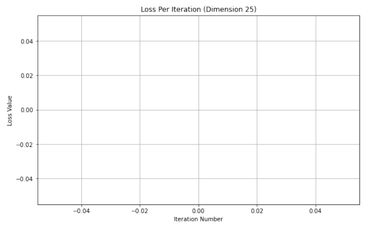


Embedding dimension: 55
PEARSON Mean similarity: 0.8655686244654691
EUCLIDEAN Mean similarity: 0.04662014587555382
COSINE Mean similarity: 0.614217355860565
Number of matching relationships--PEARSON: 243
Number of matching relationships--EUCLIDEAN: 151
Number of matching relationships--COSINE: 0

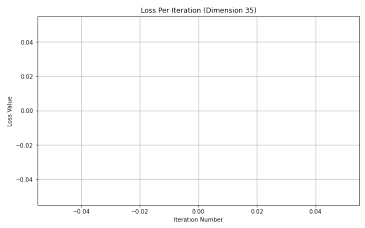
E.5 Results of Tuning Initial Learning Rate Experiments on Initial Learning Rate=0.5, iteration = 30



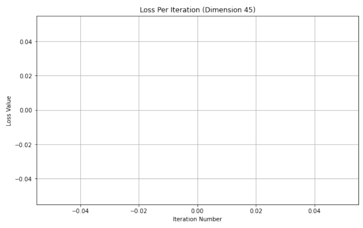
Embedding dimension: 15
PEARSON Mean similarity: 0.03807502067599726
EUCLIDEAN Mean similarity: 0.04564335060964835
COSINE Mean similarity: 0.037967638779249996
Number of matching relationships--PEARSON: 347
Number of matching relationships--EUCLIDEAN: 347
Number of matching relationships--COSINE: 347



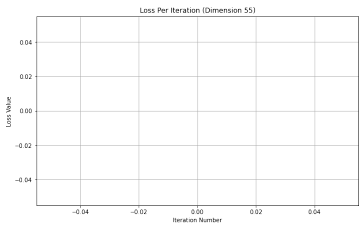
Embedding dimension: 25
PEARSON Mean similarity: 0.034831752241802356
EUCLIDEAN Mean similarity: 0.0459378528454103
COSINE Mean similarity: 0.03449330858148888
Number of matching relationships--PEARSON: 347
Number of matching relationships--EUCLIDEAN: 347
Number of matching relationships--COSINE: 347



Embedding dimension: 35
PEARSON Mean similarity: 0.033330741837246554
EUCLIDEAN Mean similarity: 0.046018737669005175
COSINE Mean similarity: 0.03316360118646213
Number of matching relationships--PEARSON: 347
Number of matching relationships--EUCLIDEAN: 347
Number of matching relationships--COSINE: 347

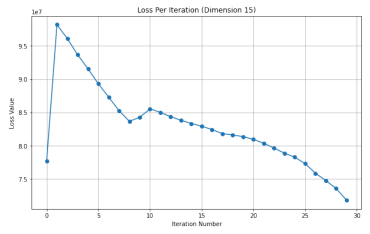


Embedding dimension: 45
PEARSON Mean similarity: 0.03209057220504062
EUCLIDEAN Mean similarity: 0.046372619865563985
COSINE Mean similarity: 0.031897199453244006
Number of matching relationships--PEARSON: 347
Number of matching relationships--EUCLIDEAN: 347
Number of matching relationships--COSINE: 347

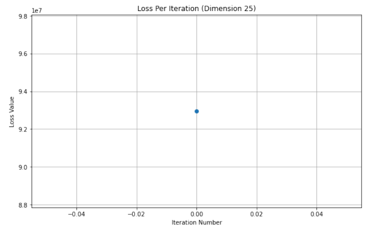


Embedding dimension: 55
PEARSON Mean similarity: 0.031240796831545133
EUCLIDEAN Mean similarity: 0.04660925590481328
COSINE Mean similarity: 0.03108460850342338
Number of matching relationships--PEARSON: 347
Number of matching relationships--EUCLIDEAN: 347
Number of matching relationships--COSINE: 347

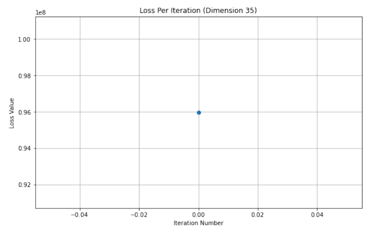
E.6 Results of Tuning Initial Learning Rate Experiments on Initial Learning Rate=0.34, iteration = 30



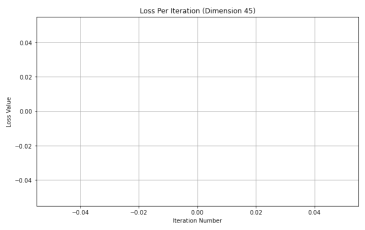
Embedding dimension: 15
PEARSON Mean similarity: 0.9497190165554932
EUCLIDEAN Mean similarity: 0.04564676083753451
COSINE Mean similarity: 0.9191424174005911
Number of matching relationships--PEARSON: 469
Number of matching relationships--EUCLIDEAN: 221
Number of matching relationships--COSINE: 271



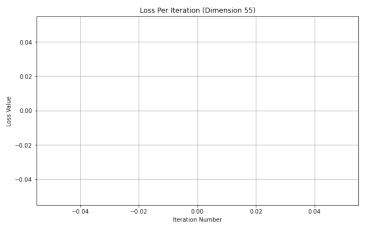
Embedding dimension: 25
PEARSON Mean similarity: 0.034831752241802356
EUCLIDEAN Mean similarity: 0.0459378528454103
COSINE Mean similarity: 0.03449330858148888
Number of matching relationships--PEARSON: 347
Number of matching relationships--EUCLIDEAN: 347
Number of matching relationships--COSINE: 347



Embedding dimension: 35
PEARSON Mean similarity: 0.033330741837246554
EUCLIDEAN Mean similarity: 0.046018737669005175
COSINE Mean similarity: 0.03316360118646213
Number of matching relationships--PEARSON: 347
Number of matching relationships--EUCLIDEAN: 347
Number of matching relationships--COSINE: 347

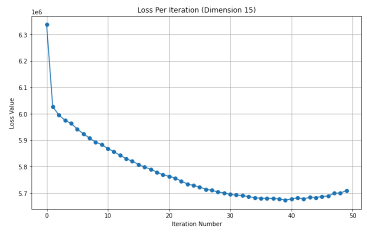


Embedding dimension: 45
PEARSON Mean similarity: 0.03209057220504062
EUCLIDEAN Mean similarity: 0.046372619865563985
COSINE Mean similarity: 0.031897199453244006
Number of matching relationships--PEARSON: 347
Number of matching relationships--EUCLIDEAN: 347
Number of matching relationships--COSINE: 347

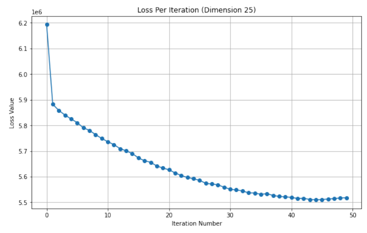


Embedding dimension: 55
PEARSON Mean similarity: 0.031240796831545133
EUCLIDEAN Mean similarity: 0.04660925590481328
COSINE Mean similarity: 0.03108460850342338
Number of matching relationships--PEARSON: 347
Number of matching relationships--EUCLIDEAN: 347
Number of matching relationships--COSINE: 347

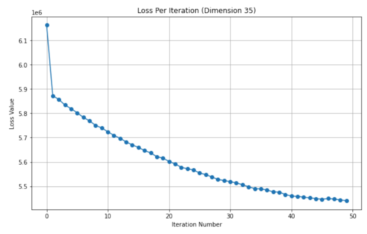
E.7 Results of Tuning Initial Learning Rate Experiments on Initial Learning Rate=0.1, iteration = 50



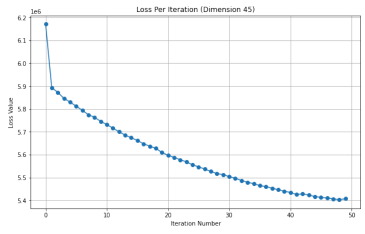
Embedding dimension: 15
PEARSON Mean similarity: 0.9690167242526304
EUCLIDEAN Mean similarity: 0.4533732370576619
COSINE Mean similarity: 0.9677710883902658
Number of matching relationships--PEARSON: 649
Number of matching relationships--EUCLIDEAN: 659
Number of matching relationships--COSINE: 645



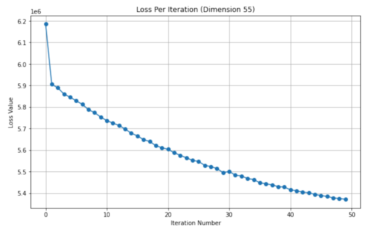
Embedding dimension: 25
PEARSON Mean similarity: 0.9507276661815108
EUCLIDEAN Mean similarity: 0.40572504518303243
COSINE Mean similarity: 0.9512493584321556
Number of matching relationships--PEARSON: 666
Number of matching relationships--EUCLIDEAN: 676
Number of matching relationships--COSINE: 667



