Semester Project

EPFL - CYD Campus

Forecasting Emerging Disruptive Technologies

Tomas Garate Anderegg

Supervisors :

Prof. Andrea Cavallaro
Julian Jang-Jaccard

February 12, 2026

Table of Contents

1 Introduction

1.1 Motivation

Technological innovation evolves at an unprecedented pace, transitioning rapidly from research laboratories to commercial and operational environments. This accelerated evolution has profound implications for national security and cyber defence. As emerging technologies continuously reshape the digital landscape, their early identification becomes critical, not only to leverage new defence opportunities such as advanced encryption methods or AI-driven detection systems, but also to anticipate novel threats, including autonomous cyberattacks and sophisticated AI-powered exploits. In this high-stakes environment, the ability to forecast technological disruptions is a cornerstone of proactive national strategy.

A disruptive technology is a type of innovation capable of transforming how an industry operates. In its early stages, such a technology often performs worse than established solutions, leading large companies to overlook it. It may initially attract only a small community of users with very specific needs [1]. Over time, however, the technology improves, gradually gains adoption, and eventually becomes competitive with mainstream alternatives. Once it reaches this threshold, it can replace existing technologies and fundamentally reshape an entire market. These transitions can significantly impact companies, economic structures, and the way technologies are used.

One well-known example of a disruptive technology is the Internet. Its emergence enabled the development of many other technologies, such as e-mail, social media platforms, smartphones, and file-sharing systems [1]. The Internet did not only introduce a new communication medium but also fundamentally changed how information is accessed, shared, and used.

Smartphones are another example of disruptive technology. Although they did not exist a few decades ago, their adoption increased rapidly after 2013. This widespread use led to the creation of new services and applications, including Instagram, Snapchat, and WhatsApp, which reshaped communication and content consumption [1].

More recent examples of disruptive technologies include autonomous vehicles, where major companies are competing to develop reliable solutions, as well as autonomous robots, which are increasingly used in industrial and service applications [1].

Because disruption often manifests as a sudden rise in attention or investment, this project adopts a strategy focused on detecting early signals of such rapid shifts. In practical terms, we are interested in companies that initially receive little attention, few investors or weak signals, but suddenly become highly attractive. This intuition guides our analysis of TechRank scores, a ranking algorithm.

We first examine the initial static ranking, paying particular attention to companies positioned at the bottom of the hierarchy. Then, after applying our temporal forecasting model, we analyze whether any of these low-ranked companies rise significantly in the predicted TechRank ranking. Such upward movements may signal the early phases of a disruptive technological trajectory.

Traditional monitoring and analytical frameworks are increasingly inadequate [21]. They struggle to keep pace with the exponential growth and inherent complexity of the global technological ecosystem [21]. While data-driven models like TechRank [19] offer a significant advancement by quantifying the influence within a static network of companies and investors, they possess a fundamental limitation. They provide a snapshot of the ecosystem's current state but lack the inherent capability to model its temporal evolution.. This static formulation ignores the temporal order of interactions, the timing of events, and the evolution of relationships between nodes, which are key factors in the emergence of disruption.

Analyzing temporal evolution is particularly challenging because technological ecosystems are driven by dynamic processes such as sudden growth phases and non-linear changes. Capturing these dynamics requires models that can account for when interactions occur, how influence accumulates over time, and how past events affect future behavior. As a result, static approaches are inherently limited in their ability to anticipate disruptive shifts, whose signals often emerge gradually before becoming visible at the structural level [19, 21].

The challenge, therefore, is to move beyond static analysis and develop methods capable of modeling temporal dynamics, in order to detect early signals of disruption before they become widely apparent.

1.2 Problem definition

This project directly addresses this gap by investigating the integration of dynamic artificial intelligence techniques with network based ranking algorithms. It is guided by the following research question:

- **How can Temporal Graph Networks enhance a model like TechRank to improve the temporal forecasting of cyber relevant disruptive technologies**

We hypothesize that modeling the technological ecosystem as a dynamic bipartite graph of investors and companies will enable more accurate and earlier identification of high-potential technologies compared to using static network analysis alone. We also assume an indirect proportional relationship between the amount of funding a company receives and the technologies it is developing. All technologies associated with a company are weighted equally.

To answer this question, the project *AI-Powered Forecasting of Emerging Disruptive Technologies for National Cyber Defence* develops a novel forecasting framework. This work builds upon the foundation TechRank algorithm and extends it by integrating a Temporal Graph Network (TGN), a state-of-the-art architecture for deep learning dynamic graphs.

The methodology involves constructing a temporal bipartite graph from real-world commercial data (Crunchbase...). The TGN then employed to learn the underlying dynamics of this ecosystem, how relationships form and evolve, and to forecast its future state. By recalculating the TeckRank scores on this predicted future graph, we aim to identify technologies that are on a trajectory to become disruptive, thereby providing a crucial early-warning signal.

1.3 Contribution

The contributions of this work are:

- **A Novel Methodological Framework**: This project proposes a new framework that combines the TechRank algorithm with a Temporal Graph Network. TechRank measures the current influence of entities in a network but does not model how this influence evolves over time, while TGN predicts future links between entities but does not directly quantify their influence. The main contribution of this work is to integrate these two approaches into a single pipeline, enabling the early detection of disruptive signals by jointly capturing current influence and predicted future interactions.
- **Adapted Loss Functions**: This work introduces different loss functions specifically designed to mitigate the effects of bipartite graph structures, such as degree bias. This approach reduces the dominance of high-degree entities and promotes low-degree entities that may have strong disruptive potential.

The goal of this project is to deliver a robust, data-driven foresight tool that can support strategic decision making in national threats. By enabling a more proactive identification of both threats and opportunities, this work aims to contribute to a more resilient and anticipatory defence posture.

1.4 Outline of the report

The remainder of this report is structure as follows: Section 2 reviews related work on forecasting models and graph-based approaches, providing context for our methodological choices. Section 3 presents a comparative analysis of centrality methods and dynamic graph models to justify our selection of TechRank and TGN. Section 4 provides a detailed description of our proposed methodology, including the formalisms of TechRank and TGN and their integration approach. Section 5 outlines the experimental setup, describes the Crunchbase dataset and presents the evaluation results alongside a discussion of the model's limitations. Finally, Section 6 concludes the project by summarizing the key contributions and suggesting promising directions for future research.

2 Related Works

2.1 Forecasting Models

Time series forecasting has long relied on statistical models [5, 20]. With the rise of deep learning, new architectures such as encoder–decoder models, attention mechanisms, Transformers, and Graph Neural Networks have enabled models to learn directly from large collections of time series [5, 20]. These models can automatically capture complex temporal dependencies, non-linear relationships, and long-term patterns that are difficult to model with traditional methods. As a result, they have led to improved forecasting accuracy, better scalability to large datasets, and increased robustness to noise and missing data across diverse forecasting tasks [5, 7].

However, many real-world problems involve interdependent series, such as sensors in a network or products in a hierarchical system. To capture these cross-dependencies, Graph Neural Networks (GNNs) have been widely adopted. Several works combine GNN layers with temporal modules (RNNs, CNNs, or Transformers) to jointly model spatial and temporal relationships. For example, spatiotemporal GNNs have shown strong results in traffic forecasting, air-quality prediction, and epidemic modeling, where correlations between nodes play a key role [5, 20].

A common limitation of these architectures is that they often assume the graph structure to be static, even though in practice interactions and dependencies evolve over time. This motivates the use of Temporal Graph Neural Networks, designed to handle dynamic graphs whose connectivity changes. The TGN model is specifically tailored for such settings: it maintains a memory of past interactions and continuously updates node representations as new events occur [5, 20].

This project builds on this line of research by applying TGN to forecasting tasks, with the goal of better capturing temporal patterns in dynamic relational environments.

2.2 Graph Approaches

Node Influence Measure and Centrality Methods: A social network is a type of graph that reflects the social structure of its nodes and their relationships, such as friendship between individuals, publication links between researchers, or professional collaborations [2]. These networks, often very large, are complex systems in which the spread and circulation of information play a key role. Understanding how influence moves through such systems requires tools that can measure the importance or impact of different actors [23].

For this reason, many metrics have been proposed in the literature to evaluate social influence. According to [23], centrality measures are among the most widely used because they quantify how strategically positioned an individual is within the network. The most common examples include degree, closeness, betweenness, and eigenvector centrality.

- **Degree**: Measures the number of direct connections a node has. A node with a high degree is very connected and can quickly influence its immediate neighborhood.
- **Closeness**: Evaluates how close a node is to all others in the network. A node with high closeness centrality can reach other nodes faster, making it an effective spreader of information.
- **Betweenness**: Indicates how often a node appears on the shortest paths between other nodes. A node with high betweenness centrality acts as a bridge or mediator in the flow of information.
- **Eigenvector**: Considers not only the number of connections of a node but also the importance of its neighbors. A node is central if it is connected to other highly influential nodes.

In the late 1990s, Larry Page and Sergey Brin developed PageRank for Google, a centrality algorithm that assigns each web page an importance score. The score is computed recursively: a page is important if other important pages link to it [22].

Several variants of PageRank later emerged, such as Weighted PageRank, which gives more weight to pages considered important rather than treating all pages equally [30] or Ratio-Based Weighted PageRank, which addresses convergence issues found in the traditional PageRank and its weighted version [4].

PageRank has also inspired algorithms used in collaborative platforms such as Wikipedia, where the quality of an article depends on the expertise of the editors who contribute to it. In return, editors develop expertise by contributing to high-quality content. Bipartite models such as PageRank use this idea to evaluate both content quality and contributor expertise. These works show that collective contributions can improve quality, but too many poorly coordinated editors may sometimes reduce it [14, 19].

This idea inspired TechRank, a tool developed at CYD Campus to measure centrality in a network. Designed to help investors navigate complex technological ecosystems, TechRank ranks companies and technologies within a bipartite network, highlighting mutual influence relationships and offering a data-driven way to identify high-potential opportunities in rapidly evolving technology sectors [19].

There also exist machine-learning-based methods that estimate node influence for classification tasks by analyzing the network structure. These algorithms can predict node importance using extracted features and are especially useful for networks that evolve over time [3].

However, a key limitation of these centrality methods is that they operated on static snapshots of the networks and cannot predict how influence will evolve as new interactions occur [3, 19].

Prediction and Dynamic Graph Neural Networks

There are two types of temporal graphs: **continuous-time** and **discrete-time**. Continuous-time graphs track events as they occur, whereas discrete-time graphs are considered as a series of static snapshots [11].

Dynamic graphs are useful for tasks such as node prediction or link prediction. This project focuses on potential future links.

JODIE (Joint Dynamic User-Item Embeddings) is a dynamic graph model that predicts links between nodes. It learns the embedding trajectories of users and items from temporal interactions. Each user and item has a static embedding representing permanent characteristics and a dynamic embedding capturing evolving properties. The model uses two coupled recurrent neural networks to update embeddings after each interaction, and a projection operator to predict the future state of user embeddings. This approach allows the model to capture system dynamics and predict future interactions in bipartite networks [16, 24].

DyRep is another dynamic graph model designed to encode the evolution of nodes over time. It considers two key processes: association (growth of long-term links) and communication (short-term interactions). The learned embeddings mediate these processes and are updated as interactions occur. DyRep is able to capture complex non-linear dynamics and predict future links in evolving graphs [24, 28].

Temporal Graph Networks (TGN), developed by Twitter, provide a general framework for learning on continuous-time dynamic graphs. They combine memory modules with graph operators to capture the evolution of node relations and features over time. Many existing models, such as JODIE or DyRep, can be seen as special cases of TGN [24].

While these dynamic models excel at predicting future interactions, they do not easily highlight the most influential nodes in the graph, a gap centrality methods are designed to address [11, 16].

