A ”tinder for startups”: building a matchmaking engine for new ventures and Venture Capital firms

Shadullah Sartaj Syed

Master of Science
Computer Science
School of Informatics
University of Edinburgh
2023

Abstract

Linking startups with potential investors is a complicated task in today's competitive entrepreneurial ecosystem. This thesis embarked on a computational exploration of this domain, aiming to unravel and predict the dynamics of investor-startup relationships. We developed two distinct models. Model 1 utilized K-Nearest Neighbors (KNN) graphs to identify similar startups and made recommendations using Graph Sage architecture. In contrast, Model 2 incorporated KNN graphs as supplementary edges within the Graph Attention Networks (GAT) architecture. Comprehensive experiments, including hyperparameter optimization using Optuna, were conducted to evaluate and refine the models. Model 1 showcased a decent performance, especially reflected in the Mean Average Precision (MAP) metric, indicating its capability to consistently deliver quality investor recommendations across varied startup groupings. Model 2, while foundational, revealed its strength in the Hit Rate metric, suggesting its potential in capturing genuine investor-startup interactions. Beyond the computational models, this research illuminates the future of investor-startup interactions—a paradigm shift from reliance on traditional networking to a structured, data-driven approach. As a roadmap for the future, avenues such as Investor Clustering, Investor Selection Metrics, Co-investor and Lead Investor Integration are proposed to further enhance the recommendation system's efficiency and accuracy. This thesis, thus, stands as a cornerstone in the realm of machine learning-driven investor-startup recommendations, aiming to bridge the gap between budding startups and the investors poised to champion them.

Research Ethics Approval

The project guidelines suggested a team of 2 to 3 students. Following the project guidelines, our team consisted of Zain and I, Sartaj. With my background in Computer Science and Zain's in Data Science. Our supervisor advised us to tackle this project together. Below is how we divided our tasks:

I, **Sartaj**, managed data to find Investor-Startup relationship, executed the Exploratory Data Analysis (EDA), and developed scripts for Investor Features. I also worked on the Textual Embedding of Investor data and addressed Co-Investors and Lead Investor data pre-processing.

On the other hand, **Zain** handled the Startup Feature data, conducted a separate EDA, and managed Text Embedding for Startups. He was also in charge of calculating Rivalry Metrics, creating the investor-startup network graph, and the Principal Component Analysis.

Our combined efforts were central to the continuous improvement and success of our model. We brainstormed, experimented, and refined our strategies collaboratively. Tasks such as model optimization, evaluation, insight generation, and conclusion drawing were approached as a unit, resulting in a cohesive and well-integrated outcome. Our synergistic collaboration not only streamlined our workflow but also propelled us to reach milestones that might have been challenging on an individual front.

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Shadullah Sartaj Syed*)

Acknowledgements

I’d like to thank my supervisor, Dr. Filippo Menolascina for trusting me with this project. Our discussions helped shape my approach, and his insights were always helpful. His expertise and knowledge greatly contributed to the quality of my work and broadened my understanding. I’ve learned a lot from him, which has improved my work and overall experience.

A hearty thanks to my team member. Your dedication played a pivotal role in our collective achievements. Facing challenges head-on, we made commendable progress together.

Special recognition goes to my friends Aman, Nithya, Bilal, and Radhikesh. Your unwavering support and encouragement have been my compass, directing and uplifting me.

To all my friends and family, your belief in my capabilities and the endless encouragement you provided have been my strongest pillars. You’ve been the bedrock of my journey.

Table of Contents

1 Introduction **1**

2 Background **3**

2.1 Matrix Factorization 4

2.2 Deep Learning with GNNs 5

2.3 GNNs in Recommender Frameworks 5

2.3.1 Homogeneous Graph Recommendations 5

2.3.2 Bipartite Graph Recommendations 6

2.3.3 Heterogeneous Graph Recommendations 6

2.4 Specific GNN Methods for Recommendation Systems 6

2.4.1 GraphSAGE 7

2.4.2 GAT 7

2.5 Case Studies 8

2.5.1 Pinterest Recommender System 8

2.5.2 Amazon Recommender System 11

3 Data Preparation **15**

3.1 Investor-Startup Relation 15

3.2 Investor Features 16

3.3 Co-investors 17

3.4 Lead Investors 18

4 Exploratory Data Analysis **20**

4.1 Temporal Analysis of Startup Categories 20

4.2 Investment Patterns Across Rounds 21

4.3 Average Funding Across Investment Rounds 21

4.4 Geographical Consistencies in Investments 22

4.5 Correlation Analysis of Numerical Features 23

5 Methodology **24**
5.1 Model 1: Knn + Graphsage 24
5.1.1 Problem Settings 24
5.1.2 Model Architecture 25
5.1.3 Training and Validation 27
5.1.4 Recommendation Process for New Startups 27
5.2 Model 2: KNN-Graph Augmented Edge Prediction for Node Linkages 29
5.2.1 Problem Settings 29
5.2.2 Model Architecture 29
5.2.3 Training and Validation Procedure 30
5.2.4 Recommendation Process for New Startups 31

6 Experiments **33**
6.1 Model 1 33
6.1.1 Hyperparameter Optimization with Optuna 33
6.1.2 Experiment Settings 33
6.1.3 Hyperparameter Optimization Results 34
6.1.4 Inference on New Startups 35
6.1.5 Evaluation Metrics 36
6.1.6 Evaluation Results 36
6.1.7 Limitations 37
6.2 Model 2 37

7 Conclusion and Future Work **39**
7.1 Conclusion 39
7.2 Future Work 40

Bibliography **41**

Chapter 1

Introduction

Societal progress, economic expansion, and technological innovation is now being driven by the global startup ecosystem. This ecosystem's fundamental strength is the symbiotic relationship between startups, which are frequently brimming with creative ideas but lacking in funding [29], and venture capitalists (VCs), who provide not only funding but also networking opportunities, industry knowledge, and guidance [30]-[26]. Due to their collaboration, early-stage startups have transformed into powerful market leaders and industry titans [12].

However, the sheer number of startups across various industries and regions has made the ecosystem complex and multifaceted. Each startup, with its own vision and difficulties, looks for a venture capitalist whose approach, risk tolerance, and expertise match its own course [29]. VCs, on the other hand, sift through a lot of startups to find those that align with their investment philosophies and have the potential to generate good returns [10]-[38]. In this vastness, it becomes crucial but difficult to make sure that the right startup is aligned with the right VC. In reality, this simple matching problem is complicated by the nuanced aspects of market dynamics, investment approaches, growth potential, and risk considerations.

When we examine the semantics of this relationship, we find that startups and VCs interact within a complex web of preferences, histories, and strategies. For instance, a startup's prior funding rounds, growth trajectory, competitive environment, and team background all play crucial roles in attracting a potential investor [38]-[16]. On the other hand, a VC's preference for a particular startup is influenced by their prior investments, sectoral focus, typical investment size, and exit strategies [28].

Additionally, there are more aspects of the ”recommendation” problem than simply compiling a list of VCs for a startup or suggesting startups to VCs. It entails comprehending the complex dynamics of the startup ecosystem, analysing previous investment trends, identifying growth potential, and forecasting future interactions based on a variety of variables [47]. The stakes are high because a strong recommendation can spur the success of a startup, while a weak recommendation could result in missed opportunities or, worse, startup failure [24].

In this complex environment, the development of advanced computational techniques, particularly machine learning and Graph Neural Networks (GNNs) , offers a practical solution. As explained in the following chapters of this thesis, GNNs offer a strong framework to model the complex interactions and relationships in the startup-investor ecosystem due to their exceptional capacity to handle graph-structured data [8]-[20]-[19]. By doing this, they open the door for producing recommendations that are more perceptive, precise, and useful, which streamlines the matchmaking procedure.

This thesis offers a systematic journey into the complex realm of investor-startup recommendations. Chapter 2 establishes the groundwork, elucidating matrix factorization, deep learning, and GNNs, particularly within the context of recommendation systems. It highlights specific GNN techniques suitable for recommendation systems and is bolstered by practical implementations from industry leaders like Pinterest and Amazon. Chapter 3 focuses on data pre-processing, extracting features and relationships in the investor-startup ecosystem. Chapter 4 delves into a detailed exploration of the data, revealing patterns and insights into the investor-startup ecosystem. Chapter 5 focuses on the research's crux, detailing the methodologies and introducing two pioneering models designed for the recommendation challenge. Chapter 6 provides a comprehensive account of the experiments conducted, covering their configurations, results, and the evaluation criteria employed to assess the model performances. The research culminates in Chapter 7, which draws insightful conclusions, emphasizing the wider ramifications, and outlining avenues for prospective research.

This study deep dives into the complexities and intricacies of the startup-VC landscape. It aims to provide valuable insights and tools that can significantly enhance the matchmaking process, benefiting academicians, industry stalwarts, startups, and venture capitalists.

Chapter 2

Background

The rapid growth of technology-focused sectors coupled with an increased appetite for tailored experiences has expanded research into recommendation systems [32]. These systems expertly match customers with goods or services by considering their prior tendencies and behaviours. They have displayed impressive adaptability, finding uses in a variety of industries like online shopping, media consumption, and financial services [17]-[34]-[14]. An efficient recommendation system, for instance, can connect startups with the right investors, potentially revolutionising the investment industry by streamlining capital acquisition and enhancing the chances that a startup will succeed.

Conventional recommendation systems generally employ one of three primary algorithms to suggest products or information tailored to a user's preferences and requirements [31]. These encompass content-driven recommendation methods [25], collaborative filtering techniques [46], and a combination of both, known as hybrid recommendation strategies [35].

Content-driven recommendation methods, which belong to the first category, operate by pinpointing products a user has shown a liking for in the past. Using items resembling these favoured products, the system formulates the suggestions [25]. Collaborative filtering recommendation methods, belonging to the second category, are further divided into two subsets: memory-based and model-based collaborative filtering [46]-[11]. The memory-based subset includes both user-centric and item-centric collaborative filtering, both of which are prevalent in recommendation system designs [33]-[48]. The model-based collaborative filtering, on the other hand, leverages machine learning algorithms to predict a user's interest in an item, relying on patterns discerned from historical data. The final category encompassing hybrid recommendation techniques merges various standalone algorithms, often yielding results superior to any individual method.

Nonetheless, their intricate nature makes them a challenging endeavor to actualize [35].