Embedding dimension: 35
PEARSON Mean similarity: 0.9391560454488329
EUCLIDEAN Mean similarity: 0.37193044918854795
COSINE Mean similarity: 0.937670834483565
Number of matching relationships--PEARSON: 655
Number of matching relationships--EUCLIDEAN: 668
Number of matching relationships--COSINE: 655



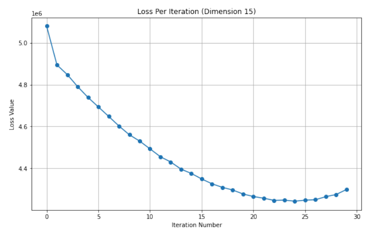
Embedding dimension: 45
PEARSON Mean similarity: 0.9275491212988534
EUCLIDEAN Mean similarity: 0.3487669774313269
COSINE Mean similarity: 0.9270987416299794
Number of matching relationships--PEARSON: 660
Number of matching relationships--EUCLIDEAN: 674
Number of matching relationships--COSINE: 659



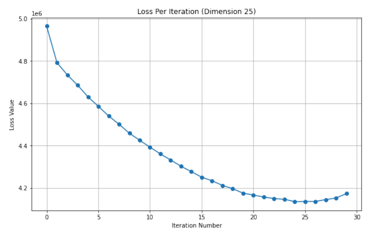
Embedding dimension: 55
PEARSON Mean similarity: 0.9182105941997849
EUCLIDEAN Mean similarity: 0.331989851660003
COSINE Mean similarity: 0.9179211586933896
Number of matching relationships--PEARSON: 660
Number of matching relationships--EUCLIDEAN: 668
Number of matching relationships--COSINE: 662

F Tuning Experiments on InOutFactor and Measure Methods of Node2Vec with KNN on Cora Dataset

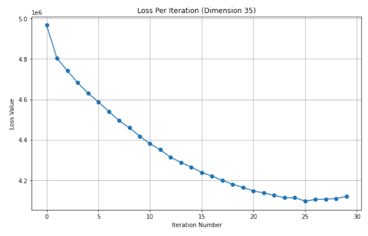
F.1 Results of Tuning InOutFactor Experiments on InOutFactor=3, iteration = 30



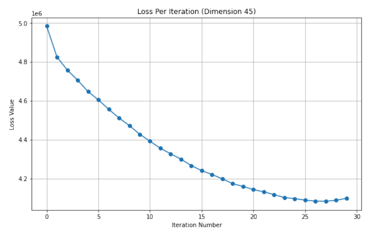
```
Embedding dimension: 15
PEARSON Mean similarity: 0.9641895184312649
EUCLIDEAN Mean similarity: 0.4247134794688823
COSINE Mean similarity: 0.9632573537671584
Number of matching relationships--PEARSON: 652
Number of matching relationships--EUCLIDEAN: 656
Number of matching relationships--COSINE: 659
```



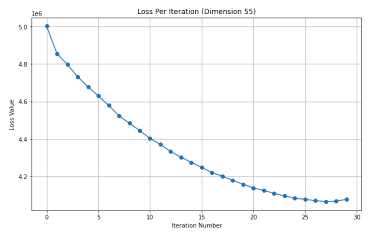
```
Embedding dimension: 25
PEARSON Mean similarity: 0.945940177444165
EUCLIDEAN Mean similarity: 0.3851913529625671
COSINE Mean similarity: 0.9456144608986431
Number of matching relationships--PEARSON: 660
Number of matching relationships--EUCLIDEAN: 666
Number of matching relationships--COSINE: 663
```



```
Embedding dimension: 35
PEARSON Mean similarity: 0.9328823760129853
EUCLIDEAN Mean similarity: 0.3579145436448924
COSINE Mean similarity: 0.9321914613687129
Number of matching relationships--PEARSON: 660
Number of matching relationships--EUCLIDEAN: 675
Number of matching relationships--COSINE: 663
```

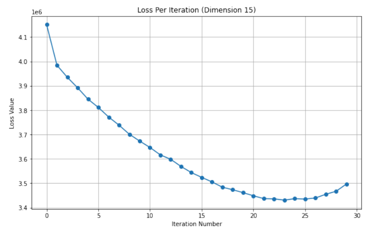


```
Embedding dimension: 45
PEARSON Mean similarity: 0.9220243008837566
EUCLIDEAN Mean similarity: 0.3387218370395362
COSINE Mean similarity: 0.9213917254697273
Number of matching relationships--PEARSON: 657
Number of matching relationships--EUCLIDEAN: 655
Number of matching relationships--COSINE: 657
```

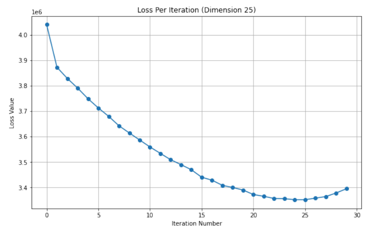


```
Embedding dimension: 55
PEARSON Mean similarity: 0.913125826160876
EUCLIDEAN Mean similarity: 0.325129794863l6135
COSINE Mean similarity: 0.912363117730776
Number of matching relationships--PEARSON: 670
Number of matching relationships--EUCLIDEAN: 671
Number of matching relationships--COSINE: 672
```

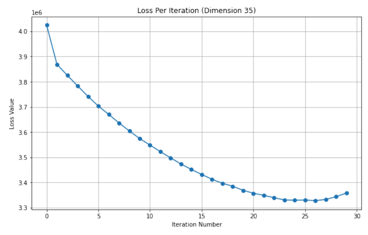
F.2 Results of Tuning InOutFactor Experiments on InOutFactor=6, iteration = 30



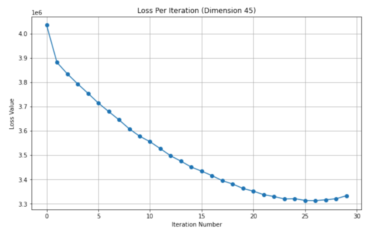
Embedding dimension: 15
PEARSON Mean similarity: 0.9610198151942018
EUCLIDEAN Mean similarity: 0.40113060562438063
COSINE Mean similarity: 0.9594406330779173
Number of matching relationships--PEARSON: 645
Number of matching relationships--EUCLIDEAN: 642
Number of matching relationships--COSINE: 654



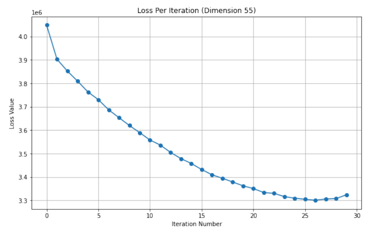
Embedding dimension: 25
PEARSON Mean similarity: 0.9418697244483576
EUCLIDEAN Mean similarity: 0.3651013708185058
COSINE Mean similarity: 0.9403807241420098
Number of matching relationships--PEARSON: 659
Number of matching relationships--EUCLIDEAN: 654
Number of matching relationships--COSINE: 659



Embedding dimension: 35
PEARSON Mean similarity: 0.927550697890131
EUCLIDEAN Mean similarity: 0.3405140116788788
COSINE Mean similarity: 0.9268961783878208
Number of matching relationships--PEARSON: 662
Number of matching relationships--EUCLIDEAN: 654
Number of matching relationships--COSINE: 655

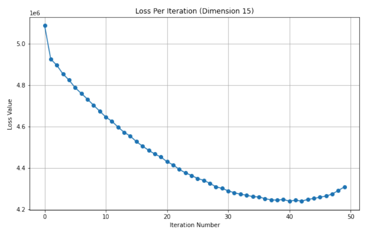


Embedding dimension: 45
PEARSON Mean similarity: 0.916404811740802
EUCLIDEAN Mean similarity: 0.3240914985735624
COSINE Mean similarity: 0.916127820205125
Number of matching relationships--PEARSON: 651
Number of matching relationships--EUCLIDEAN: 654
Number of matching relationships--COSINE: 655

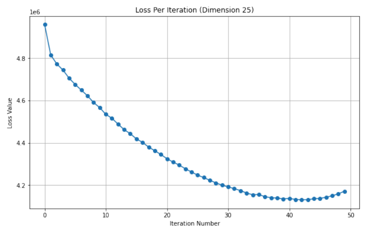


Embedding dimension: 55
PEARSON Mean similarity: 0.9074123058572654
EUCLIDEAN Mean similarity: 0.31307687498794173
COSINE Mean similarity: 0.9074157275336532
Number of matching relationships--PEARSON: 660
Number of matching relationships--EUCLIDEAN: 653
Number of matching relationships--COSINE: 665