2.3 Summary

The literature reviewed in this section highlights the importance of modeling both temporal dynamics and relational dependencies when analyzing complex systems. While deep learning has significantly improved time series forecasting by capturing non-linear and long-term temporal patterns, many approaches still treat entities independently or rely on static assumptions about their relationships. This limits their ability to model systems where interactions between entities play a central role.

Graph Neural Networks address this issue by explicitly modeling dependencies between entities, and spatiotemporal GNNs further combine structural and temporal information. However, many of these models assume a fixed graph structure, even though real-world interactions evolve continuously over time. Dynamic graph models such as JODIE, DyRep, and Temporal Graph Networks overcome this limitation by representing interactions as time-ordered events and updating node representations accordingly. TGN was chosen among other dynamic models because of its memory capabilities. This model is also compared with two other dynamic models as baselines in order to assess the relevance of this choice. However, these dynamic models have a limitation, as they do not easily highlight the most influential nodes in the graph.

In parallel, centrality methods provide a principled way to quantify influence in networks. Classical approaches such as PageRank operate on homogeneous graphs, whereas TechRank extends this idea to heterogeneous bipartite networks, allowing mutual influence between different types of entities to be captured. Nevertheless, TechRank remains a static method and can only reflect influence based on observed interactions.

Motivated by these observations, this project combines Temporal Graph Networks for predicting future interactions with TechRank for influence estimation. By applying TechRank to both the observed and the predicted graphs, the proposed approach aims to identify entities whose predicted increase in influence may signal disruptive potential. The resulting methodology and implementation choices are presented in Section 5.

3 Preliminaries

When choosing a GNN model, it is important to consider the relevance of each node in the graph and how nodes influence one another. Some models take all neighbors as information sources, as in our case. However, a key constraint is that we cannot treat all neighbors equally, since not all neighbors carry the same importance. There must be a mechanism to distinguish between neighbors. Some graph models inspired by PageRank account for neighbor influence [26]. Therefore, neighbors should be discriminated based on both their features and the graph structure (how they are connected).

We are looking for a GNN that allows us to select the most important neighbors.

3.1 Selecting a Centrality Method

A natural question is how one node can influence another. Node influence is defined as the ability of a node or group of nodes to impact other nodes or part of the network [3]. The goal is to generate a ranking of nodes according to their influence in the network.

Different types of rankings include: **Centrality-based methods** or **Machine Learning methods** as mentioned previously.

[3] demonstrate simple methods such as centrality methods are a reliable approach to adopts. To choose the most suitable centrality method for our application, several requirements are listed:

Requirements
1. Path lengths from a node to a target are not considered. 2. A global measure is preferred, since we aim to model information propagation across the whole network. 3. Both direct and indirect influences are not taken into account. 4. As suggested by [3], structural properties of the network (e.g., heterogeneity) influence the choice of method; the method must be robust to complex and heterogeneous networks. 5. The centrality method should also be suitable for heterogeneous graphs.

Table 1: Summary of requirements for selecting an appropriate centrality method.

A selection of different centrality methods are summarized in the table below to guide the selection process. Each centrality method is compared against relevant features and according to the requirements.

Centrality Methods	**Distance**	**Global Measure**	**Indirect Influence**	**Requirements**
Degree	×	Local	×	✓
Closeness	✓	✓	×	×
Betweenness	✓	✓	✓	×
Eigenvector	×	✓	✓	✓
PageRank	×	✓	✓	×
Weighted PageRank	×	✓	✓	×
TechRank	×	✓	✓	✓

Table 2: Selection of the most important centrality methods, including TechRank, to assess whether they satisfy the requirements. (✓) satisfy the method, (×) does not satisfy the methods [3, 19].

Having more centrality methods available increases the robustness of the algorithm used to detect node influence [3]. A natural choice for the centrality measure is the TechRank, an algorithm developed at the CYD campus on a bipartite graph. This alogorithm has already demonstrated good performance on the dataset used in this study, making its integration into the proposed pipeline straightforward [19], and offering additional features, such as external variable, which is not exploited in this project but could further enhance the robustness of the pipeline.

3.2 Selecting a Dynamic Graph Model

Given the dataset at our disposal, we naturally focus on a **continuous-time approach**, since the temporal data corresponds to the announced dates of funding rounds, which are considered events due to their non-periodic nature. Previous studies have compared different temporal graph models on various aspects, such as preferential attachment, recency, density, and temporal granularity [11, 24]. The model that best meets the criteria for this project is the **Temporal Graph Neural Network (TGN)**. [11] show that TGN performs consistently well under coarse temporal granularity, similarly to DyRep and JODIE. Moreover, [24] highlight the importance of an explicit memory module, which enables TGN to capture long-term dependencies and evolving node states in continuous-time dynamic graphs, making it particularly well suited for modeling funding events and their temporal dynamics, whereas DyRep and JODIE do not include an explicit memory module.

4 Methodology

Symbol / Variable	Description
C	Set of companies (nodes of type company) in the bipartite graph.
T	Set of technologies (or investors, depending on the experiment) in the bipartite graph.
I	Set of investors (used when modeling the investor–company bipartite graph).
$c \in C$	Index of a company node.
$t \in T$	Index of a technology (or investor) node.
$i \in I$	Index of an investor node.
n_c	Total number of company nodes.
n_t	Total number of technology (or investor) nodes.
$M_{c,t}^{CT}$	Adjacency matrix of the bipartite graph; equals 1 if company c is connected to technology (or investor) t, and 0 otherwise.
$w_c^{(n)}$	TechRank score of company c at iteration n.
$w_t^{(n)}$	TechRank score of technology (or investor) t at iteration n.
$w_c^{(0)}, w_t^{(0)}$	Initial TechRank scores for companies and technologies (or investors).
$G_{c,t}(\beta)$	Transition probability from technology (or investor) t to company c, modulated by parameter β.
$G_{t,c}(\alpha)$	Transition probability from company c to technology (or investor) t, modulated by parameter α.
α	TechRank parameter controlling the influence of company degree on technology (or investor) scores.
β	TechRank parameter controlling the influence of technology (or investor) degree on company scores.
n	Iteration index of the recursive TechRank update (reflection method).
u, i	Node identifiers in the TGN framework, where u denotes a company and i an investor.
$e = (u, i, ts)$	Temporal interaction (edge) between nodes u and i occurring at timestamp ts.
ts	Timestamp associated with an interaction or funding event.
$m_u(t)$	Memory state of node u at time t in the TGN.
$msg_{u,i}(t)$	Message generated from an interaction between nodes u and i at time t.
$\phi(\cdot)$	Message function used to construct messages from node states and interaction features.
$\psi(\cdot)$	Memory update function that updates node memory based on incoming messages.
$h_u(t)$	Embedding of node u at time t computed from its memory state.
$s(u, i, t)$	Predicted link score between nodes u and i at time t.
ΔR	Change in TechRank ranking position of a company between the initial graph and the predicted future graph.
$V_{\text{threshold}}$	Threshold applied to ΔR to identify companies with a significant predicted rank increase.
ρ	Temporal split ratio used to divide the test dataset into prediction and ground-truth subsets.
r	Spearman rank correlation coefficient between predicted and ground-truth rankings.
p	P-value associated with the Spearman rank correlation.
α_{focal}	Weighting coefficient for the focal loss.
α_{dcl}	Weighting coefficient for the DCL loss

Table 3: Summary of variables and parameters used in this work.

4.1 TechRank

TechRank [19] ranks companies and technologies in a bipartite network, it uncovers mutual influence patterns and provides a data-driven framework to identify high-potential opportunities in fast-evolving technology landscapes.

Model Architecture for TechRank

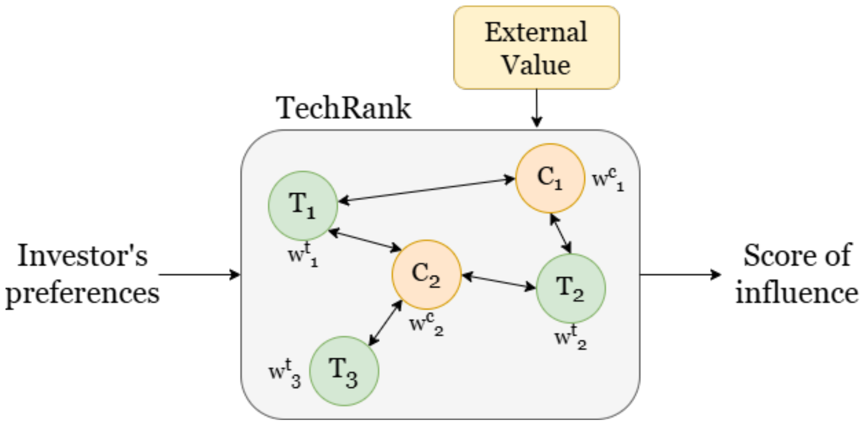


Figure 1: Framework of the TechRank algorithm, showing a bipartite graph between technologies (T) and companies (C) with their respective score w.

In graph theory, an **adjacency matrix** represents all connections in a graph. For TechRank, this matrix models the ecosystem of companies and technologies.

$$w_c^{(0)} = \sum_{t=1}^{n_t} M_{c,t}^{CT}, \quad w_t^{(0)} = \sum_{c=1}^{n_c} M_{c,t}^{CT} \tag{1}$$

$$w_c^{(n+1)} = \sum_{t=1}^{n_t} G_{c,t}(\beta)\, w_t^{(n)}, \quad w_t^{(n+1)} = \sum_{c=1}^{n_c} G_{t,c}(\alpha)\, w_c^{(n)} \tag{2}$$

where $M_{c,t}^{CT} = 1$ if company c uses technology t (0 otherwise). $w_c^{(0)}$ and $w_t^{(0)}$ are the initial scores. The external variable incorporates external context, improving TechRank's predictive accuracy.

Recursive Scoring (Reflection Method)

The core of the algorithm is an iterative process that updates the scores of companies and technologies based on the scores of their connected neighbors. Transition probabilities control how influence propagates across the bipartite network, with parameters α and β modulating the weight of node degrees.

The algorithm iteratively updates company and technology scores based on their neighbors. Transition probabilities, modulated by parameters α and β, control how influence spreads across the bipartite network.

In this project, the external variable and the computations used to obtain the optimal parameters $\alpha^\star$ and $\beta^\star$ based on that external variable are not taken into account. We only use the scoring system and the reflection method.

4.2 Temporal Graph Neural Network

Temporal Graph Networks for Deep Learning on Dynamic Graphs [24] are a framework designed specifically for deep learning on continuous-time dynamic graphs, which are represented as a sequence of timed events (e.g. a user liking a post at a specific time).

Model Architecture for Temporal Graph Networks (TGN)

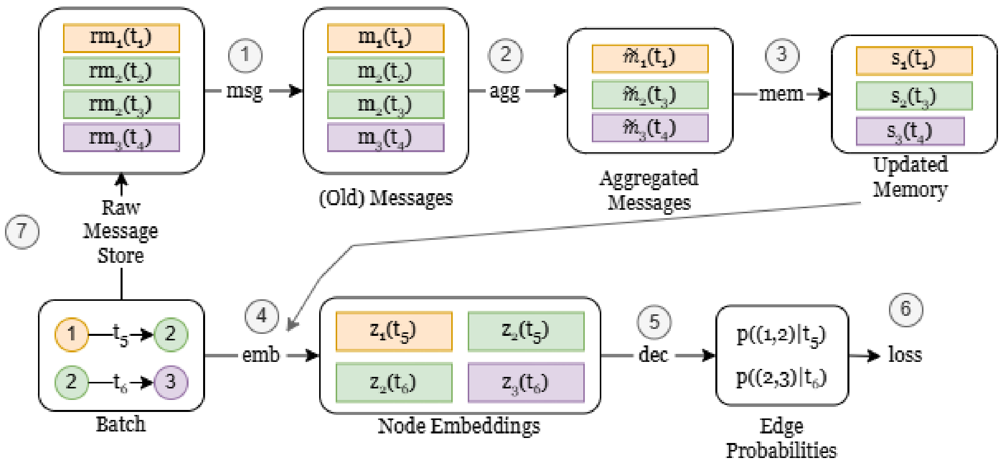


Figure 2: Framework of the TGN algorithm [24].