Collaborative filtering (CF) has become a staple in recommendation system methodologies, largely propelled by its association with the Netflix Prize challenge [13]. CF functions by analysing historical user actions to suggest products mirroring those a user previously engaged with. This method bifurcates into two branches: user-centric and item-centric [33]-[48]. While the former makes suggestions based on preferences of similar users, the latter emphasizes items akin to a user's past interests [13]-[33]-[48]. Notwithstanding its evident successes, CF grapples with the 'cold start' dilemma, a scenario where nascent users or items pose recommendation challenges due to the unavailability of past data [13].

Each algorithmic strategy has its own advantages and disadvantages. For instance, content-driven recommendation techniques have an advantage because they don't heavily rely on past user interactions and are generally resistant to problems like data scarcity or the aforementioned cold start problem. However, due to their inherent feature extraction limitations, they struggle with scalability, especially when dealing with large amounts of data. On the other hand, collaborative filtering techniques fall short when dealing with sparse data and due to the cold start problem, while excelling in terms of diversifying suggestions and intuitively understanding user preferences. Hybrid approaches promise to address these issues, but their implementation is complex.

2.1 Matrix Factorization

Matrix factorization, specifically Singular Value Decomposition (SVD) , is another technique that can be used in recommendation systems. SVD decomposes a matrix into three other matrices, capturing the underlying structure of the original matrix [13]. This makes it especially useful for recommendation systems, where it can capture the latent factors behind user-item interactions [42]. MF techniques have shown great success in many fields, such as movie recommendations and e-commerce product recommendations [42].

In the context of investor-startup recommendations, SVD could decompose the investor-startup interaction matrix to capture the latent factors underlying investment decisions, such as market trends, investor preferences, and startup growth potential. These latent factors could then be used to recommend suitable startups to investors, potentially transforming the investment landscape and streamlining the funding process.

2.2 Deep Learning with GNNs

Deep learning, a subset of machine learning, has emerged as a critical tool in numerous fields, including image processing. Its application in recommendation systems has garnered a lot of academic attention. Among the spectrum of deep learning methods, Graph Neural Networks (GNNs) stand out due to their prowess over managing intricate graph configurations and non-traditional data processing [45]-[19].

The initial graph convolutional neural network was unveiled in 2013 by [3]. By fusing principles from graph theory [7] and the convolution theorem, they articulated graph convolution within the spectral domain. Inspired by this groundbreaking discovery, [6]. introduced the concept of a frequency domain graph convolutional network. These spectral and frequency domain innovations garnered global acclaim.

Fast-forward to 2018, [2]. advanced the notion of graph networks tailored for relational inference. Their goal was to address combinatorial generalization challenges and employ relational induction biases to equip deep architectures with the capability to comprehend entities, their interrelationships, and organizational methods. Further insights into graph-based deep learning approaches were provided by [44] and [49].

2.3 GNNs in Recommender Frameworks

GNNs have recently demonstrated their superiority over conventional graph learning techniques across various sectors [9]. By continually disseminating information from interrelated items and refreshing the user vector, these networks augment user representation. Beyond nodes, GNNs can assimilate edge representations in the graph, enriching the recommendation system's performance.

Examining the graph's structure, GNNs for recommendations can be segmented into three categories: homogeneous graph recommendation, bipartite graph recommendation, and heterogeneous graph recommendation [41]. Homogeneous graphs comprise of uniform nodes and edge types, bipartite graphs include two node types and a singular edge type, whereas heterogeneous graphs encompass multiple node and edge types. As node and edge varieties increase, so does the network's intricacy [41].

2.3.1 Homogeneous Graph Recommendations

Homogeneous or single-type GNN recommendation models are defined by identical nodes and edges, streamlining operations. For instance, [22] employed GNNs for apparel suggestions by designing a style graph. Here, nodes symbolize clothing genres, with relationships between genres forming the connecting edges. Post clothing matchups, a complete outfit is conceptualized, generating a corresponding subgraph. This model subsequently promotes clothing items that best complement the selected item.

2.3.2 Bipartite Graph Recommendations

In real-world recommendation systems, the prevalent node classes are users and items, with user-item interactions typically forming a bipartite graph. Within this structure, the central challenge revolves around assimilating neighboring node data and amalgamating it with target node information [21].

2.3.3 Heterogeneous Graph Recommendations

As the name suggests, heterogeneous graphs are more intricate due to an expanded node and edge typology. Nevertheless, given their alignment with actual networks, recommendation algorithms based on heterogeneous graph neural networks often yield superior results. In such a graph, nodes can encompass item attributes alongside users and items [4]. Taking a film rating platform as an example, nodes could represent movie genres, directors, actors, and so forth. Beyond just interaction dynamics, the edge type can also encapsulate social connections [20], with the potential for multiple interaction types depending on the recommendation system's context.

2.4 Specific GNN Methods for Recommendation Systems

As mentioned earlier, GNNs have proven to be effective in various domains. By applying GNNs to recommendation systems, we can leverage their strengths in processing non-Euclidean data and their ability to understand complex graph structures [9]-[8]. The utilisation of GNN methods in the context of investor recommendation systems for startups is discussed further.

2.4.1 GraphSAGE

GraphSAGE, an innovative technique, stands out for its method of dealing with nodes within large-scale graphs. Notably, it samples a fixed-sized neighbourhood for every node and introduces a strategy employing mean, sum, or max-pooling aggregators. It also utilizes a concatenation operation for updating node information [15].

$$\text{Aggregation:} \quad n_v^{(l)} = \text{Aggregator}^l \left(h_u^{(l)}, \forall u \in N_v \right), \tag{2.1}$$

$$\text{Update:} \quad h_v^{(l+1)} = \delta \left(W^{(l)} \cdot \left(h_v^{(l)} \oplus n_v^{(l)} \right) \right), \tag{2.2}$$

Combining these unique elements, GraphSAGE facilitates the generation of embeddings for nodes, even in situations where the graph scales to millions of nodes or more. It does this by utilising the enormous amount of feature data that is linked to each node. The technique's ability to learn inductively, which enables it to create embeddings for new nodes, is its distinguishing feature. GraphSAGE can expand its application to new nodes without necessitating a thorough model retraining, by learning an aggregation function from the local neighbourhood of the nodes. The accuracy of personalised recommendations is improved by GraphSAGE's user-item similarities, which are based on local graph structures and feature data, in a variety of contexts, including recommendation systems [15].

2.4.2 GAT

Graph Attention Networks (GATs) offer an advanced approach to graph-based data analysis, distinguishing themselves through their unique attention mechanism [39]. In contrast to other models, GATs operate under the assumption that the influence of neighbours is not uniform and is not strictly determined by the graph structure. As a result, they utilize an attention mechanism to differentiate the contributions of neighbours, allowing each node to attend more to the neighbours that are more relevant to its information update.[39]

Specifically, the GAT updates the node embeddings through an aggregation operation defined as:

$$n_v^{(l)} = \sum_{j \in N_v} \alpha_{vj} h_j^{(l)}, \quad \alpha_{vj} = \frac{\exp\left(\text{Att}\left(h_v^{(l)}, h_j^{(l)}\right)\right)}{\sum_{k \in N_v} \exp\left(\text{Att}\left(h_v^{(l)}, h_k^{(l)}\right)\right)},$$

and an update operation defined as :

$$h_v^{(l+1)} = \delta\left(W^{(l)} n_v^{(l)}\right),$$

where Att$(\cdot)$ is an attention function and a typical Att$(\cdot)$ is LeakyReLU $\left(a^T \left[W^{(l)} h_v^{(l)} \oplus W^{(l)} h_j^{(l)}\right]\right)$, $W^{(l)}$ is responsible for transforming the node representations at l^{th} propagation, and a is the learnable parameter. [39]

By integrating attention mechanisms, GATs can adaptively determine the importance of neighbouring nodes, leading to more expressive and powerful node embeddings that capture the complex dependencies between nodes and their local neighbourhoods [39].

2.5 Case Studies

2.5.1 Pinterest Recommender System

2.5.1.1 Graph-Focused Convolutions

At the core of the PinSage method is the idea of graph-centric convolutions [43]. PinSage incorporates multiple convolutional layers, assimilating feature details from a node's adjacent graph environment to construct a node's embedding, pertinent to an item within Pinterest. Such features might span visual or textual details of a pin. By stacking several convolutional layers, the technique discerns the nuances of the local graph layout. Notably, the intricacy of this approach remains unchanged regardless of the size of the input graph, due to the shared parameters of these localized convolutional layers.

2.5.1.2 Problem Setup

The Pinterest platform is visualized as a bipartite graph structure. This graph entails two distinct sets of nodes: I (symbolizing pins) and C (indicating boards). Pins serve as visual pointers to online resources, such as fashion items or recipes. Users cluster these pins into boards based on thematic resonance. The mission is to derive sophisticated embeddings for pins, honed for offering pertinent recommendations [43].

2.5.1.3 Item Attributes

Each pin or item $u \in I$ is tied to continuous attributes $x_u \in R^d$. These descriptors, possibly being metadata or specific content details, might include rich textual nuances and visual elements [43].

2.5.1.4 Forward Propagation Algorithm

PinSage creates node embeddings through a localised convolution operation. In this operation, the representations of a node's neighbours are transformed using a dense neural network, and these transformed vectors are then combined using a pooling function. The local neighbourhood of a node is represented vectorially by this aggregation. The current representation of the node is combined with this aggregated neighbourhood vector before being transformed through yet another dense neural network layer. This algorithm produces a representation of the node that includes details about the node and its immediate neighbourhood in the graph [43].

Algorithm 1 Convolve [43]

```
Require: Current embedding $z_u$ for node u; set of neighbor embeddings $\{z_v | v \in N(u)\}$,
    set of neighbor weights $\alpha$; symmetric vector function $\gamma(\cdot)$
Ensure: New embedding $z_{\text{new}}$ for node u
1: $n_u \leftarrow \gamma(\{ReLU(Qh_v + q) | v \in N(u)\}, \alpha)$
2: $z_{\text{new}} \leftarrow ReLU(W \cdot \text{concat}(z_u, n_u) + w)$
3: $z_{\text{new}} \leftarrow z_{\text{new}} / ||z_{\text{new}}||_2$
```

2.5.1.5 Significance-Driven Neighbors

Within PinSage, a node's neighbouring nodes are discerned based on the top T nodes that have the highest impact on the initial node [43]. This relevance is ascertained by executing random walks originating from the initial node and tallying the L1-normalized counts of nodes encountered during the walk. Nodes with peak normalized counts are deemed to be the most impactful neighbours [43].