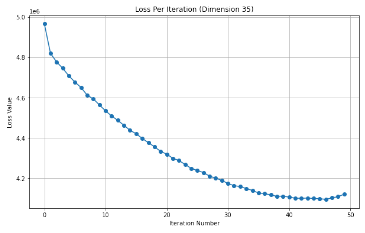
F.3 Results of Tuning InOutFactor Experiments on InOutFactor=3, iteration = 50



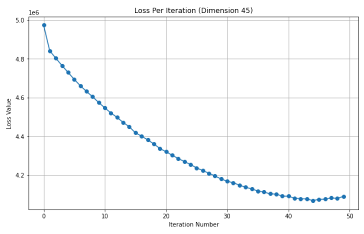
Embedding dimension: 15
PEARSON Mean similarity: 0.9651385212930654
EUCLIDEAN Mean similarity: 0.42168520578077306
COSINE Mean similarity: 0.9636697693096482
Number of matching relationships--PEARSON: 653
Number of matching relationships--EUCLIDEAN: 659
Number of matching relationships--COSINE: 657



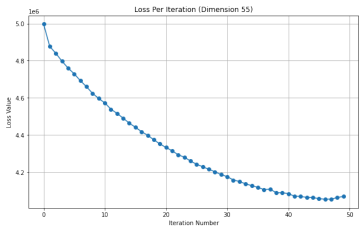
Embedding dimension: 25
PEARSON Mean similarity: 0.9461260425111215
EUCLIDEAN Mean similarity: 0.380653515383271
COSINE Mean similarity: 0.9454999194011379
Number of matching relationships--PEARSON: 663
Number of matching relationships--EUCLIDEAN: 668
Number of matching relationships--COSINE: 661



Embedding dimension: 35
PEARSON Mean similarity: 0.931260121062376
EUCLIDEAN Mean similarity: 0.35177915451973896
COSINE Mean similarity: 0.9310552115278371
Number of matching relationships--PEARSON: 677
Number of matching relationships--EUCLIDEAN: 674
Number of matching relationships--COSINE: 673



Embedding dimension: 45
PEARSON Mean similarity: 0.921586131345222
EUCLIDEAN Mean similarity: 0.33234834234330646
COSINE Mean similarity: 0.9206820337247638
Number of matching relationships--PEARSON: 666
Number of matching relationships--EUCLIDEAN: 667
Number of matching relationships--COSINE: 668



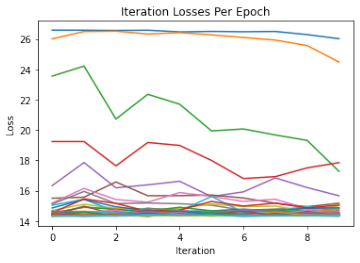
Embedding dimension: 55
PEARSON Mean similarity: 0.9107687537391922
EUCLIDEAN Mean similarity: 0.3188113203429049
COSINE Mean similarity: 0.9107360538386876
Number of matching relationships--PEARSON: 672
Number of matching relationships--EUCLIDEAN: 664
Number of matching relationships--COSINE: 672

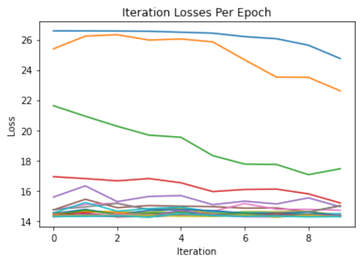
G Tuning Experiments of GraphSAGE with KNN on Cora Dataset

G.1 Tuning Experiments on Aggregator, Activation Function and Search Depth of GraphSAGE with KNN on Cora Dataset

batchSize=30, aggregator="POOL" searchDepth=5 activationFunction="Sigmoid"

```
didConverge                                          False
ranIterationsPerEpoch     [10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 1...
iterationLossesPerEpoch   [[26.578063195757903, 26.57627072419955,
26.56...
ranEpochs                                              20
epochLosses               [24.74753612616579, 22.608372509346854,
17.455...
dtype: object
[24.74753612616579, 22.608372509346854, 17.455371242311582,
15.187571670340565, 14.939775989797974, 14.999852930359754,
14.707054300212826, 15.000603704284766, 14.408599294923352,
14.48353896336844, 14.348834969366774, 14.315318934263127,
14.349448664819956, 14.335332045623607, 14.392992485390007,
14.323826669136787, 14.29078425524421, 14.29517837187679,
14.271789792633404, 14.282777729909995]
Number of matching relationships: 13
```



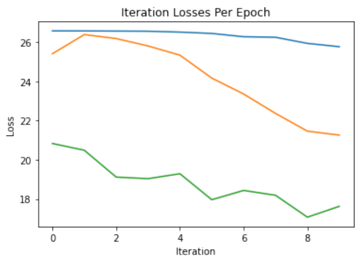


epoch=3, aggregator="POOL" searchDepth=5 activationFunction="Sigmoid"

```
didConverge                                          False
ranIterationsPerEpoch                        [10, 10, 10]
iterationLossesPerEpoch   [[26.578074975930928, 26.575951973216927,
26.5...
ranEpochs                                              3
epochLosses               [25.77068087361759, 21.257657219416203,
17.621...
dtype: object
[25.77068087361759, 21.257657219416203, 17.621216803013937]
Mean similarity: 0.999494967326808
Number of matching relationships: 26
```

epoch=40, aggregator="POOL" searchDepth=5 activationFunction="Sigmoid"

```
didConverge                                          False
ranIterationsPerEpoch     [10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 1...
iterationLossesPerEpoch   [[26.57801715110504, 26.574354178515687,
26.55...
ranEpochs                                              40
epochLosses               [26.01268229185979, 24.47529654926414,
17.2691...
dtype: object
[26.01268229185979, 24.47529654926414, 17.269124168625947,
17.844523490438235, 15.659407985092216, 15.171602216736366,
14.60643849156997, 14.704857908323067, 15.007431556742128,
14.826084757855275, 14.86252330470172, 14.768374206745241,
14.525201523907015, 15.046787627448111, 14.438423879416359,
14.557421334435798, 14.512800249286986, 14.548075958025104,
14.381526653754616, 14.412599660206677, 14.410700478728334,
14.466058581586008, 14.36738081299401, 14.45112876597656,
14.428161344103781, 14.386780784022369, 14.363662862619943,
14.431476148634946, 14.381591163833297, 14.352564364329078,
14.420722070546296, 14.381645087197688, 14.378830948533857,
14.39457526080552, 14.431403924617872, 14.360805345737063,
14.35340643685374, 14.316383161660607, 14.369591644493886,
14.330696444853803]
Mean similarity: 0.9992481583858699
Number of matching relationships: 7
```



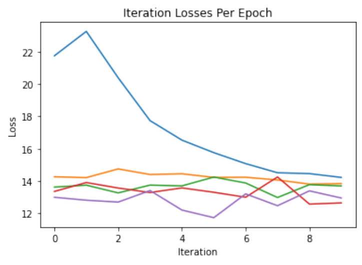
epoch=3, aggregator="MEAN" searchDepth=5 activationFunction="Sigmoid"

```
didConverge                                          False
ranIterationsPerEpoch                        [10, 10, 10]
iterationLossesPerEpoch   [[26.578405443948792, 26.57781356633784,
26.57...
ranEpochs                                              3
epochLosses               [25.72238683578421, 20.671585665462043,
19.614...
dtype: object
[25.72238683578421, 20.671585665462043, 19.6147673905415]
Mean similarity: 0.9996308795810627
Number of matching relationships: 129
```

G.2 Tuning Experiments on Sample Size of GraphSAGE with KNN on Cora Dataset

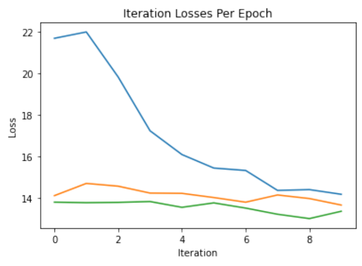
aggregator="MEAN" searchDepth=3
activationFunction="ReLu"sampleSizes=[10,10]

```
didConverge                                      False
ranIterationsPerEpoch                   [10, 10, 10, 10, 10]
iterationLossesPerEpoch   [[21.754560713863505, 23.26248933044796,
20.39...
ranEpochs                                           5
epochLosses            [14.227935121332774, 13.847646488878516,
13.69...
dtype: object
[14.227935121332774, 13.847646488878516, 13.69934965483869,
12.65047489157352, 12.953638816657573]
Mean similarity: 0.9557151991648372
Number of matching relationships: 360
```