TGN models and predicts how relationships evolve over time by combining a *memory module*, capturing each node's historical state, with *graph-based operators* that model node interactions.

The model takes as input a bipartite graph with a specific structure. The bipartite graph constructed is adapted to ensure compatibility with the TGN implementation while keeping modifications to the original code minimal. In this design, nodes do not contain any features and are therefore initialized with zero vectors. All the information is stored in the edge features.

The model processes the events stored in the edge features in chronological order. For each event involving two nodes, the memory of the corresponding nodes is updated. This memory captures the historical information of each entity over time. To make a prediction, the model generates a node embedding by aggregating information from its most recent neighbors using an attention-based layer. An MLP decoder then takes two node embeddings as input and predicts the probability that a future interaction will occur between them. The model is trained by comparing its predictions with the true interactions and updating its parameters through backpropagation.

Once the model is trained, it is used to predict future interactions. Embeddings are generated for all companies and investors, and interaction probabilities are computed for all possible pairs. These predicted interactions are then adapted into a bipartite graph and used as input for the TechRank centrality method.

4.3 Approach

First, we compute the TechRank centrality to get an initial measure of influence for both investors and companies. The graph is then used as input to a Temporal Graph Network (TGN), which predicts a new graph with potential new edges. By recalculating TechRank on the predicted graph, we obtain predicted influence scores for the nodes. To assess disruptiveness, the difference between the initial and predicted TechRank scores highlights companies that could be particularly interesting or promising by a system of threshold.

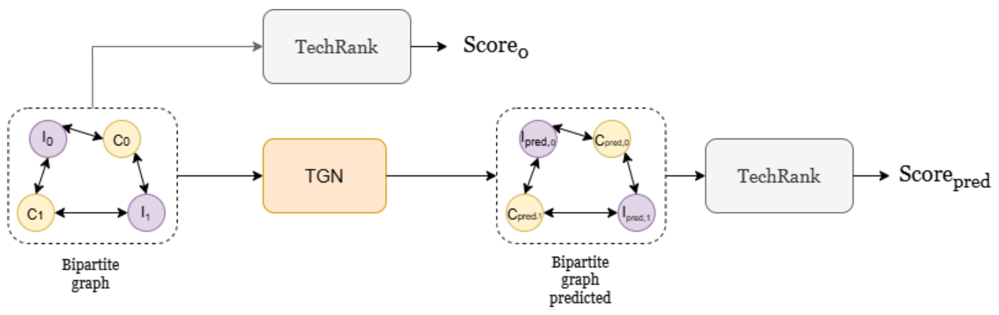


Figure 3: Schematic of the predictive algorithm including TechRank centrality method, TGN and the bipartite graph of Investor(I) and Companies (C).

This project focuses on companies that initially appear at the bottom of the TechRank ranking but experience a significant rise after prediction and score recomputation. To ensure a meaningful selection, a threshold criterion is introduced to identify companies showing a substantial increase in predicted influence.

The TechRank score is computed as a relative change with respect to its initial value in order to measure how much the score has evolved over time. This relative variation is then compared to a predefined threshold to determine its level of disruptiveness.

$$\Delta R = \frac{R_{\text{predicted}} - R_{\text{initial}}}{R_{\text{initial}} + \varepsilon} \tag{1}$$

where ε is a small constant introduced to avoid numerical instability. This could happen since TechRank initial score can be very low.

This rank-based approach enables the identification of companies that were initially weakly visible but become more influential after prediction, which directly aligns with the notion of disruptiveness studied in this work.

Also, TechRank relies on parameters α and β to control how influence propagates in the network. An analysis of these parameters allows the model to better exploit their effects and to provide a more effective forecasting tool for identifying potentially disruptive companies.

Finally, TechRank provides interpretable influence scores but remains static, while TGN captures temporal dynamics without offering a direct measure of influence. The proposed framework leverages the strengths of both methods.

5 Validation

5.1 Dataset

The choice of dataset was straightforward, given that access to data on company investments and activities is very limited and expensive. The CYD campus has access to the Crunchbase dataset, which contains a wealth of relevant information, including investments made in companies, the types of technology they develop, the investors involved, and the dates of funding rounds. We evaluate our approach using **Crunchbase** [8], a platform for private and public company data, which provides, among other, financial data focusing on innovative and technology oriented companies. The list below presents a sample of the different entities contained in the Crunchbase dataset:

- Acquisitions
- Degrees
- Event Appearances
- Investment Partners
- Funding Rounds
- Investments
- Organizations
- People

To build a bipartite graph compatible with our application and dynamic pipeline, we had to select fields that would allow us to predict future links between Investor and Company nodes. Based on our assumption about the indirect connection between companies and technologies, a new link between an investor and a company can be interpreted as an investment in the technologies developed by that company.

We therefore selected the total amount raised in funding rounds together with the names of the investors. The number of investors also appeared relevant: the more investors a company attracts, the higher the interest in what it develops.

On the company side, technologies were an obvious choice, since our project focuses on analyzing the technologies companies work on. For this project, we selected technologies related to quantum computing and quantum key distribution, which resulted in a moderately sized dataset. This allowed the model to run efficiently during data extraction, graph construction, and training.

Since graph prediction requires temporal information, we used the *announced on* field, which indicates the date of each funding round. The final lists for both types of nodes are presented below:

Investors
- Name
- Funding Round ID
- Date of the Funding Round
- Raised Amount USD
- Number of Investors

Companies
- Name
- Company ID
- Technologies

In the bipartite graph, the **Investor node** contains only the investor's name, and the

Company node contains only the company's name. All additional information is stored in the **edge** connecting the two nodes.

5.2 Experimental Setup

The goal of this experiment is to evaluate the proposed integration of TechRank and TGN for predicting emerging technologies. All experiments were conducted on a Lenovo ThinkPad P14s equipped with an NVIDIA T500 GPU (4 GB VRAM), an Intel Core i7 CPU, and 16 GB of RAM. This hardware setup provides sufficient computational capacity to train dynamic graph models of moderate size. However, the limited GPU memory constrains the batch size and the neighbor sampling depth. All training was performed on Windows using PyTorch with a CUDA version compatible with the GPU. To ensure reproducibility, all experiments were run in the same software environment, using identical datasets, fixed random seeds, and a consistent preprocessing pipeline.

Data Preparation: Crunchbase data are extracted from the investments and funding rounds tables. After standard data cleaning (i.e., removal of irrelevant columns and handling of missing values), the two tables are merged using the unique funding round uuid identifier. A domain-specific filter is then applied to retain only companies operating in the targeted fields, namely quantum computing and quantum key distribution.

The announced on column (funding announcement date) is converted to a datetime format and used as the temporal marker. The data are then sorted chronologically and split into training (70%), validation (15%), and test (15%) sets. This temporal split ensures that the model does not access future information and prevents information leakage.

The bipartite graph is constructed using NetworkX, where:

- type-0 nodes represent companies,
- type-1 nodes represent investors,
- edges represent funding events and include attributes such as total raised amount (USD), number of funding rounds, and funding round identifiers.

For each edge, the timestamp corresponds to the date of the funding event between the company–investor pair. Companies and investors are stored in dictionaries to allow later integration with TechRank. Table 4 reports the main statistics of the constructed bipartite graph. The dataset size is intentionally limited to ensure reasonable runtime and computational cost.

Finally, the data are formatted for TGN. This includes a CSV file containing the columns u (company ID), i (investor ID), ts (timestamp), and label (always set to 1), a NumPy file containing edge features (total amount raised and number of funding rounds), and a mapping file linking TGN internal IDs to real entity names.

Network	Crunchbase: I-C
#Nodes	1239
#Edges	1330
#Node Investors	1016
#Node Companies	223
#Edge Features	2
Timespan	42684 days
Chronological Split	70% – 15% – 15%

Table 4: Bipartite graph Investors (I) and Companies (C) constructed with Crunchbase.

Model Parameters: In TechRank, the parameters α and β control the transition probabilities in the random walk on the bipartite network [19].

The parameter β modulates how an investor's degree (number of companies funded) affects the weight of its contribution to company scores. Formally, investor i's contribution is weighted by $k_i^{-\beta}$, where k_i is its degree.

- $\beta = 0$: all investors contribute equally (simple degree counting)
- $\beta > 0$: penalizes highly active investors, favoring votes from niche investors
- $\beta < -1$: rewards highly active investors; with β = -5 (our setting), active investors' votes are amplified by a factor of k_i^5, giving disproportionate influence to well-connected investors

The parameter α follows analogous logic for companies voting toward investors, weighted by $k_c^{-\alpha}$ where k_c is the company's degree (number of investors).

- $\alpha > 0$: penalizes highly funded companies
- $\alpha \simeq 0$: (our setting: $\alpha = 0.3$): companies contribute nearly equally regardless of their connectivity

Our focus is on identifying initially low-visibility companies that experience sudden increases in attractiveness. Such companies initially have few investors and low TechRank scores. After TGN prediction, new links may be inferred, increasing investor count and consequently the company's score. We adopt β = -5 to strongly amplify the influence of active, established investors. When TGN predicts that a highly active investor will fund a previously low-ranked company, the addition of this new link to the graph causes a substantial TechRank increase, due to the amplified weight (k_i^5) of the investor's contribution. For instance, an investor funding 50 companies has a weight 9.7 million times larger than an investor funding only 2 companies ($50^5/2^5 \approx 9.7 \times 10^6$). This amplification effect makes TechRank particularly sensitive to new connections with established investors, enabling the identification of companies experiencing sudden shifts in perceived potential. Conversely, $\alpha \simeq 0$ ($\alpha = 0.3$) ensures that companies contribute nearly symmetrically to investor scores, avoiding bias based on company connectivity and maintaining focus on investor activity as the primary signal.

Regarding the TGN model, training was performed using the hyperparameters listed in Table 12 (see Appendix A.1).

TGN training is carried out in two stages:

- **Training** on the training set, with performance monitored on the validation set
- **Future link prediction** between investors and companies using the test set

Binary cross-entropy is used as the loss function for the link prediction task, and optimization is performed using Adam with a weight decay of 1×10^{-5}.

Evaluation Metrics: The network contains only observed investor–company interactions. To train the model, negative examples are generated through negative sampling, where random investor–company pairs are selected for each positive edge. This allows the model to distinguish true investment opportunities from false ones.

While prior work on TGN mainly relies on global classification metrics such as AUROC and Average Precision [24], the task addressed in this project is a ranking problem. The objective is not only to classify links correctly, but to rank companies likely to receive investments among the top recommendations.

Therefore, we complement classification metrics with recommendation oriented measures, including Mean Reciprocal Rank (MRR), Recall@10, and Recall@50, as commonly used in temporal prediction models such as JODIE [16]. MRR evaluates the ability of the model to rank correct links as early as possible, strongly penalizing late predictions. This metric is particularly suited for early retrieval analysis in imbalanced link prediction settings dominated by hub structures [25].

Recall@k further measures the proportion of true links appearing within the top-k recommendations for each investor, capturing the model's ability to cover relevant predictions at the top of the ranking.