2.5.1.6 Layered Convolutions

By overlaying multiple convolutional layers, the method can delve deeper into the surrounding graph framework of a node. The input for the convolutions at the k^{th} layer is derived from the output of the $k-1^{th}$ layer. The foundational representations are based on the original features of the nodes [43].

2.5.1.7 Optimization Objective

PinSage utilizes an optimization strategy rooted in a max-margin loss function for its training phase. The aim is to bolster the dot product for positive pairs (embeddings of the focal item paired with a related item) while ensuring that the dot product for negative pairs (embeddings of the focal item and an uncorrelated item) lag behind the positive pair's product by a specified margin [43]. The loss function for a singular node embedding pair $(z_q, z_i) : (q, i) \in L$ is represented as:

$$J_G(z_q, z_i) = E_{n_k \sim P_n(q)} \max\{0, z_q \cdot z_{n_k} - z_q \cdot z_i + \Delta\}$$

Here, $P_n(q)$ symbolizes the distribution of negative samples for the item q, and Δ denotes the margin's hyper-parameter [43].

2.5.1.8 Multi-GPU Training with Large Mini-batches

With PinSage, each GPU processes a portion of the minibatch using the same set of parameters using a multi-tower training approach. After backward propagation, the gradients for each parameter across all GPUs are combined, and synchronous SGD is carried out in a single step. The current iteration's GPU computation and the subsequent iteration's CPU computation are carried out concurrently using a producer-consumer pattern [43].

2.5.1.9 Sampling Negative Items

The loss function approximates the edge likelihood normalisation factor by using negative sampling. Using a curriculum training scheme, "hard" negative examples (items that are somewhat related to the query item but not as related as the positive item) are added to the list of negative items for each item [43].

2.5.1.10 Generating Node Representations through MapReduce

Once the model completes its training phase, embeddings for all items are systematically produced using a MapReduce methodology, ensuring computational efficiency, and avoiding redundant calculations [43].

2.5.1.11 Prompt Nearest-Neighbor Retrievals

The node representations fashioned by PinSage can be applied across various recommendation scenarios. In numerous contexts, these representations facilitate recommendations by executing swift nearest-neighbour searches within the cultivated embedding realm [43].

2.5.1.12 Critical Analysis

PinSage distinguishes itself through the efficient application of localised graph convolutions, which enables it to record both node features and graph structures. Additionally, it uses an importance-based neighbourhood definition to generate node embeddings that focus on the most important data. Additionally, PinSage's training process is effective and scalable. It uses producer-consumer minibatch construction, multi-GPU training, and a MapReduce method to create node embeddings.

PinSage does, however, come with some difficulties. For large graphs, its use of random walks to determine node neighbourhoods can be computationally intensive. The number of convolutional layers, the size of the embeddings, and the margin parameter in the loss function are other hyperparameters that can affect how well a model performs. The strategy also needs a lot of computational power, including multiple GPUs and possibly a lot of memory. Finally, empirical validation is required, which can be a challenging and time-consuming process, to determine the efficiency of the training process and the calibre of the embeddings that result.

2.5.2 Amazon Recommender System

DAEMON is a model designed for product recommendation systems. It learns to recommend related products by leveraging the structure of a directed graph, where nodes represent products and edges represent co-purchase relationships [40]. The challenge is to generate high-quality product embeddings that encode the information present in the graph structure and the product metadata.

2.5.2.1 Directed Graph and Initial Embeddings

The model starts by representing the product relationships as a directed graph $G = (V, E)$, where V is the set of nodes (products), E is the set of edges (co-purchase relationships), and $E \subseteq V \times V$. Each product $v \in V$ is associated with an initial embedding $x_v \in R^d$, where d is the dimension of the embedding space [40]. These initial embeddings are typically learned from the product's metadata using a separate neural network.

2.5.2.2 Graph Neural Networks

To refine these initial product embeddings, DAEMON uses a variant of Graph Convolutional Networks (GCNs) . GCNs are a type of Graph Neural Network (GNN) that can generate node embeddings which effectively capture the local neighbourhood structure of the graph around each node [19].

2.5.2.3 Graph Convolution Operation

The graph convolution operation in a GCN refines the embedding of a product by aggregating the embeddings of its neighbouring products [19]. The intuition is that related products should have similar embeddings. Formally, the graph convolution operation at the l-th layer for product v is defined as [40]:

$$h_v^{(l+1)} = \sigma\left(W^{(l)}h_v^{(l)} + \frac{1}{|N(v)|}\sum_{u\in N(v)} W^{(l)}h_u^{(l)}\right)$$

where: - $h_v^{(l)}$ is the embedding of product v at the l-th layer, - $N(v)$ is the set of neighbours of product v, - $W^{(l)}$ is a weight matrix learned at the l-th layer, - σ is a non-linear activation function, such as the ReLU function.

2.5.2.4 Stacking Multiple Graph Convolutions

By stacking multiple graph convolution operations, DAEMON can capture higher-order proximity information between products. After L layers of graph convolutions, the final product embeddings $h_v^{(L)}$ are obtained [40].

2.5.2.5 Training and Optimization

Training DAEMON involves optimizing the parameters of the model to minimize a loss function, typically a contrastive loss function that encourages the embeddings of related products to be close together and the embeddings of unrelated products to be far apart in the embedding space [40]. Specifically, for a pair of related products i and j, the contrastive loss function can be defined as:

Algorithm 2 DAEMON Product Embedding Generation (i.e. Forward Pass)

```
**Require:** Product graph $G = (P, \{E_{cp} \cup E_{cv}\})$; input product features $\{X_u, \forall u \in P\}$;
Number of GNN layers $L$; weight matrices $W_l, \forall l \in \{1,2,\ldots,L\}$
**Ensure:** Source embedding $\theta_s^u$ and target embedding $\theta_t^u, \forall u \in P$
$h_s^u[0] \leftarrow X_u$; $h_t^u[0] \leftarrow X_u$, $\forall u \in P$
**For** $l = 1$ to $L$
  **For** $u \in P$
    $h_s^u[l] \leftarrow \sigma\left(\sum_{(u,v)\in E_{cp}} h_t^v[l-1]W_l\right) + \sigma\left(\sum_{(u,v)\in E_{cv}} h_s^v[l-1]W_l\right)$
    $h_t^u[l] \leftarrow \sigma\left(\sum_{(v,u)\in E_{cp}} h_s^v[l-1]W_l\right) + \sigma\left(\sum_{(v,u)\in E_{cv}} h_t^v[l-1]W_l\right)$
  **EndFor**
  $h_s^u[l] \leftarrow \frac{h_s^u[l]}{||h_s^u[l]||_2}, \forall u \in P$
  $h_t^u[l] \leftarrow \frac{h_t^u[l]}{||h_t^u[l]||_2}, \forall u \in P$
**EndFor**
$\theta_s^u \leftarrow h_s^u[L], \forall u \in P$
$\theta_t^u \leftarrow h_t^u[L], \forall u \in P$
```

$$\mathcal{L}_{ij} = -\log\left(\frac{\exp(h_i^T h_j/\tau)}{\sum_{k\in\mathcal{N}_i}\exp(h_i^T h_k/\tau)}\right)$$

where: - h_i and h_j are the embeddings of products i and j respectively, - τ is a temperature parameter that controls the concentration of the distribution, - $\mathcal{N}_i$ is a set of negative samples for product i [40].

The negative samples $\mathcal{N}_i$ were sampled uniformly at random from the set of all products, using a more sophisticated strategy that considers the graph structure and the current model predictions.

2.5.2.6 Model Complexity

The complexity of DAEMON is determined by the number of layers in the graph neural network, the dimension of the embeddings, and the size and sparsity of the product graph. The time complexity of a single graph convolution operation is $O(|E|d)$, where $|E|$ is the number of edges in the graph and d is the dimension of the embeddings. Therefore, the total time complexity of L layers of graph convolutions is $O(L|E|d)$[40].

The space complexity of DAEMON is $O((|V|+|E|)d)$, which accounts for storing the product embeddings and the edge weights. Therefore, DAEMON can be quite memory-intensive for large-scale e-commerce datasets[40].

2.5.2.7 Critical Analysis

The DAEMON model's advantages come from the way it models product relationships using a directed graph, which gives co-purchase associations an comprehensive visual representation. Additionally, it uses a Graph Neural Network approach to generate embeddings of products that effectively capture both graph structure and product metadata. DAEMON can also incorporate detailed product metadata into the embeddings by creating initial product embeddings based on the metadata.

The model, however, has some drawbacks. For example, graph convolution operations can be computationally taxing, especially for sizable e-commerce datasets. The number of graph convolution layers and the embedding dimension are two hyperparameter choices that have a negative impact on the model's performance. Furthermore, the accuracy and accessibility of the product metadata and co-purchase information are critical to the quality of the product embeddings and the recommendations that follow. Finally, DAEMON may not be transparent or easily interpretable, like many deep learning models, which makes it difficult to communicate the recommendations to users or other stakeholders.

Chapter 3

Data Preparation

Our project is deeply rooted in the crucial processes of data preprocessing and feature extraction, especially due to the considerable inconsistencies encountered in the raw data. Some critical steps in this phase include managing missing data, encoding categorical variables, feature aggregation, and text data preprocessing.

From a wide selection of 18 csv files comprising the raw data, we chose to incorporate seven into our project. These selected files - namely, organisations, investors, investments, funding rounds, people, organization descriptions, and people descriptions - were found to be most relevant to our work. The rest of the files, despite their current lack of correlation with the others and seeming lack of relevancy, were set aside rather than discarded. We recognize their potential usefulness for possible future exploration.

3.1 Investor-Startup Relation

The relationship between investors and startups was elucidated by integrating multiple datasets. Here is a detailed step-by-step process:

Investments and Investors: The process began by merging the 'investments' dataset with the 'investors' dataset. This merge was performed on the basis of the 'investor uuid', which is a common attribute in both datasets. As a result, each investment was linked to a specific investor.

Adding Funding Rounds: The merged dataset from the previous step was then joined with the 'funding rounds' dataset. This was done using the 'funding round uuid', a common attribute in the merged dataset and the 'funding rounds' dataset. After this step, each investment was associated not only with a specific investor but also with a specific funding round.