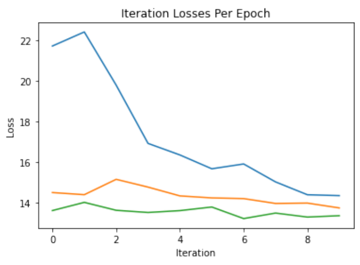
aggregator="MEAN" searchDepth=3
activationFunction="ReLu"sampleSizes=[3,3]

```
didConverge                                      False
ranIterationsPerEpoch                        [10, 10, 10]
iterationLossesPerEpoch   [[21.67280248604041, 21.978633549007345,
19.81...
ranEpochs                                           3
epochLosses            [14.161219714249686, 13.643826400342164,
13.34...
dtype: object
[14.161219714249686, 13.643826400342164, 13.345073581056699]
Mean similarity: 0.94096874845609
Number of matching relationships: 387
```



aggregator="MEAN" searchDepth=3
activationFunction="ReLu"sampleSizes=[15,15]

```
didConverge                                      False
ranIterationsPerEpoch                        [10, 10, 10]
iterationLossesPerEpoch   [[21.702246430992016, 22.393893106745043,
19.7...
ranEpochs                                           3
epochLosses            [14.340485521454346, 13.73461264471733,
13.351...
dtype: object
[14.340485521454346, 13.73461264471733, 13.35131493347294]
```



```
nodeCount                                  2708
nodePropertiesWritten                          2708
preProcessingMillis                             0
computeMillis                              2129
writeMillis                               546
configuration        {'writeProperty': 'graphSAGE11', 'modelName': ...
Name: 0, dtype: object
Mean similarity: 0.9484272521566958
Number of matching relationships: 386
```

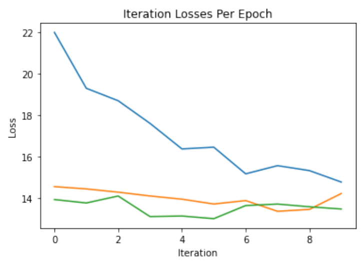
aggregator="MEAN" searchDepth=3
activationFunction="ReLu"sampleSizes=[8,8]

```
didConverge                                      False
ranIterationsPerEpoch                        [10, 10, 10]
iterationLossesPerEpoch   [[21.904654964588396, 22.83658392728238,
19.98...
ranEpochs                                           3
epochLosses            [14.23767916728001, 13.725985007856622,
12.654...
dtype: object
[14.23767916728001, 13.725985007856622, 12.654825814599912]
Mean similarity: 0.956262732467764
Number of matching relationships: 433
```

G.3 Results of Experiments on Initial Learning Rate =0.001 of GraphSAGE with KNN on Cora Dataset

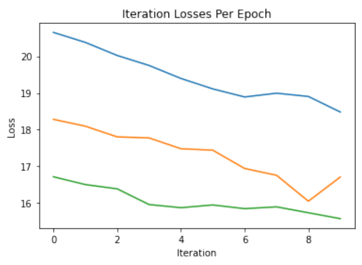
learningRate=0.001,aggregator="MEAN"
searchDepth=3
activationFunction="ReLu"sampleSizes=[8,8]

```
didConverge                                        False
ranIterationsPerEpoch                         [10, 10, 10]
iterationLossesPerEpoch    [[22.008286119030892, 19.313000253947074,
18.7...
ranEpochs                                              3
epochLosses              [14.791299331888364, 14.236805479042946,
13.48...
dtype: object
[14.791299331888364, 14.236805479042946, 13.489631265801638]
Mean similarity: 0.856513860975798
Number of matching relationships: 588
```



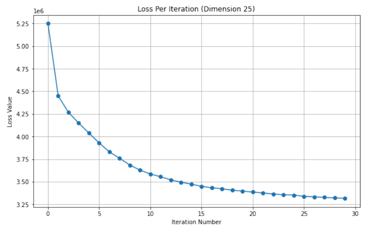
learningRate=0.001， aggregator="MEAN"
searchDepth=3
activationFunction="ReLu"sampleSizes=[8,8]

```
didConverge                                        False
ranIterationsPerEpoch                         [10, 10, 10]
iterationLossesPerEpoch    [[20.653596375141724, 20.38089396969442,
20.02...
ranEpochs                                              3
epochLosses              [18.479360185898344, 16.703690036370766,
15.56...
dtype: object
[18.479360185898344, 16.703690036370766, 15.567027167981925]
Mean similarity: 0.850995515685314
Number of matching relationships: 613
```

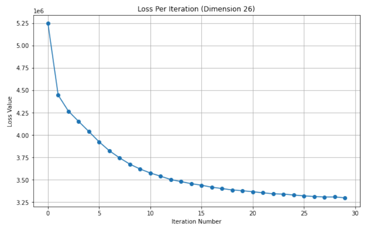


H Tuning Experiments of Node2Vec with KNN on Movielens Dataset

walklength=10

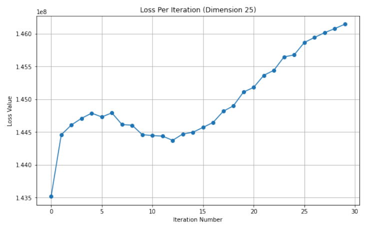


Mean similarity: 0.9179394287298098
Number of matched pairs between user_movie_recommendations.csv and test_set.txt: 5169

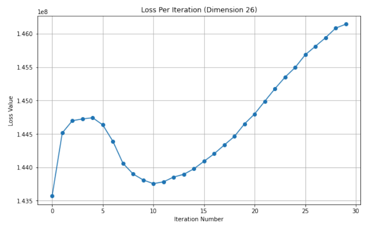


Mean similarity: 0.9143643395979072
Number of matched pairs between user_movie_recommendations.csv and test_set.txt: 5021

walklength=100

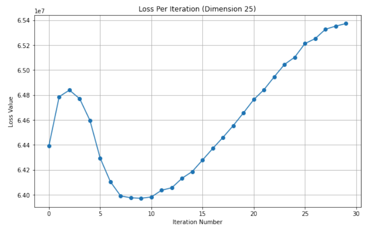


Mean similarity: 0.9485307979109621
Number of matched pairs between user_movie_recommendations.csv and test_set.txt: 4971

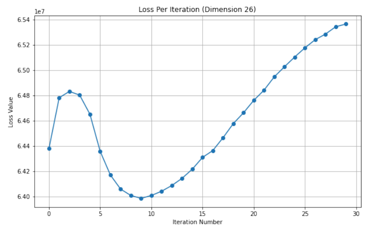


Mean similarity: 0.9395245605372539
Number of matched pairs between user_movie_recommendations.csv and test_set.txt: 4977

walklength=50

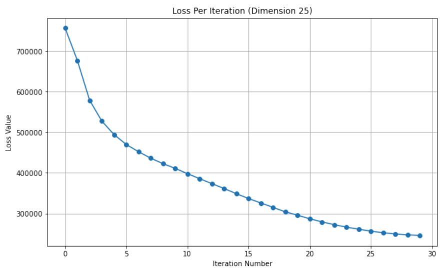


Mean similarity: 0.9353859937160655
Number of matched pairs between user_movie_recommendations.csv and test_set.txt: 4967

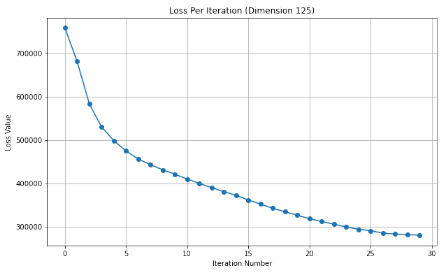


Mean similarity: 0.9326353849982498
Number of matched pairs between user_movie_recommendations.csv and test_set.txt: 4987

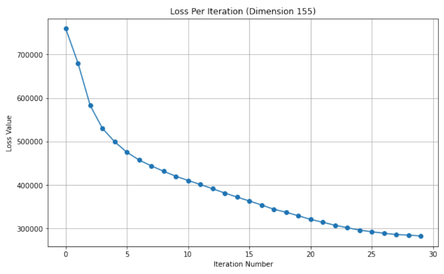
walklength=5



Mean similarity: 0.8523713693888213
Number of matched pairs between user_movie_recommendations.csv and test_set.txt: 4414



Mean similarity: 0.7380355205885119
Number of matched pairs between user_movie_recommendations.csv and test_set.txt: 4553

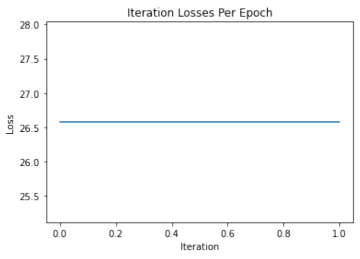


Mean similarity: 0.7286615581010578
Number of matched pairs between user_movie_recommendations.csv and test_set.txt: 4524