Baselines: The TGN model is compared against three baselines: a classical machine learning model (Random Forest) and two continuous-time dynamic graph models, JODIE and DyRep. All models are trained on the same bipartite investor–company graph and follow identical temporal constraints to ensure a fair comparison.

The Random Forest baseline uses a tabular representation of investor–company pairs, with temporal, financial, and relational features computed only from historical information available at prediction time, thus avoiding information leakage. It is evaluated using both classification metrics (AUROC, AP) and ranking-oriented metrics (MRR, Recall@k), serving as a non-graph reference to assess the benefit of temporal and structural modeling in TGN [24, 25].

Table 5 reports the transductive performance of the the selected TGN and the three baseline models.

Configuration	**Transductive**				
	AUROC	**AP**	**MRR**	**R@10**	**R@50**
Random Forest	0.519	0.602	0.086	0.225	0.500
Jodie	0.764	0.712	0.143	0.282	0.823
DyRep	0.720	0.718	0.185	0.323	0.682
TGN	**0.774**	**0.795**	**0.306**	**0.382**	**0.852**

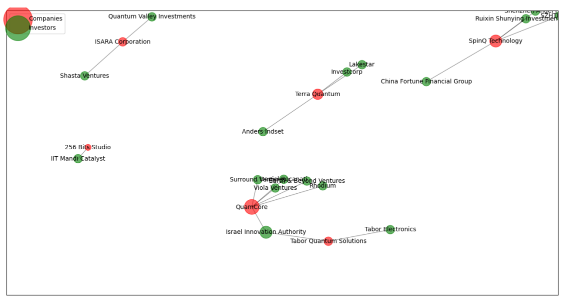
Table 5: Transductive performances of the different models on the Crunchbase dataset.

The results confirm the relevance of using TGN and its level of complexity for predicting new links in a bipartite graph. Classical machine learning models do not capture temporal dynamics [9, 10]. The dynamic graph models JODIE and DyRep can be seen as specific instances of TGN [24], and they achieve lower performance across all evaluation metrics.

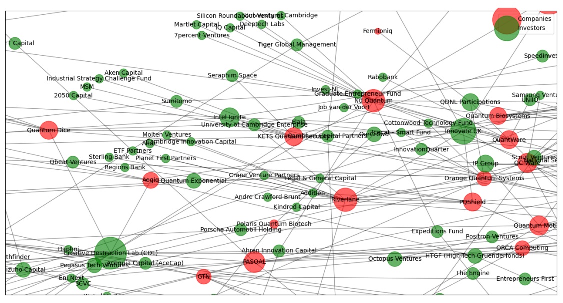
It is more relevant to focus on the model performance in the transductive setting. In a transductive task, the model predicts future links between nodes that were already observed during training, whereas in an inductive setting, links are predicted for nodes that were never seen during training. Since this project focuses on predicting potential future investments involving already existing companies, evaluating performance in the transductive setting is more appropriate.

5.3 Evaluation

Bipartite Graph: Figure 4 shows two views of the bipartite graph: one highlighting sparsely connected nodes and another illustrating a densely connected region. An image of the complete graph is provided in Appendix A.3 11.



(a) Visualization of nodes with few investors, low degree.



(b) Visualization of nodes with a lot of investors, high degree.

Figure 4: Visualization of the nodes Investors-Companies with their edges. The size of the node represent the degree of it. The bigger the node the higher the degree and vice versa.

By analyzing the construction of the bipartite graph, several structural properties can be identified that guide both the data splitting strategy and the interpretation of the experimental results. The graph shown in Fig. 10 (see Appendix A.2) is represented as an adjacency matrix between investors and companies, where a value of 1 indicates the presence of a link and 0 its absence. The matrix exhibits a very high level of sparsity, which is a common characteristic of real-world interaction graphs.

After sorting rows and columns by node degree, hub structures clearly emerge, revealing a highly asymmetric degree distribution between investors and companies. This indicates a strong structural imbalance in the graph topology, rather than a class imbalance in the strict sense of link prediction [25]. A small number of nodes concentrate a large fraction of interactions, while most nodes remain sparsely connected. Such a structure can facilitate information propagation during the training of temporal models such as TGN, but it may also introduce bias, as the model can favor predictions involving high-degree nodes at the expense of low-degree ones [25].

It is important to distinguish this structural degree bias from class imbalance. Class imbalance depends primarily on the ratio between positive and negative links in the dataset, rather than on the graph topology itself, even though [12] noted there is no strict definition of imbalance. In our case, after analyzing the global data distribution, we observe an extreme imbalance between positive and negative links, with a positive rate of 0.59 %, corresponding to a negative-to-positive ratio of 169:1. This imbalance poses two major challenges during training: first, the dominance of negative examples can bias the model toward systematically predicting negative links; second, high-degree nodes are over-represented in training, while low-degree nodes, although potentially informative, remain under-represented.

To stabilize training under this extreme imbalance, we adopt a balanced negative sampling strategy. For each positive interaction, a negative sample is generated, resulting in artificially balanced batches composed of 50 % positive and 50 % negative examples, while preserving the temporal structure of interactions [13, 17]. In the negative sampling process, the source node is fixed and the destination node is sampled uniformly at random among eligible companies. This design encourages the model to learn source-specific preferences, preparing it for the evaluation setting where, for a given company, the model must rank potential investors. In contrast, fully random source–destination sampling could lead the model to simply learn co-occurrence patterns, without capturing individual investor behavior [18].

Although uniformly sampled negative companies can sometimes be easy to distinguish (e.g., companies from unrelated sectors). More advanced approaches, such as adversarial negative sampling, explicitly select negative samples that are similar to positive ones, making the distinction more difficult for the model [18]. In our case, due to the constructive nature of the project, where companies are selected within a specific technology category, and because the dataset focuses on a technology ecosystem, randomly sampled negatives can already be considered relatively hard.

Despite balanced sampling, not all training examples have the same difficulty and degree bias remains present. To address these issues, we introduce weighted loss functions that dynamically adjust the contribution of each example during optimization. Focal Loss focuses learning on difficult examples by reducing the influence of easy, well-classified predictions. Degree Contrastive Loss (DCL) mitigates structural bias by reweighting interactions based on node degree: interactions involving high-degree nodes are downweighted, while those involving low-degree nodes receive higher importance [13]. This prevents frequent interactions from overshadowing rarer but potentially more informative ones. An explanatory pseudo-code of the DCL formulation is provided in Appendix A.4 (Algorithm 1).

In this project, no filtering is applied to remove low-degree nodes, as these may correspond to emerging or potentially disruptive companies. Excluding them would contradict the central hypothesis of the study. Instead, the degree bias is explicitly preserved and analyzed during evaluation to ensure a realistic representation of the data [25], while being mitigated at the loss level through degree-based reweighting.

Finally, an ablation study was conducted using the TGN model to evaluate the robustness of the proposed approach. We first trained the model using the standard binary cross-entropy (BCE) loss. We then evaluated the impact of Focal Loss to emphasize hard samples, followed by Degree Contrastive Loss to address degree bias. Lastly, we combined both losses to assess whether their joint use further improves training stability and ranking performance in highly imbalanced settings. The hybrid loss is a simple weighted ($\alpha_{focal} = \alpha_{dcl} = 0.5$) focal and DCL losses.

$$\mathcal{L}_{\text{Hybrid}} = \alpha_{\text{focal}}\,\mathcal{L}_{\text{Focal}} + \alpha_{\text{dcl}}\,\mathcal{L}_{\text{DCL}} \tag{2}$$

Model Performance:

Configuration	**Transductive**				
	AUROC	**AP**	**MRR**	**R@10**	**R@50**
TGN	**0.835**	**0.848**	0.530	0.645	0.870
TGN-Focal loss	0.767	0.799	**0.702**	0.745	0.835
TGN-DCL	0.763	0.779	0.527	**0.755**	**0.910**
TGN-(DCL + Focal loss)	0.769	0.819	0.631	0.685	0.865

Table 6: Ablation study on implemented loss functions compared to the baseline loss.

The model generally performs better when using the DCL loss, as shown in Table 6. These results are not surprising, given that the node and edge features are not very rich. In such a setting, the sampled negative examples are expected to be very similar to the positive ones in the embedding space, which limits the effectiveness of focal loss. In contrast, explicitly addressing degree bias leads to an improvement in model performance.

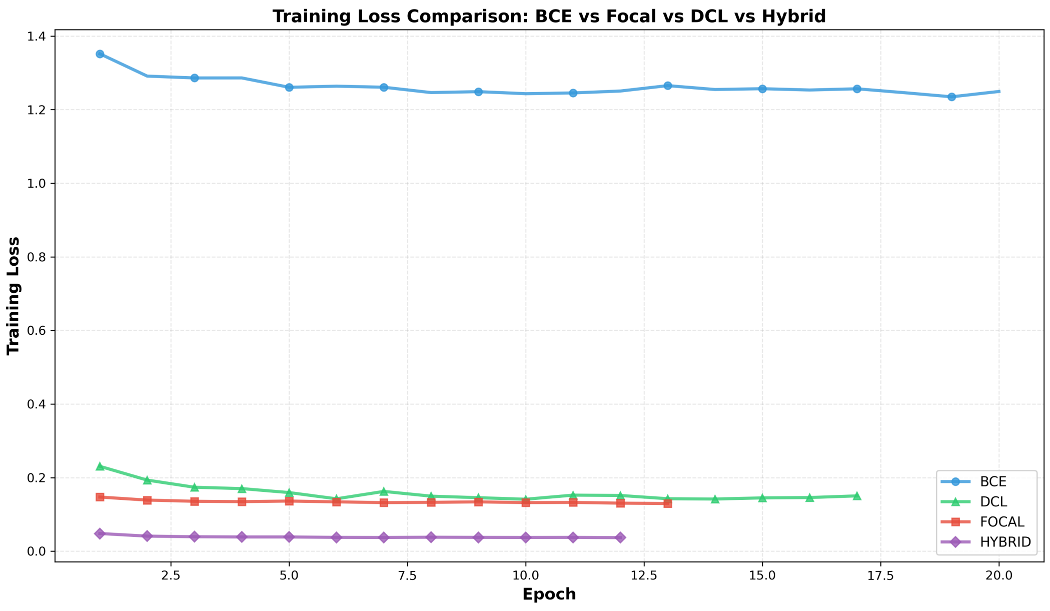


Figure 5: Evolution of the loss curves for the 4 configurations of losses.

The TGN model shows good performance according to the AUROC and AP metrics. However, these metrics are considered less relevant for our task. As pointed out in [25], AUROC can be insufficient and even misleading in highly imbalanced settings, as it is insensitive to class imbalance and may overestimate the real performance of link prediction methods.

For this reason, the main metrics of interest in this project are ranking-based metrics, namely MRR and Recall@K. According to these metrics, we observe a clear improvement, with Degree Contrastive Loss showing particularly strong performance on Recall@K. Figure 5 shows that all loss functions converge relatively quickly, but the tested losses reach significantly lower values compared to the original loss. An early stopping mechanism with a patience strategy was implemented to stop training when the improvement in loss becomes very small. Finally, the hyperparameters for the focal and DCL loss were not optimized, as hyperparameter tuning is outside the scope of this project.

Delta Ranking (ΔR): Once the predicted graph is obtained, TechRank scores are recomputed to measure changes in ranking. A threshold is applied to define significant rank variations. A stricter threshold requires larger rank increases but results in fewer candidate companies, while a lower threshold increases coverage. Figure 6 presents the top 20 companies with $\Delta R > V_{threshold}$, showing their scores before and after prediction.