Linking Startups (Organisations): In the third step, the 'organisation uuid' was obtained from the merged dataset. This unique identifier represents the startup in which the investor invested. The dataset was then joined with the 'organisations' dataset, which contains detailed information about each startup. This step linked each investment to a specific startup and added relevant features related to the startup to the dataset.

Data Cleaning: The merged dataset now contained a lot of information, some of which might not be relevant for further analysis. Therefore, irrelevant columns were dropped from the dataset. Additionally, some columns were renamed to make the dataset easier to understand.

The resulting dataset served as the primary source of information for exploring the relationship between investors and startups. It linked each investment to an investor and a startup and included detailed information about the investor, the startup, and the funding round. This dataset provided a solid foundation for subsequent analyses, such as feature extraction for investors.

3.2 Investor Features

Investor features are divided into preferences and attributes. Preferences indicate an investor's tendencies towards specific categories, locations, or rounds, whereas attributes describe the investor's properties, including roles and types.

1. **Investor Preferences:**

    (a) **Category Preferences:** Categories and groupings associated with startups were separated from strings. Investments in each category and group were tallied and normalized to represent each investor's proportional investments. These were combined into a single table.

    (b) **Geographical Preferences:** Country and region preferences were calculated by tallying investments in each area and then normalizing these counts. These were merged to form a unified geographical preference table.

    (c) **Investment Round Preferences:** The number of investments in each round was counted and normalized. All preference tables were combined for a holistic view.

2. **Investor Attributes:**
   (a) **Roles and Types:** Investor roles and types were one-hot encoded, creating binary columns for each unique role and type.
   (b) **Category and Location:** 'Category_list', 'category_groups_list', 'Country', and 'Region' columns were one-hot encoded, generating binary columns for each unique category, group, country, and region.
   (c) **Investment Count:** Missing values in the 'investment_count' column were replaced with the column's mean, followed by normalization.
   (d) **Textual Information:** Information from 'organisation descriptions' and 'people descriptions' columns was converted into a numerical format using the Sentence Transformer, which is built on the BERT model. This process transformed the textual data into 768-dimensional embeddings that capture the semantic essence of the descriptions. Subsequently, these embeddings were normalized to ensure consistent feature scaling and enhance model performance.

3.3 Co-investors

The process of identifying co-investors involves identifying investors who have jointly invested in the same funding rounds. Here are the steps followed in the provided code:

Data Joining: The 'investments' dataset was joined with itself based on the 'funding round uuid'. This attribute serves as an indicator of a funding round in which multiple investors might have participated together. The resulting joined dataset contains pairs of investors who co-invested in the same funding round.

Frequency Calculation: The frequency of co-investments for each pair of investors was calculated using a group by operation. This operation groups the dataset by investor pairs and counts the number of occurrences of each pair, providing a measure of how frequently each pair of investors co-invested.

Normalization: The frequency counts were then normalized. Normalization is the process of scaling the data to a standard range, typically 0 to 1. This step is important as it allows for fair comparison between different pairs of investors, as it accounts for pairs that might appear more frequently just because those investors are more active.

Data Cleaning: After normalization, there could be duplicate columns or rows, arising from the self join operation performed earlier. These duplicates were removed from the dataset. Moreover, pairs of investors could appear in two forms: (Investor A, Investor B) and (Investor B, Investor A). These are essentially the same pair and could be considered duplicates. These alternate pairs were also identified and removed from the dataset.

The resulting dataset provides a clean and normalized representation of co-investments between pairs of investors. Each row in this dataset represents a pair of investors who have co-invested in one or more funding rounds, and the frequency of their co-investments is given by the normalized frequency count. This dataset can be used to analyze patterns of co-investment, which can provide insights into how investors collaborate in funding rounds.

3.4 Lead Investors

Identifying lead investors involves pinpointing those investors who often take the helm in investment rounds. The process for identifying these investors is as follows:

Data Separation: The 'lead investor' column in the primary dataset was initially a single string containing multiple lead investors. This string was split into multiple columns, each representing a different lead investor.

Lead Investor Identification: For each investor, the number of times they were a lead investor was calculated. This was done by traversing through the 'investor' column and the split 'lead investor' columns, and counting the frequency of each investor appearing as a lead investor.

Frequency Normalization: The frequency counts of lead investments were then normalized. Normalization scales the data to a standard range, typically 0 to 1, facilitating fair comparison between different investors.

Data Cleaning: After normalization, any potential duplicate columns or rows were removed from the dataset. Additionally, there could be investor pairs appearing in two forms: (Investor A, Investor B) and (Investor B, Investor A). Since these are essentially the same pair, they were considered duplicates and removed from the dataset.

The final dataset provides a comprehensive view of the lead investor roles, specifying for each investor how often they acted as a lead investor. This information is crucial, as lead investors often play a pivotal role in deciding the success of a funding round and influence other investors' decisions.

Other than data pre-processing steps mentioned above, feature engineering was performed to create informative startup features, encompassing steps such as data acquisition, feature selection, data cleaning, one-hot encoding for categorical data, and geographical data aggregation. To examine competitive behavior, rivalry scores were calculated for investor pairs, quantifying 'non-overlapping' investment activity for each unique year and 'category_region' combination. This enriched the understanding of investor competition trends over time. Data mapping and cleaning processes were also employed, integrating and integer encoding unique identifiers, and generating edge indices that represented investor-startup relationships. This resulted in a consolidated dataset ready for model training and evaluation. Lastly, dimensionality reduction was applied to the dataset using variance thresholding, mean centering, and Principal Component Analysis (PCA), creating dimensionality-reduced investor and startup feature matrices. Each of these steps was crucial in preparing the data for the final recommendation models. For further details on each process mentioned here, please refer to the corresponding paper by the project member.

Chapter 4

Exploratory Data Analysis

Exploratory Data Analysis (EDA) provides the foundational understanding of the underlying structure and patterns of a dataset, setting the stage for subsequent data-driven inquiries. It offers insights into the relationships, anomalies, and unique characteristics within the data. This chapter dives deep into the dataset, highlighting its key features and shedding light on their interrelationships.

4.1 Temporal Analysis of Startup Categories

The dynamism of the startup landscape is evident when analyzing the investment trends across various categories over time.

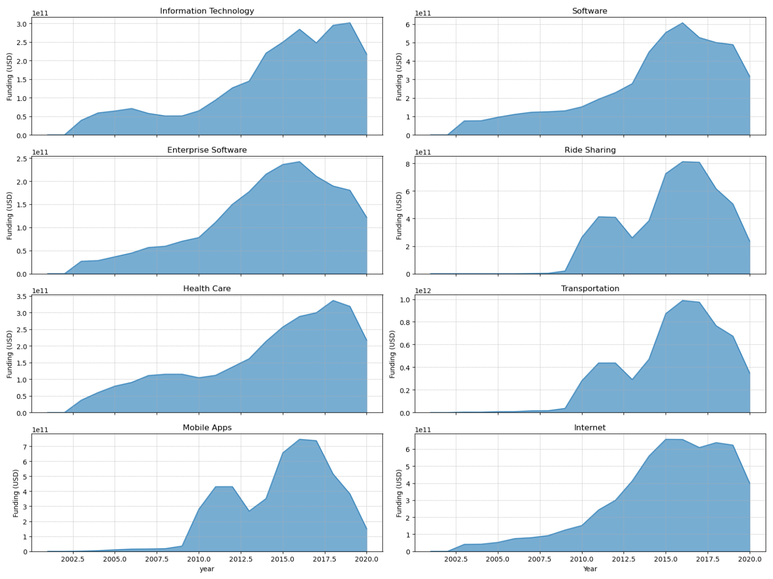


Figure 4.1: Temporal Distribution of Investments across Startup Categories

Figure 4.1 showcases how investments in the top 10 startup categories have evolved. These categories were cherry-picked based on cumulative funding and the frequency with which they garnered investments.

As inferred from Figure 4.1, the popularity of categories follows a cyclical trend. Peaks and troughs in investments mirror the shifting preferences and market dynamics.

4.2 Investment Patterns Across Rounds

Analyzing the frequency of investments across various rounds provides insights into the most favored stages of investment. Figure 4.2 illustrates the number of investments segregated by their respective rounds.

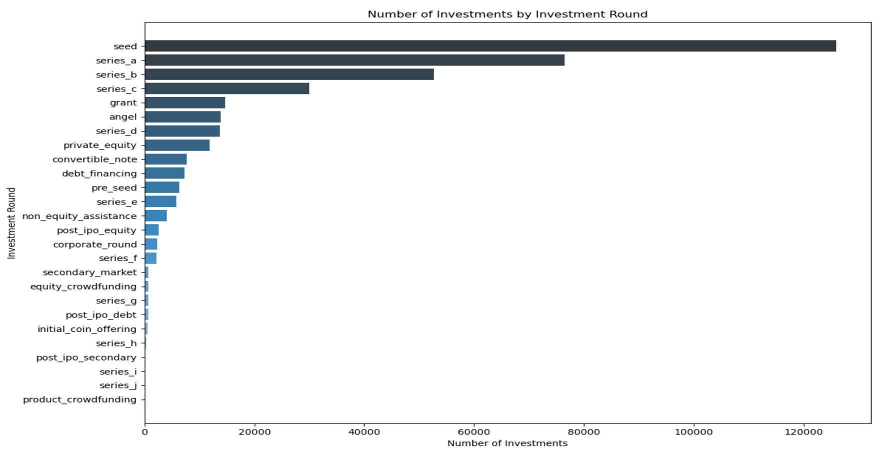


Figure 4.2: Investments Distribution Across Different Rounds

From Figure 4.2, it's clear that seed funding rounds attract the highest number of investments, followed by Series A, B, C, and so forth. This trend highlights the risk appetite and strategic focus of investors.

4.3 Average Funding Across Investment Rounds

To understand the monetary emphasis across different rounds, Figure 4.3 delineates the average amount raised during each investment phase.