I Tuning Experiments of GraphSAGE with KNN on Movielens Dataset

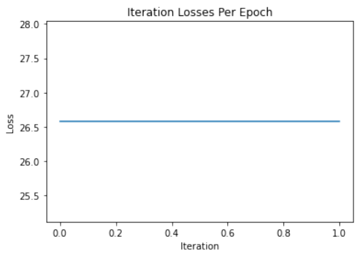
epoch=3，aggregator="MEAN" activationFunction="ReLu"   sampleSizes=[8,8]

```
didConverge                                              True
ranIterationsPerEpoch                                      [2]
iterationLossesPerEpoch    [[26.578495437882474, 26.578495437882474]]
ranEpochs                                                1
epochLosses                              [26.578495437882474]
dtype: object
[26.578495437882474]
Mean similarity: 1.0
Number of matched pairs between user_movie_recommendations.csv and
test_set.txt: 1872
```



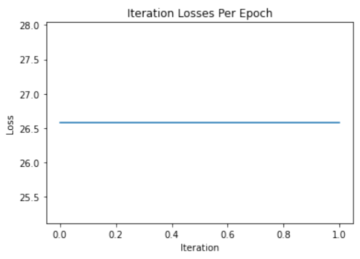
epoch=3，aggregator="MEAN" activationFunction="ReLu" sampleSizes=[10,10]

```
didConverge                                              True
ranIterationsPerEpoch                                      [2]
iterationLossesPerEpoch    [[26.578495437882516, 26.57849543788252]]
ranEpochs                                                1
epochLosses                               [26.57849543788252]
dtype: object
[26.57849543788252]
Mean similarity: 1.0
Number of matched pairs between user_movie_recommendations.csv and
test_set.txt: 3073
```



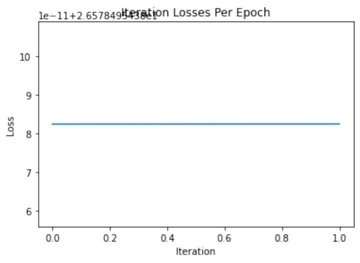
epoch=3，aggregator="MEAN" activationFunction="ReLu"sampleSizes=[30,30]

```
didConverge                                              True
ranIterationsPerEpoch                                      [2]
iterationLossesPerEpoch    [[26.578495437882488, 26.57849543788249]]
ranEpochs                                                1
epochLosses                               [26.57849543788249]
dtype: object
[26.57849543788249]
Mean similarity: 1.0
Number of matched pairs between user_movie_recommendations.csv and
test_set.txt: 2314
```



epoch=3，aggregator="MEAN" activationFunction="ReLu"sampleSizes=[15,15]

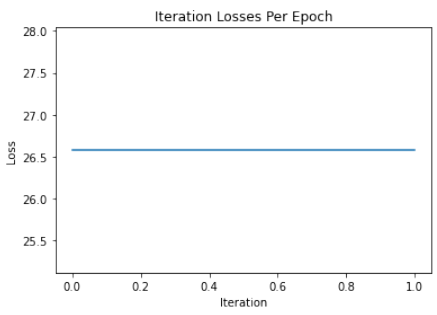
```
didConverge                                              True
ranIterationsPerEpoch                                      [2]
iterationLossesPerEpoch    [[26.57849543788246, 26.578495437882502]]
ranEpochs                                                1
epochLosses                              [26.578495437882502]
dtype: object
[26.578495437882502]
Mean similarity: 1.0
Number of matched pairs between user_movie_recommendations.csv and
test_set.txt: 2500
```



epoch=10，aggregator="MEAN" activationFunction="ReLu" sampleSizes=[10,10]

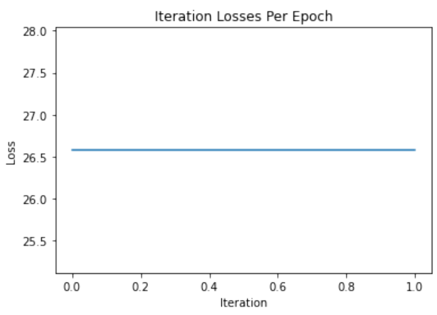
```
didConverge                                              True
ranIterationsPerEpoch                                      [2]
iterationLossesPerEpoch    [[26.578495437882488, 26.578495437882488]]
ranEpochs                                                1
epochLosses                              [26.578495437882488]
dtype: object
[26.578495437882488]
Mean similarity: 1.0
```

```
Number of matched pairs between user_movie_recommendations.csv and
test_set.txt: 1526
```



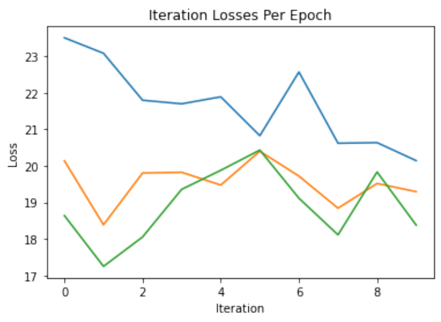
**epoch=1，aggregator="MEAN"
activationFunction="ReLu" sampleSizes=[10,10]**

```
didConverge                                          True
ranIterationsPerEpoch                                 [2]
iterationLossesPerEpoch    [[26.5784954378825, 26.578495437882488]]
ranEpochs                                              1
epochLosses                        [26.578495437882488]
dtype: object
[26.578495437882488]
Mean similarity: 1.0
Number of matched pairs between user_movie_recommendations.csv and
test_set.txt: 2975
```



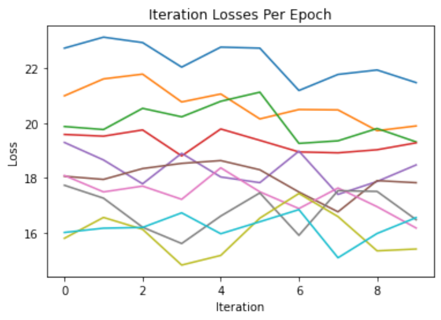
**epoch=3，aggregator="POOL"
activationFunction="ReLu"sampleSizes=[10,10]**

```
didConverge                                              False
ranIterationsPerEpoch                                [10, 10, 10]
iterationLossesPerEpoch    [[23.50903623353792, 23.0851805774743,
21.7992...
ranEpochs                                                  3
epochLosses              [20.144572770194024, 19.295002404139,
18.37821...
dtype: object
[20.144572770194024, 19.295002404139, 18.37821676935318]
Mean similarity: 0.9997699754663391
Number of matched pairs between user_movie_recommendations.csv and
test_set.txt: 5150
```



**epoch=10，aggregator="POOL"
activationFunction="ReLu" sampleSizes=[10,10]**

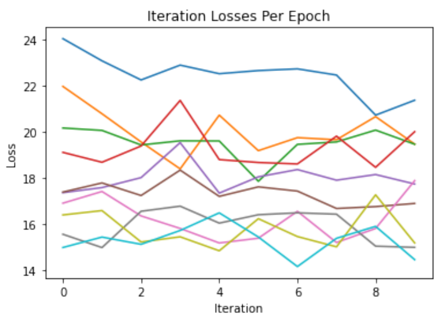
```
didConverge                                           False
ranIterationsPerEpoch         [10, 10, 10, 10, 10, 10, 10, 10, 10, 10]
iterationLossesPerEpoch    [[22.736006176992426, 23.136325308908674,
22.9...
ranEpochs                                                10
epochLosses              [21.478937469069088, 19.898557393304223,
19.31...
dtype: object
[21.478937469069088, 19.898557393304223, 19.315452252507118,
19.283626012199544, 18.4782996043101, 17.829936889595004,
16.176826538958327, 16.479512903452505, 15.410762672754009,
16.55733446603741]
```



```
Mean similarity: 0.9978706535634466
Number of matched pairs between user_movie_recommendations.csv and
test_set.txt: 5237
```

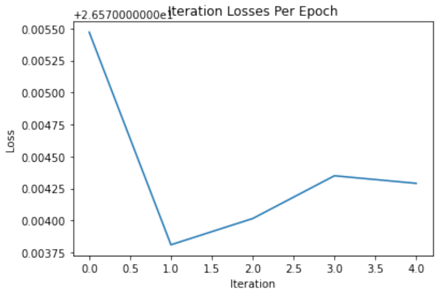
**epoch=10，aggregator="POOL"
activationFunction="ReLu"sampleSizes=[30,30]**

```
didConverge                                           False
ranIterationsPerEpoch         [10, 10, 10, 10, 10, 10, 10, 10, 10, 10]
iterationLossesPerEpoch    [[24.03143906109544, 23.07006323426566,
22.240...
ranEpochs                                                10
epochLosses              [21.36041070842736, 19.456409696101908,
19.451...
dtype: object
[21.36041070842736, 19.456409696101908, 19.45181318171054,
19.99500382299159, 17.726151012535386, 16.88280584572294,
17.87222110044179, 14.977646578400101, 15.168246856928132,
14.4433230281416]
Mean similarity: 0.9984186967849963
Number of matched pairs between user_movie_recommendations.csv and
test_set.txt: 5664
```



epoch=10，aggregator="POOL" activationFunction="Sigmoid" sampleSizes=[30,30]