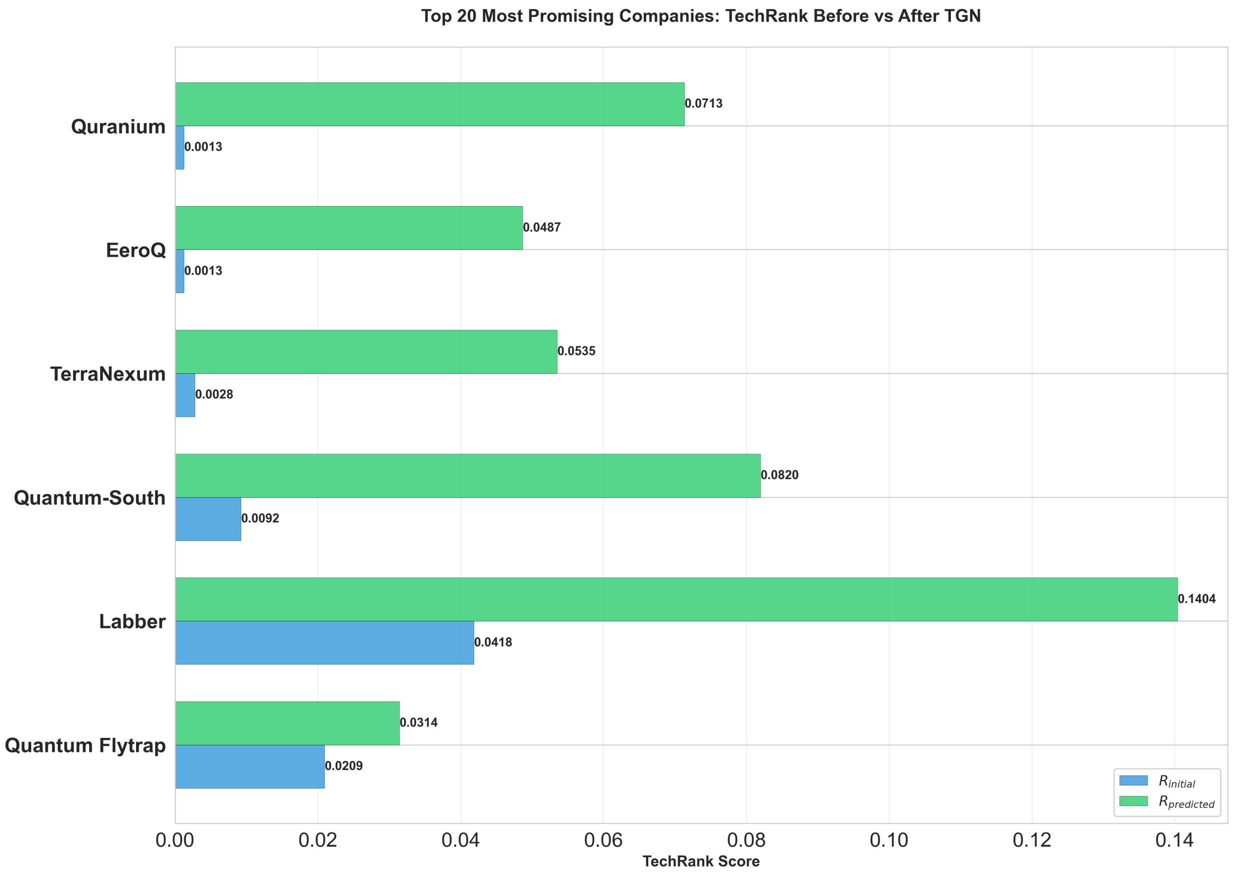


Figure 6: Ranking of companies based on the relative evolution with $\alpha = 0.3$ and $\beta = -5.0$.

Changes in ranking are also taken into account to compare the two evaluation approaches. Figure 8 illustrates the ranking increases of different companies (resp. decreasing ranking companies). We observe that the top-ranked (Quranium) company shows both a significant rank improvement and the strongest relative increase in the TechRank score.

To better visualize changes in ranking before and after prediction, Figure 7 presents a bump chart showing the scores of the top 30 companies in a more graphical manner. In addition, Figure 8 provides a more quantitative view of ranking changes for the top 30 companies, illustrating how much each company moved up or down in the ranking.

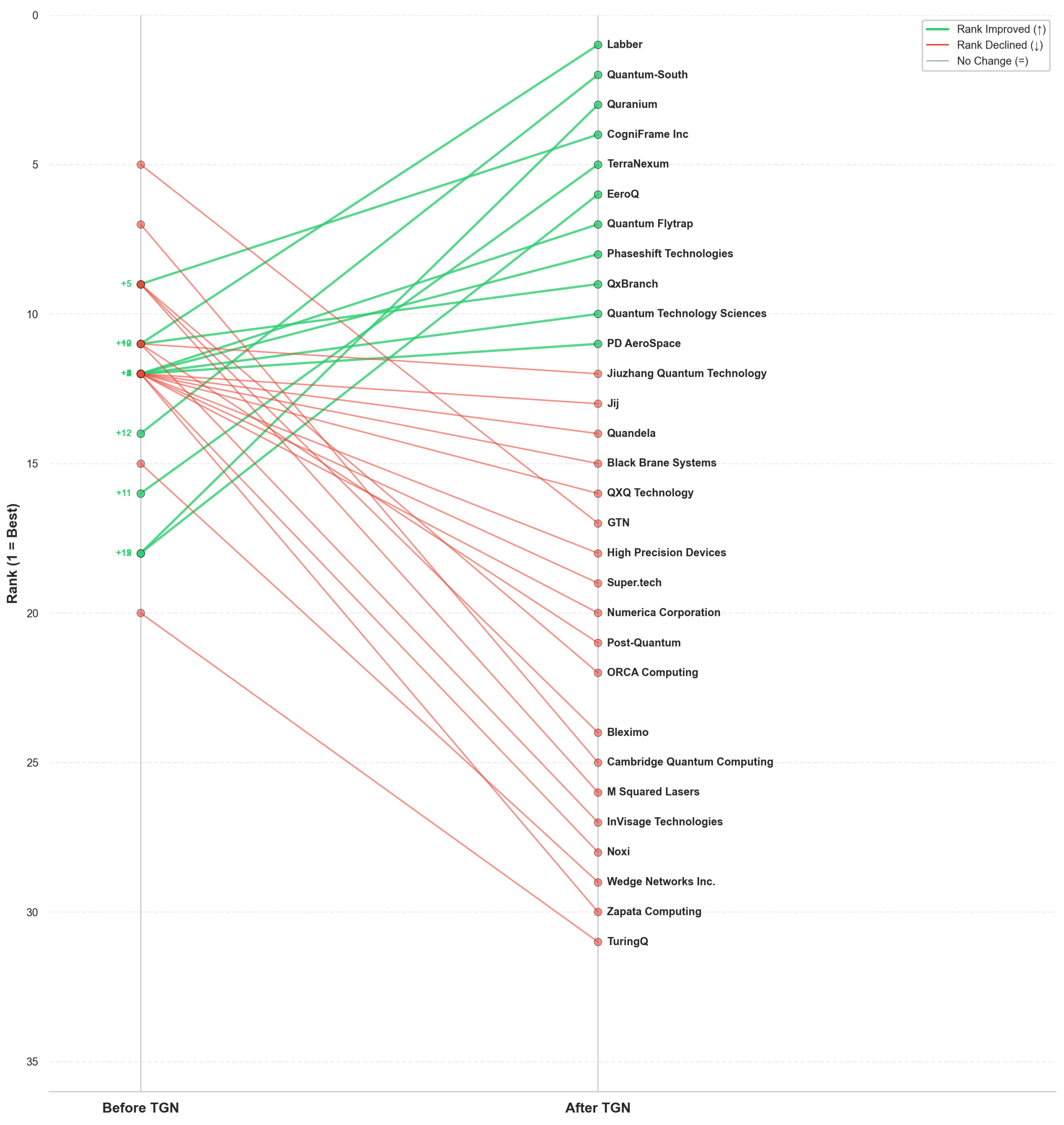


Figure 7: Bump chart of the ranking of companies based on the relative evolution.

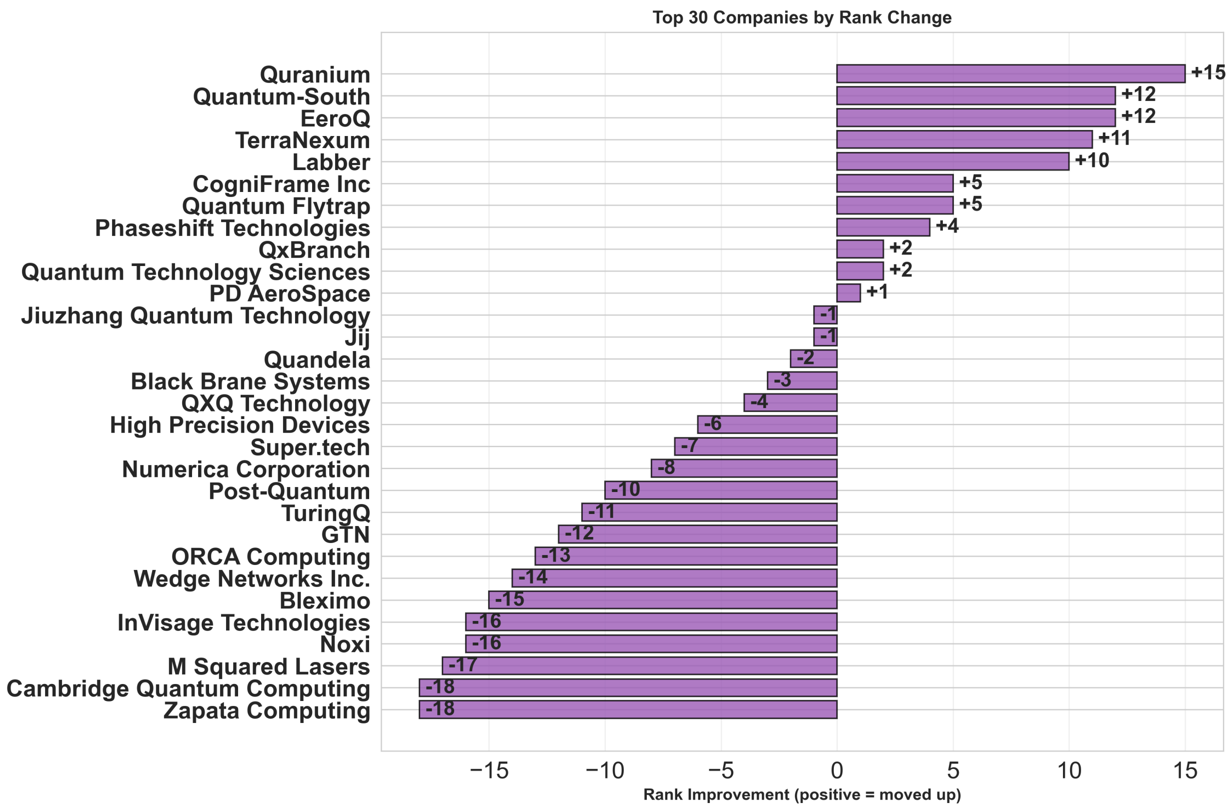


Figure 8: Diverging bar char of the ranking evolution for different companies.

Top Companies	Quranium	Quantum-South	EeroQ
Rank	1	2	3
Growth Score	65	72	73
CB Rank	60947	106889	1742
Heat Score	51	61	85

Table 7: Bipartite graph Investors (I) and Companies (C) constructed with Crunchbase.

In addition, results are compared with external indicators such as Heat Score or Growth Score (Table 7), such a comparison does not provide a methodologically sound validation. Crunchbase metrics rely on proprietary algorithms whose formulas, weights, and update mechanisms are not publicly disclosed, preventing rigorous interpretation. Moreover, the analysis conducted in this project focuses on a specific technological domain, which further limits the relevance of such comparisons [19].