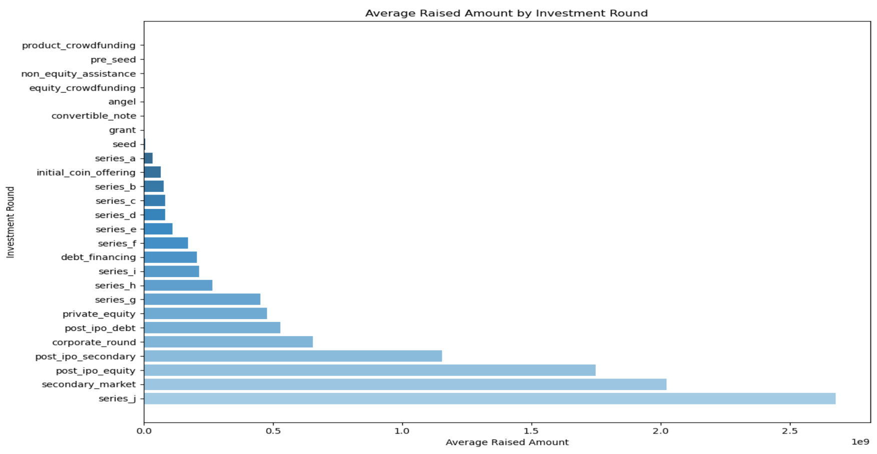


Figure 4.3: Average Raised Amount Across Investment Rounds

As observed in Figure 4.3, while the number of investments might be higher in the initial rounds, the quantum of funds raised peaks in the later stages, particularly during Series J rounds.

4.4 Geographical Consistencies in Investments

Investor behavior is also influenced by geographical considerations. The subsequent figures explore whether investments are localized (both country-wise and region-wise) or spread across borders.

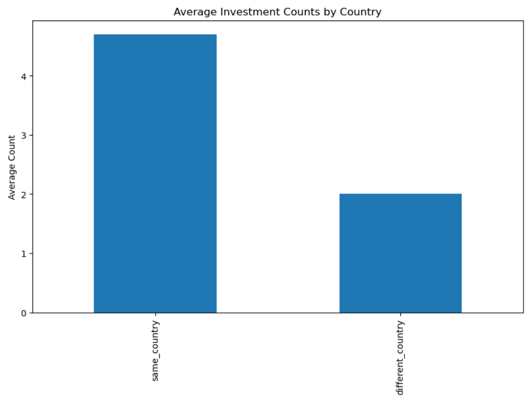


Figure 4.4: Country-based Investment Preferences

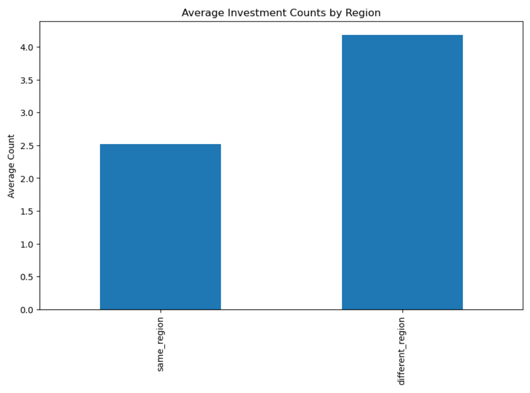


Figure 4.5: Region-based Investment Preferences

Figures 4.4 and 4.5 suggest that while investors exhibit a propensity to invest within their own countries, regional preferences are more nuanced. The inclination to invest within the same country is starkly evident, reinforcing the significance of local networks and knowledge in investment decisions.

4.5 Correlation Analysis of Numerical Features

A deeper dive into the relationships between various numerical features is facilitated by Figure 4.6, which presents a correlation matrix of features including 'investment_count', 'raised_amount', 'investor_count', 'num_funding_rounds', and 'total_funding_usd'.

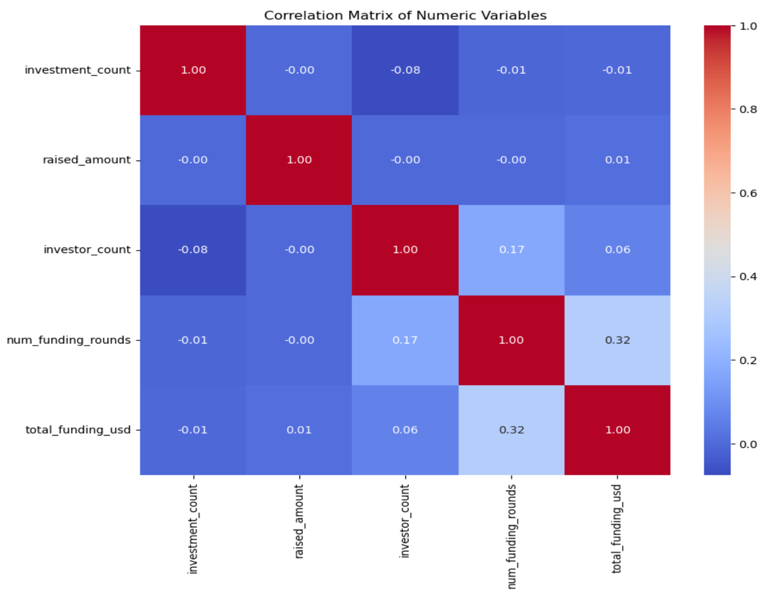


Figure 4.6: Correlation Among Key Numerical Features

The low correlation coefficients, as seen in Figure 4.6, underline the independence of these features, suggesting that they each encapsulate unique information about the startups.

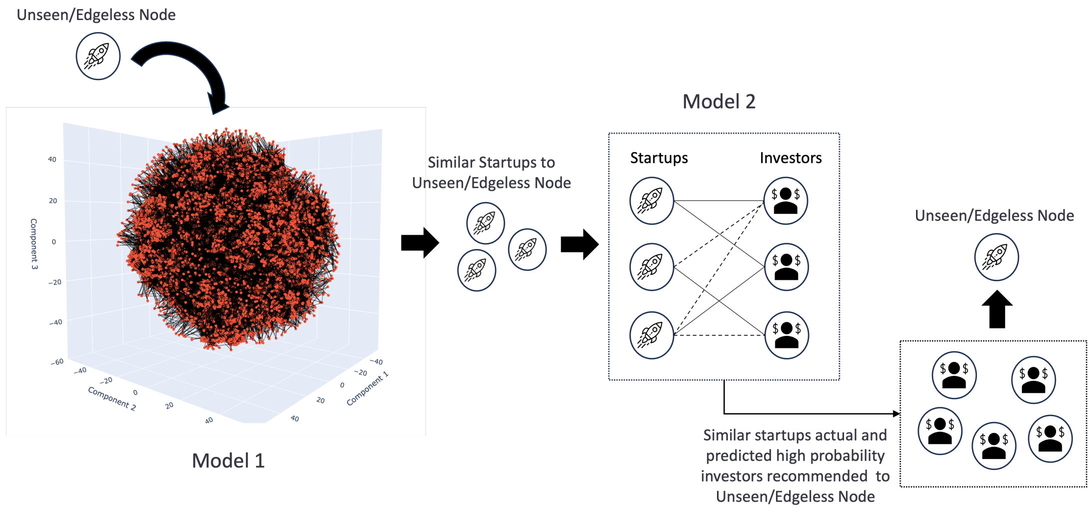
The EDA conducted provides a multi-faceted view of the dataset, empowering subsequent modeling endeavors with robust insights. The patterns unraveled and the relationships deciphered were crucial for the next phases of this research.

Chapter 5

Methodology

5.1 Model 1: Knn + Graphsage

5.1.1 Problem Settings



Unseen/Edgeless Node
Model 1
Similar Startups to Unseen/Edgeless Node
Model 2
Startups
Investors
Unseen/Edgeless Node
Similar startups actual and predicted high probability investors recommended to Unseen/Edgeless Node
Component 1
Component 2
Component 3

Figure 5.1: KNNGraph-Graphsage - Framework

Our model is designed to make investment recommendations for startups, drawing upon the interaction between a K-Nearest Neighbors (KNN) graph and a Graph Neural Network (GNN).

The model's first phase involves constructing a KNN graph using the features of the startups. This allows us to generate unique embeddings for each startup and identify clusters of similar startups. This clustering enables a nuanced understanding of the startups' respective positions within the broader ecosystem. When a new startup is introduced to the model, the KNN graph allows us to identify startups that are similar to the newcomer based on their features.

In the second phase, the model uses a GNN to predict links, focusing on learning the relationships between investors and startups within a heterogeneous graph structure. The graph comprises nodes representing investors and startups, with edges denoting investment relationships. The goal is to predict whether an investor would be likely to invest in a particular startup based on the investor's features, the startup's features, and the graph's structure.

The model's input consists of the feature vectors for investors and startups, represented as x_{inv} and x_{st} respectively. These feature vectors undergo linear transformations via separate linear layers to be embedded into a common space, resulting in investor and startup embeddings, x'inv and x'st, with a common dimension D.

These embeddings are then passed through a GraphSAGE-based GNN. The GNN takes the node embeddings and the graph's edge index as inputs and outputs updated node embeddings x''_{inv} and x''_{st}.

Once the updated node embeddings are obtained, a classifier is used to predict whether an investor would invest in a startup. The score y_{ij}, calculated for each investor i and startup j, is given by the sum of the products of corresponding dimensions of the investor's and startup's embeddings.

The model's strength lies in the synergy between these two phases. When a new startup is introduced to the model, it is first run through the KNN graph model to identify similar startups based on their cosine similarity. In the second phase, the model uses the established links for these similar startups to recommend the investors most likely to invest in the new startup. The model thus serves as a comprehensive tool for exploring and predicting investor-startup relationships in a graph-structured dataset.

5.1.2 Model Architecture

The proposed model architecture leverages the capabilities of Graph Neural Networks (GNNs) to make predictions on a heterogeneous graph. The graph consists of two types of nodes, representing startups and investors. The model architecture encompasses a multilayer Graph Convolutional Network (GCN), implemented using the GraphSAGE algorithm, along with a Classifier module and a Model module that stitches the entire pipeline together.

The first component, the GNN, is a subclass that employs GraphSAGE Convolutions (SAGEConv). GraphSAGE, an abbreviation for Graph Sample and Aggregate, employs a neighborhood aggregation scheme that derives embeddings by sampling and aggregating features from a node's local neighborhood. The GNN module features two layers of SAGEConv, with the inclusion of batch normalization (BatchNorm1d) after the initial layer to expedite learning and enforce regularization. The forward method of the GNN class undertakes the computation of graph convolutions, applies batch normalization, and then employs a rectified linear unit (ReLU) activation function to introduce non-linearity.

The Classifier module is constructed to compute the similarity between the nodes of startups and investors. In the forward method, it extracts the feature vectors for the connected startup and investor nodes using the edge_label_index tensor. The method then performs element-wise multiplication (Hadamard product) of the feature vectors of the startup and investor nodes, summing them along the final dimension. This process effectively computes the dot product between the feature vectors, which serves as a score for link prediction.