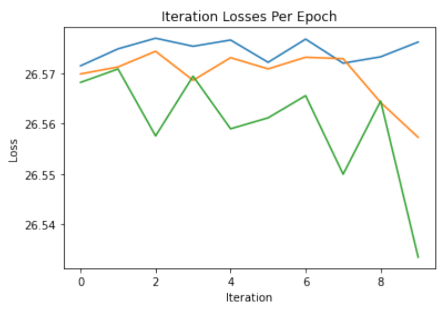
```
didConverge                                             True
ranIterationsPerEpoch                                    [5]
iterationLossesPerEpoch    [[26.575471765680476, 26.573809890217177,
26.5...
ranEpochs                                                  1
epochLosses                            [26.574290225690852]
dtype: object
[26.574290225690852]
Mean similarity: 0.999995992833287
Number of matched pairs between user_movie_recommendations.csv and
test_set.txt: 5584
```



epoch=3，aggregator="POOL" activationFunction="Sigmoid" sampleSizes=[30,30]

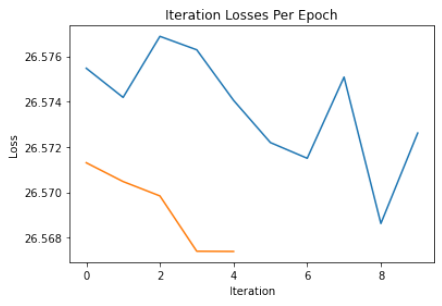
```
didConverge                                          False
ranIterationsPerEpoch                          [10, 10, 10]
iterationLossesPerEpoch    [[26.571567974896247, 26.57495718207296,
26.57...
```

```
ranEpochs                                                  3
epochLosses                 [26.576316569314447, 26.557286490361083,
26.53...
dtype: object
[26.576316569314447, 26.557286490361083, 26.53335619341953]
Mean similarity: 0.9999949564308749
Number of matched pairs between user_movie_recommendations.csv and
test_set.txt: 5494
```



epoch=10，aggregator="POOL" activationFunction="Sigmoid" sampleSizes=[30,30]

```
didConverge                                             True
ranIterationsPerEpoch                                [10, 5]
iterationLossesPerEpoch    [[26.575468209947434, 26.574183958327477,
26.5...
ranEpochs                                                  2
epochLosses                 [26.572613855279933, 26.5673951570246]
dtype: object
[26.572613855279933, 26.5673951570246]
Mean similarity: 0.999995745365961
Number of matched pairs between user_movie_recommendations.csv and
test_set.txt: 5567
```



J Quick Scan Document

Ethics and Privacy Quick Scan (version: 31 July 2023)

Section 1. Research projects involving human participants

		Yes	No
P1	Does your project involve human participants? This includes for example use of observation, (online) surveys, interviews, tests, focus groups, and workshops where human participants provide information or data to inform the research. If you are only using existing data sets or publicly available data (e.g. from Twitter, Reddit) without directly recruiting participants, please answer no.		√

If no, continue with Section 2; if yes, fill in the following questions.

<u>**Recruitment**</u>

		Yes	No
P2	Does your project involve participants younger than 18 years of age?		
P3	Does your project involve participants with learning or communication difficulties of a severity that may impact their ability to provide informed consent?[1]		
P4	Is your project likely to involve participants engaging in illegal activities?		
P5	Does your project involve patients?		
P6	Does your project involve participants belonging to a vulnerable[2] group, other than those listed above?		

If the answer to all of P2-P6 is no, continue with P8.

As you are dealing with vulnerable participants (yes to one (or more) of P2-P6) a fuller ethical review is required. Please add more detail on your participants here:

[1] For informed consent people need to be able to (1) understand information provided relevant to making the consent decision, (2) retain this information long enough to be able to make a decision, (3) weigh the information, (4) communicate the decision.

[2] Vulnerable people include those who are legally incompetent, who may have difficulty giving or withholding consent, or who may suffer highly adverse consequences if their personal data were to become publicly available or from participating. Examples include irregular immigrants, sex workers, dissidents and traumatized people at risk of re-traumatization.

		Yes	No
P7	Do you intend to be alone with a research participant or have to take sole responsibility for the participants at any point during your research activity?		

If P7 is no continue with P8, otherwise:

⚠ **As you will be alone with or solely responsible for vulnerable participants (yes to P7) a fuller ethical review is required. You may also need a Certificate of Conduct (Dutch: VOG) from the government.** Please add more detail here:

		Yes	No
P8	Does your project involve participants with whom you have, or are likely to have, a working or professional relationship: for instance, staff or students of the university, professional colleagues, or clients?		

If the answer to P8 is yes, please answer P9, otherwise, continue with PC1.

		Yes	No
P9	Is it made clear to potential participants that not participating will in no way impact them (e.g. it will not directly impact their grade in a class)?		

If the answer to P9 is yes, then continue with PC1, otherwise:

⚠ **As participants may think that not participating may harm them (yes to P8 and no to P9), participation may no longer be voluntary. Hence, a fuller ethical review is required.** Please provide more information here:

Consent Procedures		**Yes**	**No**	**Not applicable**
PC1	Do you have set procedures that you will use for obtaining *informed* consent from all participants, including (where appropriate) parental consent for children or consent from legally authorized representatives? (See suggestions for information sheets and consent forms on the website[3].)			
PC2	Will you tell participants that their participation is voluntary?			
PC3	Will you obtain explicit consent for participation?			
PC4	Will you obtain explicit consent for any sensor readings, eye tracking, photos, audio, and/or video recordings?			
PC5	Will you tell participants that they may withdraw from the research at any time and for any reason?			
PC6	Will you give potential participants time to consider participation?			
PC7	Will you provide participants with an opportunity to ask questions about the research before consenting to take part (e.g. by providing your contact details)?			

If the answer to PC1-PC7 is yes, then continue with PC8, otherwise:

Given your responses to the informed consent questions (a no on any of PC1-PC7), a fuller ethical review is required. Please provide more information regarding the questions that are causing this here:

		Yes	**No**
PC8	Does your project involve concealment[4] or deliberate misleading of participants?		

[3] uu.nl/en/research/institute-of-information-and-computing-sciences/ethics-and-privacy
[4] This may for example involve concealment of the study aim, of the identity of the researcher, or subliminal messaging during the study.

If the answer to PC8 no, continue with Section 2, otherwise:

⚠ As you plan to use concealment or misleading (yes to PC8), and this may impact participants' rights to informed consent, a fuller ethical review is required. Please provide more information on the concealment/misleading here:

Section 2. Data protection, handling, and storage

The General Data Protection Regulation imposes several obligations for the use of **personal data** (defined as any information relating to an identified or identifiable living person) or including the use of personal data in research.

		Yes	No
D1	Are you gathering or using personal data (defined as any information relating to an identified or identifiable living person[5])?		√

If the answer to D1 is yes, please answer the following questions; otherwise, continue with Section 3.

High-Risk Data

		Yes	No
DR1	Will you process personal data that would jeopardize the physical health or safety of individuals in the event of a personal data breach?		
DR2	Will you combine, compare, or match personal data obtained from multiple sources, in a way that exceeds the reasonable expectations of the people whose data it is?[6]		
DR3	Will you use any personal data of children or vulnerable individuals for marketing, profiling, automated decision-making, or to offer online services to them?		

[5] This includes people's name, postal address, unique ID, IP address, voice, photo, video etc. When a person can be identified by combining multiple data points (e.g. gender + age + job role), this also constitutes personal data. When a person can be identified by a simple search online (e.g. with the content of a tweet) this also constitutes personal data. Note that Survey tool Qualtrics by default collects IP addresses and that the survey needs to be anonymized before distribution to prevent this.

[6] This is about the combined use of data sets that have been gathered for different purposes (so not within one study), making the data more personal or sensitive. For example, combining participant data with religion or ethnic statistics data from the CBS based on zip code.

DR4	Will you profile individuals on a large scale[7]?		
DR5	Will you systematically monitor individuals in a publicly accessible area on a large scale[8] (or use the data of such monitoring)?[9]		
DR6	Will you use special category[10] personal data, criminal offense personal data, or other sensitive personal data[11] on a large scale?		
DR7	Will you determine an individual's access to a product, service, opportunity, or benefit[12] based on an automated decision or special category personal data?		
DR8	Will you systematically and extensively monitor or profile individuals, with significant effects[13] on them?		
DR9	Will you use innovative technology[14] to process sensitive personal data[15]?		