A cross-methodology was performed on Figure 9, no clear conclusion can be drawn when comparing the data processed through the project pipeline with the updated Crunchbase data.

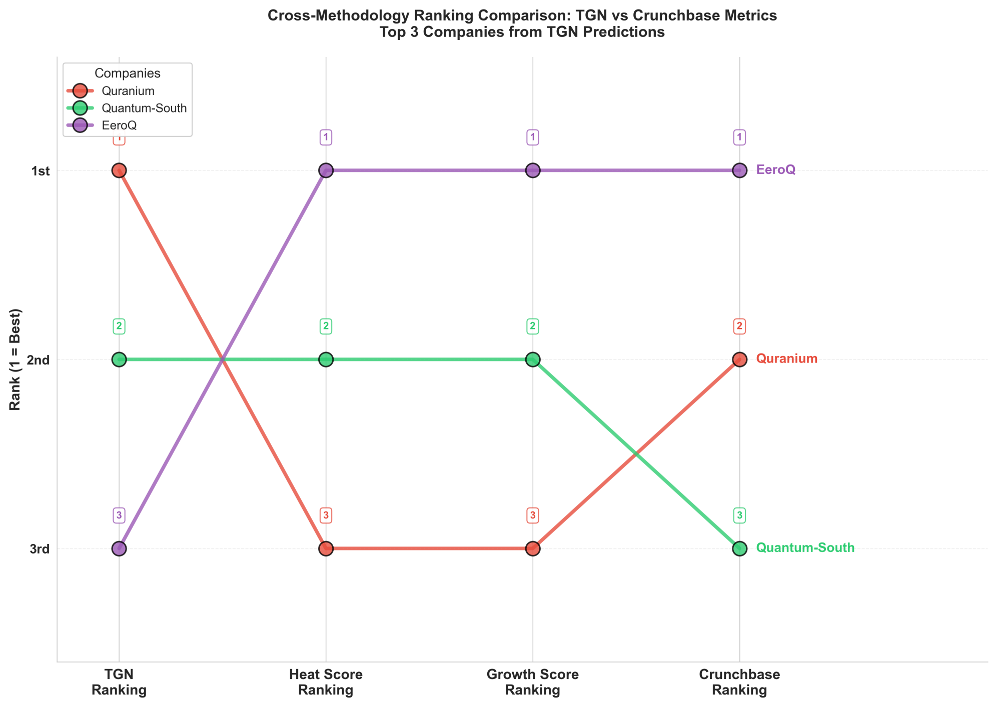


Figure 9: Cross-methodology on the ranking evolution for different companies.

A comparison using different values of the parameters α and β was conducted, as shown in table 8, to assess their influence on the ranking of companies. The values $\alpha = 0.3$ and $\beta = -5.0$ were selected, based on the reasoning done previously, for the remainder of the project. Parameter optimization is not the objective of this study.

$(\alpha;\beta)$	(0.0;-2.0)	(0.3;-5.00)	(1.0;-1.0)	(0.0;0.0)
Companie 1	Quranium	Quranium	Quranium	Labber
Companie 2	Quantum-South	Quantum-South	Labber	Quranium
Companie 3	EeroQ	EeroQ	EeroQ	Quantum-South

Table 8: Influence of the parameters (α, β) on the ranking of companies.

Pipeline To evaluate the pipeline's ability to generalize to unseen data, the test dataset, spanning from September 5, 2024 to July 3, 2025, was split using a user-defined temporal split ($\rho = 0.6$).The first portion of the split, covering interactions before March 11, 2025 and comprising 120 interactions, is used by the TGN model to generate link predictions, from which influential nodes are identified using TechRank. The second portion of the split, covering interactions after March 11, 2025 and comprising 80 interactions, contains future interactions unseen by the model and is used as ground truth. Influential nodes are also computed on this ground-truth data. Finally, Spearman's rank correlation is used to measure the agreement between the predicted ranking and the ground-truth ranking of influential companies.

Although the absolute influence scores differ in scale between the predictions and the ground truth, the results show a statistically significant Spearman correlation ($\rho = 0.3746$,

$p < 0.05$). In addition, a strong Top-k overlap is observed, indicating that the model's pipeline preserves the relative ordering of influential companies. These results suggest that the pipeline implemented is able to capture meaningful temporal patterns and maintain consistent rankings over time.

Table 9 summarizes the main results of this temporal validation.

Category	**Indicator**	**Result**
Data	Companies (predictions)	223
	Companies (ground truth)	28
	Common companies	28
Rank correlation	Correlation (r)	**0.3746**
	P-value	**0.0495**
	Statistical significance	Yes ($p < 0.05$)
Top-k overlap	Top-10 overlap	60% (6/10)
	Top-20 overlap	85% (17/20)

Table 9: Temporal validation results using TGN and TechRank

It is important to note that this validation evaluates the model's ability to preserve the relative ranking of influential companies rather that its precision in predicting individuals future links. As shown in Section 5.4, link-level prediction remains challenging Precision@$K \approx 0$ for $K < 2000$. This inconsistency between ranking performance and link-level precision could be attributed to TechRank's aggregation mechanism: because influence scores are computed by summing contributions from many predicted links, systematic prediction biases (e.g., uniform over-prediction across companies) preserve relative orderings even when absolute predictions are inaccurate. The positive Spearman correlation ($\rho = 0.3746$) suggests that the model's errors could exhibit systematic properties (by opposition to random errors).

[27] identified three types of systematic biases in link prediction algorithms: graph distance bias (over-predicting short distance links between nodes at distance 2), node degree bias and community structure bias (over-predicting intra-community links). These biases could preserve relative rankings through TechRank's aggregation mechanism even when individual link predictions are inaccurate: if the model exhibits uniform over-prediction across companies (e.g. predicting 50% more links for all nodes), the relative influence ordering remains stable despite low precision. The results 11 in Section 5.4 show the error is random rather than systematic.

5.4 Limitations

Dataset: One major limitation of the dataset is the small number of timestamps available for each company. In the dataset used for this project, the maximum number of timestamps per company does not exceed ten, and some companies have only two or three recorded events. This inconsistency and imbalance can lead to poor temporal information and weak representations for certain companies.

In addition, the dataset does not provide information about how the invested money is used inside the company. Such information would be highly valuable to better understand technology trends within companies and to improve prediction accuracy.

Future work could address these limitations by artificially generating additional chronological events, along with corresponding investment amounts, in order to provide a fixed and sufficiently large number of interactions per company. This would allow the model to learn richer temporal representations of investor–company relationships. Moreover, weighting the technologies associated with each company could help emphasize more relevant or emerging technologies.

Bipartite Graph: In this project, information was mainly incorporated into the edges rather than into the nodes, mainly for simplicity and for direct compatibility with the TGN model. However, richer node representations could improve the quality of the learned embeddings. Useful additional features could include node degree, the number of technologies a company works on or the number of investments made by an investor. These features could improve the reliability of the results but would require adapting the model architecture.

Furthermore, graph directionality was not explicitly modeled in this work. Although the direction Investor $\rightarrow$ Company is implicit, it was not enforced in the implementation. As noted in [25], ignoring edge direction can lead to a loss of important information and may negatively affect prediction performance. Incorporating directed edges would therefore be a meaningful extension of this work.

Model: Several loss functions were tested to mitigate undesirable effects caused by the structure of the graph and the strong data imbalance. These losses introduce additional hyperparameters, which were not optimized in this project, although for some hyperparameter, for the centrality method, influence and general behavior were analyzed earlier in the report.

In our evaluation framework, the model aims to predict future investor–company interactions at a given prediction time. The test set is temporally split into a history part and a future part. All future interactions are withheld during evaluation and used only as ground truth.

After updating the model memory using historical interactions, the model assigns a probability score to all possible investor–company pairs at the prediction timestamp. Low-confidence predictions can be filtered using a probability threshold τ. The remaining candidate pairs are then ranked according to their predicted scores.

Model performance is evaluated using Precision@K, defined as:

$$\text{Precision@K} = \frac{\text{Number of true future links in top-K predictions}}{K}$$

This metric measures the proportion of correct predictions among the K highest-ranked candidates. Unlike Recall@K (commonly used with negative sampling), Precision@K is appropriate when evaluating predictions over the entire search space, as it directly quantifies the reliability of top recommendations rather than coverage.

Concretely, Precision@K answers the following question: "If the model recommends K potential investments, how many of them actually occur in the future?" This metric evaluates the model's ability to correctly prioritize relevant future investors among a very large set of possible pairs (approximately 200,000 in our case), while strictly avoiding any access to future information. The corresponding pseudocode is provided in Appendix A.5 2.

K	**With threshold** ($\tau = 0.45$)		**Without threshold**	
	Precision@K	TP / K	Precision@K	TP / K
10	0.0000	0 / 10	0.0000	0 / 10
20	0.0000	0 / 20	0.0000	0 / 20
50	0.0000	0 / 50	0.0000	0 / 50
100	0.0000	0 / 100	0.0000	0 / 100
200	0.0000	0 / 200	0.0000	0 / 200
500	0.0000	0 / 500	0.0000	0 / 500
1000	0.0000	0 / 1000	0.0000	0 / 1000
2000	0.0040	8 / 2000	0.0040	8 / 2000
5000	0.0016	8 / 5000	0.0020	10 / 5000
10000	0.0008	8 / 10000	0.0016	16 / 10000

Table 10: Precision@K results with and without probability threshold ($\tau = 0.45$)

Company ID	**True Links**	**Predicted Links**	**TP**	**FP**
147	5	13	0	13
118	3	1	0	1
85	5	1	0	1
188	7	169	7	162
32	4	145	1	144
41	5	147	0	147
144	1	1	0	1
179	2	175	0	175
53	1	1	0	1
1	0	348	0	348

Table 11: Company-level link prediction results for the first 10 companies corresponding to K = 5000 with the probability threshold.

Table 10 reports the Precision@K results with and without a probability threshold ($\tau = 0.45$). Meaningful precision values appear only for large values of K ($K \geq 2000$), indicating that while the model can identify relevant future links, its top-ranked predictions remain challenging.

Table 11 presents a representative subset of company-level link prediction results. For each company, we report the number of true future links, predicted links, and the resulting true and false positives. The selected companies illustrate two key behaviors: companies with actual future activity that nonetheless receive mostly incorrect predictions and companies with no future activity that attract a large number of false positive predictions. This highlights a highly uneven and noisy distribution of prediction errors across companies. Such behavior can be attributed to a combination of strong class imbalance, the model's reliance on historical interaction patterns, the absence of rich node features and the limited size of the temporal validation set, leading to largely random, high-variance errors rather than systematic bias as [27] suggests. These results suggest that reliance on historical interaction patterns may play a role, as changing the temporal split to $\rho = 0.5$ leads to a non-significant correlation (p = 0.314, r = 0.175).

6 Conclusion

This project shows that combining TechRank with Temporal Graph Networks can improve how we forecast disruptive technologies. TechRank measures which companies are currently influential, while TGN predicts how the network will change over time. Together, they help identify companies that are likely to become much more important in the future.

However, there are important limitations. The dataset is small, funding data has biases toward well-funded companies, and predicting disruption is fundamentally difficult. The moderate correlation shows that technological disruption remains hard to forecast.

Furthermore, prediction errors in social network structures can arise from both systematic biases and random noise. Systematic biases, as identified by [27], include graph distance bias (over-predicting links between nodes at distance 2), node degree bias (favoring high-degree nodes) and community structure bias (over-predicting intra-community links). These structural biases can disproportionately affect link prediction accuracy in bipartite investor-company networks where hub structures and degree imbalances are prominent. Additionally, random errors comes from limited temporal information, sparse interaction patterns, and the inherent randomness of funding events contribute to high prediction variance. While our temporal validation results suggest that the model's errors may be largely random rather than systematically biased, evidenced by the preservation of relative rankings despite low precision at individual link level, future work should employ more rigorous diagnostic methods to disentangle systematic from random error components and develop mitigation strategies accordingly.