The Model module integrates the GNN and Classifier modules, executing feature transformation, graph convolutions, and eventually, link prediction. The init method initializes the GNN and Classifier modules, and also establishes Linear layers for transforming the input features of the startup and investor nodes. It also creates Embedding layers for the node indices. These layers are employed to generate initial feature vectors for each node type, which are subsequently fed into the GNN for further processing.

The forward method of the Model module begins by transforming the input features and node indices of the startups and investors into initial feature vectors. These feature vectors are then passed through the GNN module, which applies graph convolutions to extract high-level features that encapsulate the local graph structures around each node. The Classifier module then utilizes these high-level features to compute the similarity scores between each pair of connected nodes, returning these as predictions.

The Model module also houses a topN function that retrieves the 'n' highest-ranked investors most related to a specific startup. It computes the dot product between the feature vector of the startup and all investor feature vectors, which represents the similarity scores. The torch.topk function is then used to retrieve the 'n' highest scores, corresponding to the top 'n' most related investors.

5.1.3 Training and Validation

The training and validation stages for the model are divided into several phases:

1. **Data Preparation:** The original data are loaded and partitioned into train, validation, and test sets. A Heterogeneous graph data structure (HeteroData from PyTorch Geometric) is created, allowing different node types and edges to co-exist in a single graph. Two types of nodes are defined: ”inv” (representing investors) and ”org” (representing startups). The edge indices between the nodes are prepared.

2. **Neighbor Sampling:** Neighbor sampling is used to mitigate the computational complexity of GNNs. This is implemented using the NeighborLoader from PyTorch Geometric, which can perform mini-batch training of GNNs and allows customization of the number of neighbors sampled for each node.

3. **Model Initialization:** The model is initialized and an Adam optimizer is used for the optimization of the model's parameters. The loss function is defined as Binary Cross-Entropy with Logits (BCEWithLogitsLoss). Given the class imbalance problem, the positive weight in BCEWithLogitsLoss is set as the average negative-to-positive ratio.

4. **Training Loop:** During training, the model parameters are updated using back-propagation and the Adam optimizer. The model's performance is evaluated based on the average training loss and the macro F1 score.

5. **Validation Loop:** After each epoch of training, the model is evaluated on the validation set. The validation loss is compared to the best validation loss observed so far. If the current validation loss is lower, the best validation loss is updated, and the patience counter for early stopping is reset.

6. **Performance Monitoring:** Throughout the training and validation process, the average training and validation losses are recorded, along with the macro F1 scores. These metrics are used to monitor the model's performance over

5.1.4 Recommendation Process for New Startups

The recommendation process for new startups is a two-step process that leverages both the K-Nearest Neighbors (KNN) graph [5] and the Graph Neural Network (GNN).

1. **KNN Graph:** When a new startup is introduced to the model, it first goes through the KNN graph. Using the feature vectors of the new startup, the KNN graph identifies the most similar startups that are already present in the dataset. The Cosine similarity measure is used to determine the similarity between startups. By identifying the most similar startups, the model can better understand the position of the new startup within the overall landscape.

2. **GNN-based Recommendation:** After identifying the similar startups using the KNN graph, the model leverages the learned relationships in the GNN for recommendation. Specifically, the model looks at the investors that have invested in the similar startups and calculates the investment probabilities for these investors investing in the new startup. The top 10 investors with the highest probabilities are then recommended as potential investors for the new startup.

Through this two-step process, the model is able to provide insightful recommendations for new startups. It not only considers the features of the startups and investors, but also takes into account the structural information in the graph, thereby offering a more comprehensive and reliable recommendation. Below a pseudo code is given on how inference works on new startup.

Algorithm 3 Recommend Investors for a Startup

```
Require: Startup Features, Knowledge graph data, Model, Device configuration.
Ensure: List of recommended investors for the startup.
Extract relevant features and convert them into a tensor.
Increment the node ID.
Add the new startup node to the knowledge graph, obtaining nearest neighbors.
For each neighbor:
    Recommend potential investors.
End For
```

5.2 Model 2: KNN-Graph Augmented Edge Prediction for Node Linkages

5.2.1 Problem Settings

Our model receives investor and organization features as inputs. For a dataset containing N_{inv} investor features and N_{org} organization features, these inputs x_0 and x_1 are matrices of dimensions $N_{\text{inv}} \times D$ and $N_{\text{org}} \times D$ respectively, where D is the feature dimension.

These features are embedded separately, yielding representations x'_0 and x'_1 of identical dimensions. These embeddings are then concatenated and enhanced via a Graph Attention Network (GAT) with L layers, producing a feature-rich representation x'' that includes edge features and node connectivity.

We then decompose these enriched representations into x''_0 and x''_1, and pass them through another pair of linear transformations, obtaining final representations x'''_0 and x'''_1.

We predict the link between an investor and an organization by computing the dot product between x'''_0 and x'''_1. This operation yields a similarity matrix of size $N_{\text{inv}} \times N_{\text{org}}$, the output y of our model.

We then express the score y_{ij}, given an investor feature index $i \in [1, N_{\text{inv}}]$ and an organization feature index $j \in [1, N_{\text{org}}]$, as:

$$y_{ij} = \langle x'''_{0_i}, x'''_{1_j} \rangle \tag{5.1}$$

Here, $\langle .,. \rangle$ denotes the dot product operation.

5.2.2 Model Architecture

Our model integrates Graph Attention Networks (GAT) into a fully connected architecture. The model receives a sequence of investor and organization features, processing them through three main stages:

1. **Feature Embeddings**: Each feature of the investor and the organization is projected into a shared feature space. This is achieved through separate embedding layers, where each layer is implemented as a linear transformation (`nn.Linear()`) followed by dropout regularization (`nn.Dropout()`) to prevent overfitting. The investor and organization features are thereby transformed into x'_0 and x'_1, respectively.

2. **Graph Convolution Networks (GCN)**: The embeddings x_0' and x_1' are then concatenated and passed through a GAT layer. The GAT layer operates on the joint embeddings and uses the graph structure of the data (edge indices and edge attributes) to propagate and transform the information. This layer outputs the enriched embeddings x'', which capture both the original features and the structural information.

3. **Readout and Dot Product**: The enhanced embeddings x'' are separated back into x_0'' and x_1''. These are passed through another pair of linear transformations to obtain the final representations, x_0''' and x_1'''. A dot product operation between these final representations gives us the compatibility score matrix, which forms the output of the model. Each element of this matrix represents the compatibility score between a specific investor and an organization.

This architecture accommodates both graph-structured data and standard features effectively. It offers the flexibility to accommodate additional layers or different types of GATs, facilitating further model exploration and performance improvement.

5.2.3 Training and Validation Procedure

The training and validation procedures for the model can be divided into several stages:

Data Preparation: The initial phase involves loading the original data and splitting it into training, validation, and test sets. The 5% of edges in each training, validation, and test sets is marked as unknown so that the model can predict on nodes with no edges. A Heterogeneous graph data structure (HeteroData) from PyTorch Geometric is created, allowing different types of nodes and edges to coexist within a single graph. The edge indices between the nodes are also prepared.

KNN Graph Creation: Before feeding the data into the model, a K-Nearest Neighbors (KNN) graph is created for the startup nodes. This process ensures that an edge exists between similar startups, a crucial requirement since unknown nodes won't have any edges for message passing. Hence, the graph sent into the model contains two edge types: one between startups and investors, and another between startups.

Neighbor Sampling: To reduce the computational complexity of Graph Neural Networks (GNNs), neighbor sampling is employed. This is implemented using the NeighborLoader from PyTorch Geometric, which supports mini-batch training of GNNs and allows customization of the number of neighbors to sample for each node.

Model Initialization: The model is initialized, and the Adam optimizer is used for the optimization of the model's parameters. The Binary Cross-Entropy with Logits (BCEWithLogitsLoss) loss function is defined to handle the class imbalance problem. The positive weight in the BCEWithLogitsLoss is set as the average negative-to-positive ratio.

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot \log(p_i) + (1-y_i)\cdot \log(1-p_i)] \tag{4.1}$$

where N is the total number of samples, y_i is the true class of sample i, and p_i is the model's predicted probability for the positive class for sample i.

Training Loop: During training, the model parameters are updated using backpropagation and the Adam optimizer. The model's performance is assessed based on the average training loss and the F1 score, given by:

$$\text{F1 score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}[37] \tag{4.2}$$

Validation Loop: After each epoch of training, the model's performance is evaluated on the validation set. If the current validation loss is lower than the previously recorded best validation loss, the best validation loss is updated and the patience counter for early stopping is reset.

Performance Monitoring: Throughout the training and validation process, the average training and validation losses are recorded, as well as the F1 scores, to monitor the performance of the model over time. Early stopping is employed to prevent model from over-fitting.

5.2.4 Recommendation Process for New Startups

The recommendation strategy outlined below is engineered to discern potential collaborations between startups and prospective investors, ensuring the fidelity of predictions to real-world investor-startup interactions.

1. **Data Extraction:** Key features are extracted from both investors and test startups. Concurrently, relational data, represented by edge indices, is gathered to construct a detailed graph highlighting inter-entity connections.

2. **Batch Organization and Graphical Adjustments:** Here, the accumulated data is divided into discernible batches for streamlined processing. Each investor-oriented batch is then subjected to several graph transformations, which include:

- Isolating a 1-hop subgraph with the investor at its epicenter.
- Omitting startups already present in the training set.
- Generating a unique subgraph that comprises both the investor and the chosen startups.

3. **Projection and Graph Merging:** This pivotal phase mandates the amalgamation of different graph components into a cohesive structure. This enriched structure, augmented by connections drawn from the k-nearest neighbors, becomes the foundation upon which the model's predictions are formulated.

The model's predictions are compiled, and to ascertain their accuracy, metrics like the F1 Score, Hit-rate, Precision@k, Recall@k and Loss Value are closely examined.

In essence, this streamlined approach synergizes various phases, all converging to identify potential ties between startups and investors. Below a pseudo code is given on how inference works.