If the answer to DR1-DR9 is no, continue with DM1, otherwise:

⚠ **As high-risk data processing seems involved (yes to any of DR1-DR9), a fuller privacy assessment is required.** Please provide more information on the DR1-DR9 questions with a yes here:

[7] Large scale is for example thousands of people, all visitors to a university website, data obtained over a very large time span
[8] Large scale is for example thousands of people, all visitors to the area, data obtained over a very large time span
[9] This may also include camera surveillance and use of drones
[10] Special category personal data is information about a person's health, ethnic origin, politics, religion, trade union membership, genetics, biometrics (where used in identification), sex life or sexual orientation.
[11] Other sensitive personal data includes for instance financial data (from which people's income, capital position or spending patterns can be derived), location data (from which people's movement patterns can be derived), achievement data (e.g. outcome of course work/exams, intelligence test; this excludes performance on tasks in a research study that are unrelated to their study/job), and communication data.
[12] Examples include: access to a mortgage, insurance, credit card, smartphone contract, course or degree programme, job opportunity.
[13] Significant effects are for example impacts on somebody's legal rights, automatic refusal of a credit application, automatic rejection for a job application.
[14] Innovative technology includes e.g. machine learning (including deep learning), neuro measurement (e.g. brain activity), autonomous vehicles, deep fakes, wearables, blockchain, internet of things.
[15] Sensitive personal data includes all data mentioned in DR6.

Data Minimization

		Yes	No
DM1	Will you collect only personal data that is strictly necessary for the research?		

If you answered yes to DM1 continue with DM4, otherwise:

		Yes	No
DM2	Will you only collect not strictly necessary personal data because it is (1) technically unfeasible not to collect it when collecting necessary data[16], or (2) needed as a source of necessary data[17]?		
DM3	Will you (1) extract any necessary data as soon as possible from the collected not strictly necessary data and (2) delete the not strictly necessary data immediately after any required extraction?[18]		
DM4	Will you anonymize the data wherever possible?[19]		
DM5	Will you pseudonymize the data if you are not able to anonymize it, replacing personal details with an identifier, and keeping the key separate from the data set?		

If the answer to any of DM2-DM5 is no, see warning below, otherwise continue with DC1.

⚠ **As you do not seem to minimize data collection (no to any of DM2-DM5), a fuller privacy assessment is required.** Please provide more information on the DM2-DM5 questions with a no here:

[16] This may for instance occur when IP data is collected automatically in Qualtrics, and it is unfeasible not to do so as other personal data such as email needs to be collected.
[17] This may, for instance, occur when audio data is captured from which audio features need extracting or a transcript needs to be produced.
[18] This may for instance happen when you collect audio data, extract audio features or transcribe an audio interview as soon as possible, and delete the original audio recording once done.
[19] Possible also means given the research question. So, for example, if you have done interviews and you need to be able to at a later date link them to performance data, it is impossible to anonymize the interviews, and you will need to pseudonymize them. You can then answer yes to DM4 as you are anonymizing where it is possible, and yes to DM5 if indeed you pseudonymize. Note that in such a case you should anonymize once the linking has been done, destroying the key that links the pseudonym to the identity of the participant.

Using Collaborators or Contractors that Process Personal Data Securely

		Yes	No
DC1	Will any organization external to Utrecht University be involved in processing personal data (e.g. for transcription, data analysis, data storage)?[20]		

If the answer to DC1 is yes, please complete DC2 otherwise continue with DI1.

		Yes	No
DC2	Will this involve data that is not anonymized?		

If the answer to DC2 is yes, please complete DC3-DC5, otherwise continue with DI1.

		Yes	No	Not Applicable
DC3	Are they capable of securely[21] handling data?			▨
DC4	Has been drawn up in a structured and generally agreed manner who is responsible for what concerning data in the collaboration?			▨
DC5	Is a written contract covering this data processing in place for any organization which is not another university in a joint research project?			

If the answer to any of DC3-DC5 is no, see warning below, otherwise continue with DI1.

As you do not seem to have appropriate processes in place for sharing data with collaborators or contractors (no to any of DC3-DC5), a fuller privacy assessment is required. Please provide more information on the DC3-DC5 questions with a no here:

[20] You can answer No if this is only the use of online software, as long as this software has been deemed safe by Utrecht University. See https://tools.uu.nl/tooladvisor/ for tools that are safe/not safe to use (e.g. Microsoft Word Online Transcribe is fine, NVivo Transcription is not).

[21] Secure handling includes for example: (1) only sharing data with those who legitimately need to see it, (2) data being securely stored on password-protected employer authorized IT systems (or in the case of non-digital data: in a secure locked location), (3) if portable devices such as USB sticks are used then only encrypted and password protected with data deleted as soon as it is no longer required to be portable, (4) reporting lost or stolen data immediately, (5) deleting or disposing of data as soon as it is no longer required and in a secure manner, (6) not discussing sensitive data in public places, (7) only carrying needed data when working off-site.

International Personal Data Transfers

		Yes	No
DI1	Will any personal data be transferred to another country (including to research collaborators in a joint project)?		

If the answer to DI1 is yes, please complete DI2, otherwise continue with DF1.

		Yes	No
DI2	Do all countries involved in this have an adequate data protection regime?[22]		

If the answer to DI2 is no, please complete DI3, otherwise continue with DF1.

		Yes	No
DI3	Is a legal agreement in place?		

If the answer to DI2 and DI3 is no, see warning below, otherwise, continue with DF1.

⚠ **As you do not seem to have appropriate safeguards in place for international data transfers (no to DI2 and DI3), a fuller privacy assessment is required.** Please provide more information on intended international data transfers here:

Fair Usage of Personal Data to Recruit Participants

		Yes	No
DF1	Is personal data used to recruit participants?[23]		

[22] Countries with an adequate data protection regime include EU countries, Andorra, Argentina, Canada (only commercial organizations), Faroe Islands, Guernsey, Israel, Isle of Man, Jersey, New Zealand, Switzerland, Uruguay, Japan, the United Kingdom, and South Korea.

[23] Intended here is the direct use of personal data to target a specific person. If you are using personal data indirectly to address a group of people, for example, sending a message via a *pre-existing* Microsoft Team, Blackboard course, Discord Channel, WhatsApp group, or crowd-sourcing platform, that is fine and will not be regarded as the use of personal data here. If you are asking friends or family members this will also not be regarded as use of personal data here.

If the answer to DF1 is yes please answer DF2-DF4, otherwise continue with DP1

		Yes	No	N/A
DF2	Have potential participants provided this personal data voluntarily to be contacted about the research or is the data publicly available?			▨
DF3	If contact details have been provided by a third party, would participants expect their details to be passed on to the university and to be used in this way?			
DF4	If contact details have been gathered for a purpose other than research, would participants expect their details to be used in this way?			

If the answers to DF2-DF4 are yes or N/A continue with DP1, otherwise:

As there seem to be issues with your use of personal data for recruitment (no to one or more of DF2-DF4), a fuller privacy assessment is required. Please provide more information on the intended use of personal data for recruitment here:

Participants' data rights and privacy information

		Yes	No	Not Applicable
DP1	Will participants be provided with privacy information? (Recommended is to use as part of the information sheet: For details of our legal basis for using personal data and the rights you have over your data please see the University's privacy information at www.uu.nl/en/organisation/privacy.)			▨
DP2	Will participants be aware of what their data is used for?			▨
DP3	Can participants request that their personal data be deleted?[24]			▨

[24] This only concerns requests for personal data that you still hold. If you can no longer link the data to a participant due to anonymization, you can no longer delete, rectify or provide access to it. This should be clear to participants in the consent form. If the data is pseudonymized and you cannot access the key but the participant can (for example when the key is a WorkerID from a crowd-sourcing platform), participants should be able to request deletion on the provision of the key.

		Yes	No	Not Applicable
DP4	Can participants request that their personal data be rectified (in case it is incorrect)?[24]			▨
DP5	Can participants request access to their personal data?[24]			▨
DP6	Can participants request that personal data processing is restricted?			▨
DP7	Will participants be subjected to automated decision-making based on their personal data with an impact on them beyond the research study to which they consented?			▨
DP8	Will participants be aware of how long their data is being kept for, who it is being shared with, and any safeguards that apply in case of international sharing?			▨
DP9	If data is provided by a third party, are people whose data is in the data set provided with (1) the privacy information and (2) what categories of data you will use?			

If the answer to DP1-DP6, DP8, DP9 is yes and DP7 is no, continue with DE1, otherwise:

⚠ **As there seem to be issues with the data rights of your participants or the provision of privacy information (no to one or more of DP1-DP6, DP8, DP9, or yes to DP7), a fuller privacy assessment is required.** Please provide more detail regarding data rights and/or privacy information here:

Using data you have not gathered directly from participants

		Yes	No
DE1	Will you use any personal data[25] that you have not gathered directly from participants (such as data from an existing data set, data gathered by a third party, data scraped from the internet)?		

If the answer to DE1 is no please continue with DS1.

		Yes	No
DE2	Will you use an existing dataset in your research?		

If the answer to DE2 is yes please answer DE3-DE5, otherwise, continue with DE6.

[25] Defined as any data related to an identified or identifiable living person. This includes people's name, postal address, unique ID, IP address, voice, photo, video etc. When a person can be identified by combining multiple data points (e.g. gender + age + job role), this also constitutes personal data.