Despite these challenges, this work provides a useful foundation for early identification of emerging technologies in cyber defence.

6.1 Summary of contributions

This project makes several contributions to technology forecasting. The primary contribution is combining TechRank with Temporal Graph Networks to address their complementary limitations. TechRank measures current influence but lacks temporal awareness, while TGN predicts future network evolution but does not quantify influence. Together, they identify companies positioned for significant influence growth.

The project established a temporal validation procedure using the Crunchbase dataset on quantum computing technologies. The results have potential evidence that combining centrality methods with temporal graph learning could identify companies on disruptive trajectories.

Additionally, the work provides implementation guidance, including the adaptation of bipartite graph structure for TGN compatibility, strategies for handling extreme data imbalance through degree contrastive loss and methods for interpreting ranking changes (ΔR) as disruptive signals. The comparative analysis of different loss functions (BCE, Focal Loss, DCL) offers practical insight into training temporal models on sparse, imbalanced investor-company networks.

6.2 Future work

This project represents an initial exploration of combining TechRank with Temporal Graph Networks for forecasting disruptive technologies. Several promising directions emerge for future research.

The current implementation uses basic TechRank parameters without extensive optimization. Future work should explore systematic hyperparameter tuning and alternative temporal graph models. Incorporating additional data sources such as patent databases and academic publications would provide richer representations of technological emergence. Automatic weighting of technologies based on strategic importance would also improve accuracy.

Expanding beyond quantum computing to larger and more diverse technology domains would demonstrate broader applicability. A more comprehensive evaluation should include expert assessments and validation against realized market outcomes.

Inspired by approaches used in weather forecasting models such as Aurora, future iterations could implement data assimilation methods. These techniques would allow the model to continuously correct predicted values by incorporating real-time observation as they become available. In the context of technology forecasting, this would mean updating predicted investment flows and company rankings whenever new funding rounds are announced, thereby improving forecast accuracy over time through a feedback loop between predictions and observations. This approach has proven highly effective in meteorological forecasting and could significantly enhance the temporal reliability of disruption detection [6].

An alternative to TGN could explore models from the Joind Embedding Predictive Architecture (JEPA) family or its vairants. JEPA-based models learn representations by predicting in abstract feature space rather than directly in input space, which could potentially capture higher level patterns of technological emergence that are not immediately visible in raw funding interaction data. This self-supervised approach might prove particularly valuable when labeled disruption examples are limited.

Finally, the integration of Agentic AI principles could enhance forecasting through continuous monitoring, autonomous parameter adjustment, and real-time alerts about emerging threats [15, 29]. These directions aim to transform the current proof-of-concept into a reliable tool for identifying emerging disruptive technologies.

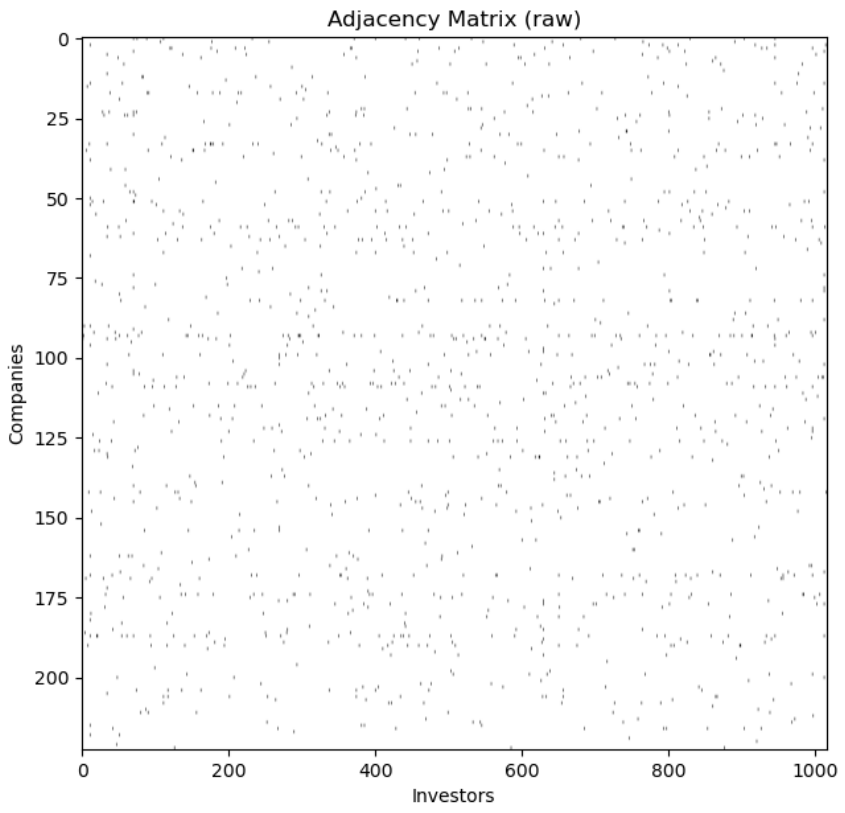
A Appendices

A.1 Hyperparameters Settings

Parameter	Value	Description
`--data`	crunchbase	Dataset name (custom)
`--bs`	200	Batch size
`--prefix`	tgn-attn	Prefix for naming checkpoints
`--n_degree`	10	Number of sampled neighbors (default value)
`--n_head`	2	Number of attention heads
`--n_epoch`	100	Number of epochs
`--n_layer`	1	Number of graph layers (default value)
`--lr`	0.0001	Learning rate
`--patience`	10	Patience for early stopping
`--n_runs`	6	Number of independent runs
`--drop_out`	0.1	Dropout rate
`--node_dim`	200	Node embedding dimension
`--time_dim`	200	Temporal embedding dimension
`--message_dim`	200	Message dimension
`--use_memory`	True	Enable memory module
`--embedding_module`	graph_attention	Graph attention embedding module
`--message_function`	identity	Identity message function
`--memory_updater`	gru	GRU memory updater
`--aggregator`	last	Last neighbor aggregation (default value)
`--memory_dim`	200	Memory dimension per node
`--use_source_embedding_in_message`	False	Use source embedding in message
`--use_destination_embedding_in_message`	False	Use destination embedding in message
`--use_wandb`	True	Enable Weights & Biases for experiment tracking
`--wandb_project`	tgn-experiments	WandB project name
`--prediction_threshold`	$\tau = 0.45$	Filter noisy probability after the prediction.
`--threshold`	0.001	$V_{threshold}$
`--alpha --beta`	(0.3, -5.0)	Parameters for TechRank.

Table 12: List of parameter values used during the training phase.

A.2 Adjacency Matrix Visualization



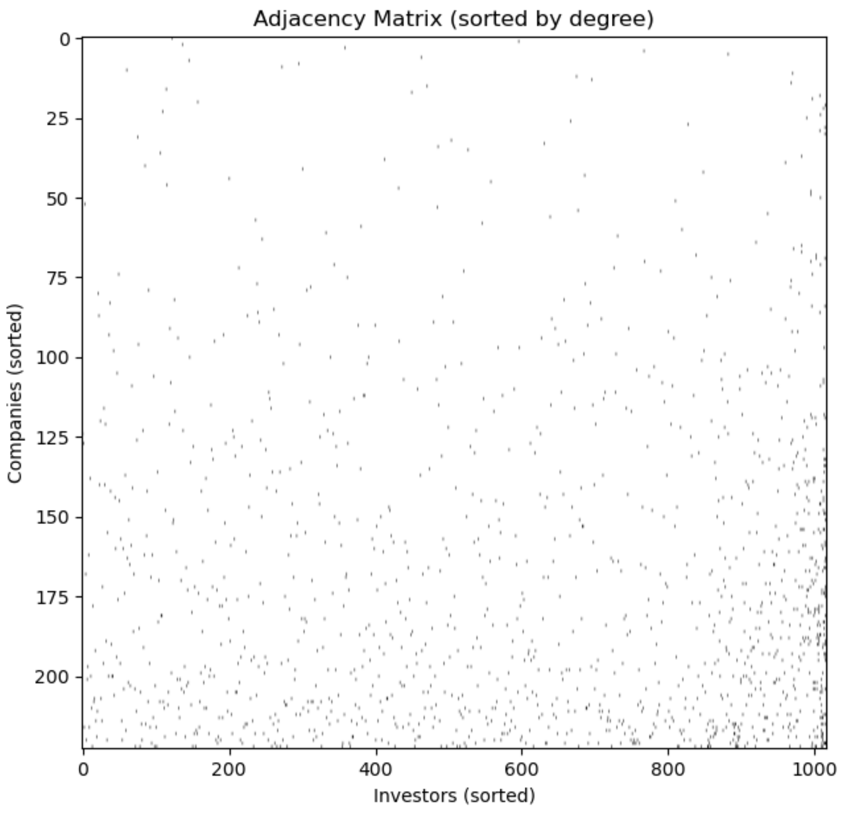


Figure 10: Visualization of the adjacency matrix of the investor company bipartite graph before and after degree reordering. Black dot represent a connection whereas white none.

A.3 Full Graph Visualization

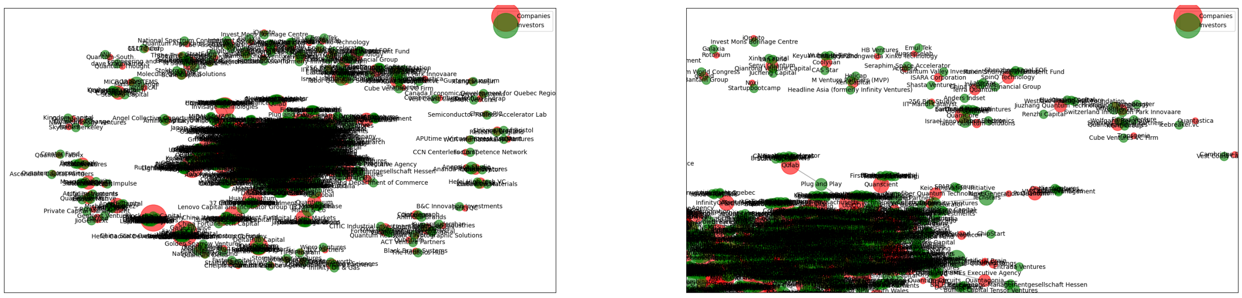


Figure 11: Visualization of all the nodes Investors-Companies with their edges. The size of the node represent the degree of it. The bigger the node the higher the degree and vice versa.