Algorithm 4 Recommend Investors for Startups

```
Require: Pre-trained model, dataset, device setup, and parameters (batch_size, PATH, etc.)
Ensure: List of recommended investors for the given startup.
Procedure Recommend(provided_startup_features):
Load model, data, and initialize parameters.
Parse required arguments using parse_args().
Initialize empty list Y_pred for predictions.
For each batch of investors:
    Extract features and compute K-hop subgraph.
    Compute nearest neighbors for the new startups.
    Create a combined graph with known and new edges.
    Predict using the model for the current batch.
    Append predictions to Y_pred.
End For
Process predictions and filter based on a threshold.
Sort and select the top N investors.
Return the recommended investors.
End Procedure
```

Chapter 6

Experiments

This chapter dives into the experiments undertaken within this project. The primary focus lies on hyperparameter optimization and the inference mechanisms devised for new startups.

6.1 Model 1

6.1.1 Hyperparameter Optimization with Optuna

To find the best hyperparameters for model, Optuna—a dedicated Python library for hyperparameter optimization—was leveraged. This systematic approach to hyperparameter tuning not only ensures enhanced model accuracy but also aids in understanding the sensitivity of the model to different parameter values [1].

6.1.2 Experiment Settings

The hyperparameter search space, delineated by Optuna, encompassed three pivotal parameters:

Learning Rate (lr): The rate of adjustment made to model weights during training [36]. It's pivotal in ensuring the model neither overshoots nor gets stuck in local minima. For this project, a log-uniform distribution was chosen, ranging from 1×10^{-5} up to 1×10^{-2}.

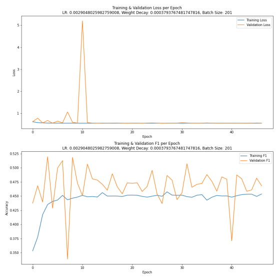
Weight Decay: Often used as a regularization technique, weight decay constrains the magnitude of model weights to prevent overfitting [27]. Its value was sampled from a log-uniform distribution between 1×10^{-5} and 1×10^{-1}.

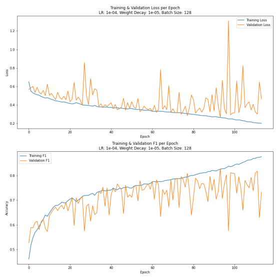
Batch Size: Refers to the number of training samples used in one iteration. It directly influences the granularity and stability of gradient updates [18]. For this experiment, batch sizes were considered from a lower limit of 32 up to a ceiling of 256.

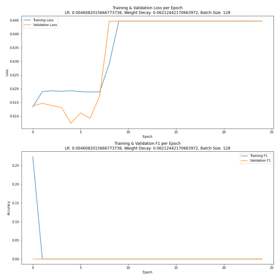
Due to constraints in computational resources, only 7 trials were conducted. The high training time meant that exhaustive hyperparameter tuning was challenging.

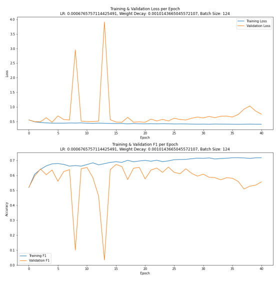
6.1.3 Hyperparameter Optimization Results

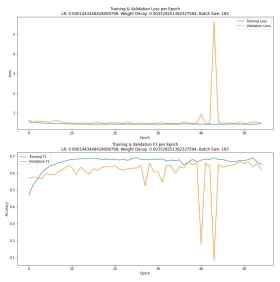
Throughout the optimization phase, Optuna meticulously probed various permutations of the hyperparameters, its goal being the discovery of the combination that minimized the objective function.

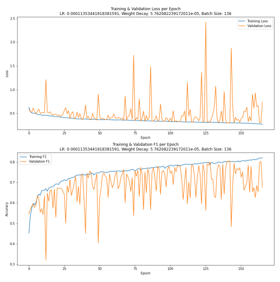

(a) Trial 1


(b) Trial 2


(c) Trial 3


(d) Trial 4


(e) Trial 5


(f) Trial 6

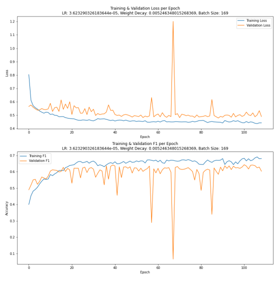

(g) Trial 7

Figure 6.1: Training and Validation Performance Metrics: F1 Score and Loss Over Epochs

The outcomes of hyperparameter tuning using Optuna are captured in the table below. Each row represents a trial with a unique combination of learning rate, weight decay, and batch size. The corresponding F1 score and loss values offer insights into the model's performance for each combination.

Trial	**LR**	**Weight Decay**	**Batch Size**	**F1**	**Loss**
Trial 1	0.0029	0.00038	201	0.4711	0.5590
Trial 2	1×10^{-4}	1×10^{-5}	128	0.7398	0.4579
Trial 3	0.0046	0.0621	128	0.6451	0.6448
Trial 4	0.00068	0.00101	124	0.5413	0.7755
Trial 5	0.00014	0.00353	183	0.6241	0.4720
Trial 6	0.00011	5.76×10^{-5}	136	0.6690	0.7536
Trial 7	3.6233×10^{-5}	0.00525	169	0.6079	0.4915

Table 6.1: Hyperparameter Tuning Results

An examination of the presented plots and results highlights that the model from Trial 2 outperforms its counterparts, boasting a superior F1 score and a lower loss. While there are observable inconsistencies in the validation plots across epochs, further refinement of the model could potentially address these issues. However, due to time and resource constraints, we elected the model from Trial 2 as our optimal choice for subsequent inferences.

6.1.4 Inference on New Startups

Upon training completion, the model was tasked with suggesting potential investors for new startups. The startup-investor graph was loaded, and the model was initialized, updated with pre-trained weights, and set to evaluation mode. This setup, adaptable to both GPU and CPU, incorporated essential investor and startup ID mappings. Additionally, a KNN Graph of startups, based on features and descriptions, was utilized, with unrepresented startups and their attributes loaded for comprehensive inference.

The model excels at producing a prioritized list of top investors for a specific startup, with rankings shaped by the embeddings to reflect investor-startup congruence. The recommendation algorithm extracts features of the startup, integrates it into the data graph, and identifies its nearest neighbors. Recommendations from these neighbors are aggregated into a unique list, which is then compared with the startup's actual investors to evaluate the model's accuracy.

6.1.5 Evaluation Metrics

Post-recommendation, various metrics are employed to gauge the quality of the recommendations:

- **Precision@K**: Measures the fraction of recommended investors that are relevant [23].
- **Recall@K**: Assesses the fraction of relevant investors that are truly recommended [23].
- **Hit Rate**: Indicates if any of the actual investors were captured in the recommendations.
- **Mean Average Precision (MAP)**: Provides an aggregate measure of recommendation quality across all startups [23].

Finally, the average values for the aforementioned metrics are computed, offering a holistic view of the model's recommendation efficacy.

6.1.6 Evaluation Results

Upon inference, the model's recommendations were subjected to rigorous assessment, validated against a slew of metrics. Post-inference, the model's recommendations were assessed using various metrics. The combined results for random 100 and 500 startups are presented in the table below:

Table 6.2: Evaluation Results for 100 and 500 Startups at K = 10

Metric	**100 Startups**	**500 Startups**
Average Precision@K	0.001	0.0006
Average Recall@K	0.005	0.0014385964912280703
Average Hit Rate	0.02	0.026
Mean Average Precision (MAP)	0.349	0.348

Practical evaluation showcased varied results. For ”11 Biomics”, the model adeptly identified ”SOSV” as a mutual investor. Conversely, in the case of ”2nd Address”, while the model recognized ”Foundation Capital” as a potential investor, it missed several others like ”Pierre Lamond” and ”GV”. And for most eventhough potential investors were the ones which invested in those type of startups they weren't from the actual. These discrepancies show the model's uneven performance, suggesting potential complexities in the data that may not be fully captured by the current configuration.

The empirical outcomes shed light on the model's adeptness at investor recommendation across diverse startup groupings. The MAP metric, being particularly enlightening, reveals that the model consistently delivers a commendable quality of recommendations, irrespective of the size of the startup group.

6.1.7 Limitations

While the model showcased commendable proficiency in making investor-startup recommendations, it is important to underscore its limitations, which might guide future refinements:

1. **Inference Inconsistency**: The model's recommendations vary across startups, missing key investors in certain instances.
2. **Dependency on Prior Interactions**: Its efficacy leans heavily on existing interactions between startups and investors, limiting accuracy for startups without any investments.
3. **Scalability Issues**: As data grows, the model's current configuration might require enhancements to maintain its efficacy.

In summary, our model performs effectively when there's at least one investment interaction between a startup and an investor. It has a good accuracy in predicting potential investment links, but faces challenges during the inference stage. To address this, we used a KNN approach to recommend investors based on similarities with other startups.

6.2 Model 2

The second model presented a mixed bag of outcomes when evaluated against key performance indicators. Here's a closer examination:

- **F1 Score Stability:** An intriguing observation was the unvarying **F1 Score**, persistently recorded at 0.0022 for both $k = 10$ and $k = 20$.

- **Hit Rate Progression:** The **Hit Rate** provided decent results. With an increase from 16.67% at $k = 10$ to 35.42% at $k = 20$. This indicates that as we increase the number of recommendations, startups are more likely to encounter an investor with whom they have genuinely interacted.

- **Average Precision Nuances:** The **Average Precision** showcased a modest rise, transitioning from 1.67% at $k = 10$ to 1.98% at $k = 20$. This elevation signifies the model's ability to prioritize genuine recommendations as the list expands.

- **Recommendation Analysis:** Checking the model results revealed mixed outcomes. For ”Kids on 45th”, the model's acumen shone through, rightfully pinpointing ”Sesame Street” as a shared investor. However, this was contrasted by the model's suggestions for ”Scout”, where none aligned with the startup's real investors. These variances underscore the model's inconsistent performance and hint at intricate patterns potentially eluding the current model.

To conclude, while Model 2 flaunts certain strengths, especially concerning the hit rate, there are areas for refinement. For an in-depth analysis of these outcomes, I recommend referring to the paper authored by the project team.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Owing to the complexities inherent in linking startups with potential investors, the use of conventional methods to understand their interactions has not been very effective. This study clarified the underlying dynamics of investor-startup relationships using computational investigation. While using computational investigation turned out to be difficult, the final results were fruitful.

From the outset, the importance of understanding and predicting potential partnerships between startups and investors was clear. As the global entrepreneurial landscape becomes increasingly competitive, the capability to accurately match startups with the right investors can be the difference between success and failure.