		Yes	No
DE3	Do you have permission to do so from the owners of the dataset?		
DE4	Have the people whose data is in the data set consented to their data being used by other researchers and/or for purposes other than that for which that data set was gathered?		
DE5	Are there any contractual conditions attached to working with or storing the data from DE2?		

		Yes	No
DE6	Does your project require access to personal data about participants from other parties (e.g., teachers, employers), databanks, or files[26]?		

If the answer to DE6 is yes please answer DE7-DE8, otherwise, continue with DE9.

		Yes	No
DE7	Do you have a process in place to gain informed consent from these participants?		
DE8	Are there any contractual conditions attached to working with or storing the data from DE5?		

		Yes	No
DE9	Does the project involve collecting personal data from websites or social media (e.g., Facebook, Twitter, Reddit)?		

⚠ **As there may be issues with the use of existing data (no to DE3, DE4, DE7 or yes to DE9), a fuller privacy assessment is required.** Please provide more detail regarding the use of existing data here:

[26] For example, do you get a student's grade from the teacher, in addition to data gathered directly in your study or data in an existing research data set?

Secure data storage

		Yes	No
DS1	Will any data be stored (temporarily or permanently) anywhere other than on password-protected University authorized computers or servers?[27]		

If the answer to DS1 is yes, please answer DS2, otherwise, continue with DS4.

		Yes	No
DS2	Does this only involve data stored temporarily during a session with participants (e.g. data stored on a video/audio recorder/sensing device), which is immediately transferred (directly or with the use of an encrypted and password-protected data-carrier (such as a USB stick)) to a password-protected University authorized computer or server, and deleted from the data capture and data-carrier device immediately after transfer?		

If the answer to DS2 is yes, continue with DS4, otherwise answer DS3.

		Yes	No
DS3	Does this only involve data stored with a collaborator or contractor?		
DS4	Excluding (1) any international data transfers mentioned above and (2) any sharing of data with collaborators and contractors, will any personal data be stored, collected, or accessed from outside the EU[28]?		

If the answer to DS2 and DS3 is no, or the answer to DS4 is yes, see the warning below, otherwise continue with Section 3.

⚠ **As there may be issues with secure data storage (no to DS2 and DS3, or yes to DS4), a fuller privacy assessment is required.** Please provide more detail regarding data storage here:

[27] OneDrive business, Qualtrics, Microsoft Forms are ok. Do not use Google Drive/Sheets/Docs/Forms, Dropbox, OneDrive personal. See https://tools.uu.nl/tooladvisor/ for tools that are safe/not safe to use. Bachelor and master students are authorized to use a password-protected personal computer, as long as that computer is not shared with other people.
[28] This may happen, for instance, when data is collected and stored on a Utrecht University laptop whilst abroad.

Section 3: Research that may cause harm

Research may harm participants, researchers, the university, or society. This includes when technology has dual-use, and you investigate an innocent use, but your results could be used by others in a harmful way. If you are unsure regarding possible harm to the university or society, please discuss your concerns with the Research Support Office.

		Yes	No
H1	Does your project give rise to a realistic risk to the national security of any country?[29]		√
H2	Does your project give rise to a realistic risk of aiding human rights abuses in any country?[30]		√
H3	Does your project (and its data) give rise to a realistic risk of damaging the University's reputation? (E.g., bad press coverage, public protest.)		√
H4	Does your project (and in particular its data) give rise to an increased risk of attack (cyber- or otherwise) against the University? (E.g., from pressure groups.)		√
H5	Is the data likely to contain material that is indecent, offensive, defamatory, threatening, discriminatory, or extremist?		√
H6	Does your project give rise to a realistic risk of harm to the researchers?[31]		√
H7	Is there a realistic risk of any participant experiencing physical or psychological harm or discomfort?[32]		√
H8	Is there a realistic risk of any participant experiencing a detriment to their interests as a result of participation?		√
H9	Is there a realistic risk of other types of negative externalities?[33]		√

[29] For example, research that can be used for autonomous armed vehicles/drones/robots, research on automated detection of objects, research on AI-enhanced forgery of video/audio data.
[30] For example, research on natural language/video/audio processing for automated identification of people's identity, sentiments, or opinions.
[31] For example, research that involves potentially violent participants such as criminals, research in likely unsafe locations such as war zones, research on an emotionally highly challenging topic, research in which the researcher is alone with a not previously known participant in the participant's home.
[32] For example, research that involves strenuous physical activity, research that stresses participants, research on an emotionally challenging topic.
[33] A negative externality is a harm produced to a third party, society in general, or the environment. For instance, intended or unintended negative ethical (e.g. bad governance or management practices), social (e.g. consumerism, inequality) or environmental effects (e.g. large CO2 footprint or e-waste production) of your project.

If the answer to H1-H9 is no continue with Section 4, otherwise:

⚠ **As you replied yes to one (or more) of H1-H9, a fuller ethical review is required.** Please provide more detail here on the potential harm, and how you will minimize risk and impact:

Section 4: Conflicts of interest

		Yes	No
C1	Is there any potential conflict of interest (e.g. between research funder and researchers or participants and researchers) that may potentially affect the research outcome or the dissemination of research findings?		√
C2	Is there a direct hierarchical relationship between researchers and participants?		√

If the answer to C1-C2 is yes, continue with Section 5, otherwise:

⚠ **As you replied yes to C1 or C2, a fuller ethical review is required.** Please provide more information regarding possible conflicts of interest and how you mitigate them here:

Section 5: Your information

This last section collects data about you and your project so that we can register that you completed the Ethics and Privacy Quick Scan, sent you (and your supervisor) the summary of what you filled out, and follow up where a fuller ethics review and/or privacy assessment is needed. For details of our legal basis for using personal data and the rights you have over your data please see the University's privacy information. Please see the guidance on the ICS Ethics and Privacy website on what happens on submission.

Z0. Which is your main department?

[✓ ○ Information and Computing Science]
○ Freudenthal Institute
○ Pharmacy
○ Other, namely:

Z1. Your full name: Tinting Zhang

Z2. Your email address:t.zhang5@students.uu.nl

Z3. In what context will you conduct this research?

- ○ 1. As a student on a course with course coordinator:
- ○ 2. As a student for my bachelor thesis, supervised by:
- [✓ ○] 3. As a student for my master thesis, supervised by: Behrisch Michael
- ○ 4. As a PhD student, supervised by:
- ○ 5. As an independent researcher (e.g. research fellow, assistant/associate/full professor)

In case the answer to Z3 is 2:

Z4. Bachelor programme for which you are doing the thesis:

- ○ Artificial Intelligence (Kunstmatige Intelligentie)
- ○ Computing Science (Informatica)
- ○ Information Science (Informatiekunde)
- ○ Other:

In case the answer to Z3 is 3:

Z5. Master programme for which you are doing the thesis:

- ○ Applied Data Science
- ○ Artificial Intelligence
- ○ Business Informatics
- ○ Computing Science
- ○ Data Science
- ○ Game and Media Technology
- [✓ ○] Human-Computer Interaction
- ○ Other:

In case the answer to Z3 is 1, 2, 3, or 4:

Z6. Email of the course coordinator or supervisor (so that we can inform them that you filled this out and provide them with a summary): m.behrisch@uu.nl

In case the answer to Z3 is 2 or 3:

Z7. Email of the moderator (as provided by the coordinator of your thesis project):

Z8. Title of the research project/study for which you filled out this Quick Scan:

Design and Implementation of a Paper Recommendation System Based on Neo4j and Knowledge Graph

Z9. Summary of what you intend to investigate and how you will investigate this (200 words max):

This research aims to integrate an intelligent recommendation system based on knowledge graphs into the MiniConf framework, enhancing the experience of virtual academic conferences. We will construct a comprehensive knowledge graph encompassing conference papers, authors, topics, and details, employing an optimized graph query language to facilitate personalized recommendations for papers, individuals, and session schedules. The system will implement user tracking and feedback mechanisms to achieve real-time optimization. Anticipated outcomes include a well-structured knowledge graph supporting semantic queries and a context-aware graph query language tailored to meet the individualized needs of conference participants. Additionally, we will employ an explainable and updated recommendation system to enhance user experience and foster trust. Through this innovation, we aim to strengthen the recommendation system, deliver personalized suggestions, and provide valuable insights into user behaviour and preferences for conference organizers. Ultimately, our goal is to transform MiniConf into a more semantically-aware platform, shaping the future of virtual academic conferences and promoting global academic collaboration.

In case the answer to Z3 is 2 or 3:

		Yes	**No**	**Not Applicable**
Z10.	In case you encountered warnings in the survey, does your supervisor already have ethical approval for a research line that fully covers your project?		√	

In case the answer to Z10 is yes:

Z11. Provide details on the ethical approval (e.g. ethical approval number):