A.4 DCL Loss

Algorithm 1 DCL Loss Implementation

```
Require: Positive scores $\mathbf{p} \in \mathbb{R}^N$, negative scores $\mathbf{n} \in \mathbb{R}^N$
Require: Source node degrees $\mathbf{d}_s \in \mathbb{R}^N$
Require: Positive target degrees $\mathbf{d}_t^+ \in \mathbb{R}^N$, negative target degrees $\mathbf{d}_t^- \in \mathbb{R}^N$
Require: Temperature $\tau$, degree exponent $\alpha$
Ensure: Scalar loss value $\mathcal{L}$
 1: Compute degree weights
 2: for $i = 1$ to $N$ do
 3:     $w_s[i] \leftarrow \max(\mathbf{d}_s[i], 1)^{-\alpha}$
 4:     $w_t^+[i] \leftarrow \max(\mathbf{d}_t^+[i], 1)^{-\alpha}$        ▷ Positive destination
 5:     $w_t^-[i] \leftarrow \max(\mathbf{d}_t^-[i], 1)^{-\alpha}$        ▷ Negative destination
 6:     $w_{\text{pair}}^+[i] \leftarrow w_s[i] \cdot w_t^+[i]$
 7:     $w_{\text{pair}}^-[i] \leftarrow w_s[i] \cdot w_t^-[i]$
 8:     $w_{\text{avg}}[i] \leftarrow \frac{w_{\text{pair}}^+[i] + w_{\text{pair}}^-[i]}{2}$
 9: end for
10: Normalize scores by temperature
11: $\mathbf{p}_{\text{norm}} \leftarrow \mathbf{p}/\tau$
12: $\mathbf{n}_{\text{norm}} \leftarrow \mathbf{n}/\tau$
13: Compute contrastive loss
14: for $i = 1$ to $N$ do
15:     $\ell_{\text{base}}[i] \leftarrow -\log\left(\frac{\exp(\mathbf{p}_{\text{norm}}[i])}{\exp(\mathbf{p}_{\text{norm}}[i]) + \exp(\mathbf{n}_{\text{norm}}[i])}\right)$
16:     $\ell_{\text{weighted}}[i] \leftarrow \ell_{\text{base}}[i] \cdot w_{\text{avg}}[i]$
17: end for
18: Aggregate loss
19: $\mathcal{L} \leftarrow \frac{1}{N}\sum_{i=1}^{N} \ell_{\text{weighted}}[i]$
20: return $\mathcal{L}$
```

A.5 Temporal Validation Procedure

Algorithm 2 Temporal Validation Procedure for TGN with Configurable Split Ratio and Probability Threshold.

```
Require: Train set $\mathcal{D}_{train}$, validation set $\mathcal{D}_{val}$, test set $\mathcal{D}_{test}$
Require: Temporal Graph Network (TGN) model with memory
Require: Link predictor
Require: Split ratio $\rho \in (0, 1)$
Require: Probability threshold $\tau \in [0, 1]$
Require: List of evaluation cutoffs $\mathcal{K}$
 1: Sort $\mathcal{D}_{test}$ by timestamp
 2: $N \leftarrow |\mathcal{D}_{test}|$
 3: $N_{hist} \leftarrow \lfloor \rho \cdot N \rfloor$
 4: $\mathcal{D}_{history} \leftarrow \mathcal{D}_{test}[1 : N_{hist}]$
 5: $\mathcal{D}_{future} \leftarrow \mathcal{D}_{test}[N_{hist} + 1 : N]$
 6: $split\_timestamp \leftarrow \max(\mathcal{D}_{history}.\text{timestamps})$
 7: Build history neighbor finder (no data leakage)
 8: $\mathcal{D}_{hist} \leftarrow \mathcal{D}_{train} \cup \mathcal{D}_{val} \cup \mathcal{D}_{history}$
 9: $history\_neighbor\_finder \leftarrow \textsc{BuildNeighborFinder}(\mathcal{D}_{hist})$
10: Reset TGN memory
11: for each interaction $(u, v, t)$ in $\mathcal{D}_{train} \cup \mathcal{D}_{val}$ (chronological) do
12:     Compute temporal embeddings using TGN
13:     Update memory of nodes $u$ and $v$
14: end for
15: for each interaction $(u, v, t)$ in $\mathcal{D}_{history}$ (chronological) do
16:     Compute temporal embeddings using $history\_neighbor\_finder$
17:     Update memory of nodes $u$ and $v$
18: end for
19: Prediction time $t_p \leftarrow split\_timestamp$
20: Initialize empty list $\mathcal{P}$
21: for each candidate pair $(u, v)$ do
22:     Compute embeddings of $u$ and $v$ at time $t_p$
23:     $s_{u,v} \leftarrow \textsc{LinkPredictor}(u, v)$                ▷ link probability
24:     if $s_{u,v} \geq \tau$ then
25:         Append $(u, v, s_{u,v})$ to $\mathcal{P}$
26:     end if
27: end for
28: Build ground truth
29: $\mathcal{G} \leftarrow \{(u, v) \mid (u, v, t) \in \mathcal{D}_{future}\}$
30: Sort $\mathcal{P}$ by score in descending order
31: for each $K \in \mathcal{K}$ do                                      ▷ Precision@K
32:     $\mathcal{P}_K \leftarrow$ top-$K$ predictions from $\mathcal{P}$
33:     TP $\leftarrow |\mathcal{P}_K \cap \mathcal{G}|$
34:     Precision@$K \leftarrow$ TP$/K$
35: end for
```

References

[1] Airini Ab Rahman, Umar Zakir Abdul Hamid, and Thoo Ai Chin. “Emerging Technologies with Disruptive Effects: A Review”. In: *PERINTIS eJournal* 7.2 (2017). Corresponding authors: acthoo@utm.my; umar@moovita.com, pp. 111–128.

[2] Charu C Aggarwal. “An introduction to social network data analytics”. In: *Social network data analytics*. Springer, 2011, pp. 1–15.

[3] Seyed Amir Sheikh Ahmadi et al. “A Comparative Study of Methods for Measuring Node Influence in Complex Networks”. In: *Engineering Applications of Artificial Intelligence* (2025). Department of Mathematical Sciences and School of Engineering, RMIT University, Melbourne, Australia. DOI: 10.1016/j.engappai.2025.111088. URL: https://doi.org/10.1016/j.engappai.2025.111088.

[4] Fayyaz Ali, Irfan Ullah, and Shah Khusro. “An Empirical Investigation of PageRank and Its Variants in Ranking Pages on the Web”. In: *2016 International Conference on Frontiers of Information Technology (FIT)*. 2016, pp. 354–359. DOI: 10.1109/FIT.2016.071.

[5] Konstantinos Benidis et al. “Deep learning for time series forecasting: Tutorial and literature survey”. In: *ACM Computing Surveys* 55.6 (2022), pp. 1–36.

[6] Cristian Bodnar et al. *A Foundation Model for the Earth System*. 2024. arXiv: 2405.13063 [physics.ao-ph]. URL: https://arxiv.org/abs/2405.13063.

[7] Emmanuel Chris, Anita Johnson, and Grace Phonix. “Deep Learning vs. Traditional Machine Learning: Key Differences”. In: (Nov. 2024).

[8] *Crunchbase*. https://www.crunchbase.com. Accessed: 18-Oct-2025.

[9] GeeksforGeeks. *Decision Tree in Machine Learning*. Last updated: 30 Jun 2025. Accessed: 2025-02-17. June 30, 2025. URL: https://www.geeksforgeeks.org/machine-learning/decision-tree/.

[10] GeeksforGeeks. *Random Forest Algorithm in Machine Learning*. https://www.geeksforgeeks.org/machine-learning/random-forest-algorithm-in-machine-learning/. Last updated: 31 Oct 2025. Accessed: 2025-02-17. 2025.

[11] Abigail J. Hayes, Tobias Schumacher, and Markus Strohmaier. “What Do Temporal Graph Learning Models Learn?” In: *arXiv preprint arXiv:2510.09416* (2025). arXiv: 2510.09416 [cs.LG].

[12] Zhan ao Huang et al. “A neural network learning algorithm for highly imbalanced data classification”. In: *Information Sciences* 612 (2022), pp. 496–513. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2022.08.074. URL: https://www.sciencedirect.com/science/article/pii/S0020025522009847.

[13] Wei Ju et al. *Towards Graph Contrastive Learning: A Survey and Beyond*. 2024. arXiv: 2405.11868 [cs.LG]. URL: https://arxiv.org/abs/2405.11868.

[14] Maximilian Klein, Thomas Maillart, and John Chuang. “The Virtuous Circle of Wikipedia: Recursive Measures of Collaboration Structures”. In: *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. CSCW

'15. Vancouver, BC, Canada: Association for Computing Machinery, 2015, pp. 1106–1115. ISBN: 9781450329224. DOI: 10.1145/2675133.2675286. URL: https://doi.org/10.1145/2675133.2675286.

[15] Nir Kshetri. "Transforming cybersecurity with agentic AI to combat emerging cyber threats". In: *Telecommunications Policy* 49.6 (2025), p. 102976. ISSN: 0308-5961. DOI: https://doi.org/10.1016/j.telpol.2025.102976. URL: https://www.sciencedirect.com/science/article/pii/S0308596125000734.

[16] Srijan Kumar, Xikun Zhang, and Jure Leskovec. "Predicting dynamic embedding trajectory in temporal interaction networks". In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pp. 1269–1278.

[17] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. 2018. arXiv: 1708.02002 [cs.CV]. URL: https://arxiv.org/abs/1708.02002.

[18] Tiroshan Madushanka and Ryutaro Ichise. *Negative Sampling in Knowledge Graph Representation Learning: A Review*. 2024. arXiv: 2402.19195 [cs.AI]. URL: https://arxiv.org/abs/2402.19195.

[19] Anita Mezzetti et al. "TechRank: Modelling Portfolios of Cyber-Related Emerging Technologies". In: *arXiv preprint arXiv:2210.07824* (2022). URL: https://arxiv.org/abs/2210.07824.

[20] John A Miller et al. "A survey of deep learning and foundation models for time series forecasting". In: *arXiv preprint arXiv:2401.13912* (2024).

[21] Abdolreza Momeni and Katja Rost. "Identification and monitoring of possible disruptive technologies by patent-development paths and topic modeling". In: *Technological Forecasting and Social Change* 104 (2016), pp. 34–46. DOI: 10.1016/j.techfore.2015.12.002.

[22] Lawrence Page et al. *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report 1999-66. Previous number = SIDL-WP-1999-0120. Stanford InfoLab, Nov. 1999. URL: http://ilpubs.stanford.edu:8090/422/.

[23] Sancheng Peng et al. "Influence analysis in social networks: A survey". In: *Journal of Network and Computer Applications* 106 (2018), pp. 17–32.

[24] Emanuele Rossi et al. "Temporal Graph Networks for Deep Learning on Dynamic Graphs". In: *arXiv preprint arXiv:2006.10637* (2020). URL: https://doi.org/10.48550/arXiv.2006.10637.

[25] Niladri Sett, Bhargavi Kalyani I, and A. Rama Prasad Mathi. *Evaluating Link Prediction: New Perspectives and Recommendations*. Feb. 18, 2025. arXiv: 2502.12777v1 [cs.SI]. URL: https://arxiv.org/abs/2502.12777 (visited on 12/13/2025).

[26] Yong Shi et al. "Graph Influence Network". In: *IEEE Transactions on Cybernetics* 53.10 (2023), pp. 6146–6160. DOI: 10.1109/TCYB.2022.3164474. URL: https://doi.org/10.1109/TCYB.2022.3164474.

[27] Aakash Sinha, Rémy Cazabet, and Rémi Vaudaine. *Systematic Biases in Link Prediction: comparing heuristic and graph embedding based methods*. 2018. arXiv: 1811.12159 [cs.SI]. URL: https://arxiv.org/abs/1811.12159.

[28] Rakshit Trivedi et al. “Representation learning over dynamic graphs”. In: *arXiv preprint arXiv:1803.04051* (2018).

[29] Minhao Xiang, Dian Fu, and Kun Lv. “Identifying and Predicting Trends of Disruptive Technologies: An Empirical Study Based on Text Mining and Time Series Forecasting”. In: *Sustainability* 15.6 (2023). ISSN: 2071-1050. DOI: `10.3390/su15065412`. URL: `https://www.mdpi.com/2071-1050/15/6/5412`.

[30] W. Xing and A. Ghorbani. “Weighted PageRank algorithm”. In: *Proceedings. Second Annual Conference on Communication Networks and Services Research, 2004.* 2004, pp. 305–314. DOI: `10.1109/DNSR.2004.1344743`.