Model 1 The subsequent results from Model 1, especially the MAP metric, stood testament to the model's capability. It consistently delivered quality recommendations across diverse startup groupings, underscoring the importance of hyperparameter tuning.

Model 2, on the other hand, provided a foundational understanding of this domain. Although it exhibited consistent F1 Scores, its true strength emerged in the realm of the Hit Rate. The model's ability to better capture genuine investor-startup interactions with broader recommendation scopes illuminated its potential. While the results were promising, it also highlighted the need for further refinement and optimization.

Beyond the metrics and models, this research highlighted the immense potential of computational techniques to navigate the investor-startup ecosystem. The ability to recommend potential investors systematically and accurately to startups can revolutionize the way entrepreneurial ventures seek and secure funding. It can streamline the process, ensuring that startups are matched with investors who not only provide capital but also align with their vision, ethos, and growth trajectory.

7.2 Future Work

Building upon the foundation laid by this research, several promising avenues can be explored to enhance the efficiency, accuracy, and applicability of the investor-startup recommendation system:

- **Investor Clustering**: Grouping similar investors into clusters can offer computational efficiency. By targeting clusters instead of individual investors, the model can make quicker yet relevant recommendations.
- **Investor Selection Metrics**: Introducing advanced metrics or functions can help determine the superiority of one investor graph over another. This can aid in refining the recommendation process, ensuring that only the most relevant investors are considered.
- **Co-investor and Lead Investor Integration**: Incorporating insights from co-investor and lead investor analyses can further refine recommendations. Recognizing patterns of co-investments and lead roles can provide a deeper understanding of investor dynamics, assisting in making more informed recommendations. Leveraging co-investor and lead investor data can also help in reducing the sparsity inherent in large datasets. By understanding frequent collaborations and leadership roles, the system can impute or predict missing data points with greater confidence.
- **Hybrid Models**: Incorporating a mix of classification tools alongside the existing models can offer a more holistic recommendation system. By leveraging the strengths of various algorithms, the model can potentially achieve higher accuracy and relevance in its predictions.

In conclusion, the investor-startup ecosystem, with its myriad of interactions and potentialities, offers a rich ground for machine learning exploration. The strides made in this research, coupled with the proposed future directions, pave the way for a more informed and nuanced understanding of this dynamic landscape. The ultimate vision remains to seamlessly bridge the gap between startups in need and the investors best suited to propel them forward.

Bibliography

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. *CoRR*, abs/1907.10902, 2019.

[2] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018. Version: v3.

[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013. Version: v3.

[4] Mengru Chen, Chao Huang, Lianghao Xia, Wei Wei, Yong Xu, and Ronghua Luo. Heterogeneous graph contrastive learning for recommendation. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, page To be specified, Singapore, Singapore, 2023. ACM.

[5] Padraig Cunningham and Sarah Jane Delany. k-nearest neighbour classifiers: 2nd edition (with python examples). *CoRR*, abs/2004.04523, 2020.

[6] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, volume 29, 2016. NIPS 2016 final revision.

[7] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. 7 2016.

[8] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, and Yong Li. A survey of graph

neural networks for recommender systems: Challenges, methods, and directions. *ACM Trans. Rec. Sys.*, 1(1):49, September 2022. Code repositories available at: https://github.com/tsinghua-fib-lab/GNN-Recommender-Systems.

[9] Jian Gao, Jianshe Wu, Xin Zhang, Ying Li, Chunlei Han, and Chubing Guo. Partition and learned clustering with joined-training: Active learning of gnns on large-scale graph. *Knowledge-Based Systems*, 258:110050, 2022.

[10] Paul Gompers, Anna Kovner, Josh Lerner, David Scharfstein, and Morgan Hall. Nber working paper series skill vs. luck in entrepreneurship and venture capital: Evidence from serial entrepreneurs skill vs. luck in entrepreneurship and venture capital: Evidence from serial entrepreneurs, 2006.

[11] SongJie Gong, HongWu Ye, and HengSong Tan. Combining memory-based and model-based collaborative filtering in recommender system. In *2009 Pacific-Asia Conference on Circuits, Communications and Systems*, pages 690–693, 2009.

[12] Will Gornall, Ilya A Strebulaev, Shai Bernstein, Kim Furlong, Akitada Kasahara, Julien Krantz, Kenji Kutsuna, Shinzo Nakano, Kristian Rydqvist, Stephen Schaefer, and John Taylor. The economic impact of venture capital: Evidence from public companies *.

[13] Stephen Gower. Netflix prize and svd, 2014.

[14] Sajal Halder, Md. Hanif Seddiqui, and Young-Koo Lee. An entertainment recommendation system using the dynamics of user behavior over time. In *2014 17th International Conference on Computer and Information Technology (ICCIT)*, pages 41–46, 2014.

[15] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*, 2017. Published in NIPS 2017; version with full appendix and minor corrections.

[16] Steven N Kaplan and Per Strömberg. Venture capital and corporate governance. *The Review of Financial Studies*, 22(8):2939–2970, 2009.

[17] Farida Karimova. A survey of e-commerce recommender systems. *European Scientific Journal*, 12(34):75, December 2016.

[18] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[19] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

[20] Xiao Li, Li Sun, Mengjie Ling, and Yan Peng. A survey of graph neural network based recommendation in social networks. *Neurocomputing*, 549:126441, 2023.

[21] Xin Li and Hsinchun Chen. Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach. *Decision Support Systems*, 54(2):880–890, 2013.

[22] Xingchen Li, Xiang Wang, Xiangnan He, Long Chen, Jun Xiao, and Tat-Seng Chua. Hierarchical fashion graph network for personalized outfit recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, page To be specified, Virtual Event, China, 2020. ACM.

[23] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval.* Cambridge University Press, 2008.

[24] Ethan Mollick. The dynamics of crowdfunding: An exploratory study. *Journal of Business Venturing*, 29:1–16, 1 2014.

[25] Sri Hari Nallamala, Usha Rani Bajjuri, Sarvani Anandarao, Dr. D. Durga Prasad, and Dr. Pragnaban Mishra. A brief analysis of collaborative and content based filtering algorithms used in recommender systems. *IOP Conference Series: Materials Science and Engineering*, 981(2):022008, dec 2020.

[26] Ramana Nanda and Matthew Rhodes-Kropf. *Financing High-Potential Entrepreneurship.* Center for International Development at Harvard University, 2016.

[27] Andrew Y Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Twenty-first international conference on Machine learning–ICML '04*. ACM, 2004.

[28] Manju Puri and Rebecca Zarutskie. On the life cycle dynamics of venture-capital- and non-venture-capital-financed firms. *Journal of Finance*, 67:2247–2293, 12 2012.

[29] Eric Ries. *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. Crown Business, 2011.

[30] Marco Da Rin, Giovanna Nicodano, and Alessandro Sembenelli. Public policy and the creation of active venture capital markets, 2005.

[31] Debasmita Roy and Mainak Dutta. A systematic review and research perspective on recommender systems. *Journal of Big Data*, 9(59):59, 2022. Received: 04 October 2021, Accepted: 28 March 2022, Published: 03 May 2022.

[32] Deepjyoti Roy and Mala Dutta. A systematic review and research perspective on recommender systems. *Journal of Big Data*, 9, 12 2022.

[33] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. *GroupLens Research Group/Army HPC Research Center, Department of Computer Science and Engineering, University of Minnesota*, 2001. Emails: fsarwar, karypis, konstan, riedlg@cs.umn.edu.

[34] Marwa Sharaf, Ezz El-Din Hemdan, Ayman El-Sayed, and Nirmeen El-Bahnasawy. A survey on recommendation systems for financial services. *Multimedia Tools and Applications*, 81:1–21, 03 2022.

[35] Sanya Sharma, Aakriti Sharma, Yamini Sharma, and Manjot Bhatia. Recommender system using hybrid approach. In *2016 International Conference on Computing, Communication and Automation (ICCCA)*, pages 219–223, 2016.

[36] Leslie N Smith. Cyclical learning rates for training neural networks. *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472, 2017.

[37] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, f-score and roc: A family of discriminant measures for performance evaluation. volume Vol. 4304, pages 1015–1021, 01 2006.

[38] Morten SØrensen. How smart is smart money? a two-sided matching model of venture capital. *Journal of Finance*, 62:2725–2762, 12 2007.

[39] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2018. To appear at ICLR 2018.

[40] Srinivas Virinchi, Anoop Saladi, and Abhirup Mondal. Recommending related products using graph neural networks in directed graphs. *International Machine Learning, Amazon*, 2020.

[41] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: A survey. *ACM Computing Surveys*, 37(4):Article 111, April 2022. Affiliations: Peking University, China (Shiwen Wu, Wentao Zhang, Xu Xie, Bin Cui); Alibaba Group, China (Fei Sun).

[42] Robert Bell Yehuda Koren Yahoo Research and Chris Volinsky ATT Labs—Research. Matrix factorization techniques for recommender systems.

[43] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 10. ACM, ACM, New York, NY, USA, 2018.

[44] N. Zhang, H. W. Baker, and M. R. Elliott. Zhang et al. respond. *American journal of public health*, 103(10):e10, 2013.

[45] Xiao-Meng Zhang, Li Liang, Lin Liu, and Ming-Jing Tang. Graph neural networks and their current applications in bioinformatics. *Frontiers in Genetics*, 12, 2021. Research Topic: Artificial Intelligence in Bioinformatics and Drug Repurposing: Methods and Applications.

[46] Zhen Zhang, Taile Peng, and Ke Shen. Overview of collaborative filtering recommendation algorithms. *IOP Conference Series: Earth and Environmental Science*, 440:022063, 2020. Published under licence by IOP Publishing Ltd.

[47] Xiaoxue Zhao, Weinan Zhang, and Jun Wang. Risk-hedged venture capital investment recommendation. *Department of Computer Science, University College London*, 2023. Emails: x.zhao, w.zhang, j.wang@cs.ucl.ac.uk.

[48] Zhi-Dan Zhao and Ming-sheng Shang. User-based collaborative-filtering recommendation algorithms on hadoop. In *2010 Third International Conference on Knowledge Discovery and Data Mining*, pages 478–481, 2010.

[49] Zhifeng Zhou, Guanshan Zhu, Ahmad R. Hariri, et al. Zhou et al. reply. *Nature*, 458(7238):E7–E7, 2009.